Rapport de stage

Du 03 Avril 2023 AU 12 Mai 2023

Entreprise : BY-SYSTEMS SRL

Adresse : Avenue Charles Plisnier 13, 1070 Anderlecht



**Auteur** : Corentin Maillard

**Maître de stage** : Youssef Boujraf

**Superviseur** : Quentin Lurkin

Année 2022-2023 Une image contenant texte

Description générée automatiquement

**Remerciements**

Je tiens à remercier Monsieur BOUJRAF, notre maitre de stage et CEO de l’entreprise By-Systems. Il m’a permis de réaliser ce stage avec deux autres étudiants de l’ECAM au sein de son entreprise. Il nous a encadré et nous a conseillé tout au long de ce stage. Je le remercie également pour la confiance qu’il nous a accordée, son précieux temps qu’il nous a consacré. et de nous avoir transmis son expertise.

Je tiens également à remercier notre professeur, Monsieur Quentin Lurkin pour avoir été mon superviseur de stage et nous avoir suivi et conseillé durant ce stage ainsi qu’au temps qu’il nous a consacré lors des présentations.

Enfin, j’exprime aussi mes remerciements à tous mes professeurs de l’ECAM pour les enseignements qu’ils m’ont prodigués durant ma formation.

Table des matières

[1. Introduction 4](#_Toc136285995)

[2. Objectifs du stage 4](#_Toc136285996)

[3. By-Systems et l’équipe projet 5](#_Toc136285997)

[3.1. Présentation de l’entreprise 5](#_Toc136285998)

[3.1.1. Business challenges 5](#_Toc136285999)

[3.1.2. Technical challenges 5](#_Toc136286000)

[3.2. Présentation de notre équipe 6](#_Toc136286001)

[4. Présentation du projet 7](#_Toc136286002)

[4.1. Introduction 7](#_Toc136286003)

[4.2. Gestion du dépôt via Github 7](#_Toc136286004)

[4.3. Authentification à Jira et Jira service Management 8](#_Toc136286005)

[4.4. ETL diagram 11](#_Toc136286006)

[4.5. Jira-Connector 11](#_Toc136286007)

[4.5.1. Requêtes HTTP 11](#_Toc136286008)

[4.5.2. Webhook 14](#_Toc136286009)

[4.5.3. Difficultés rencontrées 17](#_Toc136286010)

[4.5.4. Pistes d’améliorations 17](#_Toc136286011)

[5. Conclusion 18](#_Toc136286012)

[6. Bibliographie 19](#_Toc136286013)

# Introduction

Au terme du programme de bachelier d’ingénieur industriel, il est demandé d’effectuer un stage d’immersion en entreprise d’au minimum de 6 semaines. L’objectif étant de découvrir les différents aspects de la vie en entreprise, de ses défis, des recherches permanentes pour trouver la solution la plus adaptée pour le client. Ce stage a permis de mettre en pratique les connaissances acquises durant le bachelier mais aussi d’en apprendre de nouvelles.

Le rapport se structure comme suite : présentation des objectifs du stage, suivi par une description de l’entreprise By-Systems et de l’équipe mise en place pour réaliser ce projet. Ensuite une description du projet réalisé ainsi que les outils utilisés durant ce stage. Ce rapport se termine par la conclusion.

# Objectifs du stage

Avant le début du stage, notre maître de stage, Monsieur BOUJRAF, a fixé deux réunions préparatoires afin de définir le projet à réaliser et fixer les objectifs du stage. Suite à ces 2 réunions, les objectifs suivants ont été retenus pour la mise en place du projet :

* Définir la gouvernance liée au projet, y compris une présentation à réaliser chaque vendredi sur le travail effectué durant la semaine écoulée.
* Répartir les trois parties du projet (Front end, ETL et back-end) entre les 3 personnes de l’équipe
* Ecrire une API permettant d’extraire des informations de Jira et Jira Service Management.
* Pouvoir réagir à la modification d’une information sur Jira et Jira Service Management.
* Pouvoir stocker les informations dans la base de données.
* Pouvoir afficher les informations extraites de la base de données.
* Documenter le projet avec ce qui aura été réalisé.
* Stocker le code du projet dans github avec des commits vérifiés.

Une fois que les objectifs du stage ont étés clairement définis, il a fallu mettre en place une équipe projet afin de réaliser le projet. Le chapitre suivant décrit l’entreprise By-Systems où le stage s’est effectué et l’équipe projet mise en place pour réaliser ce projet.

# By-Systems et l’équipe projet

# Présentation de l’entreprise

L’entreprise By-Systems est située à Anderlecht. « Elle intègre des systèmes de montage non linéaires, des graphiques de diffusion en temps réel, des solutions de serveurs vidéos, des systèmes de gestion de réseau, des systèmes de contrôle de diffusion, y compris des informations, des sports, de la télé-réalité, des élections, des prévisions météorologiques et de la radio TV. » [1]

By-Systems a aussi réalisé des projets comme fournir une configuration complète pour 5 camions de production télévisée afin de diffuser les XXII Jeux Olympiques d’hiver qui se sont déroulés du 7 au 23 février 2014 à Sotchi en Russie.

# Business challenges

Les défis de l’entreprise d’un point de vue métier sont :

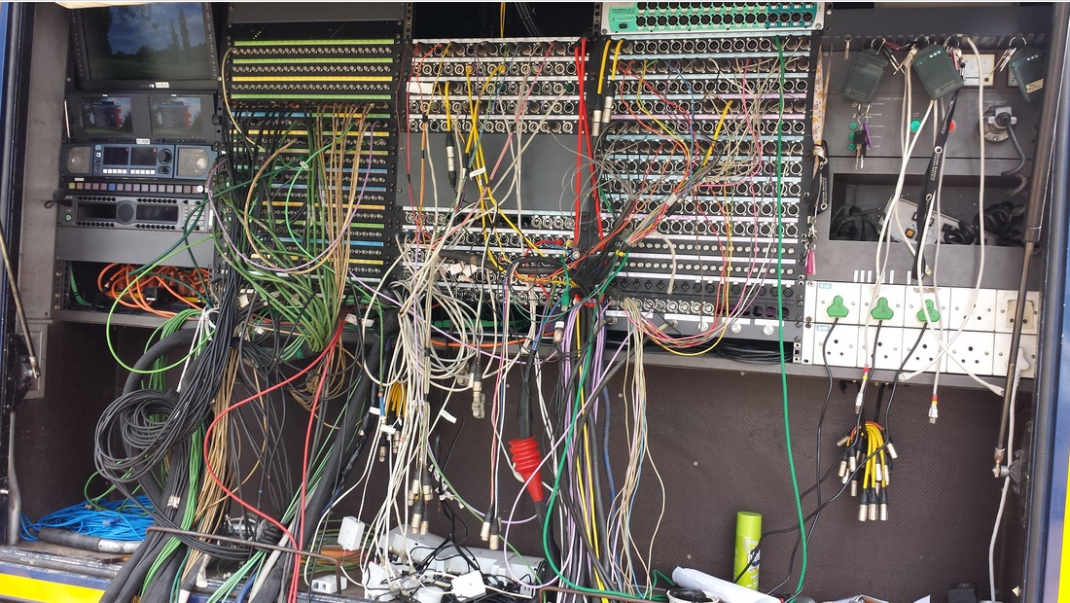
* Recueillir les besoins et les exigences commerciales concernant la diffusion en temps réel.
* Proposer une infrastructure robuste qui réponde aux besoins métiers de la diffusion en temps réels.

# Technical challenges

Les défis de l’entreprise d’un point de vue technique sont :

* Intégrer et coordonner plusieurs fournisseurs dans la mise au point de la solution
* Intégrer un contrôle à distance de tous les systèmes interconnectés
* Le contrôle des routers vidéos et audios
* Intégrer et gérer de multiples protocoles d’échanges
* Intégrer plusieurs types d’appareils et différentes plateformes afin de former un tout cohérent et fonctionnel.

Les photos suivantes illustrent la complexité d’une des solutions mise en place.



*Figure 1 : Câblage du camion [2]*



*Figure 2 : Intérieur du camion finalisé [3]*

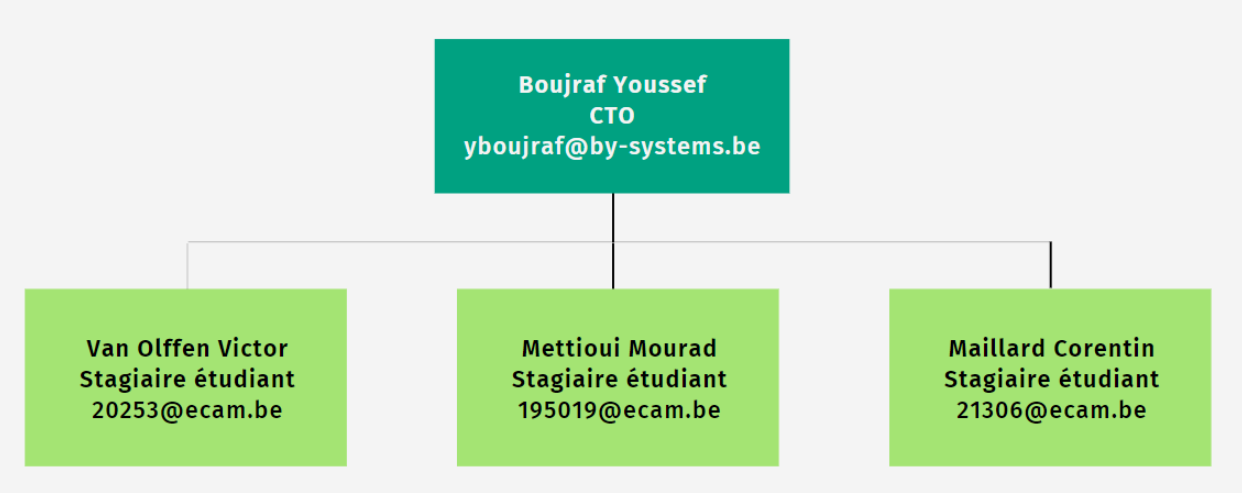
# Présentation de notre équipe

Pour mener à bien ce projet, une équipe de quatre personnes est mise en place avec la répartition des responsabilités suivantes :

* Youssef BOUJRAF : chef de projet & CTO (Chief Technology Officer). Il s’occupe de fixer les objectifs du projet, de l’encadrer. Il s’assure également au fur et à mesure de l’avancement du projet que ce dernier atteigne bien le but final, sans déviation.
* Victor VAN OLFFEN : stagiaire étudiant à l’ECAM en 3ème bachelier d’ingénieur industriel option électronique et informatique. Il gère la partie front-end du projet et s’occupe de l’affichage toutes les données provenant du back-end.
* Mourad METTIOUI : stagiaire étudiant à l’ECAM en 3ème bachelier d’ingénieur industriel option électronique et informatique. Il gère la partie back-end du projet et s’occupe de faire le lien entre le Jira-connector et le front-end.
* Corentin MAILLARD : stagiaire étudiant à l’ECAM en 3ème bachelier d’ingénieur industriel option électronique et informatique. Il gère la partie extraction de données du projet et s’occupe de faire un Jira-connector qui va extraire les données de Jira et les envoyer au back-end.

Etant donné que les trois parties sont interconnectées, il est primordial d’avoir une excellente communication entre les différents membres de l’équipe afin de s’assurer que les parties développées forment un tout cohérent et remplissent les objectifs exprimés au départ par Monsieur BOUJRAF.

L’organigramme suivant représente l’équipe mise en place.



*Figure 3 : Structure de l’équipe*

# Présentation du projet

# Introduction

Le projet qui nous a été confié, consiste à créer un connecteur qui va récupérer les informations de Jira Service Management et les envoyer au back-end afin que ce dernier les stocke dans une base de données. Le front-end, quant à lui, va demander au back-end les données à afficher sur une page web pour les clients. Le back-end va aller chercher les informations dans la base de données et envoyer les informations demandées par le front-end. Le back-End se charge également de faire la transformation des données de façon à ce que le front-end n’aie plus qu’à les afficher.

Durant le stage, il a été demandé d’utiliser Sphinx-Doc pour faire la documentation et comme générateur de document. Celui-ci présente l’avantage de pouvoir générer plusieurs formats de documentation comme latex, html ou même de pouvoir publier directement la documentation sur une page confluence. Pour ce faire, il faut écrire la documentation dans un fichier en reStructuredText (rst) qui est un langage de balisage utilisé pour la documentation.[4]

# Gestion du dépôt via Github

Avant de commencer les développements, s’est posée la question de l’endroit du stockage des sources et comment gérer les différentes versions. Il a été finalement décidé de choisir l’open-source github car le produit a déjà été utilisé dans le passé pour d’autres projets dans l’entreprise, il permet aux différents développeurs de partager le code informatique du projet afin de travailler de manière collaborative. Il permet aussi de gérer les versions et donc de pouvoir facilement annuler un changement en cas de souci et de revenir à une version précédente qui fonctionne. Et enfin, il est gratuit.

Avant d’utiliser Github, nous avons ajouté une connexion SSH et GPG afin de décompexifier le travail.

* SSH : est un protocole qui permet de se connecter à Github sans avoir à donner un nom et mot de passe. Il permet de faire des git push sécurisés dans le dépôt du projet. Lors de la création de la clé SSH, il est possible d’ajouter optionnellement un passphrase qui sera demandé à chaque « git push » pour confirmer que la clé n’a pas été volée. Il ne restera plus qu’à enregistrer la clé sur Github.[5]
* GPG : quand à lui permet de signer un commit afin d’être vérifié sur Github. Pour ce faire, il faut créer une clé GPG et l’enregistrer sur Github. De plus, pour renforcer la sécurité, il est possible d’ajouter lors de création de la clé GPG, un passphrase qui sera demandé lors de chaque commit.[6]

Pour structurer les commits sur github, la version sémantique et commitizen ont été retenus.

* Commitizen : permet d’avoir une norme d’écriture des commits. Un commit contiendra un « type » avec un « scope » suivi d’un « short et long message », il peut y avoir un « Breaking Point » et un « Issues Close ». Ils seront enregistrés avec la commande « git commit -S -m ’’type(Scope) : Short’’ -m ‘’Long’’ -m ‘’Breaking point’’ -m ‘’Issues Close’’. Il est à rappeler que le -S veut dire que le commit est signé avec la clé GPG.
* La version sémantique est un système qui indique la version d’un logiciel grâce à 3 chiffres (Major.minor.Patch). Les 3 chiffres représentent chacun un type de modification du logiciel :
* Le premier chiffre représente une modification majeure (Breaking Change) : ce qui veut dire que la mise à jour n’est pas rétro- compatible (ne fonctionne plus sur les versions antérieures). Ex : changer l’API.
* Le deuxième chiffre représente une modification mineure : il est utilisé lors de l’ajout d’une fonctionnalité rétro-compatible.
* Le dernier chiffre représente un patch : il est utilisé lors d’une correction de bugs.

Il faut savoir que, si un chiffre augmente, alors tous les chiffres sur la droite de celui-ci retournent à 0 et les autres chiffres sont conservés. Par exemple : la version actuelle est 2.3.4 et une modification majeure est effectuée, alors la nouvelle version deviendra 3.0.0. Maintenant, si la modification est mineure, alors la nouvelle version deviendra 2.4.0. Enfin, si la modification est un patch, alors la nouvelle version deviendra 2.3.5.

# Authentification à Jira et Jira service Management

Pour l’extraction des données de Jira et Jira Service Management, il a été décidé de d’abord tester l’extraction des données de Jira Service Management en utilisant des requêtes HTTP via Postman. Postman est une interface graphique permettant facilement de manipuler et tester le fonctionnement d’une API, sans devoir développer du code spécifique pour afficher les résultats. En plus, il est possible d’enregistrer la réponse d’une ou plusieurs requête(s) HTTP pour pouvoir les consulter ultérieurement. Postman permet également de déjà tester l’API alors la partie Front-end n’est pas encore finalisée.

Voyons un peu plus en détail comment extraire concrètement des données de Jira Service Management. La première étape est de s’authentifier lors des requêtes HTTP. Pour ce faire, les 2 méthodes d’authentification suivantes ont été utilisées et comparées :

* API token : La méthode d’authentification par API token pour Atlassian, consiste à générer un API token sur le compte Atlassian[7] pour pouvoir ensuite l’utiliser via la méthode d’authentification Basic Auth. Cette méthode permet de pouvoir commencer les requêtes directement en utilisant juste l’adresse email du compte Atlassian comme username et L’API token comme password. Voir l’exemple de la Figure 4 ci-dessous.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement*Figure 4 : Postman Basic Auth*

* OAuth 2.0 : Avant de pouvoir utiliser la méthode d’authentification OAuth 2.0, il faut d’abord aller sur le site d’Atlassian Developer et créer une nouvelle « apps ». Cette application permet de définir les permissions et le type d’autorisation (ici OAuth 2.0)[8].
* Une image contenant texte, capture d’écran, logiciel, nombre

  Description générée automatiquement*Figure 5 : Atlassian Developer Overview My Apps*

Il a fallu ajouter certaines permissions pour pouvoir faire certaines requêtes HTTP en OAuth 2.0. [9]

Une fois les permissions accordées, il faut récupérer le « client id » et « client secret » dans les Settings.

Dès lors, sur Postman, il est possible de créer un token temporaire qui aura les permissions indiquées dans le scope avec un durée de vie temporaire. Comme illustré sur la figure 4, pour créer ce token temporaire, il faut entrer les informations suivantes : le client id, client secret, l’URL d’accès du token (ici celui d’atlassian) et comme les scopes sont déjà définis sur Atlassian, il n’est pas utile de compléter le champ. C’est ce token qui sera utilisé lors des requêtes HTTP en OAuth2

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

*Figure 6 : Création d’un token en OAuth2*

Différentes requêtes ont été testées comme les :

* « GET Users » : Pour récupérer les utilisateurs.
* « GET Organizations » : Pour récupérer les organisations.
* « GET Customer in Organization » : Pour récupérer les clients dans les organisations.

Maintenant, intéressons-nous à la partie ETL proprement dite.

# ETL diagram

Pour mener à bien ce projet, la stratégie ETL a été utilisée. Le « E » veut dire EXTRACT : Comme le montre le diagramme ci-dessous, le Jira-Connector va extraire toutes les données utiles du Jira cloud avant de les envoyer au back-end. Le « T » veut dire TRANSFORM : il va s’occuper de transformer les données extraites via jira connector et de les mettre dans la base de données. Et finalement, le « L » veut dire LOAD : le front-end va demander au back-end les données de la base de données et les afficher.

Une image contenant texte, croquis, diagramme, Dessin technique

Description générée automatiquement

*Figure 7 : Schéma global du connecteur, back-end et front-end*

La partie mentionnée dans ce rapport concerne le fonctionnement du Jira-Connector. Les parties du projet concernant le backend et le front end ayant été développées par Mourad et Victor, ne seront donc pas développées dans ce rapport.

La section suivante est consacrée au jira-connector.

# Jira-Connector

# Requêtes HTTP

Le but du Jira-Connector, comme il a été mentionné dans le ETL Diagram, est d’extraire toutes les données de Jira et de Jira Service Management afin de les mettre à disposition du back-end.

Avant de commencer, il faut d’abord savoir quelle méthode d’authentification sera utilisée pour faire des requêtes HTTP. Comme évoqué dans la section Authentification, deux possibilités se présentaient : une authentification via API token ou l’authentification par OAuth 2.0. Pour tester ces deux méthodes, le logiciel Postman a été utilisé. L’interface graphique de Postman, permet de tester facilement et simplement quelques requêtes HTTP vers Jira via les deux méthodes d’authentifications pré-citées.

Comparons ces deux méthodes d’authentification afin de retenir celle qui sera la plus appropriée pour le projet :

* API token : permet de faire facilement des requêtes avec juste un token (créé sur Jira) et l’adresse mail utilisée pour la création du compte Atlassian à partir duquel on peut faire des requêtes HTTP.
* OAuth 2.0 : permet de faire des requêtes plus sécurisées car au lieu de donner l’API token créé sur Jira, un token temporaire va être fourni grâce à une requête HTTP. Ce qui veut dire que, même si, par mégarde, le token venait à devenir public, il ne serait plus valable au bout de 1h. De plus, cette méthode utilise un cloud id qui permet de cacher l’adresse du site et d’utiliser une adresse générale lors des requêtes. L’OAuth 2.0 doit avoir ses permissions (scopes) définies à l’avance. Ainsi, si lors d’une requête, celle-ci utilise un ou plusieurs scopes qui n’ont pas été autorisés, alors cette requête ne pourra pas s’effectuer.

Malgré que la méthode d’authentification OAuth 2.0 est la plus sécurisée, c’est finalement la méthode d’authentification API token qui sera retenue dans le cadre du projet pour sa simplicité de mise en œuvre.

Une fois la méthode d’authentification choisie, des requêtes HTTP vont être faites vers Jira pour pouvoir récupérer les données souhaitées. Ici le but était de récupérer les Issues avec leurs worklogs, les organizations ainsi que leurs customers et users.[10] Ces réponses sont ensuite formatées pour ressembler aux interfaces suivantes :

|  |  |
| --- | --- |
| Une image contenant texte, Police, capture d’écran  Description générée automatiquement | Une image contenant texte, Police, capture d’écran, ligne  Description générée automatiquement |
| Une image contenant texte, Police, capture d’écran, nombre  Description générée automatiquement | Une image contenant texte, capture d’écran, Police  Description générée automatiquement |
| Une image contenant texte, Police, capture d’écran  Description générée automatiquement |

HTTP request : UML organization

Une image contenant texte, diagramme, Parallèle, Plan

Description générée automatiquement

*Figure 8 : Diagramme UML Org*

Le diagramme UML Org schématise le processus suivant :

En lançant le programme, le Jira connector va faire un POST sur l’url de jira avec le username qui est l’adresse email du compte jira et le password qui est l’api token afin de recevoir un token d’authentification temporaire.

Ensuite, il va faire un GET des organisations de Jira Service Management avec l’url des organisations en utilisant le token précédemment obtenu.

Pour pouvoir comparer ce qui vient d’être obtenu avec ce qui se trouve déjà dans la base de données (data base), un GET contenant l’url de l’organisation va être fait vers le back-end. Le back-end va ensuite faire un GET sur la base de donées pour obtenir les organisations présentes dans la base de données avant de les renvoyer au Jira-Connector.

Enfin, une boucle for est réalisée pour comparer chaque organisation venant de Jira avec celle venant de la base de données et si elle n’existe pas déjà dans la base de données, alors elle sera envoyée au back-end via un POST qui va les ajouter en base de données. Le même processus est appliqué pour les Users, les Customers, les Worklogs et les Issues.

Les requêtes HTTP utilisent des promises (async/await) pour permettre l’exécution de plusieurs parties du code en simultané, sans attendre le résultat de la précédente (async = comportement asynchrone), ce qui rend le code plus rapide vu qu’il ne doit plus attendre la réponse avant de pouvoir passer à la suite du code. L’expression await, permet quant à elle d’interrompre l’exécution de la fonction asynchrone et attend la résolution de la promise passée. Les requêtes sont réalisées grâce à Axios qui est un client HTTP permettant de communiquer avec des API.

La section suivante présente le fonctionnement d’un webhook.

# Webhook

« Un webhook est un moyen pour une application de fournir en temps réel des informations. » [11]

Contrairement à l’API, qui fournit les informations quand on l’interroge, le webhook, quant à lui transmet l’information dès qu’elle est disponible. Ce qui peut faire économiser un temps précieux.

L’écran suivant montre comment est configuré le webhook jira-connecter.

Une image contenant texte, capture d’écran, logiciel

Description générée automatiquement

*Figure 9 : Configuration du webhook*

Avant de pouvoir utiliser le webhook, il faut d’abord le configurer. Pour ce faire, comme le mentionne la Figure 9, il faut lui donner une adresse de callback (adresse sur laquelle il va envoyer tous les events configurés). L’adresse du callback doit être une adresse publique, or, le Jira-Connector n’en n’a pas. C’est pourquoi, Ngrok a été utilisé, car ce logiciel permet de créer rapidement et facilement une adresse ip publique temporaire qui va permettre de transférer toutes les informations reçues via le webhook sur un port local au choix. Une fois l’url de callback défini, il ne reste plus qu’à préciser le type d’évènements pour lesquels le webhook va réagir. Dans notre cas, il s’agit des évènements suivant :

* Issue : création, mise à jour, suppression d’une issue d’un user ou customer.
* Worklog : création, mise à jour, suppression d’un worklog d’un user.
* User : création, mise à jour, suppression d’un user.

Le schéma suivant (figure 10), nous illustre le fonctionnement du webhook utilisant le Jira-Connector et l’IP publique obtenue via Ngrok :

Une image contenant texte, Police, capture d’écran, conception

Description générée automatiquement

*Figure 10 : Schéma du webhook*

À chaque création, modification ou suppression (CRUD) sur Jira Service Management d’un des events configurés dans le webhook (ici user, issue, comment, worklog), le webhook va envoyer l’event en question sur l’url de callback qui est une adresse publique (ici c’est l’ip généré par Ngrok). Ngrok va s’occuper de transférer l’event reçu sur son adresse publique vers le port 3000. Et le Jira-Connector va lire les informations reçues sur le port 3000 grâce à express.

Le schéma UML suivant représente la création d’un event de type worklog lors de la mise œuvre du webhook.

Webhook: UML Create Worklog

Une image contenant diagramme, ligne, Rectangle, blanc

Description générée automatiquement

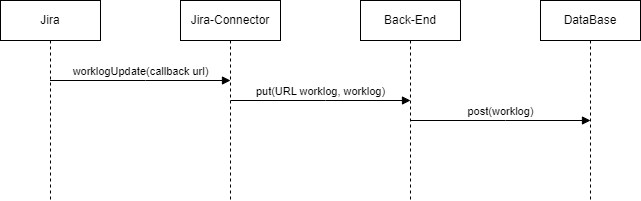
*Figure 11 : UML create worklog*

Quand une modification est faite sur l’un des events définis dans le webhook, Jira va envoyer cet event (ici la création d’un worklog) sur l’adresse publique fournie par Ngrok. Ngrok va l’envoyer vers le port 3000 où le Jira connector pourra lire cet event.

En fonction du cas de l’event, il sera formaté avant d’être envoyé au back-end par un POST. Ici, comme le cas était celui de la création d’un worklog, les informations du worklog sont envoyées en utilisant un POST au back-end sur l’url du worklog.

Le Back-end va ensuite faire un POST sur la base de données afin de mettre à jour celle-ci.

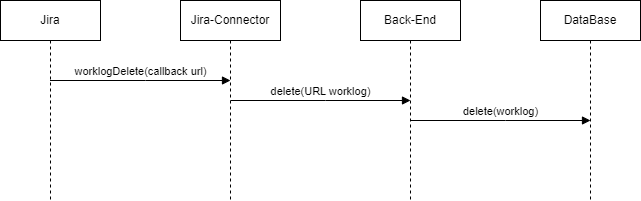
Webhook: UML Update Worklog

**

*Figure 12 : UML update worklog*

Pour la mise à jour d’un worklog, Jira va envoyer un event Update Worklog sur l’URL de callback qui va être transféré sur le port 3000 par ngrok. Ensuite le Jira-connector va lire le contenu de l’event et va exécuter le case Update Worklog. Jira connector va faire la même requête que pour le POST mais avec un PUT qui va être utiliser pour dire que l’on veut remplacer le worklog par celui envoyé. Pour ce faire, il faut donner dans l’URL du worklog son id et le contenu du worklog qui va être envoyé au back-end. Le back-end va ensuite chercher dans la base de données, le worklog correspondant, grâce à son id avant de faire un PUT sur la base de données pour remplacer le worklog correspondant.

Webhook : UML Delete Worklog



*Figure 13 : UML delete worklog*

Et pour la suppression d’un worklok ou d’une issue, le déroulement est un peu différent car il faudra utiliser un DELETE avec l’id du worklog ou de l’issue à supprimer dans la base de données.

Le point suivant est consacré aux difficultés rencontrées lors du stage.

# Difficultés rencontrées

Un projet se déroule rarement sans difficultés. Lors de la mise en œuvre du projet, les défis suivants se sont présentés :

* OAuth 2.0 ne permettait pas de récupérer les issues de Jira Service Managament. C’est pourquoi, la méthode d’authentification API token a été retenue comme méthode d’authentification finale.
* Le webhook n’envoie pas d’event lors de l’ajout ou de la suppression d’un customer ou d’un user
* Le webhook ne permet pas de configurer d’event lors de la création, mise à jour ou suppression d’une organisation.

Le point suivant décrit les pistes d’améliorations pour une seconde phase du projet.

# Pistes d’améliorations

Suite à ce stage, les pistes d’améliorations suivantes ont été identifiées :

* La mise en place d’un pare-feu pour Ngrok car actuellement il n’y en a aucun, ce qui permet à n’importe qui d’envoyer n’importe quoi vers Jira.
* Utiliser aussi le webhook pour les customers et les users.
* Voir si une future mise à jour de OAuth 2.0 permettrait de pouvoir récupérer les issues.

# Conclusion

Ce stage a permis de découvrir les aspects de la vie en entreprise. Notre maître de stage, Monsieur BOUJRAF a fixé les objectifs du projet et a assuré la gestion de projet. Le scope du projet a été défini afin de rencontrer les objectifs fixés et les tâches ont été réparties entre les personnes de l’équipe.

Ce rapport de stage s’est focalisé sur la partie concernant le développement de la partie Jira connector, permettant d’aller chercher des informations dans Jira. Pour accèder à jira, deux méthodes d’authentification ont été testées pour finalement retenir la méthode d’authentification API token.

Les outils suivants ont été utilisés lors de la réalisation du projet : Jira, confluence, Postman, afin de tester les API et les webhooks via son interface graphique, Ngrok pour obtenir une IP publique et de transférer l’event reçu sur le port 3000. L’api permettant d’obtenir les informations lorsqu’elle est appelée. Par contre, le webhook permet de transférer l’information sans y faire appel. Il suffit d’être à l’écoute. La notion d’ETL a aussi été utilisée pour le projet. Ensuite en fin de stage, les tests d’intégrations ont permis de vérifier que les différentes parties prévues au départ (Jira connector, Front-end, back-end et les interactions avec la base de données) communiquent correctement. Express permet d’écouter sur un port défini en localhost.

Sphinx-Doc a également été utilisé pour rédiger la documentation. Cet outil a été retenu afin de pouvoir générer plusieurs formats de fichier et de pouvoir directement publier la documentation dans confluence. Les sources ont été déposées sur Github au fur et à mesure des développements.

Comme tout projet, certaines difficultés sont survenues au cours de sa réalisation, et pour lesquelles une solution a pu être trouvée. Ce travail se termine par des pistes d’optimisations possibles qui pourraient être prises en compte lors d’une seconde phase éventuelle du projet.

# Bibliographie

**[1]** [By-system] by-system : *By-systems, broadcasting technologies*, en ligne : [HTTP://by-systems.weebly.com/](https://by-systems.weebly.com/). Consulté le 11 avril 2023.

**[2]** [By-system] by-system : *Dimension Television Facilities*, en ligne : <https://by-systems.weebly.com/dimension-facilities-tv.html>. Consulté le 11 avril 2023.

**[3]** [By-system] by-system : *Ano sports broadcasting*, en ligne : [HTTP://by-systems.weebly.com/ano-sports-broadcasting.html](https://by-systems.weebly.com/ano-sports-broadcasting.html). Consulté le 11 avril 2023.

**[4]** [Sphinx-Doc,2023] Sphinx-Doc : *Documentation : Getting Started*, en ligne :  [https://www.sphinx-doc.org/en/master/](HTTP://www.sphinx-doc.org/en/master/). Consulté le 4 avril 2023.

**[5]** [GitHub,2023] GitHub : *Génération d’une nouvelle clé SSH et ajout de celle-ci à ssh-agent : Une fois que vous avez vérifié les clés SSH existantes, vous pouvez générer une nouvelle clé SSH à utiliser pour l’authentification, puis l’ajouter à l’agent ssh*, en ligne : <https://docs.github.com/fr/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>. Consulté le 3 avril 2023.

**[6]** [GitHub,2023] GitHub : *Generating a new GPG key : If you don't have an existing GPG key, you can generate a new GPG key to use for signing commits and tags*, en ligne :  [https://docs.github.com/en/authentication/managing-commit-signature-verification/generating-a-new-gpg-key](HTTP://docs.github.com/en/authentication/managing-commit-signature-verification/generating-a-new-gpg-key). Consulté le 3 avril 2023.

**[7]** [Atlassian,2023] Atlassian : [*Keep your Atlassian account secure*](https://support.atlassian.com/atlassian-account/docs/keep-your-atlassian-account-secure/) *: Manage API tokens for your Atlassian account*, en ligne :  [https://support.atlassian.com/atlassian-account/docs/manage-api-tokens-for-your-atlassian-account/](HTTP://support.atlassian.com/atlassian-account/docs/manage-api-tokens-for-your-atlassian-account/). Consulté le 11 avril 2023.

**[8]** [Atlassian,2023] Atlassian : *Setting up OAuth 2.0 integration*, en ligne : [Setting up OAuth 2.0 integration | Atlassian Support | Atlassian Documentation](https://confluence.atlassian.com/jiracore/setting-up-oauth-2-0-integration-1005784173.html). Consulté le 12 avril 2023.

**[9]** [Atlassian,2023] Atlassian : *Jira Cloud Platform : REST API v2*, en ligne : <https://developer.atlassian.com/cloud/jira/platform/rest/v2/intro/#about>. Consulté le 12 avril 2023.

**[10]** [Atlassian,2023] Atlassian : *Jira Cloud Platform : REST API*, en ligne : <https://developer.atlassian.com/cloud/jira/platform/rest/v3/intro/#version> . Consulté le 12 avril 2023.

**[11]** [Mailchimp] Mailchimp : *Webhooks*, en ligne : <https://mailchimp.com/fr/marketing-glossary/webhook/#:~:text=Webhooks%20are%20a%20way%20for,constantly%20polling%20for%20new%20data>. Consulté le 12 avril 2023.