Projet SIE

Devènes Romain

April 2024

1 Introduction

La société MyGameServ qui propose des serveurs de jeux à plusieurs centaines de joueurs par jour souhaite integrer au seins de son ERP (odoo) un module qui lui permet de suivre en direct le status et le nombre de joueur présents sur chaque serveurs qu'ils possèdent.

Le problème qu'ils ont avec leur stratégie actuelle est qu'ils doivent pour chaque jeu accéder à des plateformes différentes pour obtenir les informations dont ils ont besoin.

L'idée est d'utiliser des API pour accéder aux données sur toutes les plateforme et de les regrouper sur un même module Odoo.

Dans la partie qui va suivre, nous allons analyser la situation actuelle de l'entreprise MyGameServ et réfléchir à une solution pour résoudre le problème présenté.

2 Analyse

2.1 Description du problème

Dans le cas de MyGameServ qui est une entreprise d'environ 30 personnes, le management à quelque difficulté à accéder à toutes les statistiques sur leurs différent serveurs de jeux. Ce retard les freines pour avancer dans le développement de leur entreprise et prendre les bonnes décision d'un point de vue organisationnel. Il y a une difficulté pour eux d'utiliser toutes les plate-forme pour accéder aux informations, malgré les effort par le service informatique pour regrouper au mieux tout les services au même endroit, ces efforts semble toujours insuffisant. C'est pourquoi ils souhaitent maintenant regrouper ces informations au seins de leur ERP.

2.2 Analyse du problème

Le processus actuel pour récupérer les données statistiques est presque entièrement manuel, ils doivent chercher eux même pour chaque jeux les informations de status et de statistique. Ce système les limites dans leur prise de décision rapide et leur font perdre du temps et des ressources qui pourraient être utilisée pour faire croître l'entreprise.

2.3 AS-IS

Actuellement, le processus se passe manuellement, ils récupère les informations des status des serveurs pour chaque jeu et les insère dans un excel qui est ensuite importé dans odoo leur ERP.

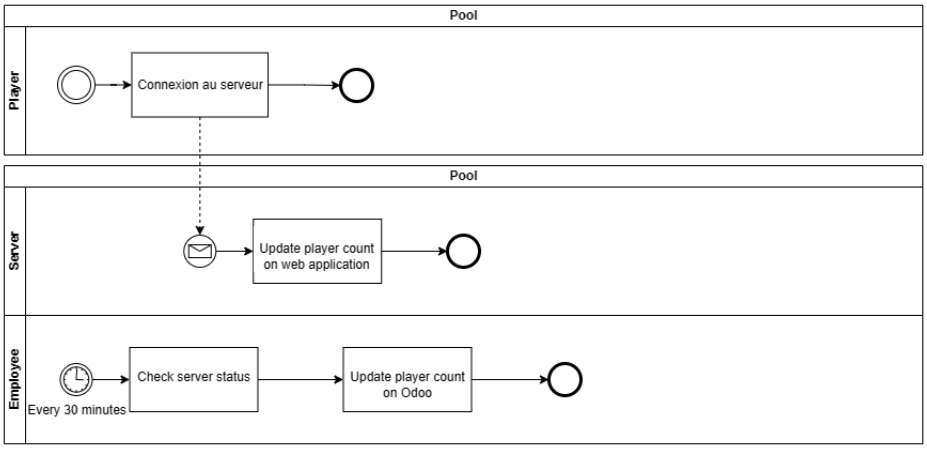

Figure 1: AS-IS process

Dans la partie qui va suivre, nous allons travailler sur la création d'une solution au problème de MyGameServ et nous allons présenter le design retenu pour améliorer la performance générale, la diminution des coûts et la prise de performance des prises de décision.

3 Proposition

3.1 Process design

Les acteurs qui seront utilisateurs du nouveau module à développer sont les départements de direction et business qui, souhaite accéder plus facilement aux données technique pour diriger la prise de décision. Le département informatique fait aussi partie des acteurs, leur rôle sera de fournir les accès au différentes API pour permettre de regrouper l'ensemble des serveurs.

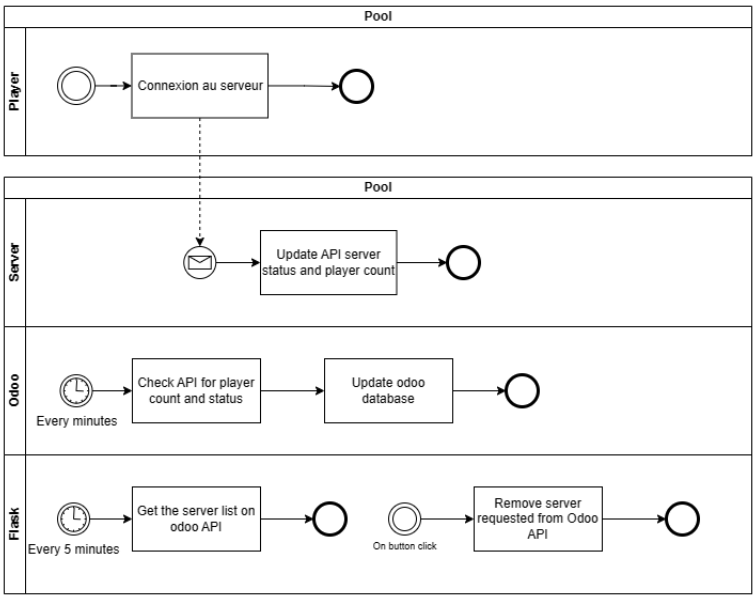

Figure 2: TO-BE process

Ce processus va permettre d'éliminer l'utilisation de ressource humaine dans le traitement des données de statut des serveurs, en permettant via les API de passer directement les données dans l'ERP Odoo. Il va aussi permettre de proposer une plate-forme web Flask qui va permettre d'accéder au donnée dans odoo directement sans passer par le module. Cette plate-forme permet aussi de supprimer un serveur dans le module.

3.2 Implémentation

3.2.1 Module odoo

L'implémentation du module s'est fait en suivant la documentation et les recomendation pour le développement d'un module odoo.

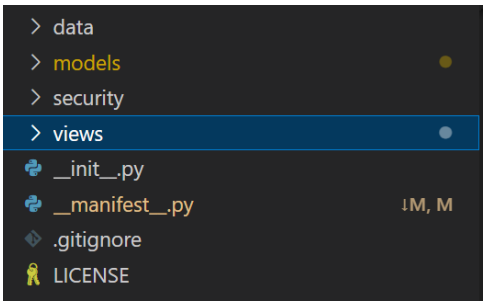

Figure 3: Structure du module

On retrouve les composant de donnée, modèle, sécurité et views qui suivent les directives de odoo en terme de module.

La partie la plus importante du code est celle qui permet de récupérer les informations via les API's des serveurs de jeux.

```python
def fetch_external_data(self):
        if not self.ip_addr:
            print("no_ip_addr")
            return
        url = f"https://api.mcsrvstat.us/3/{self.ip_addr}"
        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()
            self.update_data(data)
        else:
            _logger.error(f"Failed to fetch data for IP {self.ip_addr}: {response.status_code}")
```

Figure 4: Récupération des données

Le code ci-dessus est exécuté chaque minutes par le module via une routine. Il utilise les api's des serveurs pour récuperer les informations importante. La fonction update_data va mettre à jour les informations dans la base de donnée odoo. Dans ce cas particulier pour le status d'un serveur minecraft.

```python
def update_data(self, data):
        #updating data
        self.is_online = data.get('online', False)
        self.num_player = data.get('players', {}).get('online', 0)
        self.name = data.get('motd', {}).get('clean', [self.name])[0]
```

Figure 5: Mise à jour des données

3.2.2 WebApp Flask

La web app flask est une simple page avec une liste de serveur qui se trouve dans la base de donnée odoo. Il accède aux donnée odoo via l'API d'odoo et il est composé de deux simple fonction qui permettent d'afficher la pâge d'acceuil et de gérer la connexion à la base de donnée mais aussi de la logique qui permet en utilisant un bouton de supprimer un serveur de la base de donnée.

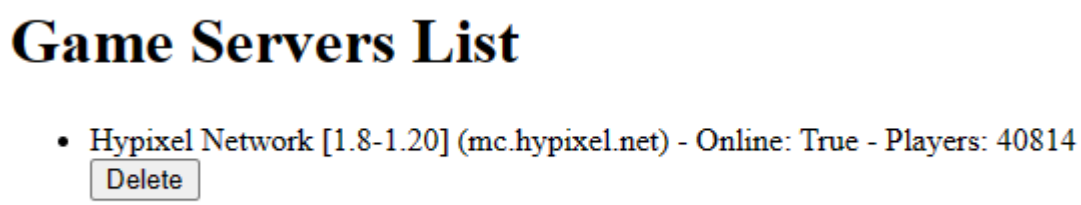

Figure 6: Application web

```python
@app.route('/')
def index():
    # Fetch all game servers
    ids = models.execute_kw(db, uid, password, 'server_status', 'search', [[]])
    servers = models.execute_kw(db, uid, password, 'server_status', 'read', [ids], {'fields': ['name', 'ip_addr', 'is_online', 'num_player']})
    return render_template('index.html', servers=servers)

@app.route('/delete/<int:server_id>', methods=['POST'])
def delete_server(server_id):
    # Delete a server
    models.execute_kw(db, uid, password, 'server_status', 'unlink', [[server_id]])
    return redirect(url_for('index'))
```

Figure 7: Code de l'application flask

3.3 Déploiement

Le déploiement du système est réalisé simplement en faisant un clone des repo github : Module Odoo, Flask app

Ensuite il faut installer le module dans le dossier de module externe de odoo et l'installer depuis l'interface administrateur de odoo.

La partie de gestion des API se fait directement par le module qui interagit avec odoo directement. Il n'y a pas d'installation nécessaire autre que le module pour le bon fonctionnement de celui-ci.

Il faut par ailleurs créer une action planifiée depuis les paramètres en mode développeur. Les 3 figures suivantes montre la marche à suivre pour automa-

tiser la mise à jour des serveur.

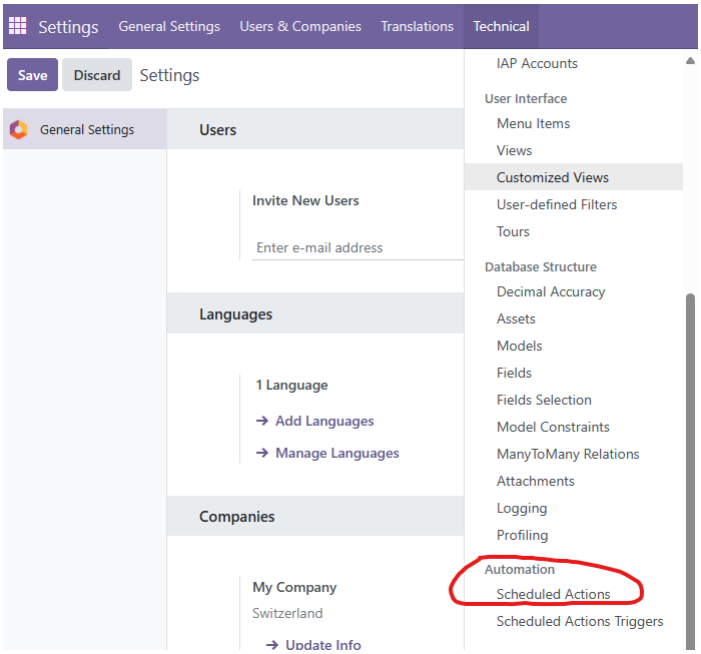

Figure 8: Ajout d'une tâche planifiée

Premièrement, il faut créer la tâche. Le model à exexuter s'appelle log the status of servers et on peut décider de la fréquence de mise à jour.

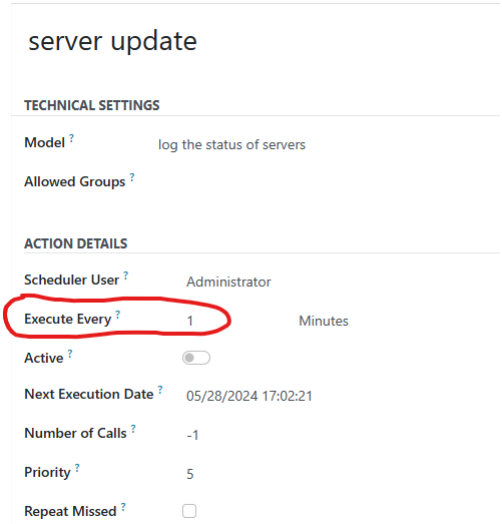

Figure 9: Crétation de la tâche

Une fois crée, la tâche devra executer la fonction fetch_external_data() qui va récupérer les informations de tout les serveurs enregistré. La portion de code est la suivante :

- for record in model.search([]): # Add specific domain if necessary record.fetch_external_data()

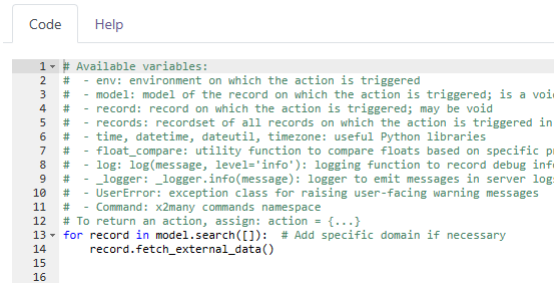

Figure 10: Code à executer

Pour l'application Flask, il suffit de l'executer et elle se connectrea automatiquement à l'API si les identifiants entrés sont corrects.

Dans la partie suivante, nous allons terminer par une conclusion et une analyse des résultats obtenu par le module. Nous allons analyser si les résultats obtenu par le module développé est capable d'atteindre les attentes initiales.

4 Conclusion

Le module d'intégration conçu pour MyGameServ représente une avancée significative dans la centralisation et l'automatisation des données des serveurs de jeux. Cette solution, qui s'appuie sur des API pour alimenter directement l'ERP Odoo, offre une vision unifiée et en temps réel qui est cruciale pour une prise de décision rapide et éclairée au sein de l'entreprise.

La mise en œuvre de ce module a permis d'éliminer les processus manuels fastidieux, réduisant ainsi les risques d'erreurs et le temps consacré à la collecte des données. Bien que le projet ait rencontré des défis, notamment l'adaptation des API aux spécificités des différents serveurs de jeux, les solutions développées ont prouvé leur efficacité.

Rétrospectivement, le choix de ce projet était ambitieux mais judicieux, car il répondait à un besoin critique de l'entreprise. Les difficultés rencontrées ont été des occasions d'apprentissage précieuses, et l'engagement de l'équipe informatique a été déterminant pour la réussite de l'implémentation. Malgré quelques obstacles initiaux lors du déploiement, les ajustements effectués ont permis de réaliser une intégration réussie.

En conclusion, le projet a non seulement amélioré les opérations internes de MyGameServ, mais a également posé les bases pour des améliorations continues dans la gestion de leurs ressources informatiques. La collaboration entre les départements de direction, business et informatique a été exemplaire, illustrant l'importance d'une vision partagée et d'une communication efficace dans la réalisation des objectifs stratégiques de l'entreprise.