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 professionel 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.

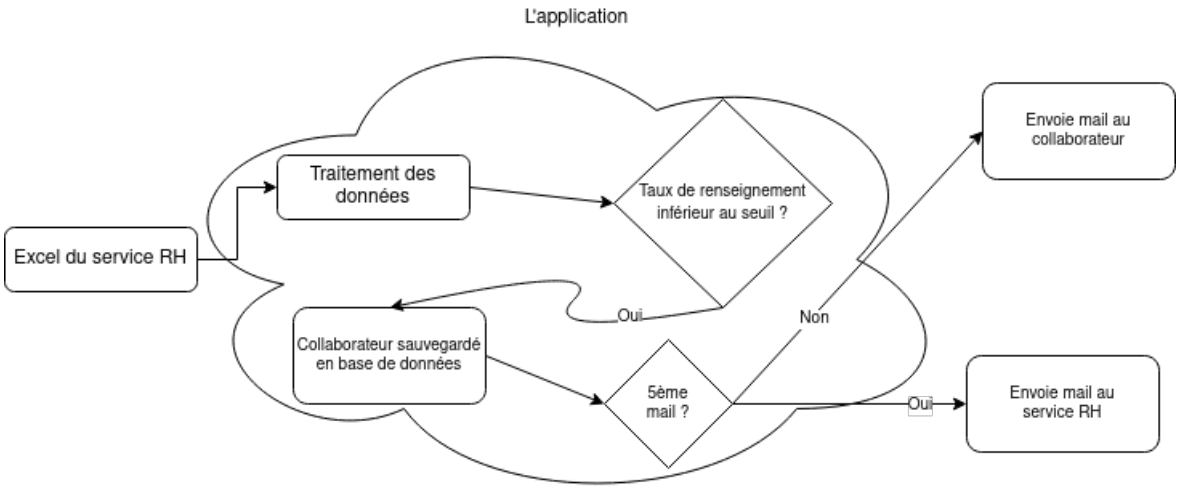

Figure 1: Diagram du fonctionnement de MPP Dashboard

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