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

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

ECOLE INTERNATIONALE DES SCIENCES DU TRAITEMENT DE L'INFORMATION

Résumé

Diplôme d'ingénieur en Informatique

Développeur chez Figaro Classifieds

by Thomas GIOVANNINI

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Remerciements

Les remerciements et les gens à remercier.....

Contents

List of Figures

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	rads^{-1}

For/Dedicated to/To my...

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 Morel. 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 Description objective de l'entreprise

Petit tour rapide du Figaro

2.1.1 L'entreprise en général

2.1.1.1 Secteur d'activité

concurrents, marché, évolution, ... 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. Cette entreprise est présente sur 3 gros secteurs: l'Emploi, la Formation et l'Immobilier. N°1 des "Quality Classifieds" en France, elle a pour ambition de proposer aux internautes/mobinautes et aux professionnels "le meilleur des médias et des solutions d'annonces classées". 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 nos 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.

2.1.1.2 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. Je faisais partie, en ce qui me concerne, du pôle Digital.

2.1.1.3 Chiffre d'affaire

60M€

2.1.1.4 Effectifs

PME de 350 collaborateurs

2.1.1.5 Organisation interne

FIGARO CLASSIFIEDS est dirigé par Thibaut GEMIGNANI et est organisé autour de trois gros secteurs que sont l'Emploi, la Formation et l'Immobilier. Autour de ces trois grands axes, on retrouve 5 grandes directions transverses:

2.1.1.6 Relations du groupe

blablabla Figaro, blablabla Serge Dassault

2.1.2 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.

2.1.2.1 Outils, technologies et méthodes

La plupart des employés a accès à un ordinateur, et un compte est attribué à l'arrivé, lui permettant de gérer notamment ses mails ou ses 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, ce qui

permet 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. C'est via Jenkins, un outil d'intégration continu que les différentes release d'une application sont gérés. Il s'agit d'un outil simplement mentionné en cours que je n'avais jamais utilisé auparavant.

2.1.2.2 Outils de travail collaboratifs

Des outils sont aussi disponibles pour permettre aux équipes de travailler ensemble plus facilement. Redmine est un logiciel de gestion de projet qui est 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 est aussi utilisé pour mettre en commun le code écrit par toute l'équipe. Il s'agit d'un outil beaucoup utilisé à l'EISTI mais d'une façon très simpliste, et j'ai appris au cours de mon stage de nombreuses nouvelles possibilité pour cet outil. 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. 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.

2.2 L'environnement de travail

2.2.1 Les relations humaines au sein de l'entreprise

Les membres des nombreuses équipes se côtoient sereinement, se respectent et s'entraident.

2.2.2 Relations de la direction avec le reste du personnel

Mails, ils viennent nous voir, ...

2.2.3 Relations entre services, départements et divisions

On se côtoit souvent pour échanger, souvent par le biais de Laëtitia, open space, entente cordiale, on va faire du sport ensemble

2.2.4 Relations entre les différentes catégories de personnel et les différents niveaux hiérarchiques

Partie qui semble inutile, déjà traité

2.3 Mon intégration dans l'entreprise

Je dit bonjour à tout le monde et je tiens la porte.

Chapter 3

Les apports techniques

Le site Cadremploi.fr est découpé en différentes parties. En effet, il dispose d'une partie publique, qui 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 client. J'ai travaillé durant ce stage sur la partie concernant les recruteurs, ou "Espace Recruteur". L'espace recruteur est aujourd'hui trop ancien, difficilement maintenable et utilisable. 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.

3.1 Cahier des charges et fonctionnement général du cycle de développement

3.1.1 Contenu du stage

Le sujet principal de mon stage concernait la refonte de l'espace recruteur du site cadremploi.fr. CADREMPLOI.fr a conçu un espace réservé aux professionnels afin de les aider dans leurs 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 BLABLA ans et une refonte totale en était nécessaire, de manière à proposer des services nouveaux ainsi qu'un design plus actuel. 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

Les fonctionnalités attendues comprenaient notamment la mise en place du pattern Event Sourcing, que je détaillerai plus tard, de manière à pouvoir suivre de manière rigoureuse l'évolution de chaque utilisateur dans l'espace recruteur, ainsi que la simplification du système de classification des offres en créant un produit d̈e baseäuquel se grefferont des options.

3.1.2 Méthode de management

L'équipe Cadremploi, sur le projet Espace Recruteur notamment, suit une méthodologie Agile.

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?" 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.

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 que je traiterai plus tard permet d'éviter de nombreuses erreurs au cours du développement de l'application.

- 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 les représentant dessus et déplacent le ticket dans la catégorie à laquelle il correspond.

Le tableau sur lequel sont affiché les tickets sont plutôt grands 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.

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.

3.2 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 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.

3.2.1 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 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.

3.2.2 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.

3.2.3 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 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.

3.2.4 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.

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

3.3 Aperçu des technologies et des techniques utilisées

3.3.1 Langages

3.3.1.1 Backend

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

3.3.1.2 Frontend

Du côté front, l'équipe a utilisé le framework AngularJS ainsi que la librairie D3.js qui permettent des animations dynamiques.

3.3.2 Environnements et outils

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 les 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. 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. L'équipe Cadremploi utilise aussi Jenkins comme outil d'intégration continue pour déployer les nouveautés sur les différents environnement de production utilisés. Cet outil avait été brièvement présenté lors du cours d'Outils de Développement mais jamais utilisé. 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

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 mon utilisation de ces outils, particulièrement pour Git puisque j'ai redécouvert l'outil puisque le projet auquel je participais differais grandement en taille de ceux auquels j'avais contribué auparavant.

3.3.3 Techniques utilisées

3.3.3.1 Organisation Git

3.3.3.2 Concertations d'équipe

3.4 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.

3.4.1 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 troubles 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 ce mode de fonctionnement avec mes projets précédents. En effet, mes projets passés se faisaient dans des équipes plus

réduites, et le temps manquait pour la revue de code. Cela ne nous empêchait pas de suivre la règle du boyscoot, qui prescrit de toujours laisser une fonction, un module dans un meilleur état que celui dans lequel on l'a trouvé, et qui évite la "possession" du code par un membre de l'équipe. Cette règle est plus ou moins comprise dans la revue de code de Cadremploi qui offre plus d'avantages encore.

3.4.2 Utilisation de Git

Git est un outil de contrôle de version flexible et puissant. Il propose énormément d'options de workflow et il est parfois compliqué Git is a flexible and powerful version control system. While Git offers significant functionality over legacy centralized tools like CVS and Subversion, it also presents so many options for workflow that it can be difficult to determine what is the best method to commit code to a project. The following are the guidelines I like to use for most software projects contained within a Git repository. They aren't applicable to every Git project (especially those hosted on drupal.org or GitHub), but I've found that they help ensure that our own projects end up with a reasonable repository history.

3.5 Architecture de l'application

L'espace recruteur 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.

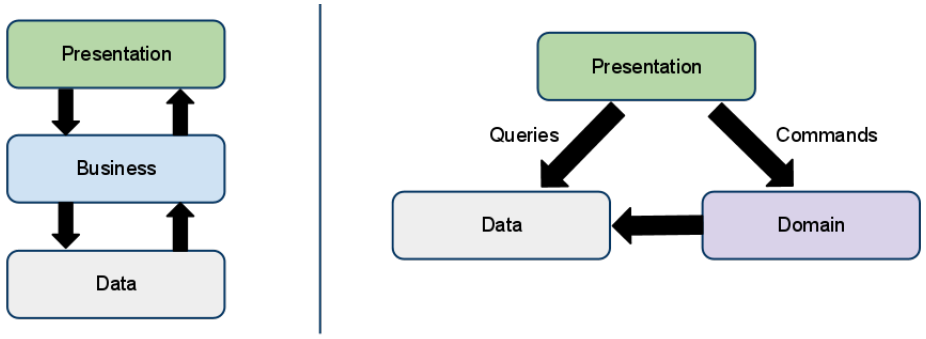
CQRS: Command Query Responsibility Seggregation CQRS (Command and Query Responsibility Segregation) est un modèle d'architecture plutôt récent dont le principe repose, comme son nom l'indique, sur la séparation entre l'écriture et la lecture de l'information. Dans les systèmes de gestion de donnée traditionnels, 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. 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 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 événement et que ces objets, eux-même stockés dans le même ordre que celui dans lequel ils ont été appliqués, permettent de connaître l'état de cette application à tout instant t. Il est ainsi possible, en plus de seulement requêter ces événements, de pouvoir reconstruire 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.

L'architecture CQRS se base sur plusieurs concepts tirés du Domain Driven Design (DDD). Il me semble important, avant de détailler le contenu des différentes couches formant cette architecture, de clarifier 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 pour appliquer les règles métiers et pour appliquer les contrôles nécessaires à chaque actions, des agrégats et des services notamment.

Les aggrégats Un aggrégat est un objet modifiable lors d'une transaction. Il est généralement repésenté par un classe unique 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 ("root class") et tout accès à l'aggrégat par l'extérieur se fait via cette classe et toutes les autres restent interne à l'agrégat. Ces classes contiennent la logique du domaine (3.5.1.1) et permettent 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 (**??**) et la base de donnée. On retrouve dans ces objets des méthodes du type 'create' permettant de créer une nouvelle instance d'un aggrégat ou 'save' qui permet de sauvegarder les changements relatifs à un aggrégat.

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 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. De tels noms sont ainsi très expressifs et permettent 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 avec 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 applique son effet sur l'aggrégat, ce qui génère des événements de modification. Les découpages et cette logique permettent de maîtriser l'impact de chaque action sur le système.

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 **??**subs:Le domaine)domaine 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.

Les projections sont des objets écoutant des événements particuliers émanant des modifications venant de la couche de modification des données. Leur but est de retirer certaines informations particulières de ces événements de manière à construire les DTO qui seront utilisés dans la couche lecture. Les projections écoutent alors les événements apparaissant dans le bus de manière asynchrone et répercutent les changements qu'ils demandent, mettant ainsi à jour la couche de lecture symétriquement à la couche d'écriture.

L'architecture CQRS possède plusieurs avantages blablaabl

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 contient ainsi la donnée relative à un événement, la date d'application de cet événement sur le système ainsi que l'identifiant de l'aggrégat sur lequel s'applique l'événement. 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 finalement une liste de tous les événements ayant eu lieu depuis le démarrage de l'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.

Migration des événements Evidemment, l'application Espace Recruteur est vouée 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. J'ai ainsi travaillé sur un outil de migration des événements de cette base dans le cas d'une modification de la sorte. 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. Il s'agit donc de la seule nuance à la non-modification des données de cette base.

L'utilisation d'une base PostgreSQL en append only permet la réalisation de l'Event Sourcing

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.

Cet index est mis à jour de manière asynchrone à la réception d'une intention de l'utilisateur. Parallèlement à l'ajout d'un événement en base de donnée, un message Akka est envoyé sur un bus et sera utilisé pour mettre à jour l'index ElasticSearch. Ainsi, les données présentes dans cet index sont 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. De plus, cette mise à jour asynchrone permet de garantir au recruteur une interaction la plus synchrone possible avec l'application: un feedback immédiat lui est envoyé lorsqu'il exécute une action puisque la mise à jour de l'index n'est pas bloquante.

En somme, les parties d'écriture et de lecture sont physiquement séparées dans l'application Espace Recruteur de Cadremploi.

3.5.2.2 Bus d'événements: Kafka

Kafka est un service offrant la fonction de système de message 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.

Vocabulaire Les données du cluster Kafka sont stockées dans divers "topics". Les entités publiant des messages dans un topic Kafka sont appelés "producteurs". Celles qui souscrivent aux topics et lisent les flux sont appelés "consommateurs".

Utilisation Sans détailler le fonctionnement intrinsèque de Kafka, retenons qu'à plus haut niveau, des producteurs envoient des messages dans le cluster Kafka via le réseau et ce dernier les envoit à son tour aux consommateurs. Kafka est utilisé pour la communication entre l'Espace Recruteur, le Backoffice et divers autres services au sein de FIGARO CLASSIFIEDS. C'est notamment via ce service que les événements sont envoyés depuis la couche Command à la couche Query: ici, le producteur de message est le CommandExecutor tandis que les projections sont les consommateurs. La vitesse de transmission que propose Kafka permet de traiter la donnée en temps réel.

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. Généralement, cette fonction effectue 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 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

Si une erreur survient lors de l'exécution d'une commande par le CommandExecutor, elle est utilisée pour informer le front directement. Autrement, l'aggrégat résultat de la modification opérée par l'utilisateur est retourné. Cette valeur de retour est la seule exception au pattern CQRS opéré par l'équipe Cadremploi. En effet, il existe des cas où il est nécessaire pour le backend de se tenir au courant de la mise à jour directement après l'exécution d'une commande. Le pattern CQRS recommande d'exécuter une query pour se faire, mais la mise à jour de l'index n'étant pas synchrone mais asynchrone, comme mentionné précédemment, cela peut poser des problèmes. Des contraintes techniques ont donc obligé l'utilisation de cette valeur de retour.

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

Projections Les projections sont des objets permettant de dériver un état d'un flux d'événements. Dans l'Espace Recruteur, elles écoutent dans le bus des événements particuliers et mettent à jour en fonction l'index ElasticSearch. La transmission des messages étant rapide via Kafka, les projections sont cohérente à quelques millisecondes près avec la partie de lecture. Ce décalage est peu contraignant: la gestion des projections étant faite de manière asynchrone, on peut toujours assurer au recruteur un feedback rapide. C'est pour maximisercette réactivité que l'on accepte de rendre cohérente la couche Query de notre application plus tard.

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.

3.6 Event sourcing

Une des demandes pour le nouvel espace recruteur est de pouvoir suivre avec précision les actions de l'utilisateur. Ainsi chaque modification qu'il apportera à une de ses offres ou à son profil par exemple doit être enregistrée. *"L'idée fondamentale derrière l'Event Sourcing est d'assurer que chaque modification de l'état d'une application est capturée dans un objet événement et que ces événements sont eux-mêmes stockés dans l'ordre dans lequel ils ont été appliqués à l'application."* (Martin Fowler)

3.6.1 Fonctionnement désiré

Il est nécessaire de pouvoir suivre l'évolution de l'annonce que créé un utilisateur. Concrètement, à chaque fois que l'utilisateur modifie son annonce, on veut générer un événement qui informe notre système de cette modification. Toutes ces données pourront ainsi être utilisées pour faire des statistiques ainsi que des contrôles. Une des utilités de ce suivi est par exemple de pouvoir contrôler les éditions abusives d'annonces déjà en ligne. Il est ainsi possible de suivre de très près la façon dont l'utilisateur procède, ou les champs qui posent problème.

3.6.2 Fonctionnement technique

3.6.2.1 Enregistrement de l'événement

Concrètement, derrière chaque champ formulaire que l'utilisateur remplit, un watcher d'Angular est présent. Ainsi lorsque ce watcher indique que le champ a été modifié, on récupère la valeur qu'il contient et envoyons cette demande de modification vers le backend. Cette "demande de modification" est traitée comme une commande et génère alors un ou plusieurs événements qui résument la modification effectuée et sont stockés dans une base de donnée, conformément au modèle de Write du pattern CQRS décrit ci-dessus. Chacune des modifications faite par l'utilisateur est ainsi enregistrée, et il est possible de connaître l'état du système à tout instant t donné en interrogeant la base de données.

3.6.2.2 Utilisation de l'événement

Si l'enregistrement en base se passe bien, l'événement est ensuite publié sur un EventBus, un objet Akka qui permet entre autre d'envoyer un message à des groupes d'acteurs, pour être écouté par des objets de type Projection. Ces projections écoutent seulement une partie des événements selon leurs responsabilité et modifient en conséquence un objet DTO qu'elles indexent. Ce traitement est fait de manière asynchrone, via le système d'acteurs Akka. Il permet donc d'optimiser le modèle CQRS vu précédemment: la commande est exécutée rapidement puisque l'événement est sauvegardé de suite et le reste du traitement est effectué de manière asynchrone. Cela permet d'informer quasi instantannément l'utilisateur que son action a été prise en compte. D'autre part, en acceptant de rendre asynchrone et donc pas forcément instantannée la mise à jour de la partie Read du modèle, on améliore grandement la performance ressentie par l'utilisateur. Le

système est donc tenu à jour en lecture et ainsi, à l'appel d'une query, les projections peuvent être interrogées pour récupérer la donnée de manière très rapide.

3.7 Domain Driven Development

3.8 Organisation

comparaison entre les plannings prévisionnels et réels

3.9 Auto-évaluation

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

3.10 Résultats et prolongements possibles

le site tourne et continuera de fonctionner encore

Chapter 4

Le stage dans ma formation

4.1 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

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

Git, veille technologique, architecture, scala

Appendix A

Appendix Title Here

Write your Appendix content here.