Déclaration des nouvelles variétés dans PHIS et récupération de leur nom et URI

*Développement d’un outil externe à PHIS*

*Solution retenue pour répondre au cahier des charges*

Pour développer cet outil, la première étape a consisté à imaginer comment aller être utilisé cet outil pour répondre au besoin identifié dans le cahier des charges. Après une phase de réflexion et de test, l’idée retenue est la suivante : à partir d’une liste de variétés contenue dans une colonne d’un fichier Excel (figure 24), l’outil génère en sortie un fichier CSV respectant le format pour importer des variétés dans PHIS (figure 25).

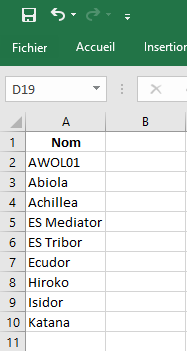


Figure 24 : Exemple d'une liste de variétés utilisable en entrée de l’outil

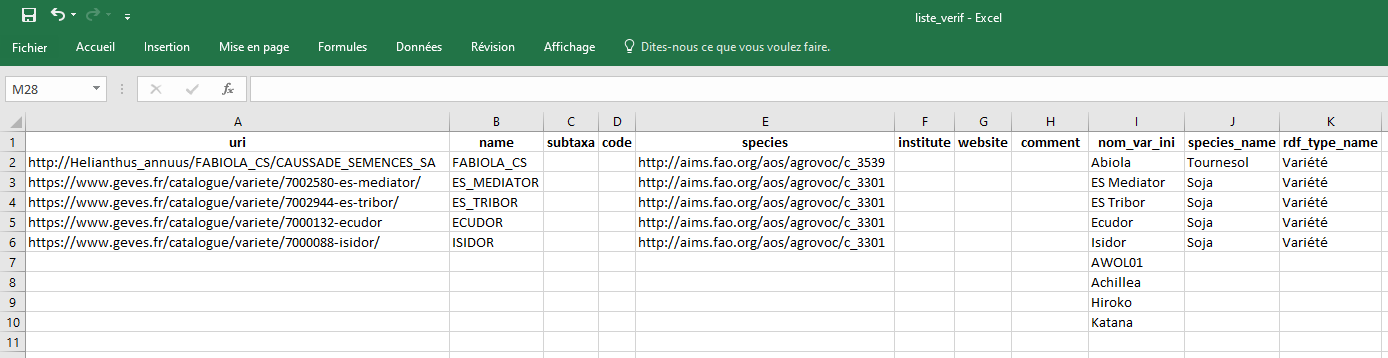


Figure 25 : Exemple de fichier CSV obtenu en sortie de l'outil ouvert dans Excel

Ce fichier (figure 25) est prérempli avec les informations des variétés déjà déclarées dans PHIS. Cela permet à l’utilisateur de vérifier si la correspondance est exacte pour les variétés trouvées et de remplir directement le fichier pour les variétés non déjà déclarées. Une fois ce fichier complété avec les informations manquantes, il peut être utilisé tel quel pour l’import des nouvelles variétés dans PHIS.

Une autre utilité de ce fichier est qu’il va pouvoir servir de fichier de correspondance entre le nom des variétés tel qu’il est dans la liste variétale fournie par le commanditaire (champ *nom\_var\_ini*), le nom tel qu’il est déclaré dans PHIS (champ *name*) et leur URI dans PHIS (champ *uri*). Pour ce faire, il suffit, lors de la génération du fichier CSV, d’y ajouter un champ contenant le nom des variétés tels qu’ils sont dans la liste variétale fournie en entrée (champ *nom\_var\_ini*).

*Développement du script de l’outil*

Après avoir fixé ce que devait permettre de faire cet outil, je suis passé à la phase de développement d’un script répondant au cahier des charges.

Ce script utilise l’API de PHIS (librairie *opensilexClientToolsPython*) pour rechercher l’existence de variété dans la base de données à partir du nom fourni dans liste variétale. Pour pouvoir utiliser cet API, il faut veiller à bien configurer le client. La fonction de recherche de PHIS étant peu performante (correspondance exacte seulement), j’ai ajouté dans le script la possibilité de rechercher un nom de variété sans espace (avec « \_ » à la place) pour un nom initialement avec un espace et inversement pour un nom initialement avec un « \_ ». La librairie *pandas* est utilisée pour lire les données tabulaires issu d’un fichier Excel, pour réorganiser ces données et pour les exporter au format CSV. Pour exploiter les résultats des requêtes de l’API de PHIS, la librairie *json* a été utilisé.

*Développement de l’application (App\_Verification\_Germplasm)*

Une fois le script fonctionnel, il a été nécessaire de l’inclure dans une application exécutable avec une interface utilisateur de façon à ce que n’importe qui puisse paramétrer et exécuter le script. La première étape consistait donc à créer l’interface graphique en Python, mon script étant déjà en Python. Après quelques recherches, j’ai opté pour le module PyQt5 étant donné la multitude de ressources disponibles en ligne et la possibilité d’utiliser le logiciel Qt Designer, un outil qui permet de générer simplement le code Python des interfaces graphiques (\*.ui) à partir d’un éditeur graphique.

Les interfaces utilisateur ont donc été réalisé avec Qt Designer en disposant les éléments de différents types (bouton, barre de texte, label …) par glisser-déposer (figure x). Chaque élément de l’interface est nommé pour pouvoir le relier au script.

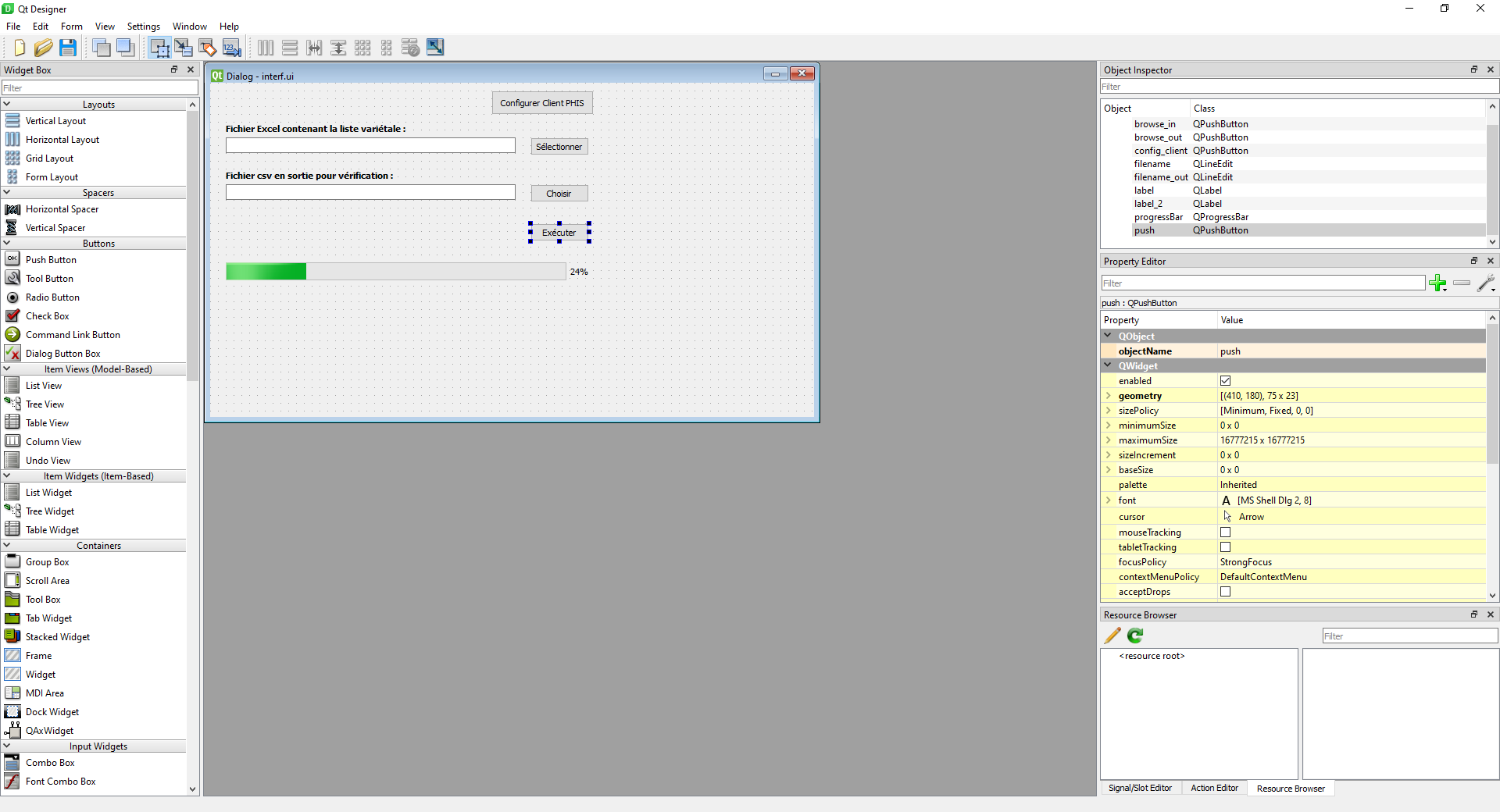


Figure 26 : Exemple de création d'une interface graphique avec Qt Designer

L’application comporte quatre interfaces graphiques :

* la principale permettant de paramétrer le chemin du fichier Excel en entrée et l’emplacement du fichier CSV en sortie, d’accéder au formulaire de paramétrage du client pour l’API de PHIS, d’exécuter le script et de donner à l’utilisateur une idée de l’état d’avancement du traitement avec une barre de progression.

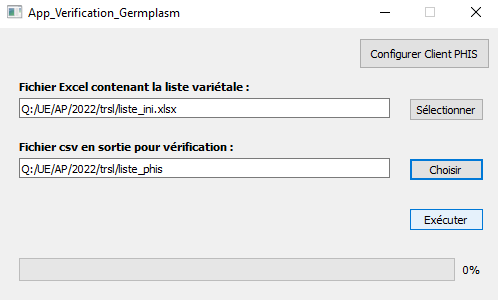


Figure 27 : Interface principale de l'application App\_Verification\_Germplasm

* une autre correspondant au formulaire de paramétrage du client pour l’API de PHIS (avec la possibilité de sauvegarder la valeur de ces champs et de revenir à l’interface principale en cliquant sur OK)

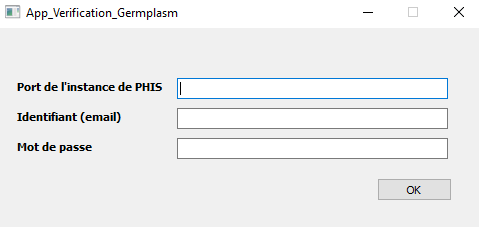


Figure 28 : Interface de l'application App\_Verification\_Germplasm permettant de configurer le client de l'API de PHIS

* une correspondant à un pop-up pour signaler la bonne exécution du script (avec la possibilité de revenir à l’interface principale en cliquant sur OK)

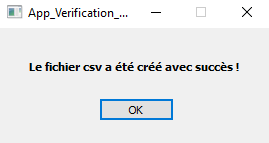


Figure 29 : Interface de l'application App\_Verification\_Germplasm permettant de signaler le succès de l'exécution

* une correspondant à un pop-up pour signaler une erreur lors de l’exécution du script (avec la possibilité de revenir à l’interface principale en cliquant sur OK)

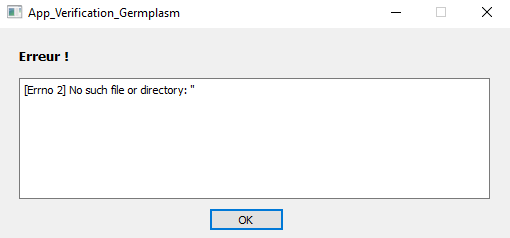


Figure 30 : Interface de l'application App\_Verification\_Germplasm permettant de signaler une erreur lors de l'exécution

L’étape suivante a été de faire le lien entre les interfaces graphiques présentées ci-dessus et le script en utilisant les modules de PyQt5. Étant donné ma faible expérience en programmation orienté objet sur Python je me suis inspiré de ressources disponibles en lignes. Ces ressources utilisées étant des vidéos accessibles sur YouTube, les liens de ces dernières sont donnés dans la partie Bibliographie ou ressources en fin de rapport.

Je ne peux pas expliquer en détail ici le code de l’application mais je mets à disposition le script commenté de l’application en annexe (annexe X). Globalement, cette application offre la possibilité à l’utilisateur de naviguer entre les différentes interfaces (avec les modules *QDialog*, *loadUi*), d’utiliser l’explorateur de fichier pour renseigner les chemins de fichiers (avec le module *QFileDialog*), de garder en mémoire les champs du formulaire contenant les informations de connexion à l’API de PHIS (avec le module *QSettings*) et bien-sûr de paramétrer et exécuter le script.

La dernière étape consistait à trouver une solution pour que l’application soit exécutable depuis n’importe quel ordinateur et par n’importe qui. En effet, à ce stade l’application ne pouvait être lancée qu’en exécutant le fichier ‘.py’ de l’application dans un environnement disposant des packages nécessaires au bon fonctionnement de l’application.

Après plusieurs pistes étudiées, la solution finalement retenue a été de faire un exécutable ‘.exe’ à partir du script ‘.py’ avec un environnement virtuel contenant Python et toutes les packages nécessaires au bon fonctionnement de l’application. Comme pour le développement de l’application, je me suis appuyé sur des vidéos disponibles sur YouTube pour mettre en œuvre la solution retenue. La procédure précise pour la création du ‘.exe’ est disponible en annexe (annexe liste étape création .exe). Globalement, cette procédure consiste à créer un environnement virtuel grâce au package Python *Virtualenv* et d’y installer les packages nécessaires au bon fonctionnement de l’application puis de générer l’application ‘.exe’ avec le package Python *Pyinstaller*.

On obtient finalement un dossier contenant de nombreux fichiers et l’application ‘.exe’ (annexe X dossier avec .exe). L’application peut donc être diffusée sur n’importe quel poste ou sur le réseau en exportant ce dossier.

Déclaration de l’ITK de Geofolia dans PHIS (lien Geofolia-PHIS)

Pour rappel, l’objectif de cette étape est de pouvoir renseigner dans PHIS la façon avec laquelle les µp ont été conduites au cours de l’expérimentation à partir de la saisie des interventions réalisés dans Geofolia à l’échelle des unités culturales (UC). Le fait de vouloir déclarer une information à l’échelle de la µp alors que l’on dispose seulement de l’information à l’échelle de l’unité culturale a été le problème à résoudre pour cette étape.

L’écriture du cahier des charges pour cette étape a permis d’identifier comment solutionner ce problème. L’idée est de déclarer dans PHIS des objets correspondant aux UC de Geofolia pour renseigner ensuite l’ITK mené sur ces dernières. On utilise finalement la capacité de PHIS à établir des connexions entre objets pour indiquer le lien de parenté entre les objets correspondant aux µp et les objets correspondant aux UC.

La mise en œuvre de cette solution se concrétise par 3 étapes au niveau du nouveau processus.

L’étape permettant la déclaration des OS correspondant aux UC a déjà été expliqué précédemment. Cela correspond à l’étape  du processus global permise par le développement du modèle *gen\_gabarit\_phis\_uc* utilisé lors de l’étape  de la chaîne de traitement sur ArcGIS. Il a fallu pour l’occasion créer dans PHIS un nouveau type d’objet scientifique « *ManagementUnit* » dédié aux UC.

L’ étape permettant d’indiquer le lien entre les objets correspondant aux µp et les objets correspondant aux UC lors de la déclaration des OS des µp a déjà été expliqué précédemment. Cela correspond à l’étape  du processus global permise par le développement du modèle *ajout\_uri\_uc* utilisé lors de l’étape  de la chaîne de traitement sur ArcGIS et du modèle *gen\_gabarit\_phis\_mp* utilisé lors de l’étape  de la chaîne de traitement sur ArcGIS.

Les réalisations techniques pour mener à bien l’étape permettant de déclarer les interventions dans PHIS à partir des fichiers exportables de Geofolia sera explicité ci-dessous.

*Solution retenue pour répondre au cahier des charges*

Comme pour l’application App\_Verification\_Germplasm, j’ai décidé de développer un outil externe à PHIS.

Une application permettant de faire le lien entre Geofolia et PHIS a déjà été développée par des membres de l’INRAE Clermont-Ferrand. Néanmoins, cette application ne répondait pas exactement aux besoins de l’UE étant donné qu’elle a été développée pour l’UE du centre de Clermont-Ferrand dont la gestion du parcellaire et la façon de nommer les parcelles sont différentes par rapport à l’UE.

L’idée était donc de s’inspirer de cette application pour l’adapter au besoin identifié dans le cahier des charges.

Cette application doit donc permettre à partir d’un fichier contenant les listes des interventions exportées de Geofolia de déclarer ces interventions dans PHIS comme des « Évènements » à l’échelle des objets scientifiques de type « Unité culturale ». Le prérequis étant que les UC sur Geofolia et les OS correspondant dans PHIS porte exactement le même nom.

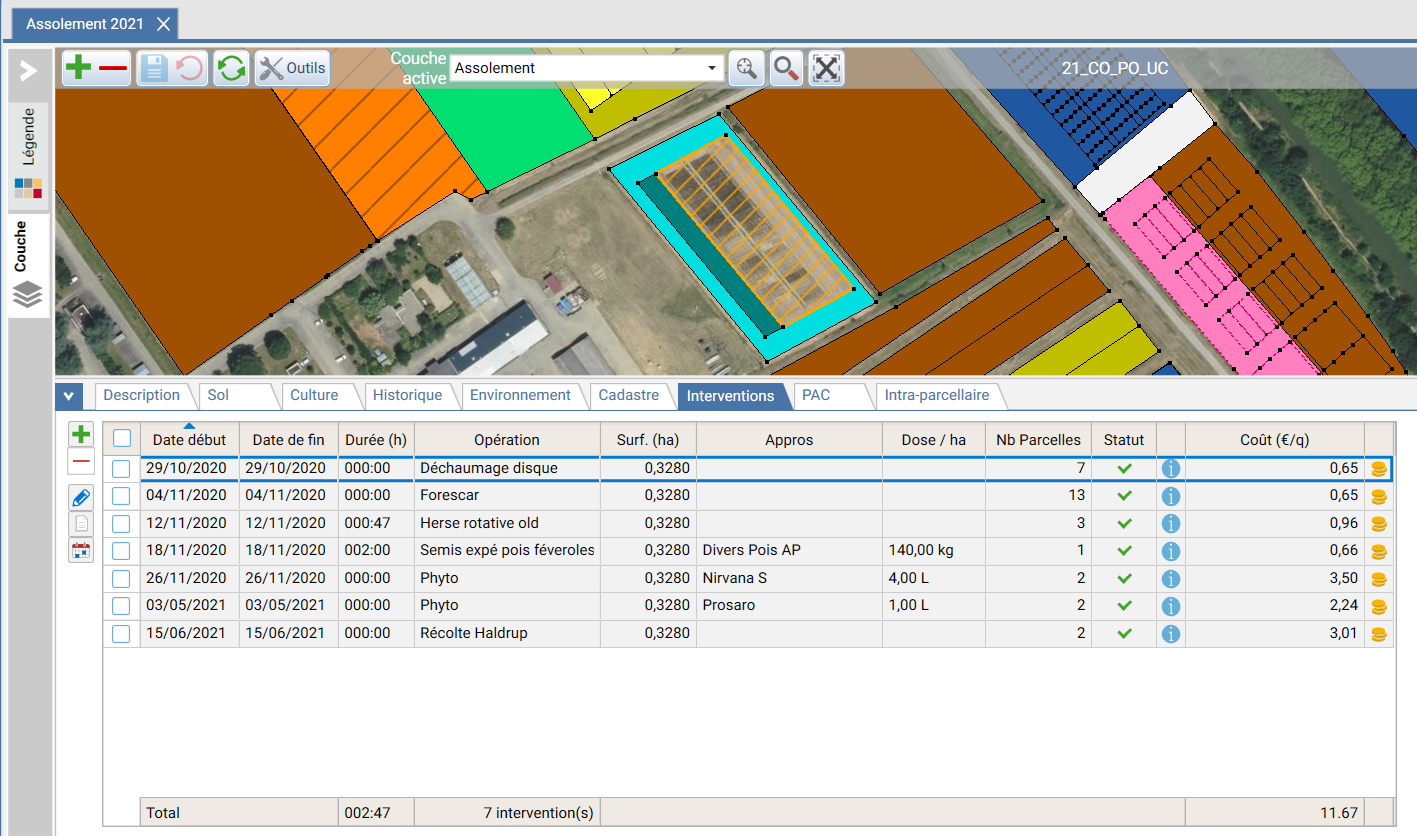
Le but est que pour chaque « Évènement » le type dans PHIS soit défini par le type d’intervention de Geofolia, la date de début et de fin correspondent à celles saisie dans Geofolia et que dans la description de l’ « Évènement » on puisse retrouver le nom de l’opération et si l’intervention correspond à l’apport d’un produit, on puisse en plus retrouver le nom du produit ainsi que la dose par hectare apportée avec la bonne unité (figure 31).

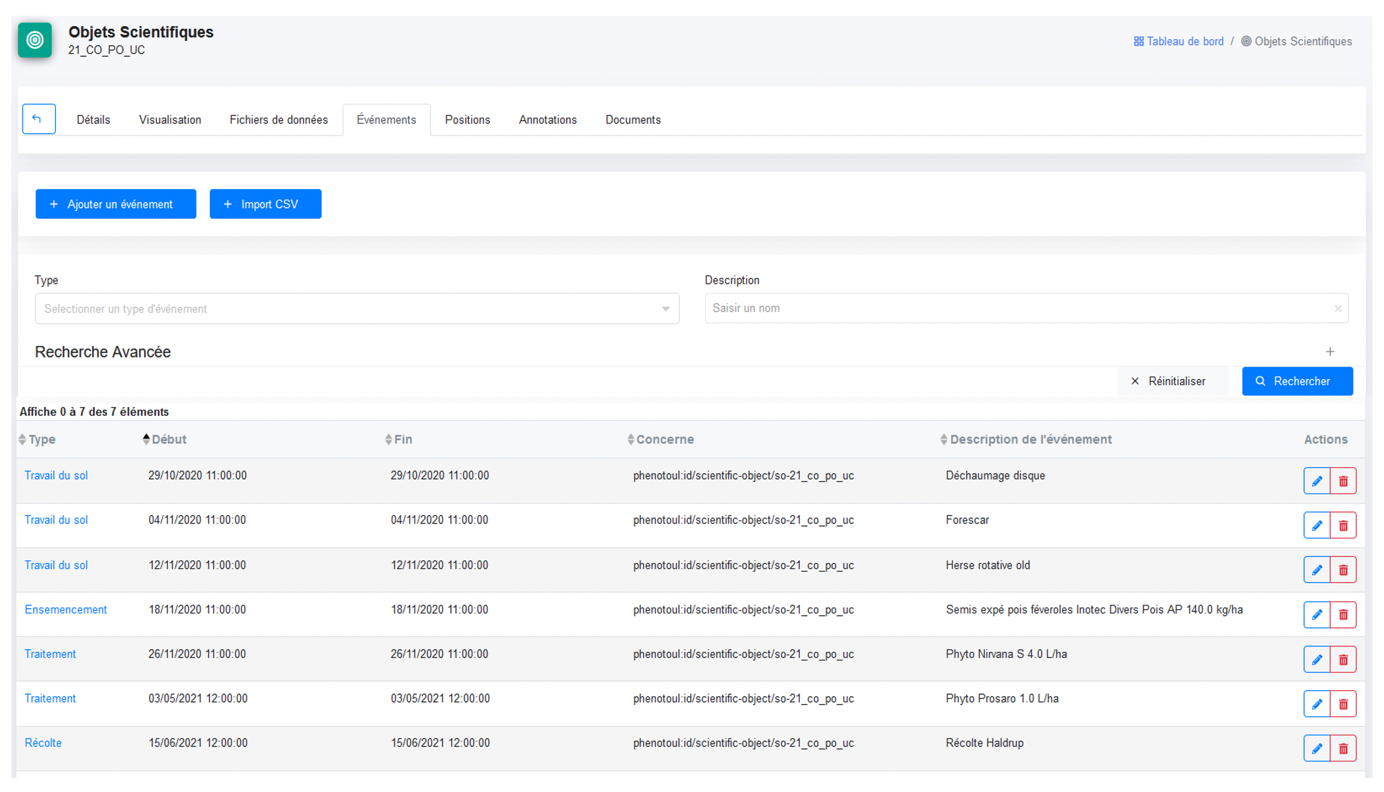
figure 31 ci-dessous :

Illustration utilité de l’application :

Liste interventions dans Geofolia

Liste des mêmes interventions dans PHIS





*Développement du script de l’outil*

Le script de l’outil s’est fait en s’inspirant de celui de l’application développé par des membres de l’INRAE Clermont-Ferrand. Il a donc fallu dans un premier faire tout un travail de décryptage du code de cette application pour identifier quelles parties permettait la déclaration de l’ITK de Geofolia et de comprendre leur fonctionnement.

En parallèle, le travail de compréhension de la structure des fichiers JSON exportés de Geofolia a été primordial pour savoir comment en extraire les informations nécessaires.

Après ce travail préliminaire, j’ai pu passer à la phase de développement du script. Ce script utilise les packages *zipfile* et *json* pour lire et récupérer le contenu du fichier ‘.zip’ exporté de Geofolia. La librairie *datetime* a été utilisé pour mettre les dates au format requis par PHIS. Les packages *opensilexClientToolsPython* (API de PHIS), *request* et *urllib* sont utilisés pour récupérer les URI des UC à partir de leur nom.

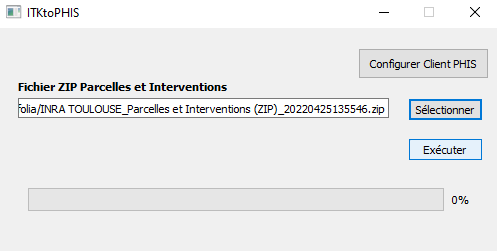
Une différence dans ce script par rapport celui dont je me suis inspiré est que la récupération des URI des OS d’UC se base sur la correspondance exacte des noms entre PHIS et Geofolia. Une autre différence est que ce script permet de remplir la description des « Évènements » déclarés de façon plus précise (nom du produit, dose calculée à l’hectare avec la bonne unité).

L’API de PHIS est aussi utilisée pour déclarer les évènements qui ne le sont pas encore. En effet, ce script permet de ne pas redéclarer des interventions qu’il le serait déjà. La librairie *pandas* est aussi exploité pour gérer et mettre en forme les données.

*Développement de l’application* (ITKtoPHIS)

Le processus de création de l’application sous forme exécutable avec des interfaces utilisateurs est identique à celui présenté pour l’application *App\_Verification\_Germplasm*.

Il a seulement fallu adapter l’interface principale pour n’avoir que l’emplacement pour renseigner le chemin du fichier ZIP issu de Geofolia (figure X).



Le contenu du pop-up signalant la bonne exécution du script a été modifié pour permettre de donner les détails des déclarations à la fin de l’exécution. Cela permet à l’utilisateur de savoir combien d’interventions ont été déclarées et de savoir combien d’interventions avaient déjà été déclarées avec les OS concernés (figure X).

