Stage Atos

Alexis Aoun

1 Objectifs et réalisation de mes missions

1.1 MPP Dashboard

1.1.1 Le contexte

Atos a une plateforme en ligne, My Atos, sur laquelle tous les employés doivent renseigner leurs informations personnelles ainsi que leurs parcours professionnel et académique. Le problème de cette procédure, qui est en théorie obligatoire, est qu' une partie importante des salariés ne remplissent pas ces informations, et la plupart du temps cela est dû à un simple oubli. Pour y remédier, le service RH et la direction général d'Atos décidèrent de lancer un projet interne qui permetterait de relancer les employés automatiquement à partir d'un fichier excel contenant les informations de tous les collaborateurs, fournit par le service RH. Ce projet fut baptisé MPP Dashboard.

Le projet est une application web(TODO) dont le fonctionnement repose sur trois processus principaux :

1. Le traitement de l'excel : Le service RH fournit un fichier excel sous format XSLX (Le format par défaut des fichiers excels microsoft). Celui-ci doit être traité de manière à ce que les informations qu'il contient puissent être manipuler programmatiquement.

2. Le filtrage : Une fois les donnés des collaborateurs dans notre application, on sauvegarde dans une base de donnée uniquement ceux dont le taux de renseignement d'informations et en-dessous d'un seuil défini par les administrateurs de l'application. Le taux par défaut est de 90%.

3. L'envoie de mail : L'application enverra des mails à tous les salariés présent dans la base de données après la phase de filtrage. Un chiffre correspondant au nombre de mails envoyés au collaborateurs est associé à chacun d'entre eux. Au bout du 5ème mail l'employé n'ayant pas renseigner les informations nécéssaires est signalé automatiquement au service RH.

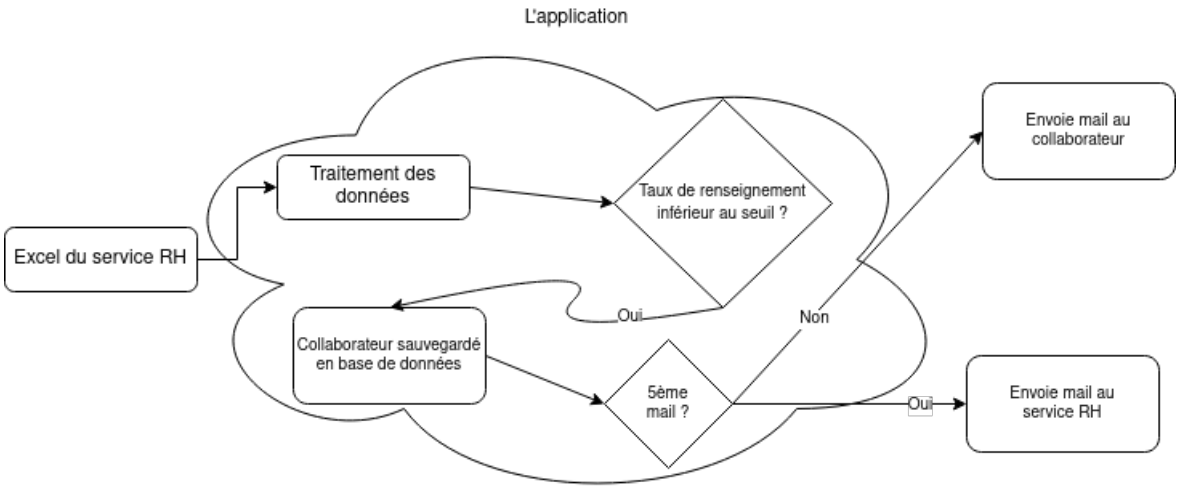

Figure 1: Diagramme du fonctionnement de MPP Dashboard

Les technologies utilisés pour la réalisation du projet sont :

- Pour la base de données : PostgreSQL (TODO)
- Pour le serveur backend (TODO) : ExpressJS (TODO)
- Pour l'interface grahique : ReactJS (TODO)

1.1.2 La problèmatique

Lorsque je suis arrivé sur le projet, celui-ci a déjà été développé mais avait un problème avec l'un des processus, celui d'extraction de l'excel. En effet la librairie (TODO) utilisée pour remplir cette tâche, du nom d'Excellente, a été développée en interne par des employés d'Atos et comporte certains inconvenients :

- Il comporte plusieurs bugs (TODO). Cela est notamment dû au fait que cette librairie n'est pas connue du grand publique et ne reçoit donc pas un grand nombre de tests et de contributions
- La librairie devient exponentiellement lente avec la grandeur du fichier excel. Hors celui qu'on a à extraire fait plus de 40 000 lignes

Ma tâche est donc de trouver une solution qui puisse répondre à la fois aux problèmes de vitesse et de fiabilité.

1.1.3 Les solutions possibles

Afin d'accomplir ma mission j'ai étudié plusieurs possibilités :

- Garder la librairie Excellente et essayer de gommer ces défauts en modifiant son implémentation, voir même la librairie en elle-même. Le risque est que cela prenne trop de temps ou même que cela n'aboutisse tous simplement pas. Du fait de l'utilisation limitée d'Excellente et du manque de documentation techique, corriger la librairie pourrait prendre des mois alors que la direction d'Atos attendait des résultats dans les semaines suivant mon arrivé.

- Utiliser un script Python (TODO). Lors de mes différents cours en Data au sein de l'IMT Nord Europe, l'une des compétences que j'ai acquis est le traitement de données par le biais de script Python en utilisant des librairies tels que Panda (TODO). Les performances seraient des ordres de grandeurs meilleurs que celle de n'importe quelle autre solution implémentée en Javascript (TODO), et l'écriture du script peut se faire en une journée. La difficulté est l'implémentation du pont entre le serveur ExpressJS et le script Python. Celui-ci doit être exécuter au bon moment par le serveur, et ce dernier doit pouvoir récupérer le résultat final. Il y avait aussi la problématique des dépendances(TODO) spécifique à Python dont a besoin un tel script qui rajouterait une couche de compléxité au projet ExpressJS, au développement mais surtout à la maintenance.

- Utiliser une autre librairie Javascript, ExcelJS(TODO). ExcelJS est sans aucun doute la librairie Javascript la plus utilisée pour le traitement d'excel. Un avantage très important d'une librairie aussi connue est sa documentation riche qui m'a permis de faire des premiers tests rapidements. De ces expérimentations j'en avais conclu que la fiabilité était satisfasante et que la vitesse était tout à fait convenable pour notre application, même si plus lente que le script Python. Cette solution fut donc retenu.

1.1.4 La réalisation de la mission

La mise en œuvre de la solution fut assez simple dans son ensemble en grande partie grâce à l'excellente documentation d'ExcelJS. La seule difficulté rencontrée a été la gestion des erreurs dans l'excel. En effet celui-ci comporte des formules qui dans certains cas retournent des erreurs. Mon implémentation a donc dû prendre en compte ces cas de figures là. Après cela j'ai pu procéder à une série de tests sur mon ordinateur personnel, le traitement de l'excel comportant 40 000 lignes a duré dans les alentours des 1 minutes et 45 secondes. Il était temps de déployer ma solution en production.

1.1.5 Résultats et bilan

Après quelques ajustements mon algorithme de traitement obtenait les résultats escomptés. Le temps d'exécution de moins de 2 minutes jugés satisfasant. Avec plus de temps une implémentation du script Python aurait été possible, permettant de réduire encore plus le temps de traitement. Néanmoins je pense avoir pu obtenir le meilleur résultat possible compte tenu de la contrainte temporelle.

Cette première mission fut l'opportunité pour moi de prouver à la fois mes compétences techniques mais surtout mes capacités à prendre des initiatives et à les mettre en œuvre.

1.2 Tickeratops

1.2.1 Le contexte

Comme mentionné dans la partie décrivant le contexte de réalisation de mon stage, j'ai évolué dans l'entité Cloud Apps & Data, ou CAD. Celle-ci est composée principalement de deux corps de métier, les développeurs et les devOps(TODO) qui travaillent ensemble afin de concevoir et déployer les solutions efficacement.

Cependant un problème récurrent apparait au sein de l'entité : la communication entre ces deux groupes. En théorie un canal de communication sur la plateforme Teams (TODO) est dédié à l'échange entre développeur et devOps, mais le manque de formalisme et la forte pression que peuvent subir les devOps engendre un temps d'attente prolongé pour le traitement des demandes des développeurs qui en conséquence décident souvent de court-circuiter Teams et de communiquer directement avec les devOps soit en personnne soit par message privé. Cela à pour effet d'aggraver les problèmes d'organisations et de suivi des demandes.

Il y a aussi un second axe d'amélioration qui est l'optimisation du temps de travail des devOps. Un certains nombres de leurs tâches s'avèrerent être répétitives, tel que le redémarage, la suprresion, ou le déploiement de serveurs.

Afin de répondre à ces deux problématiques les équipes du CAD ont décidé de développer une application web, Tickeratops, qui aura un système de tickets pour les demandes des développeurs, lesquels pourront être traités et suivis par les devOps avec plusieurs status (à faire, en cours de traitement, bloqué, terminé) et qui seront affichés sous forme de kanban. L'autre fonctionnalité principale qui la distingue des nombreuses solutions de gestions de projets existantes est l'intégration direct d'exécution de script dans l'application. Prenons l'exemple d'un développeur qui a besoin qu'un serveur de pré-production soit redémarrer : il créera un ticket décrivant son besoin et le devOps, qui aura prédifini cette action avec un script, n'aura qu'a appuyé sur un boutton pour relancer le serveur. Une fois le redémarrage accompli le ticket sera catégorisé comme étant accompli. La demande du développeur est suivi de bout en bout et le devOps gagne du temps grâce à l'automatisation intégrée dans l'application.

Les techonologies utilisées pour ce projet sont :

- Pour la base de données : MongoDB(TODO)
- Pour le serveur backend : NestJS(TODO) et GraphQL(TODO)
- Pour l'interface graphique : NextJS (TODO)

Dans le cadre de ce projet j'ai eu à accomplir plusieurs tâches, au niveau du serveur et au niveau de l'interface graphique. Je vais me concentrer sur une seule mission, que je juge avoir été la plus intéressante. Lorsqu'un développeur crée un ticket il a le choix de rajouter une description afin d'expliquer plus en détail sa demande. Dans cette description on souhaite également pouvoir insérer des images afin d'illustrer nos propos, qui pourrait être par exemple une capture d'écran. La consigne qui m'a été donnée était de pouvoir stocker ces photos nos pas sur le même serveur où foncitonne le backend de l'application mais sur un serveur tiers détenu par l'un des partenaire d'Atos, Google, avec leur service de Bucket GCP (TODO, ou explique ici)

1.2.2 La problématique

Cette tâche requiert une bonne connaissance de l'environnement backend or je n'avais pas utilisé NestJS par le passé. Je n'ai également jamais utilisé la plateforme GCP de google, qui d'ailleurs devait être configurer sur mesure pour notre projet. Je devais donc passer par les devOps, leur transmettre les besoins de l'application afin qu'ils mettent en place l'instance de Bucket GCP.

1.2.3 La réalisation de la mission

Dans un premier temps je pris le soin de m'autoformer et d'acquérir les compétences nécessaires à l'accomplissemnt de la tâche. Une fois assez familier avec NestJS je me suis penché sur l'excelllente documentation de l'API de Google pour la plateforme GCP afin de pouvoir implémenter correctement la solution à l'application. Ensuite j'ai communiqué avec les devOps, pour la première fois depuis le début de mon stage, pour la mise en place de la plateforme GCP de production. J'ai pu alors vivre par moi-même la raison de l'existence du projet sur lequel je travail.

1.2.4 Résultats et bilan

Après quelques ajustements et correctifs, la fonctionnalité produisait le résultat attendu. On pouvait sauvegarder, supprimer et afficher des images de notre Bucket GCP à partir de l' application web.

A travers cette mission j'ai pu faire preuve d'adaptation en m'autoformant aux technologies requises. Mais là n'est pas la leçon la plus importante. Contrairement au projet MPP où j'étais seul, j'ai travaillé au sein d'une équipe qui suivait les principes des méthodes agiles ou SCRUM(TODO), une première pour moi. J'ai donc appris les fondamentaux avec les sprints et les users story. J'ai également pu participé aux sprint review, sprint planning, retro, daily etc. Des connaissances basiques d'organisations d'équipes que je ne connaissais qu'en théorie.

1.3 UGAP

1.3.1 Le contexte

Pour ma 3ème mission j'ai eu la chance de travailler sur un projet client externe. Une occurence rare chez Atos qui préviligie les projets internes pour les stagiares et alternants.

> "L'UGAP est un établissement public à caractère industriel et commercial français créé en 1985 et placé sous la double tutelle du ministre chargé du Budget et du ministre chargé de l'Éducation nationale. Elle est une centrale d'achat publique généraliste et constitue, en raison d'un mode opératoire quasi-exclusif en achat pour revente, un acteur spécifique de l'achat public, dont le rôle et les modalités d'intervention sont définis par le code de la commande publique (qui est entré en vigueur en avril 2019) relative aux marchés publics5. Son activité globale concerne plus de cinq milliards d'euros hors taxes, répartis en 4,67 milliards d'euros d'activité de grossiste (achat pour revente) et 520 millions d'euros d'achats réalisés en direct par les entités publiques sur les marchés de gaz et d'électricité passés par l'UGAP pour leur compte en tant qu'intermédiaire." https://fr.wikipedia.org/wiki/Union_des_groupements_d'achats_publics

Pour résumer en une phrase, la raison d'être d'Ugap est la vente de biens aux établissements publics.

Aujourd'hui la majorité des ventes se font à travers leur site internet. Celui-ci dattant depuis une décennie maintenant requiert d'être remis à niveau, à la fois visuellement et dans la gestion du backend.

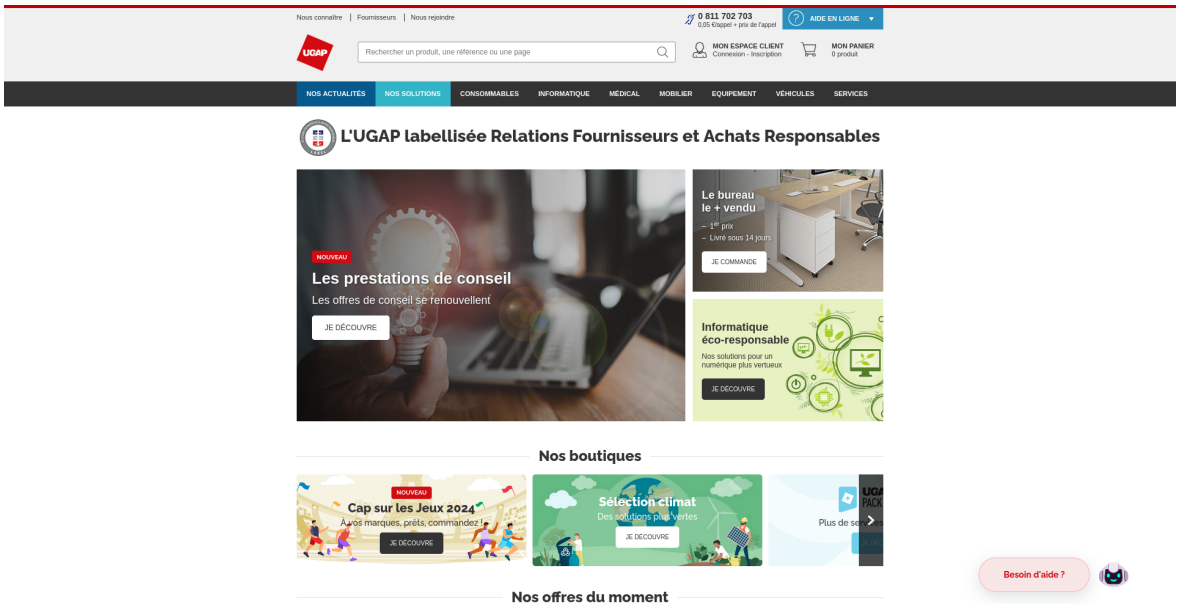

Figure 2: Site actuel de l'Ugap

C'est dans ce contexte qu'intervient Atos en tant que préstataire extérieur avec pour mission de rénover le site. J'ai fait parti de l'équipe chargée de la refonte graphique.

Je n'avais par le passé jamais participé à un projet d'une tel envergure, l'architecure et les outils techniques employés reflète parfaitement bien la compléxité de l'application. L'interface graphique est composé de plusieurs microfrontend. Pour comprendre ce qu'est un microfrontend on peut faire l'analogie avec les micro-services utilisés communement dans les serveurs backend. La raison d'être des microservices est qu'au lieu d'avoir un gros serveur monolithique on ai plusieurs petites applications regroupés sous un même domaine(TODO). Cela permet de conteneuriser les fonctionnalités. Par exemple un supermarché peut décider d'avoir un microservice pour la gestion des produits, un autre pour la gestion des employés, un autre pour la gestion de l'inventaire etc. Un microfrontend se base sur le même principe à la différence qu'ici on ne conteneurise pas des fonctionnalités backend mais des interfaces graphiques.

Le site d'Ugap est composé de trois grandes sections :

- Section produit, pour l'achat de biens
- Section offre, présentant les réductions et soldes du moment
- Section magazine, contenant des articles sur l'actualité du service publique

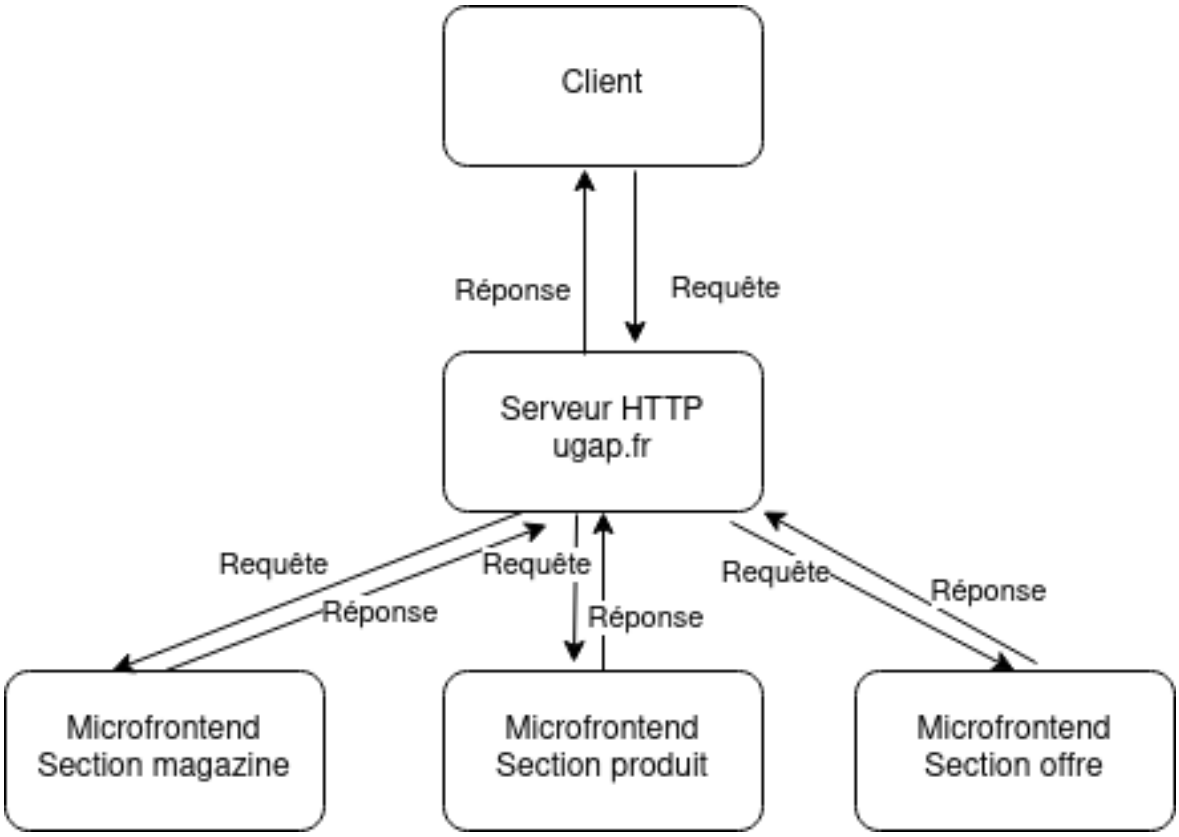

Figure 3: Diagramme de l'architecture frontend du nouveau site d'Ugap

L'interface de chacune de ces sections est un microfrontend indépendant. Les avantages d'une telle architecture sont :

- Un site plus robuste au bug et autres problèmes : Si l'un des microfrontend vient à ne plus fonctionner le reste du site n'en sera pas impacté
- Une maintenance plus simple : Pour les même raisons que le premier point, si l'on veut mettre à jour l'une des sections on peut le faire sans avoir à désactiver l'ensemble du site

- Un développement en équipe facilité : L'indépendance inter microfrontend fait que les développeurs des différentes sections n'apporteront pas de modifications qui peuvent rentrer en conflit

L'interface en elle-même est developpée avec NuxtJs et VueJS(TODO les deux). L'ensemble des tâches que j'ai effectué pour le projet Ugap s'est concentré sur la section produit, plus précisement la page produit. Cette page comporte la description du produit sélectionné et offre la possibilité de l'ajouter au panier pour l'acheter.

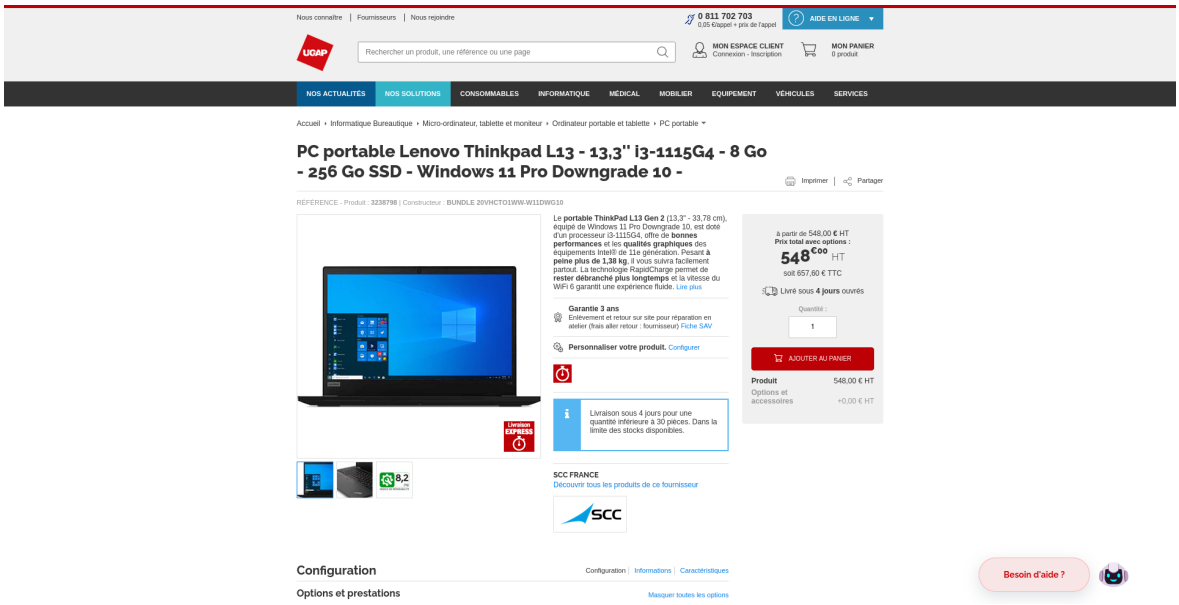

Figure 4: Capture d'écran de la page produit du site actuel d'Ugap

1.3.2 La problématique

Il est évident que de comprendre une architecture aussi complexe avec une base de code aussi grande est une épreuve en soit, surtout qu'après avoir était sur deux autres projets différents en moins de 3 mois une certaine fatigue commençait à apparaitre. Autre problème qui était général à l'ensemble du projet et dont je fis état assez rapidement fut le manque de communication entre l'équipe frontend et backend. Or celle-ci est indispensable pour que l'on puisse savoir comment l'API (TODO) d'Ugap fonctionne et surtout être au courant des futures changements auquel on doit s'adapter au niveau de l'interface, par exemple le format dans lequel l'ont recevait les données. A noter que l'équipe s'occupant du backend était constituée et dirigée en majorité par des employés d'Ugap et non d'Atos, ce qui ne facilite pas la communication.

1.3.3 Réalisation de ma mission

J'ai commencé par des tâches simples comme l'amélioration ou la modification de composants graphiques pré-existants. En parallèle je m'étais autoformé aux technologies utilisés pour la gestion des microfrontend afin d'avoir une compréhension plus profonde du projet surlequel je travail. Je suis ensuite passé à des missions plus complexe comme la création d'un tableau affichant les différentes variantes d'un produit, par exemple un ordinateur avec plusieurs processeurs différents. Quant aux problèmes de communication entre équipes frontend et backend j'ai essayé d'être à l'initiative avec mes collègues en créant des cannaux de conversations Teams et en remontant le problème aux différentes réunions afin d'initier la discussion autour du sujet.

1.3.4 Résultats et bilan

Au final j'ai réussi à accomplir la totalité des tâches qui m'ont été assignées. Le débat autour de la communication qu'on a provoqué avait commencé à porter ses fruits vers la fin de mon stage, les échanges devenait plus fréquents, fluide et spontannés.

J'ai grandement bénéficié de ce projet, évidement d'un point de vue technique, mais surtout d'un point de vue humain. D'abord j'ai encore plus approfondit mes connaissances en méthode Agile, avec notamment la notion de swimlane et de weekly(TODO?), mais j'ai aussi gagné à travailler dans une équipe aussi grande

avec tant de métiers différents : développeurs, testeurs, designeurs d'interface, scrum master etc. Ce qui contraste avec Tickeratops où l'on était une équipe de 5 développeurs et d'un scrum master. Je dois aussi mentionner le sentiment unique à ce projet de travailler sur un produit qui se retrouvera à la disposition de milliers, voir de dixaine de milliers d'utilisateurs, contrairment au deux précédentes applications dont l'utilisation sera limitée aux employés d'Atos. Ce sentiment de devoir se surpasser, de ne pas avoir le droit à l'erreur et d'être perfectionniste. Aujourd'hui j'attend avec impatience la mise en production du nouveau site prévu pour fin janvier et de voir le projet auquel j'ai contribué disponible au grand public.

2 Réflexion sur le métier d'ingénieur et la suite de mon projet professionnel

2.1 La contribution des ingénieurs au sein d'Atos

2.2 Réflexion sur la suite de ma carrière