Ecole Internationale des Sciences du Traitement de l'Information

Rapport de stage

Développeur chez Figaro Classifieds

Auteur:
Thomas GIOVANNINI

Tuteur:
Florent DEVIN

Un rapport présentant le déroulement du stage au sein de FIGARO Classifieds pour le diplôme d'ingénieur de l'EISTI

September 2015

Remerciements

Merci à tous TODO ...

Contents

List of Figures

Chapter 1

Recherche du stage

1.1 Introduction

Le stage est une période très attendue de notre cursus au sein de l'EISTI car il nous permet de découvrir le monde du travail, notre futur métier et bien sûr d'appliquer les compétences que nous avons acqéries tout au long de notre scolarité. De plus, le stage de dernière année se déroule sur une période de 20 à 22 semaines, une durée longue qui permet d'exploiter les avantages d'un stage et va nous permettre d'avoir un recul important à l'issue de celui-ci.

1.2 La façon dont j'ai procédé

Le marché des offres de stage est bien plus caché que celui des offres d'emploi. En effet, beaucoup d'offres ne sont pas publiées, ce qui rend la recherche bien compliqué. Je vais expliquer dans la partie qui suit la façon dont je m'y suis pris pour trouver mon stage de fin d'étude.

1.2.1 Mes motivations

J'ai effectué l'an dernier un stage dans une très grosse entreprise, comprenant des équipes importantes et utilisant des méthodes de management anciennes. J'ai beaucoup travaillé seul en parfaite autonomie mais sans réel contacts avec les autres équipes. Je souhaitais ainsi cette année effectuer mon stage dans une structure plus petite, de la taille d'une startup ou d'une PME. Toujours en opposition avec ce stage précédent, je souhaitais

travailler en équipe, sur tes technologies nouvelles et en suivant des méthodes de management agiles. Enfin, Scala est un langage qui m'a particulièrement intéressé cette année et l'utiliser pour les projets que nous avons effectués m'a motivé à chercher un stage dans lequel je pourrais m'y améliorer.

1.2.2 Les entreprises contactées

Le réseau de l'EISTI m'a proposé de nombreuses offres de stage différentes. J'ai ainsi contacté des entreprises dont je recevais les offres par mail, qui étaient pour la plupart de grosses ESN. Ces dernières sont connues pour employer assez facilement les eistiens; ainsi, bien que complètement éloignées des souhaits que j'ai énnoncé, ces entreprises m'assuraient. J'ai aussi contacté quelques startup et PME, trouvées sur internet, en postulant à des offres de CDI ou en déposant des candidatures ouvertes. J'ai contacté des entreprises implantées sur Pau, Toulouse et Bordeaux et quelques une sur Paris.

1.2.3 Les résultats de mes recherches

J'ai reçu assez rapidement des réponses de la part des ESN contactées et même effectué quelques entretiens avec certaines, mais les sujets que l'on me présentait ne correspondaient pas à mes critères du tout. La plupart des projets se faisaient en J2E ou en C#. Les entreprises plus petites que j'ai contacté refusaient souvent mes demandes en tant que stagiaire puisqu'un CDI était recherché. Une partie de ces entreprises m'ont recontactées pendant mon stage via le mail que je leur avait laissé ou bien via Linkedin. FIGARO CLASSIFIEDS est l'une de ces PME que j'avais contacté pour un stage à la place d'un CDI. L'entreprise m'a recontacté et nous avons organisé une rencontre via Skype très rapidement avec une responsable Ressources Humaines et monsieur Marc Morel, directeur informatique du pôle Carrière. J'ai su rapidement, une semaine après l'envoi de mon premier mail, que j'étais pris pour ce stage.

1.3 Conclusion

Grâce aux projets innovants réalisés, au stage de deuxième année et à mon expérience universitaire en Ecosse, j'ai pu présenter aux entreprises que je rencontrais un CV intéressant pour un étudiant en fin de cursus, puisque mes expériences n'étaient pas uniquement théoriques et qu'elles étaient enrichissantes. Ainsi j'ai eu la possibilité de choisir un stage qui m'intéressait parmi plusieurs offres.

Chapter 2

L'environnement du stage

2.1 L'entreprise en général

FIGARO CLASSIFIEDS, filiale du GROUPE FIGARO, est une des sociétés Internet les plus importantes en France, avec 60 M€de C.A., 350 collaborateurs et 3,5 millions de visiteurs uniques dédupliqués par mois sur l'ensemble de leurs sites. Anciennement connue sous le nom d'ADENCLASSIFIEDS, fusion des sociétés CADREMPLOI, KELJOB et EXPLORIMMO, cette société est depuis sa création dirigée par Thibault GEMIGNANI. Elle est présente sur 3 gros secteurs: l'Emploi, la Formation et l'Immobilier et elle a pour ambition de proposer aux internautes et aux professionnels le meilleur des médias et des solutions d'annonces classées en France. Ses marques-phares (CADREMPLOI, KELJOB, LE FIGARO ETUDIANT, EXPLORIMMO, PROPRIETES DE FRANCE...) allient puissance, affinité CSP+ et influence, comme autant de facteurs de différentiation par rapport à leurs concurrents. FIGARO CLASSIFIEDS réalise 80% de son chiffre d'affaires sur Internet, contribuant au développement numérique du GROUPE FIGARO, dont plus de 20% du chiffre d'affaires total est réalisé sur Internet.

2.1.1 Secteur d'activité

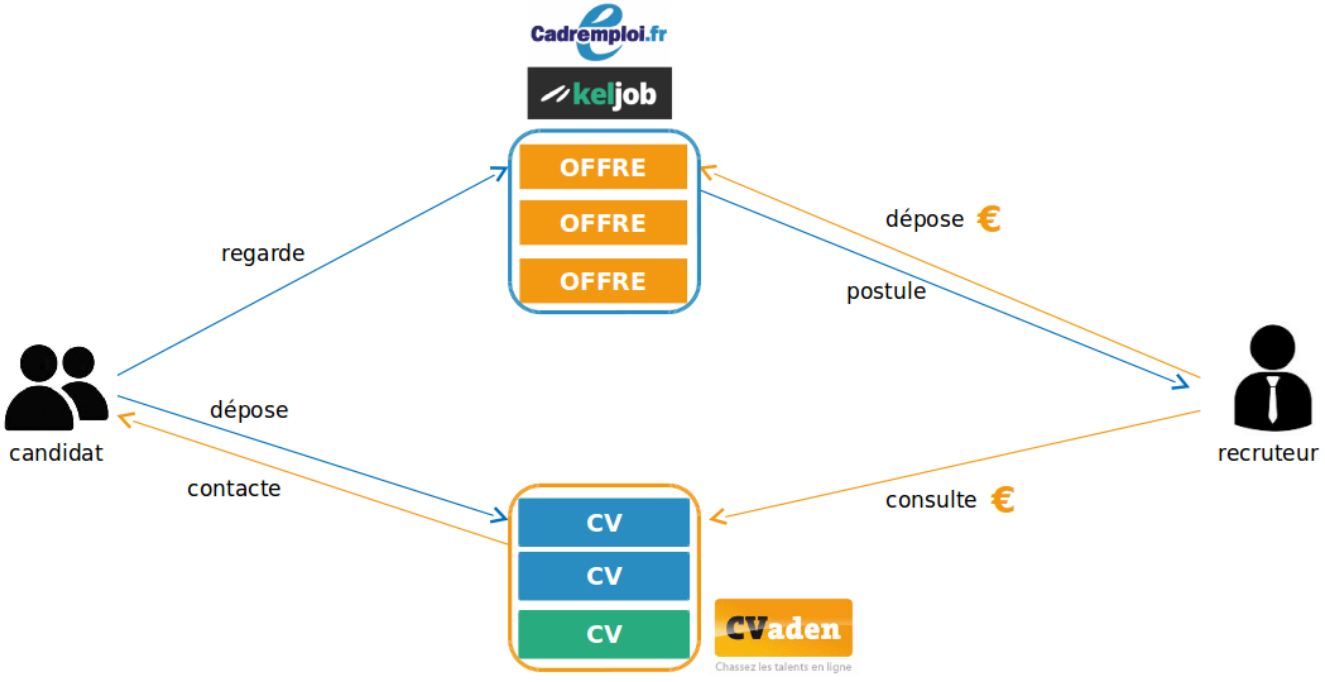
Le groupe est donc présent sur trois grand secteurs que sont l'Emploi, la Formation et l'Immobilier.

2.1.1.1 Emploi

Pionnier du marché des annonces Emploi (print, web, mobile...), FIGARO CLASSIFIEDS propose une offre unique sur le marché, fondée sur une approche multi-marques et multi-produits, lui permettant d'être à tous les carrefours de rencontres entre candidats et recruteurs:

- N°1 de l'Emploi privé sur Internet, avec les marques CADREMPLOI, KELJOB et CADRESONLINE
- N°1 de l'Emploi privé dans la presse nationale, avec LE FIGARO ECONOMIE
- N°1 de l'Emploi privé sur le Mobile, avec le succès des applications de CADREMPLOI et de KELJOB

Elle s'entoure aussi d'acteurs puissants dans le domaine de l'emploi, comme par exemple THE NETWORK, réseau international de recrutement N°1 (présent dans 136 pays).



Enfin, FIGARO CLASSIFIEDS se présente aussi comme un fournisseur de solutions RH, avec des produits comme CVAden (4 millions de CV), Adensourcing, CVmail ou Adenweb.

2.1.1.2 Formation

Fort de son leadership et de son expertise sur le marché de l'Emploi, FIGARO CLASSIFIEDS a été l'un des pionniers sur le marché des annonces de formation sur Internet depuis 2004. FIGARO CLASSIFIEDS peut ainsi accompagner les individus tout au long de leur carrière : formation initiale, en alternance, stages, recherche d'emploi, formation continue ou mobilité professionnelle via ses produits:

- Le site Internet et l'application mobile de KELFORMATION
- Les rubriques « Formation » des sites CADREMPLOI, KELJOB et CADRESONLINE
- La CV-thèque Alternance de KELFORMATION
- Les titres de presse LE FIGARO, LE FIGARO ECONOMIE et LE FIGARO ETUDIANT
- La plateforme de vidéos CAMPUS CHANNEL

2.1.1.3 Immobilier

À travers l'ensemble de ses médias, FIGARO CLASSIFIEDS dispose aussi d'une offre complète sur l'immobilier:

- L'Ancien, avec EXPLORIMMO, PROPRIETES DE FRANCE, et LE FIGARO
- Le Neuf, avec EXPLORIMMONEUF
- Les Loisirs, avec BELLES MAISONS A LOUER
- Les Solutions, avec IMMOVISION (logiciel de transactions, référencement, digital agency. . .)

2.1.2 Métiers

On retrouve différents corps de métiers dans cette entreprise, puisque 5 grandes directions se côtoient: Digital, Marketing, Communication et Édition, Ressources Humaines et Contrôle de Gestion. Toutes sont présente au même étage des locaux du FIGARO, ce qui permet une intéraction simplifiée. Je faisais partie, en ce qui me concerne, du pôle Digital dirigé par Laurent CHOLLAT-NAMY.

2.2 L'environnement de travail

Situés au cœur de Paris, dans le quartier de l'Opéra, les locaux du siège social de FIGARO CLASSIFIEDS reflètent une volonté commune : des locaux fonctionnels, créateurs d'échanges et de rencontres pour favoriser le travail collaboratif sous la forme d'open space à taille humaine. Puisque FIGARO CLASSIFIEDS est la fusion de plusieurs startups, un effort de maintient de la proximité pareil à celui caractérisant les cultures d'entreprise originelles a été fait. Toutes les strates de la hiérarchie se croisent quotidiennement et partagent des moments conviviaux notamment lors de divers évènements. Que ce soit à Paris, Lille, Lyon, Strasbourg, Toulouse, Bordeaux, Nantes, Rennes, Aix-en-Provence ou Sophia-Antipolis, FIGARO CLASSIFIEDS se vante de pouvoir compter 88% de collaborateurs satisfaits de leur environnement de travail et 85% plébiscitant une bonne ambiance au sein de leur service.

2.3 L'entreprise et son informatique

L'informatique occupe une place très importante dans le département Digital de FIGARO CLASSIFIEDS. Cette branche a majoritairement vocation a développer des applications web, c'est pourquoi de nombreux outils sont disponibles.

2.3.1 Outils, technologies et méthodes

La plupart des employés a accès à un ordinateur, et un compte est attribué à l'arrivé, permettant de gérer notamment les emails ou la pose de jours de congés. Les développeurs sont pour la majorité sous Ubuntu et développent via l'IDE IntelliJ Idea pour lequel une license est disponible. L'utilisation de l'ordinateur à disposition est plutôt libre, puisqu'il est permis d'installer des logiciels sans avoir à passer par des Demandes de Prestation Informatique. La majorité des équipes échelonne sa mise en production sous plusieurs étapes en déployant une nouveauté sur des serveurs internes particuliers avant

de procéder au déploiement public. Cela permet de sécuriser les nouvelles versions du logiciel puisqu'il est ainsi possible de s'apercevoir d'un problème avant qu'il ne soit visible par le public.

2.3.2 Outils de travail collaboratifs

Des outils sont aussi disponibles pour permettre aux équipes de travailler ensemble plus facilement.

Redmine Redmine est un logiciel de gestion de projet open source utilisé chez Figaro CLASSIFIEDS pour notamment lister les demandes client et qui permet de rendre compte en ligne du travail effectué. Lorsque l'on s'occupe d'une tâche, qu'on la termine ou que l'on a besoin de plus de renseignement, on le mentionne sur Redmine, ce qui permet de centraliser l'information et de la rendre disponible au reste de l'équipe.

Git Git est un outil de gestion de versions. Il est aussi utilisé au sein du pôle Média pour mettre en commun le code écrit par toute l'équipe. Il s'agit d'un outil que j'avais déjà beaucoup utilisé à l'EISTI mais d'une façon très simpliste, et j'ai appris au cours de mon stage de nombreuses nouvelles possibilité d'utilisation pour cet outil que nous n'avions jamais exploité à l'école.

Slack Au cours de mon stage, l'équipe Cadremploi dans laquelle je travaille s'est aussi mise à utiliser Slack, une sorte d'application de messagerie assez informelle qui permet de centraliser les échanges de l'équipe en un seul endroit. Slack dispose d'un outil de recherche par mot clef très performants qui le rend donc très utile pour rechercher une information. L'application permet aussi l'intégration de nombreux services dont Git, Jenkins ou Hangout, ce qui permet aussi une centralisation d'informations connexes utiles au projet en général. Il s'agit d'un outil que j'ai utilisé avec mon équipe au cours de mon PFE et qui s'est avéré très simple d'utilisation et très utile pour partager de l'information.

2.4 L'environnement de travail

Le pôle média de FIGARO CLASSIFIEDS se présente comme un énorme open space dans lequel plusieurs équipes évoluent. Bien que le nombre de service dans le bâtiment soit important, les relations n'y ont jamais été tendues. Les membres des nombreuses équipes, se côtoient sereinement, se respectent et s'entraident. J'ai réelement apprécié cette ambiance, et tout au long de mon stage je suis venu travailler dans un bon état d'esprit, sans contrainte mais avec au contraire beaucoup de plaisir.

2.4.1 Mon intégration

Mon intégration au sein de l'équipe Cadremploi s'est très bien déroulée et l'accueil a été réussi. L'organisation de séances sportives (escalade, Insanity, ...) ou d'afterworks a d'ailleurs aidé à souder cette équipe jeune et m'a aussi permi de rencontrer des gens dans des circonstances qui facilitent la mise en place de bonnes relations. D'autre part, il est courant de voir les membres de la direction et bien qu'un respect certain se fasse sentir, le contact est plutôt détendu.

2.4.2 Mon équipe: Cadremploi

J'ai effectué mon stage au sein de la très sympathique équipe Cadremploi qui est constituée de 9 personnes. Laëtitia Bukiatme est la Project Owner de l'équipe, Arnaud Brégère et Mathias Balussou sont deux développeurs front-end, Mickaël Barroux, Ludovic Iggiotti, Vincent Damery et Brice Friedrich sont développeurs Backend, tout comme Gaël Lazzari, techlead de l'équipe et Laurent Pageon mon maître de stage. Cette équipe avait pour but à mon arrivée de sortir une nouvelle version pour l'espace recruteur du site Cadremploi. Il s'agit d'une plateforme permettant aux recruteurs de déposer leurs annonces pour que celles-ci soient visibles sur l'espace public du site cadremploi.fr. A ce projet s'est ajouté la refonte de la Home de l'espace public du site cadremploi.fr fin août auquel j'ai aussi participé.

Chapter 3

Être développeur à Cadremploi

Le site Cadremploi.fr est découpé en différentes parties qui sont en réalité des applications à part. La partie publique est celle dans laquelle un utilisateur peut rechercher une offre en fonction de critères et y postuler. Une seconde partie, destinée aux recruteurs, leur permet de saisir une offre d'emploi qui sera ensuite consultable depuis la partie publique. J'ai travaillé durant ce stage sur la partie concernant les recruteurs, ou "Espace Recruteur". Il était en effet à mon arrivée difficilement maintenable, utilisable et son interface était obsolète. Une refonte totale en était nécessaire.

Mon stage consistait ainsi à participer avec l'équipe Cadremploi à la création d'un nouvel Espace Recruteur, plus moderne. Je vais détailler dans cette partie le fonctionnement du nouvel espace recruteur ainsi que la façon dont l'équipe s'est organisé pour le développer.

3.1 Description générale de la nouvelle application

L'application Espace Recruteur permet à des recruteurs d'entreprise de publier dans la partie publique du site Cadremploi une offre destinée à des cadres et cadres supérieurs. Il s'agit néanmoins de plus qu'un simple formulaire. En effet, en plus du remplissage d'un formulaire esthétique permettant de décrire l'offre qu'il souhaite déposer, l'espace recruteur propose plusieurs services. Le recruteur ayant un compte sur l'espace recruteur du site Cadremploi dispose d'un panneau de contrôle lui permettant de gérer ses offres et d'en consulter les informations associées. De plus, une équipe de Figaro CLASSIFIEDS supervise via une seconde application dédiée, la publication des offres.

3.1.1 Backoffice

Le backoffice est une application interne à Cadremploi permettant de gérer toutes les offres existantes. Chaque offre créé par un recruteur se doit d'être complètement sous contrôle de manière à pouvoir gérer tout cas inattendu. Ainsi, cette application est développée en parallèle à l'espace recruteur par notre équipe, comme un panneau de gestion de toutes offres rédigées par les recruteurs. Un backoffice existe déjà pour l'espace recruteur courant, mais les nouveautés apportés dans cette nouvelle version demandent une telle adaptation que le développement d'un nouveau backoffice est nécessaire. Les

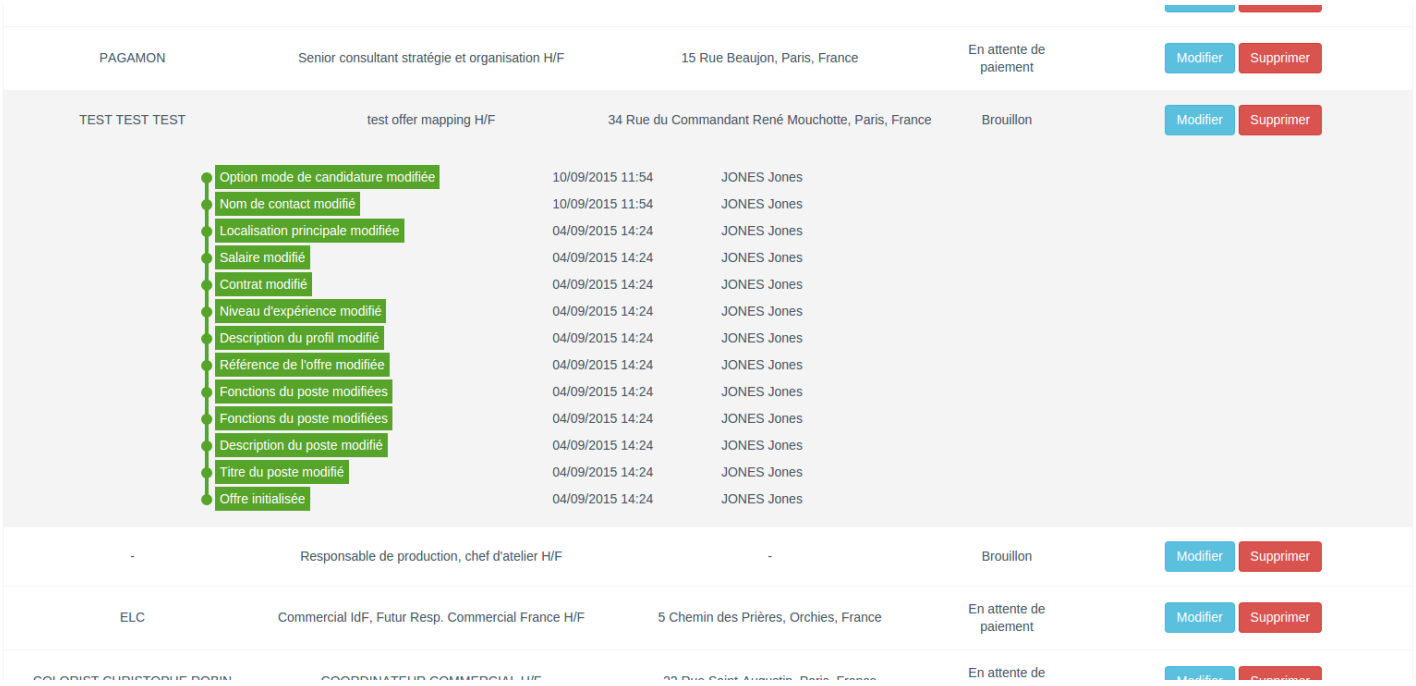


FIGURE 3.1: Le suivi de l'évolution du client dans le backoffice

fonctionnalités nécessaires sur ce nouveau backoffice sont:

- Le login "en tant que" un autre recruteur: Il est possible depuis le backoffice de se connecter pour éditer une offre comme si on était le recruteur qui l'avait rédigé et ainsi y apporter tous les changements désiré. Cela permet un contrôle parfait sur une offre puisque toutes les actions permises pour le recruteur le sont aussi pour le superviseur. Il est ainsi possible de gérer les problèmes potentiels en effectuant les modifications nécessaires à la place du recruteur.
- La gestion de la discrimination des offres en fonction de leur contenu: Une des fonctionnalité de l'espace recruteur est la détection de termes discriminants dans les champs textuels d'une offre. Ainsi, si une offre est détectée comme présentant des termes discriminants, elle n'est pas directement publiée sur le site. Une offre discriminante ne sera visible que sur le backoffice d'où elle sera relue par des responsables de la relation client (RC). Les RC sont chargés de traiter, si nécessaire avec le client, les offres discriminantes afinde les rendre acceptables dans les plus brefs délais. Le but de cette procédure est de ne pas handicaper le client et de régler les problèmes rencontrés rapidement.
- Le suivi de l'évolution du client au cours de la création de son offre (**??**). Cette fonctionnalité permet à un RC de pouvoir se rendre compte du parcours du client lorsque celui-ci rédige son offre (l'ordre dans lequel il a rempli les champs, le temps qu'il a mis à les remplir, ...) et donc de pouvoir l'aider au mieux lors de sa prise de contact. Les événements de modification effectués par le recruteur au cours de sa saisie sont donc visibles par le RC consultant une offre donnée.

3.1.2 Statistiques

Un système de statistiques mesurant la rentabilité de l'annonce est mis en place pour que le recruteur puisse suivre l'évolution de son offre. Ce système de statistiques est totalement externalisé de l'application et il permet ainsi à tout Figaro Classifieds de récupérer des statistiques sur l'utilsation de l'espace recruteur de Cadremploi et il a été développé par une équipe différente de celle de Cadremploi. L'incorporation de cette API de statistiques ainsi que la récupération des informations nécessaires était un point important dans le développement du nouvel Espace Recruteur.

3.1.3 Label qualité

Une des autres manière de garantir la qualité des offres présente sur Cadremploi est la création d'un label qualité. Il s'agit en somme d'un système de classement des offres qui

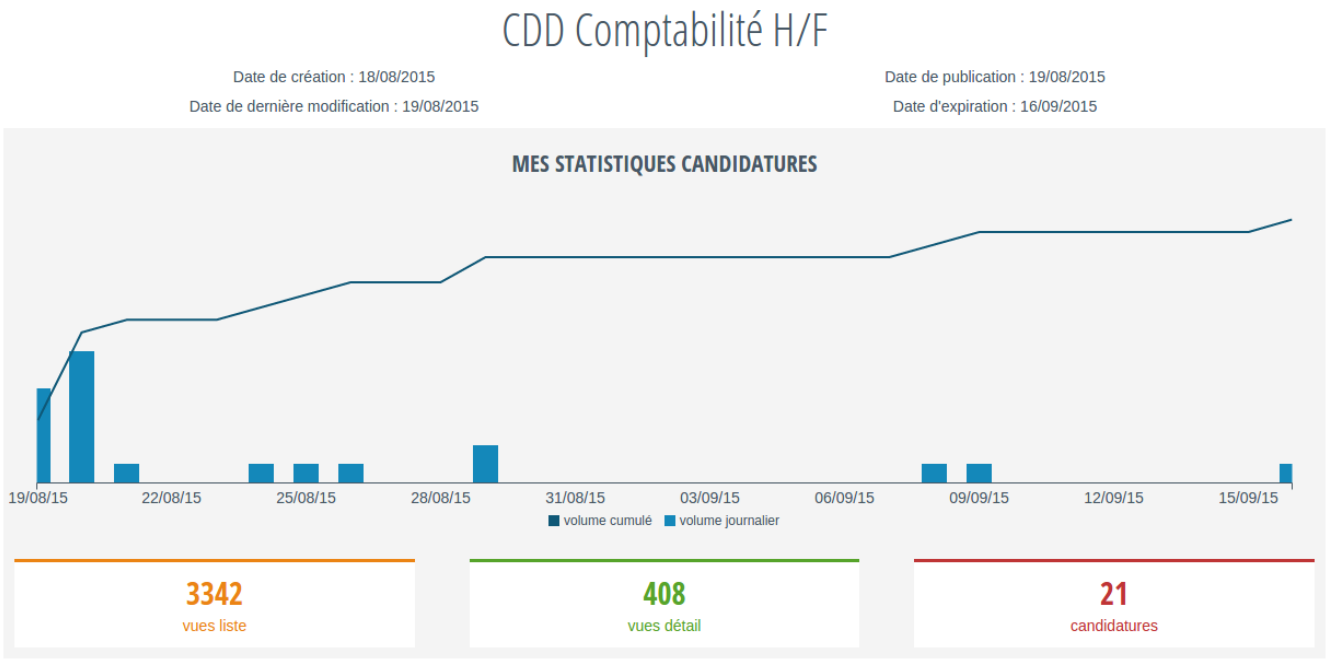


FIGURE 3.2: Les mesures statistiques proposées sur l'espace recruteur

fera passer les offres considérées comme étant de qualité devant celles ne l'étant pas dans la liste visible sur l'espace public. Il s'agit d'un système permettant d'inciter le recruteur à proposer des annonces avec des descriptions complètes ou des vidéos présentant l'entreprise, de manière à proposer un taux d'offres attractives plus important. Cette fonctionnalité n'est pas implémentée à ce jour.

3.1.4 Event sourcing

Enfin, une gestion fine des modifications apportés par les recruteurs à leurs offres sera faite. En effet, chaque modification faite par un recruteur sur son annonce sera enregistré de manière à suivre son évolution au cours du processus de mise en ligne d'annonce. Elle permettra par exemple aux RC traitant avec les recruteurs de pouvoir comprendre clairement la façon dont ils ont procédés et ainsi mieux les aider. Cette gestion permet de plus, comme je l'expliquerai plus tard, de pouvoir revenir à un état précédent d'une offre, dans le cas d'un recruteur ayant fait une erreur de manipulation.

Seule la mise en place du pattern Event Sourcing avait débuté à mon arrivée et j'ai ainsi participé au développement des autres fonctionnalités.

3.2 Cahier des charges et fonctionnement général du cycle de développement

3.2.1 Contenu du stage

Le sujet principal de mon stage concernait ainsi la refonte de l'Espace Recruteur du site cadremploi.fr. L'application actuelle se présente comme un espace réservé aux professionnels afin de les aider dans leur campagnes de recrutement. Grâce à ce service de e-commerce, il leur est possible d'accéder à tous les produits et services qu'offre Cadremploi et de payer directement en ligne avec leur carte de crédit ou par chèque après réception de la facture. Cette application web avait déjà fait l'objet d'un développement il y a plusieurs années et une refonte totale en était nécessaire, de manière à proposer des services nouveaux, un design plus actuel et des éléments de maintenabilité plus importants. Concrètement, les principaux objectifs de ce stage étaient les suivants:

- Développer de nouvelles fonctionnalités dans un environnement technique dynamique (Play! Scala, ElasticSearch, AngularJS)
- Maintenir le haut niveau de qualité et de tests présents
- Participer aux ateliers d'architecture et de cadrages techniques
- Echanger autour de bonnes pratiques avec les équipes de développement

3.2.2 Méthode de management

L'équipe Cadremploi, sur le projet Espace Recruteur notamment, suit une méthodologie Agile. Rapidement, les méthodes de type agile se veulent plus pragmatiques que les méthodes traditionnelles. Elles impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes. Les méthodes agiles prônent 4 valeurs fondamentales: l'équipe, qui doit être soudée et dans laquelle la communication est fondamentale, le fonctionnement de l'application, la collaboration avec les clients au delà de la négociation contractuelle ainsi que la flexibilité de la structure du logiciel, acceptant ainsi aisément le changement. Pour l'équipe Cadremploi, cela se caractérisait par des meetings quotidien devant un tableau rassemblant des tâches écrites sur des post-its, une mise en production des nouveautés développées chaque jeudi, la mise en place de tests unitaires et end-to-end ainsi que des revues de code pour chaque tâche effectuée.

Le stand-up meeting Comme dans de nombreuses méthodes Agiles, la communication au sein de l'équipe passe notamment par un stand-up-meeting quotidien ainsi qu'un affichage mural à base de posts-its représentant les tâches relatives à l'évolution de l'application. Le système de tickets post-it, des stand-up-meeting était très bien suivi bien que leur pratique ne soit pas un exercice figé. Le contenu des stand-up-meeting est par exemple passé au cours de mon stage de "Qu'est-ce que j'ai fait?, Qu'est-ce que je vais faire?, Ai-je des problèmes?" à des questions plus pratiques: "Qu'est-ce que j'ai fait de réellement impactant pour le reste de l'équipe? Quels dysfonctionnements ai-je remarqué?". Finalement, plutôt que l'évolution de chacun dans sa tâche, c'est l'évolution de l'application elle-même qui était alors discutée durant les stands-up, réduisant ainsi le temps de l'exercice, ce qui était nécessaire dans cette équipe de 10 personnes. De plus, suite à un amoncellement de tâche trop important qui produisait un confus global lors de la lecture du tableau, la mise en place de règles permettant de limiter le nombre de tickets en cours de traitement on été mises en place, permettant une meilleure organisation de l'équipe.

Cycle de vie d'une tâche Les post-its suivaient un cycle de vie dont les étapes rentraient dans les catégories suivantes:

- "Nouveau", il s'agit du statut des tickets qui viennent d'arriver sur le tableau et qui seront donc à traiter prochainement
- "En cours", il s'agit d'un ticket qu'un membre de l'équipe est en train de traiter.
- "Recette" équipe, lorsqu'une tâche est terminée, elle doit être validée par un membre de l'équipe. Ce système de team review permet d'éviter de nombreuses erreurs

au cours du développement de l'application: une vérification du code ainsi que des tests écrits sont effectués.

- "Recette" PO lorsque le ticket passe l'étape de recette équipe, il arrive en recette PO. Le Project Owner s'occupe donc de vérifier que les modifications apportées correspondent bien aux attentes et qu'elles ne perturbent pas les anciennes fonctionnalités. Si un problème est rencontré, le PO rédige ses retours et la personne responsable du ticket se charge de corriger les problèmes. Le ticket reste dans le même état mais un jeu d'étiquettes "Retours Recette" et "Corrigé" est déposé sur le ticket pour qu'il soit possible de suivre l'évolution des corrections.
- "Validé", le PO valide le ticket et les modifications seront mises en production sous peu
- "En attente", le ticket ne peut être traité pour le moment; une synchronisation avec une autre équipe est nécessaire ou bien des questions restent sans réponse.

Ainsi les membres de l'équipe choisissent arbitrairement un ticket qu'il traitent en collant une étiquette, un autocollant les représentant dessus et déplacent le ticket dans la catégorie à laquelle il correspond. Les tickets validés sont généralement mis en production chaque semaine, le jeudi, à l'exception de quelques mises en productions (MEP) exceptionnelles.

Organisation du tableau Le tableau sur lequel sont affiché les tickets est plutôt grand et plusieurs méthodes sont utilisés pour permettre à l'équipe de s'y retrouver rapidement. En effet, en plus des colonnes de catégories dans lesquelles sont rangés les tickets, un code couleur permettait de différencier les tâches techniques des tâches fonctionnelles, ainsi que les tâches traitant de parties de Cadremploi autre que l'espace recruteur (projet de refonte de la Home, ...). De plus, de petites étiquettes, en plus de celles représentant les membres de l'équipe, sont placées sur les tickets et permettent d'indiquer facilement si un des posts-its représente une tâche qui doit être mise en production dans la semaine (Cible MEP), ou si des retours suite à une recette (équipe ou PO) sont à traiter.

Référencement des tâches Enfin, les post-its représentant nos tâches rédigés sur un tableau se trouvent aussi référencés sur l'outil Redmine, une application web de gestion de projets, qui stocke toutes les informations concernant une tâche donnée. Ce stockage double de l'information est un peu lourd; en effet, il est nécessaire de déplacer le ticket sur le tableau et d'effectuer le même changement en ligne sur Redmine. Le point positif est qu'il permet à toute l'équipe, même en télétravail, d'accéder facilement à l'information

concernant une tâche, notamment aux différentes discussions ayant eu lieu concernant cette tâche. Il est cependant nécessaire de conserver l'affichage au tableau puisqu'il est bien plus visuel et permet à l'équipe d'échanger bien plus facilement que si le support était uniquement digital.

3.3 Une équipe nombreuse

Cadremploi, a mon arrivée, comptait alors 10 membres dont 9 développeurs. Il s'agit d'une équipe d'une taille importante pour un projet Agile. Ainsi, de l'organisation ainsi qu'une bonne coordination étaient nécessaires.

3.3.1 Revue de code

Le développement d'applications demande très souvent à une équipe de personnes de travailler ensemble. Plus l'équipe grandit plus il est compliqué de garantir une coordination correcte et une bonne organisation. Les équipes d'ingénieur sont spécialement soumises à ce genre de problèmes puisque du code est quotidiennement partagé entre plusieurs personnes au sein d'une entreprise. La revue de code aide à partager le savoir et les bonnes pratiques.

Le flux de production de l'équipe Cadremploi demande à chaque membre de l'équipe à passer un ticket terminé dans un état "R7 Equipe". Le ticket doit alors être revu par un autre développeur de l'équipe. Ce dernier doit analyser les changements fait au code, s'assurer qu'ils n'introduisent pas de récession dans l'application et vérifier que le code produit suit bien les normes de code imposées par l'équipe.

La revue de code par d'autres membres de l'équipe offre de nombreux avantages autres qu'une simple vérification du code. En effet, il est clair que n'avoir qu'un unique membre de l'équipe responsable d'un point critique sur l'application est dangereux. La revue de code distribue le savoir au sein de l'équipe et permet d'informer plusieurs membres d'un changement. De plus, elle stimule les conversations portant sur la structure du code, l'architecture de l'application et permet ainsi à de nouveaux venus, comme je l'étais, de s'adapter rapidement et donc de devenir productif plus rapidement.

Il ne m'est pas possible de réelement comparer le mode de fonctionnement de la revue de code de l'équipe Cadremploi. En effet, mes projets passés se faisaient dans des

équipes plus réduites, et du temps était rarement pris pour la revue de code. Cela ne nous empêchait pas de suivre des règles évitant la "possession" du code par un membre de l'équipe, mais rien de plus.

La revue de code effectuée avec rigueur se montre extrêmement utile, il s'agit d'une des activités qui m'a permi d'apprendre le plus au sein de l'équipe Cadremploi. En effet, en favorisant la communication mais aussi les critiques, positives, au sein de l'équipe, elle permet une amélioration continue des méthodes de l'équipe.

3.3.2 Concertation d'équipe

3.3.2.1 L'exemple de l'utilisation de Git

Git est un outil de contrôle de version flexible et puissant. Il permet notamment de mettre en commun du code de manière à ce que plusieurs personnes puissent contribuer à un même projet. Les options qu'il propose sont nombreuses et il est parfois compliqué de déterminer la meilleure façon de contribuer à un projet sans nuire aux autres contributeurs, la mise en commun de code étant parfois, si ce n'est souvent, source de conflits entre deux versions écrites par différentes personnes. Notre équipe est constituée de 9 développeurs, et nous ne pouvions échapper à de tels scénarios. C'est pourquoi quelques semaines après mon arrivée, nous avons convenu avec l'équipe de bonnes pratiques. Leurs buts étaient multiples, il s'agissait ici de limiter les problèmes de mise en commun de code souvent trop nombreux et bloquants, de simplifier la relecture par les autres membres de l'équipe lors des revues de code mais aussi de clarifier la lecture de l'historique d'évolution de l'application.

3.3.2.2 Les réunions

L'utilisation de Git a ainsi été sujet à une concertation de toute l'équipe, mais cela a été le cas pour de nombreux autres points. J'avais mentionné précédemment que la communication était un point très important dans une méthodologie Agile, et c'est pourquoi des points ponctuels étaient nécessaires. Il est souvent arrivé que notre équipe se réunisse dans le but de discuter autour de bonnes pratiques au niveau du code, de l'utilisation du fonctionnel qui est nouveau pour la plupart des membres de l'équipe issus d'un univers Java, mais aussi sur des sujets d'architecture et de modélisation. Ces réunions étaient souvent organisées surtout dans le but de présenter une ligne directrice, mais tout participant était écouté et l'idée principale se trouvait alors renforcée par les discussions qui émergeaient.

3.4 Aperçu des technologies et des techniques utilisées

3.4.1 Langages

3.4.1.1 Backend

Le framework Play Le nouvel espace recruteur est géré en backend via le framework Play! dans sa dernière version (2.4.2) et sous le langage Scala. Il s'agit d'un couple qui permet d'écrire du code rapidement et de façon maintenable. Play est l'un des framework les plus connus dans le monde Java et offre de nombreux avantages:

- Il supporte le rechargement à chaud. Il suffit de lancer Play via sa console en mode développement et il prendra automatiquement en compte à chaud les changements effectués sur le code, mais aussi les templates ou le routage. Cela contribue grandement au gain de productivité qu'offre Play.
- En plus de s'appuyer sur du code à typage statique, Play propose la sécurité du typage à d'autres endroits et notamment sur les templates ou sur le routage des différents contrôleurs. Un certain nombre de problèmes sont alors mis en lumière directement à l'étape de la compilation.
- Il permet d'exécuter des tests, notamment sur la couche web à plusieurs niveaux: On peut par exemple tester un contrôleur en démarrant un serveur web (donc via HTTP) ou, sans le démarrer, en appelant simplement le contrôleur avec le contexte qui va bien, le tout de manière simple et rapide à l'exécution.
- Il est sans état et basé sur des entrées sorties non bloquantes et permet ainsi une capacité à monter en charge très intéressante
- Il supporte nativement REST, JSON, Websocket entre autres et se présente donc comme un framework moderne

Un des points négatifs de ce choix en backend est l'utilisation de fait obligatoire de SBT qui nous a posé de nombreux problèmes. En effet, cet outil était souvent lent, compliqué d'utilisation et ralentissait souvent le développement notamment via ses résolutions de dépendances interminables.

Scala Scala a été le langage choisi pour le développement de l'espace recruteur pour sa flexibilité et sa richesse. Il s'agit d'un langage permettant d'implémenter des fonctionnalités complexes, notamment liées à l'asynchronisme, tout en maintenant un code clair et de qualité. Ce langage a aussi permi d'utiliser la puissance d'Akka ainsi que de nombreuses librairies propres à son écosystème et à celui de Java.

Akka Cadremploi utilise Akka pour diffuser des événements de manière asynchrone au sein des applications Espace Recruteur et Backoffice. Akka est aussi utilisé pour programmer l'exécution de scripts récurrents.

3.4.1.2 Frontend

Du côté front, l'équipe a utilisé le framework AngularJS ainsi que la librairie D3.js qui permettent des animations dynamiques. Ce choix de technologie peut être soutenu pour les points suivants:

- Angular permet une maintenabilité de l'application aisée, notamment en demandant une structure de type MVC au développeur, ainsi qu'en utilisant du HTMl, qui est déclaratif, pour définir l'interface utilisateur.
- Il s'agit de plus d'un framework flexible puisqu'il est possible de composer une application en

Cette stack reactive assure à l'équipe Cadremploi une consistence ainsi qu'une clareté architecturale. Elle permet la création d'une application responsive et fiable, donnant aux utilisatuers et aux clients une confiance résidant dans son architecture et son implémentation.

3.4.2 Environnements et outils

IntelliJ Le développement de l'espace recruteur s'est fait sous le framework Play! et IntelliJ Idea était l'IDE utilisé par l'équipe pour développer, puisqu'il offre une bonne intégration de ce framework. J'avais déjà souvent utilisé cet IDE pour différents projets que j'ai eu à rendre tout

au long de ma scolarité à l'EISTI, ainsi son utilisation ne m'a posé aucun problème.

Git L'équipe étant nombreuse et l'application étant sujete à grossir avec le temps, un outil de gestion de versions a été utilisée. Git est l'outil le plus répandu et le plus efficace connu, c'est ce que l'équipe Cadremploi a utilisé. Il s'agissait aussi d'un outil que j'avais beaucoup utilisé mais de manière très simpliste et j'ai appris à m'en servir d'une toute autre façon, bien plus complète, pendant ce stage.

Jenkins L'équipe Cadremploi utilise aussi Jenkins comme outil d'intégration continue pour déployer les nouveautés sur les différents environnement de production utilisés (recette, intégration), les mises en production et préproductions n'étant pas gérées directement par l'équipe. Cet outil avait été brièvement présenté lors du cours d'Outils de Développement mais jamais utilisé. Jenkins est un outil open source permettant de compiler et de tester un projet de manière continue. Il aide de fait les développeurs à intégrer facilement des changements à une application. Bien que nous avions été tenté de mettre en place un Jenkins avec mon équipe de Projet de Fin d'Étude, cela ne s'est jamais fait par manque de temps et puisqu'un besoin concret ne s'est jamais montré. Je n'ai pas eu de mal à l'utiliser puisqu'une fois configuré, cet outil est réelement simple d'utilisation. Plusieurs environnements de développement sont utilisés par Cadremploi comme pour les autres équipes de Figaro Classifieds:

- Environnement de développement, où les développeurs postent régulièrement les améliorations qu'ils mettent en place. Jenkins n'est pas utilisé pour cet environnement puisque les changements y sont publiés directement via Git.
- Environnement de recette, où la Project Owner vérifie que les changements apportés fonctionnent effectivement et les valide.
- Environnement de pré-production qui est un environnement qui ressemble autant que possible à l'environnement de production et où sont surtout testés les architectures
- Environnement de production qui est la dernière étape, contenant l'application utlisée réelement par les utilisateurs
- Environnement de'intégration, il s'agit d'un environnement supplémentaire qui est préféré pour les tests fait sur l'infrastructure du projet.

J'ai été habitué durant ma scolarité à l'EISTI à utiliser la majorité des outils présentés précédemment puisque j'étais uniquement étranger à Jenkins. J'ai malgré tout grandement progressé dans leur utilisation et particulièrement pour Git que j'ai redécouvert; le projet auquel je participais differais grandement en taille de ceux auquels j'avais contribué auparavant.

3.5 Architecture de l'application

L'espace recruteur est dispose d'une architecture interne assez singulière et je n'avais jamais eu à faire à ce genre d'application auparavant. En effet, l'équipe Cadremploi a mis en place un modèle d'architecture de type CQRS couplé à une gestion d'état basé sur de l'Event Sourcing.

CQRS: Command Query Responsibility Seggregation Dans les systèmes de gestion de donnée traditionnels, les commandes, c'est à dire la mise à jour des données, et les requêtes son exécutées sur un seul regroupement d'entitées regroupées dans une unique base de donnée. CQRS est un modèle d'architecture plutôt récent dont le principe repose, comme son nom l'indique, sur la séparation entre l'écriture et la lecture de l'information. Nous avons suivi lors du développement de l'espace recruteur ce pattern puisque la séparations des composants de traitement (les "commands") et de restitution (les "queries") de l'information offrait une architecture très intéressante de laquelle nous avons tiré de nombreux bénéfices tels que la suppression du risque d'effets de bord ou l'allègement des classes de service.

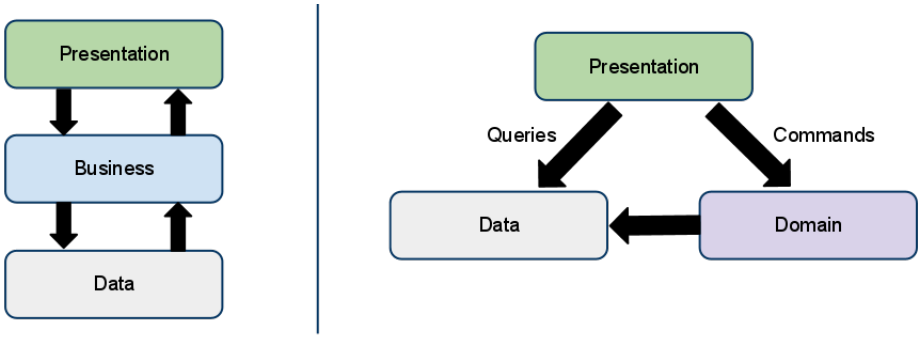
Event Sourcing La gestion commune de l'état d'un système consiste à enregistrer l'état courant des objets le composant et de reporter chaque changement effectué sur le système en modifiant directement son état, notamment via une mise à jour en base de donnée. L'idée fondamentale de l'Event Sourcing est d'assurer que chaque changement appliqué à l'état d'une application peut être capturée dans un objet de type événement. Ce n'est plus l'état du système qui est enregistré mais les événements qui ont mené le système à l'état dans lequel il est actuellement. Cette façon de penser permet notamment d'obtenir un log complet de tous les changements effectués, et donc une traçabilité ainsi qu'une aisance de debug importante. Il est ainsi possible de pouvoir reconstruire tous les états passés de l'application.

Je vais présenter dans cette partie ce type singulier d'architecture, montrer la façon dont il permet de mettre en place un système extensible et distribuable et enfin expliquer

la façon dont elle a été mise en place sur la nouvelle version de l'espace recruteur de cadremploi.fr.

3.5.1 Organisation d'une application basée sur une telle architecture

Dans une application, les besoins fonctionnels et non fonctionnels peuvent être différents selon que l'on s'intéresse à ses composantes de lecture ou d'écriture. Dans le cas d'une IHM, il est important que l'information que l'on souhaite afficher soit disponible à l'utilisateur sans qu'il n'ait à croiser lui-même différentes informations présentes sur différents écrans. Il est alors nécessaire d'aggréger et de filtrer les données et donc de les dénormaliser lorsqu'elles sont destinées à la consultation. D'autre part, lorsqu'un utilisateur souhaitera exécuter une action d'écriture, l'ensemble des données manipulées sera plus réduit et la problématique ne sera plus la même. La complexité se trouvera plutôt dans la vérification du respect de règles et il sera nécessaire d'utiliser des entités d'association pour ce faire. Ainsi, on privilégie dans ce cas une normalisation des données ainsi que leur intégrité. On voit alors que les besoins en écriture sont globalement transactionnels et une garantie de cohérence des données ainsi que leur normalisation est nécessaire, tandis qu'en lecture, les besoins sont davantage en dénormalisation des données ainsi qu'en scalabilité.



Architecture 3 Tiers

Architecture CQRS

Par opposition à une architecture du type 3 tiers, dont les services permettant d'accéder aux données se confondent avec ceux qui vont agir sur ces même données, l'architecture CQRS sépare volontairement les composants requêtant les données de ceux qui les modifient. Une telle séparation facilite l'organisation de l'application puisque des composants différents sont utilisés pour des problématiques différentes, mais permet aussi de profiter des avantages de différentes technologies sur ces composants et ainsi d'optimiser les performances de l'application.

Afin de pouvoir expliquer le fonctionnement d'une architecture de type CQRS, il est nécessaire de comprendre quelques concepts tirés du Domain Driven Design (DDD). Clarifions ainsi quelques termes que j'ai appris à connaître au cours de mon stage.

3.5.1.1 Les concepts tirés du Domain Driven Design

Le domaine Le domaine est la zone où est concentré toute la connaissance métier de l'application. On y trouve ainsi les objets nécessaires à l'application des règles métiers (la taille minimale pour un nom d'utilisateur, le nombre de chiffre dans un numéro de SIRET, ...) et pour des contrôles nécessaires à chaque actions ("je ne peux pas modifier une offre qui ne m'appartient pas"), des agrégats et des services notamment.

Les aggrégats Un aggrégat est un regroupement d'objets tirés du domaine qui est traité comme une unité. Il est généralement repésenté par une classe possédant un identifiant unique dans toute l'application mais on peut aussi trouver des représentations d'aggrégats via plusieurs classes. Dans ce second cas, une d'entre elle est dite classe racine et tout accès à l'aggrégat par l'extérieur se fait via cette classe tandis que toutes les autres restent interne à l'agrégat. Cette entité contient ainsi la logique du domaine (3.5.1.1) et permet d'effectuer des contrôles.

Les repositories (ou dépôts) Les différentes instances des aggrégats sont stockées dans un objet appelé repository (dépôt en français). Ces objets servent d'intermédiaires entre le domaine (3.5.1.1) et la base de donnée. On retrouve dans ces objets des méthodes du type 'save' qui permet de sauvegarder les changements relatifs à un aggrégat ou 'getById' qui permet de récupérer un aggrégat particulier.

3.5.1.2 La couche de modification des données: Command

Cette couche concentre toutes les modifications des données, qu'il s'agisse de création, de suppression ou de mise à jour. Une commande représente une action destinée à être exécutée, une intention, et n'est pas une simple demande d'altération de donnée. Généralement, une commande est représentée comme un appel de méthode encapsulée dans un objet; elle porte un nom explicite et ses champs contiennent les différents paramètres de l'action. Dans le cas de l'espace recruteur du site Cadremploi.fr, les commandes ont pour but d'enregistrer les altérations effectuées par les événements provenant du côté client, et on retrouve des commandes nommées 'ModifierTitrePosteCommand' ou 'PayerOffreCommand' par exemple, permettant de modifier le titre du poste dans

l'annonce ou de payer l'offre saisie respectivement. Ces commandes doivent être le plus atomique possible et l'expressivité de leur nom est important puisqu'il permet de clarifier la cause de la modification des données.

Une commande récupère via des repositories les agrégats qu'elle vise. Via les informations contenues dans ces agrégats et potentiellement à l'aide de différents services, les contrôles nécessaires à la validation de la commande peuvent être appliqués. Une fois que les différents contrôles sur la commande faits, on génère des événements de modification. Ces événements sont utilisés pour appliquer l'effet de la commande sur l'aggrégat, mais aussi pour informer la couche de lecture des données qu'une modification a eu lieu. Cette procédure permet de maîtriser totalement l'impact de chaque action sur le système. La façon d'effectuer la répercussion des modifications propres à la commande peut varier selon les architectures, mais c'est ainsi que cela fonctionne chez Cadremploi. L'essentiel ici est de ne pas oublier de mettre à jour la partie de lecture lorsqu'un changement survient pour éviter toute incohérance entre les couches de lecture et d'écriture.

Les méthodes de commandes prennent ainsi en paramètre des identifiants d'entités ainsi que des valeurs simples. Elles ne doivent rien renvoyer mais peuvent générer des exceptions. La mise en commun des patterns CQRS et Event Sourcing demande l'envoi des événements générés dans un bus en même temps que l'enregistrement des modifications sur l'aggrégat. Ces événements seront écoutés et utilisés pour la construction de la seconde couche de ce pattern, la couche de lecture des données.

3.5.1.3 La couche de lecture des données: Query

La partie Query de ce modèle se base sur le fait que les objets du domaine (3.5.1.1) sont volumineux et qu'il est possible de s'en passer. Cette couche fonctionne ainsi uniquement en lecture seule et aucune modification n'est apportée aux données. En effet le besoin représenté ici est celui d'une lecture dans un cas d'utilisation bien précis, l'objectif est d'aller extrêmement vite. Le pattern Query consiste alors à exécuter une requête précise en base et de restituer un objet de type Data Transfer Object (DTO) concis qui pourra être utilisé directement. Cela signifie que les contrôles sont réduits au minimum et que les DTOs utilisés sont des objets représentant exactement et uniquement le besoin en lecture, généralement des besoins IHM. Cette méthode permet de récupérer seulement les données dont on a besoin en une fois, en se passant ainsi de parcourir plusieurs tables à travers desquelles les données seraient éparpillées.

Projections L'utilisation de projections permet de construire ces DTO taillés sur mesure pour l'IHM. La projection est un concept aussi simple qu'important de l'Event Sourcing. Le but d'une projection est de dériver un état d'un flux d'événements. La projections souscris à un ou plusieurs flux d'événements de manière à aggréger les données qui l'intéresse uniquement. Elle dérive alors des informations particulières des événements émis depuis la couche de modification des données de manière à construire les DTO qui seront utilisés dans la couche lecture. En répercutant les changements qu'elles perçoivent, les projections mettent ainsi à jour la couche de lecture symétriquement à la couche d'écriture. Un de leur intérêt et qu'il est possible de projeter, pour tout flux d'événement, ces données à faire persister sur tout type de représentation structurelle, qu'il s'agisse d'une base de donnée SQL, NoSQL ou même un stockage en mémoire qui sera reconstruit depuis un flux d'événement local au redémarrage du serveur.

La construction de l'interface utilisateur se fait ainsi en requêtant ces projections, qui retournent des objets taillés pour être directement utilisés côté front. Les queries encapsulent alors ces requêtes aux projections, de manière similaire aux commandes encapsulant les actions. Néanmoins, contrairement à ces dernières, elles n'effectuent strictement aucune modification et renvoient un objet résultat de la requête.

L'architecture CQRS couplés à de l'Event Sourcing présentée théoriquement ici présente de nombreux avantages. Tout d'abord, le risque d'effet de bord est supprimé puisque les méthodes des services de lecture de donnée ne modifient jamais l'état du système. Le développeur peut donc les utiliser sans aucune crainte puisqu'aucun comportement anormal ne se fera ressentir sur le reste du système. De plus, le découpage en Command et Query, en plus de rendre indépendant l'implémentation de l'API de l'application de sa dynamique interne, facilite l'exposition des services puisque. En effet, les méthodes de lecture sont particulièrement adaptées pour être exposées en HTTP GET et celle d'écriture en POST. Cette approche demande néanmoins un certain effort d'adaptation et ajoutes certainement des lignes de code à écrire à l'équipe. Cela est facilement rattrapé par les bénéfices au niveau de la maintenance du code qui est très expressif, ainsi qu'au niveau de l'utilisation.

3.5.2 Implémentation

L'équipe Cadremploi a été en mesure de mettre en place cette architecture particulièrement intéressante dans le projet Espace Recruteur en se basant sur plusieurs concepts tirés du Domain Driven Design (DDD) et en utilisant diverses technologies récentes. Notamment, l'utilisation des acteurs Akka a permi une gestion confortable et performante

des événements que je traiterai dans les parties suivantes et l'utilisation d'ElasticSearch a offert une rapidité en lecture importante.

3.5.2.1 La séparation effective de la lecture et de l'écriture

L'espace recruteur utilise deux moyens différents pour accéder à la donnée en lecture et en écriture. En effet, les données reçues des commandes sont stockées dans une base PostreSQL sous forme d'événements tandis que les données utilisées en lecture sont accessibles via un index ElasticSearch. Cette utilisation de technologies dédiées est ainsi bien plus performant pour chaque type d'action.

L'écriture Les données sont stockées sous forme d'événements dans une base de données PostgreSQL et toute la partie écriture de l'application se fait sur cette base. Les données ne peuvent y être qu'insérées (append only), c'est à dire que les données déjà présente ne sont jamais modifiées. Une ligne de cette base représente un événement et contient ainsi la donnée relative à cet événement, sa date d'application sur le système ainsi que l'identifiant de l'aggrégat sur lequel il s'applique. C'est ainsi sur cette base que repose la structure d'Event Sourcing de notre application. En effet, l'idée fondamentale de l'Event Sourcing est qu'en enregistrant chaque événements survenant sur un système, on peut retrouver l'état de ce système à tout instant; cette base est la liste de tous les événements ayant eu lieu depuis le démarrage de l'application et est donc centrale à notre application.

Cohérence de la base Comme expliqué précédemment, plusieurs contrôles sont effectués avant la validation d'une commande pour assurer la validité de l'information insérée et donc la cohérence de la base. En effet, cette "liste d'événement" est la base de l'application de l'Espace Recruteur et l'information qui y est présente est utilisée pour construire/reconstruire l'application. Une corruption de la donnée qui y est présente n'est donc pas envisageable. Ce type de stockage est néanmoins extrêmement rapide, ce qui nous intéresse du point de vue des commandes puisqu'on essaye de donner à l'utilisateur un retour quasi immédiat sur son action.

La lecture Les données disponibles en lecture le sont depuis un index ElasticSearch qui est mis en place au démarrage de l'application. ElasticSearch est un moteur de recherche open source qui permet de disposer en quelques minutes seulement d'un moteur de recherche clusterisé, automatiquement sauvegardé et répliqué et interrogeable via une API REST.

Mise à jour asynchrone de la couche Cet index est mis à jour de manière asynchrone à la réception d'une intention de l'utilisateur. En effet, parallèlement à l'ajout d'un événement en base de donnée (Postgre), un message Akka est envoyé sur un bus interne à l'application et sera utilisé pour mettre à jour l'index ElasticSearch. L'envoi et le traitement de ce message sont fait, par les projections, de façon totalement asynchrone, de manière à ne pas ralentir l'envoi d'un feedback à l'utilisateur. Nous avons vu précédemment que l'exécution d'une commande est soumise à plusieurs contrôles, mais une fois que ceux-ci sont passés, l'action est validée et il est possible d'envoyer un retour à l'utilisateur de manière synchrone pour que celui-ci sache que son action a bien été prise en compte. Le traitement interne de l'action ne doit pas ralentir l'envoi de ce feedback.

Consistence des données Il est nécessaire que les données de la couche de lecture soient constamment mises à jour de manière à rester consistantes avec les données présentes dans la base de données PostgreSQL sans avoir pour autant à l'interroger. La souplesse offerte par Akka et sa gestion asynchrone des acteurs permet une mise à jour non bloquante de l'index ElasticSearch. Cela permet d'offrir à l'utilisateur un feedback immédiat, mais que la mise à jour de l'index ElasticSearch est différée. Concrètement, on peut affirmer que l'équipe Cadremploi a choisi d'avoir parfois du retard au niveau de sa couche de lecture. Par exemple, une modification du titre d'une offre depuis son espace recruteur ne pourra être visible sur le backoffice qu'une ou deux secondes après la modification effective. En effet, pour garantir un retour rapide à l'utilisateur, donc maximiser la réactivité de mon système, la cohérence des données est remise à plus tard (quelques milli-secondes plus tard).

En somme, les parties d'écriture et de lecture sont physiquement séparées dans l'application Espace Recruteur de Cadremploi. On en retire une réactivité forte du point de vue de l'utilisateur recruteur, mais aussi d'une puissance de requêtage importante apportée par l'index ElasticSearch.

3.5.2.2 Reste de l'infrastructure

Plusieurs outils sont utilisés de manière à faire fonctionner ce modèle, je vais les lister rapidement avant d'expliquer le fonctionnement concret du mécanisme des commandes et des queries au sein de l'application Cadremploi.

Akka Akka est un outil se basant sur le modèle des acteurs de manière à offrir une plate-forme permettant de construire des applications concurrentes et scalables. Ce modèle d'Acteurs permet une abstraction de haut niveau pour implémenter des services concurrents et parallèles, des modèles événementiels performants et non bloquants. C'est justement ce dont Cadremploi a besoin, puisque le nombre d'utilisateur et donc d'interactions avec le système tendent à être importants. L'abstraction qu'offre Akka et son modèle d'acteurs couplé à l'utilisation du langage Scala a permi d'écrire un code élégant et performant.

Kafka Kafka est un service offrant la fonction de système de message de type publish/subscribe partitionné et répliqué. Il permet un traitement en temps réel de la donnée mais offre aussi la possibilité de gérer des files d'attentes entre autres. Cet outil est utilisé au sein de Cadremploi puisqu'il offre un bus de données permettant, à la réception d'une commande, de publier un événement (publish), et à chaque projection ayant souscrit à ce type d'événement (subscribe), de le recevoir et d'agir en fonction de son contenu. Akka gère généralement ça grâce à son implementation d'un EventBus, mais l'application Cadremploi a besoin de discuter avec des applications externes, notamment le Backoffice mentionné précédemment, et c'est Kafka qui fait le lien entre ces différentes applications.

3.5.2.3 Les commandes

Dans l'espace recruteur de Cadremploi, une commande représente une action venant de l'utilisateur visant à modifier les données le concernant. Cela peut s'agir de son profil utilisateur, des entreprises qu'il gère ou bien des annonces qu'il a écrites.

L'interface

Envoi d'une commande L'interface de l'espace recruteur est pensée de sorte à ce que le recruteur qui l'utilise envoit des commandes à l'application de manière transparente. En effet, à chaque champ du formulaire rempli, une requête Ajax contenant les informations relatives à la modification effectuée est envoyée à l'application. Concrètement, on utilise un élément d'AngularJS appelé watcher permettant d'enregistrer un callback à exécuter lorsqu'une expression ciblée est modifiée. Cette technologie permet

des appels asynchrones à la fonction de callback et offre donc une fluidité d'utilisation à l'application. Le but de la fonction de callback est d'effectuer des vérifications côté client de la saisie du recruteur avant de générer un appel REST de type POST à destination du serveur de l'application. Cet appel sera interprété et traduit en une commande que l'on tentera d'exécuter.

Réception de la réponse La réponse retournée par le serveur après une requête REST permet d'informer le client de la bonne réception de son action. Lors de la saisie d'une offre, une bulle d'information est affichée uniquement si la réception s'est mal passée puisqu'on ne souhaite pas bombarder le client d'information. L'information

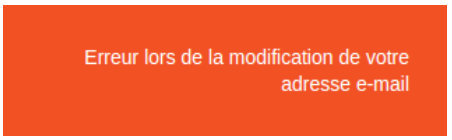


FIGURE 3.3: Message d'erreur

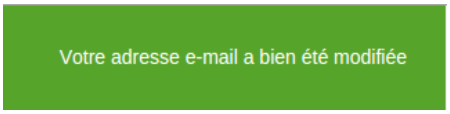


FIGURE 3.4: Message de succes

sur le retour est affichée dans tous les cas quand la modification concerne le profil de l'utilisateur. Ce retour est très rapide puisque le gros du traitement côté serveur, est fait de manière asynchrone.

Backend L'utilisation du pattern Command est rendu possible grâce à plusieurs objets.

Les objets Command Comme expliqué précédemment, l'équipe Cadremploi a représenté une commande est un appel de méthode encapsulé dans un objet. Nous avons choisi de décorréler l'exécution de l'action et la description de l'intention de l'utilisateur. Concrètement, cela signifie qu'une classe se charge de la définition de la commande, tandis qu'une seconde traitera l'exécution de l'action. Ce découpage en Command et CommandHandler est légèrement plus verbeux, mais permet néanmoins de réduire la taille des classes que nous écrivons et offre la possibilité d'un traitement asynchrone. En effet, on peut imaginer placer les commandes sur une queue qui sera traitée par les CommandHandlers dès que la ressource nécessaire à leur exécution se trouve disponible. Une logique similaire qui n'est pas encore exploitée par Cadremploi serait d'envisager un

mode déconnecté puisqu'il est possible d'exécuter des commandes en local qui seront envoyés sur le serveur et donc traitées effectivement dès que la connexion est récupérée.

Le command executor Un exécuteur de commande (ou "CommandExecutor") est une entitée créée pour être utilisée à l'exécution de chaque commande. Cette classe expose une méthode permettant d'exécuter une commande donnée. Cet exécuteur se charge de:

- retrouver l'aggrégat visé par la commande s'il existe
- vérifier que l'auteur de l'action est autorisé à l'effectuer
- récupérer le CommandHandler associé à la commande et appliquer son effet
- enregistrer les événements générés en base et les diffuser sur les différents bus de donnée.

Si une erreur survient lors de l'exécution d'une commande par le CommandExecutor, elle est utilisée pour informer le front directement via des fonctions de récupérations offertes par le framework Play.

3.5.3 Les Queries

Une query est un objet permettant à l'utilisateur de requêter de la donnée provenant du serveur. Chaque information provenant de son profil ou de ses annonces provient d'une query.

Backend

Les objets query De même que les commandes, les queries sont aussi des actions encapsulées dans des objets dans l'Espace Recruteur. Elles sont pour leur part découpées en 3 objets différents:

- Query qui est la définition de la requête, il s'agit généralement d'un objet vide, uniquement caractérisé par son nom qui est, tout comme la commande, très explicite.
- QueryResult, qui est un conteneur pour le résultat de la requête. Il s'agit donc d'une classe contenant un ou plusieurs objets.

- QueryHandler, qui tout comme les CommandHandler, sont des objets qui contiennent la méthode d'application de la requête. Il s'agit d'une méthode requêtant un index ElasticSearch directement et encapsulant son résultat dans un objet QueryResult. Nous avons en effet vu que les données contenus dans les indexes sont utilisable directement et qu'aucun contrôle n'est fait.

Le résultat des queries est sérialisé puis directement envoyé au front.

Le query executor De même que le CommandExecutor, le QueryExecutor est une entité exposant une méthode permettant d'exécuter une requête donnée. Puisqu'aucune modification n'est faite par une "query", la QueryExecutor se charge uniquement de récupérer le handler associé à la requête puis de renvoyer le QueryResult associé.

Les exécuteurs de requêtes et de commandes sont utilisées comme des courroies par lesquelles passent toutes les actions de lecture et d'écriture faite sur le système. Il est ainsi très simple de contrôler à haut niveaux toutes ces transactions.

3.5.4 Conclusion

Le modèle CQRS permet d'orienter l'architecture vers un système capable de gérer la partie écriture de l'application en ACID (de l'anglais Atomicity, Consistency, Isolation, Durability) et la partie lecture en BASE (de l'anglais Basically Available Soft-State with Eventual consistency). Cet effort permet d'offrir plus de confort à l'utilisateur, mais aussi plus de flexibilité et de performance à l'application via l'utilisation d'outils dédiés.

L'introduction d'événements, via le pattern Event Sourcing, offre de nombreux bénéfices à ce type d'architecture puisqu'il permet de décrire le comportement de l'application avec encore plus de clarté en mettant en avant le dynamisme du modèle. De plus, cela permet une isolation des responsabilités encore meilleure: ils permettent notamment d'isoler les différents cas de lectures (via les projections). L'utilisation d'outil récents comme Akka et Kafka permettent une communication intra et inter applications simplifiée, même dans des cas complexes, tout en assurant une scalabilité importante et une clarté du code.

3.5.5 Critiques du modèle

Lors de l'implémentation d'une telle architecture, l'équipe Cadremploi s'est heurté à des problèmes que le modèle théorique ne laissait pas réellement présager. En effet, le modèle implémenté par notre équipe est plutôt récent et peu d'autres entreprises s'y sont risqués. La documentation et les retours d'expérience à ce sujet sont ainsi limités. Je vais tenter de présenter les difficultés entraînées par ce modèle d'architecture, au delà de son apparence inhabituelle et du changement de mode de réflexion qu'elle demande.

3.5.5.1 L'exemple de la migration des événements

Une application reste rarement figée au cours du temps, et le jeune Espace Recruteur de Cadremploi n'échappe pas à cette règle. Ainsi, les événements écrits à la naissance de l'application sont voués à changer au cours du temps et des modifications y seront apportées. Il est quasiment certain que les événements existant aujourd'hui seront modifiés, enrichis voire même supprimés pour certain, et qu'ils ne peuvent donc rester totalement statique en base. La migration des événements contenus dans la base est nécessaire de manière à ne pas figer le comportement de l'application ni devoir être gêné continuellement par des comportements obsolètes dans le futur. J'ai ainsi travaillé, avec le reste de l'équipe, sur un outil de migration des événements de notre base dans le cas d'une modification de la sorte.

L'équipe Cadremploi a écrit ses événements sous forme de classes Scala, expressives, utilisées ainsi lors de la réception de la commande, de la réception par les projections, de l'écriture en base... Il s'agit d'objets à usage multiple, et leur modification entraîne des changements importants. En effet il est nécessaire de modifier les événements déjà présents en base de manière à ce qu'ils soient désérialisables même dans leur nouvelle version. Pourtant, cette pratique est contraire à la non-modification des données de cette base.

Une solution consistante avec le modèle serait de rajouter des événements "de migration" à la suite de ceux existant permettant de modifier les projections. Mais si une projection, suite à un problème technique, se doit d'être rejouée, il sera nécessaire de lire les données depuis la base des événements pour la reconstruire. À ce moment là, la désérialisation des événements pour que l'application puisse les interpréter sera nécessaire; leur version ayant changé, cette désérialisation ne sera plus possible. Il est alors

impossible de rejouer correctement une projection via cette méthode. L'équipe Cadremploi se voit donc contrainte de mettre à jour les événements de cette base, modifiant leur structure sérialisée.

La solution à ce problème serait de ne plus utiliser de classe et d'utiliser uniquement du Json (ce en quoi sont aujourd'hui sérialisé les événements en base). Cela entraînerait certainement des complications à chiffrer puisque le code perdrait en élégance, l'utilisation des traits, pattern matching et autres bénéfices offerts par le langage Scala n'étant plus accessibles.

3.5.5.2 Bilan

Le modèle CQRS couplé à de l'Event Sourcing offre de très nombreux bénéfices. Il offre notamment beaucoup de clarté de code mais multiplie pour ce faire les lignes nécessaire à l'implémentation de la solution. Manquant de documentation sur le sujet, certains éléments de simplification peuvent paraître intéressants à première vue, mais avoir une conséquence inattendue et gênante. En plus de ce cas de migration des événements demandant la modification de la base et donc de l'historique, qui devrait demeurrer immutable, de l'application, des dérives du modèle telles que la création de commandes avec une valeur de retour par exemple, sont survenues au cours du développement de l'Espace Recruteur.

Ce type d'architecture étant récente, l'implémentation du modèle CQRS + Event Sourcing a demandé à notre équipe la création d'une sorte de framework maison. Cette création, quasi expérimentale et parallèle au développement de l'application Espace Recruteur, s'est retrouvé rapidement bloquante pour la validation de nouvelles tâches. En effet, des éléments innatendus, car peu ou pas mentionné dans les documents sur lesquels se base notre équipe pour mettre en place ce framework, ont demandé parfois plusieurs semaines de développement.

Néanmoins, lorsque les différents outils permettant de migrer les événements ou rejouer les projections seront mis en place, notre équipe pourra enfin profiter des nombreux avantages offerts par ce modèle.

Chapter 4

Le stage dans ma formation

4.1 Auto-évaluation

respect des délais, autonomie, qualité du travail, apports à l'équipe

4.2 Résultats et prolongements possibles

le site tourne et continuera de fonctionner encore

4.3 Ce qui m'a été le plus utile dans ma formation

J'ai pu voir lors de mon stage l'utilité des cours suivis à l'EISTI qui se sont montrés tant utiles qu'actuels. En effet, je n'ai utilisé que très peu d'outils ou de technologies dont je n'avais jamais entendu parler Scala, Akka, Veille technologique

4.4 Ce que m'a apporté ce stage pour le reste de ma carrière

Git, veille technologique, architecture, scala