BORGES DE ANDRADE
Vanessa Caroline
BTS SIO 2ème année

BREVET DE TECHNICIEN SUPÉRIEUR SERVICE INFORMATIQUE AUX ORGANISATION

OPTION B : SOLUTIONS LOGICIEL ET APPLICATION MÉTIER

- Epreuve E6 -

<u>Projet 2 :</u> Automatisation du transfert de données GLPI local vers le GLPI national via script Python

SESSION: 2025

Sommaire

Introduction

1.1 Contexte général

Dans les administrations publiques françaises, la gestion efficace du parc informatique et des services d'assistance est un enjeu crucial. Pour répondre à ce besoin, de nombreuses structures utilisent **GLPI** (Gestionnaire Libre de Parc Informatique), un logiciel libre permettant de centraliser les informations relatives aux équipements, utilisateurs, incidents et interventions techniques.

Dans une logique de modernisation, de mutualisation des ressources et de sécurisation des données, l'État a initié une **centralisation des outils locaux vers une instance nationale**. Cette nouvelle **plateforme GLPI nationale**, administrée par la DSI centrale (Direction des Systèmes d'Information), regroupe les données des différentes préfectures et directions sous un seul et même environnement sécurisé, accessible par API et segmenté par entités.

Cette mutualisation vise à :

- Uniformiser les processus de gestion informatique,
- Réduire les coûts de maintenance,
- Améliorer la sécurité des données,
- Offrir une supervision globale du SI à l'échelle nationale.

1.2 Contexte local : La Préfecture de Guyane

J'ai réalisé mon stage au sein du **service informatique de la Préfecture de Guyane**, située à Cayenne. Cette préfecture possédait jusqu'alors sa **propre instance locale de GLPI**, hébergée sur un serveur Linux interne. Elle s'en servait pour gérer :

- le parc informatique (ordinateurs, imprimantes, écrans...),
- les comptes utilisateurs et leurs affectations matérielles,
- la gestion des tickets (incidents, demandes de matériel...),
- le suivi des interventions techniques.

Dans le cadre de la centralisation engagée au niveau national, la préfecture a été sollicitée pour **abandonner son GLPI local au profit de l'instance GLPI nationale**. Cela impliquait de **migrer l'intégralité des données locales** (équipements, utilisateurs, historiques de tickets, etc.) vers la nouvelle plateforme, sans perte ni altération.

Cependant, **aucune procédure automatique** ou outil clé-en-main n'était fourni pour réaliser ce transfert. La saisie manuelle des milliers d'entrées aurait été chronophage, fastidieuse, et sujette à erreurs. Face à cette difficulté, la solution retenue a été de **concevoir un script d'automatisation du transfert de données**, reposant sur l'**API REST de GLPI national**.

1.3 Présentation du projet

Le projet qui m'a été confié visait à **développer un script Python** capable de :

- **Lire les données extraites de GLPI local** (au format CSV),
- **Se connecter à l'API du GLPI national** avec authentification par jeton,
- **Créer automatiquement les objets** (ordinateurs, logiciels, utilisateurs, tickets...) dans la base nationale,
- **Enregistrer les erreurs et doublons dans un fichier journal**.

Ce projet m'a permis d'aborder plusieurs compétences clés :

- La **compréhension d'une API REST** et son intégration dans un script Python,
- La **gestion de fichiers CSV** et le traitement de données structurées,
- La **programmation orientée automatisation** dans un contexte réel,
- La **collaboration avec une équipe technique** sur un projet métier.

Ce travail s'inscrivait dans une démarche de transition numérique de l'administration et a permis d'outiller la migration vers GLPI national de manière fiable, rapide et réutilisable.

Présentation de l'organisation

2.1 Activités de l’organisation

La Préfecture de la Guyane est un service déconcentré de l’État français situé dans un département d’outre-mer. Placée sous l’autorité du préfet, elle coordonne l’action des services de l’État, veille à la sécurité publique, gère les crises, et met en œuvre les politiques gouvernementales dans les domaines économique, social et environnemental.

Face aux spécificités territoriales de la Guyane – comme les flux migratoires importants, les enjeux environnementaux ou encore l’aménagement des infrastructures – la préfecture s’organise autour de plusieurs directions (sécurité publique, affaires juridiques, etc.) qui collaborent pour répondre aux besoins du territoire.

En soutien à ces missions, la Direction du Système d’Information (DSI) joue un rôle essentiel en assurant la gestion, la sécurisation et la modernisation des outils numériques nécessaires au fonctionnement quotidien de l’administration.

2.2 Situation géographique de l’organisation

La préfecture de la Guyane est située à **Cayenne**, chef-lieu du département, sur la côte atlantique du nord-est de l’Amérique du Sud.

Cette localisation particulière implique une gestion spécifique des moyens logistiques et informatiques, notamment en raison de l’éloignement de la métropole et des contraintes climatiques locales (humidité, chaleur, instabilités réseau).
Son implantation stratégique lui permet de centraliser les services de l’État pour l’ensemble du territoire guyanais.

2.3 Personnel de l'organisation

La Direction des Systèmes d'Information (DSI) de la préfecture de Guyane est structurée autour de plusieurs pôles spécialisés, chacun ayant des missions précises. Elle est dirigée par deux responsables principaux. Le responsable général de l'administration est Monsieur le préfet **Antoine POUSSIER**.

Elle se compose des équipes suivantes :

Direction	Cellule SSI / SN Sécurité Numérique	Cellule suivi	Projets Transformation Numérique	Environnement numérique de travail- ENT	Infrastructure	SIC Transmission
2	0	4	2	15	5	5

Les équipes de la DSI assurent l'ensemble des missions informatiques de la préfecture : support technique, cybersécurité, développement applicatif, gestion de projets et innovation numérique.

Services / Missions :

Gestion et maintenance des infrastructures informatiques
La DSI veille au bon fonctionnement des réseaux, serveurs, ordinateurs et autres équipements informatiques. Elle s'assure que le personnel de la préfecture dispose d'outils performants pour mener à bien ses missions.

Sécurité des systèmes d'information
La protection des données et des systèmes contre les menaces (piratage, attaques, fuites de données) est une priorité. La DSI met en place des protocoles de sécurité, gère les accès et supervise la protection des informations sensibles.

Support technique et assistance aux utilisateurs
Le service informatique apporte une aide quotidienne aux agents en cas de problèmes techniques (pannes, erreurs système, problèmes logiciels). Il assure également la formation du personnel à l'utilisation des outils numériques.

Développement et gestion des applications

La DSI conçoit ou adapte des logiciels pour répondre aux besoins spécifiques des services de la préfecture, notamment dans la gestion des titres de séjour, des flux migratoires ou encore dans la coordination des services de sécurité. Elle assure la maintenance et la mise à jour de ces applications.

Pilotage des projets informatiques

Elle gère les projets informatiques de leur conception à leur déploiement. Cela comprend l'analyse des besoins, la coordination avec les autres services de l'État et la supervision de la mise en œuvre des solutions techniques.

Innovation et transformation numérique

La DSI contribue activement à la modernisation de l'administration en introduisant des technologies innovantes (dématérialisation, intelligence artificielle, automatisation) dans le but d'optimiser les services publics.

2.4 Organisation matérielle du Système d'Information

Le **Système d'Information (SI) de la Préfecture de Guyane** repose sur une infrastructure hybride, mêlant des équipements physiques sur site et des solutions virtualisées pour garantir la **continuité de service, la flexibilité et la sécurité** des données et des flux informatiques.

Parc informatique

Le parc est constitué de **plusieurs centaines de postes de travail**, répartis entre les différents services administratifs. Ces machines sont principalement équipées de **systèmes d'exploitation Windows**, avec une configuration standardisée afin de :

- faciliter les opérations de maintenance,
- accélérer le déploiement de nouveaux postes,
- assurer une compatibilité homogène avec le domaine **Active Directory** de la DSI.

Les utilisateurs disposent d'un profil associé à leur fonction, avec des droits adaptés selon les règles établies par la hiérarchie.

Serveurs et virtualisation

Les serveurs sont hébergés dans une **salle technique sécurisée**, équipée de dispositifs de contrôle d'accès, de climatisation redondante et de protections électriques.
L'environnement serveur repose sur une **architecture virtualisée**, exploitant des technologies comme **VMware** (ou **Hyper-V**, selon le cas réel rencontré). Cette virtualisation permet de :

- créer, modifier ou déplacer facilement des machines virtuelles,
- optimiser l'utilisation des ressources physiques,
- réduire les coûts d'exploitation et de maintenance.

Certains services critiques (Active Directory, GLPI, antivirus, DNS) sont hébergés sur ces machines virtuelles, avec des plans de bascule en cas de panne.

Réseau et sécurité

Le réseau est **segmenté en plusieurs VLAN**, pour isoler les flux sensibles et limiter la propagation d'éventuelles attaques. On distingue notamment :

- un **réseau administratif interne** pour les agents,
- un **réseau invité sécurisé** destiné aux visiteurs,
- des VLAN spécifiques pour les **serveurs**, **la téléphonie IP**, ou encore les **imprimantes**.

L'ensemble est protégé par un **pare-feu matériel**, complété par un **système de détection/prévention d'intrusion (IDS/IPS)**. Des **switchs managés** permettent un contrôle fin du trafic, et des **routages segmentés** garantissent la cohérence des accès.

Sauvegarde et supervision

Des **solutions de sauvegarde automatisées** sont en place pour protéger les données critiques. Des copies régulières sont stockées localement et/ou sur des supports externes chiffrés, avec des tests de **restauration périodiques** pour valider les plans de reprise d'activité.
La supervision du système est assurée par des outils comme **GLPI (pour l'inventaire et les tickets)** et potentiellement **Centreon** ou équivalents, pour le monitoring des ressources et l'alerte en cas d'incident.

2.5 Problématiques rencontrées

Avant le début du projet, plusieurs difficultés ont été identifiées :

- La **volumétrie importante des données** à migrer (plusieurs centaines de fiches matériels, utilisateurs, historiques de tickets),
- L'absence de **fonction d'import native vers le GLPI national**,
- La **complexité de l'API REST**, nécessitant une bonne compréhension de son fonctionnement (authentification, création d'objets, gestion des erreurs),
- Le **risque de doublons ou d'incohérences** entre les données locales et la structure attendue par le GLPI national.

Ces éléments ont justifié le besoin d'une **solution automatisée, fiable et sécurisée**, d'où le développement d'un script Python sur mesure.

Solution

3.1 Présentation du projet

Face au besoin de **centraliser les données GLPI** de la préfecture de Guyane vers l'instance **nationale**, le projet consistait à **concevoir un script automatisé** permettant de migrer de grandes quantités de données depuis l'ancienne instance locale (GLPI on-premise) vers le nouveau système centralisé, accessible via une **API REST**.

Ce projet avait un double objectif : **éviter la saisie manuelle** des données (longue, fastidieuse et sujette à erreur), et **garantir l'intégrité et la cohérence** de l'ensemble des objets (matériels, utilisateurs, tickets, etc.).

3.2 Objectifs du projet

Les objectifs principaux :

- Automatiser le **transfert de données** depuis l'export GLPI local (CSV) vers le GLPI national via API.
- Assurer la **compatibilité des formats de données** avec l'API REST.
- Éviter les **doublons**, erreurs de saisie ou pertes d'informations.
- Créer un outil **réutilisable** pour d'autres services ou futures migrations.
- Permettre un **suivi clair** des opérations (fichier de logs, gestion des erreurs).

3.3 Fonctionnement du système

Le fonctionnement du système repose sur les étapes suivantes :

1. **Export des données locales** depuis GLPI local au format CSV (matériels, utilisateurs, tickets...).

2. **Traitement des données CSV** dans un script Python (nettoyage, vérification, adaptation aux formats requis).

3. **Connexion à l'API REST de GLPI national** via un **jeton d'authentification**.

4. **Envoi des requêtes POST** vers les différents endpoints de l'API pour créer les objets.

5. **Enregistrement des erreurs** (entrées déjà existantes, champs manquants...) dans un **fichier log**.

6. **Vérification manuelle ponctuelle** pour contrôler la qualité de la migration.

3.4 Architecture technique

Langage utilisé : Python 3.x

Librairies principales : `requests` pour les appels API, `csv` pour le traitement des fichiers, `json` pour la sérialisation des données.

Source des données : fichiers CSV extraits de GLPI local.

Cible : GLPI national via son API REST.

Authentification : jeton utilisateur fourni par la DSI centrale.

Environnement : script exécuté sur une machine Linux (Ubuntu), via terminal SSH.

3.5 Spécifications fonctionnelles

Le script devait permettre :

- L'importation automatique des matériels, utilisateurs, groupes, logiciels, tickets, etc.
- La détection des doublons avant création.
- La gestion des dépendances (ex. : associer un utilisateur à un matériel).
- Le journal d'activité détaillé (succès, erreurs, doublons).
- La reprise automatique en cas d'interruption.

3.6 Spécifications techniques

- Lecture des fichiers CSV ligne par ligne.
- Construction des objets JSON pour l'API.
- Envoi des requêtes HTTP avec gestion des statuts (200, 400, 401, 409...).
- Gestion des délais d'attente (`time.sleep()` pour éviter la surcharge serveur).
- Script modulaire et commenté pour être facilement maintenable.

3.7 Avantages attendus

- **Gain de temps important** par rapport à une saisie manuelle.
- **Réduction des erreurs humaines**.
- **Traçabilité complète** du transfert.
- Possibilité de **réutiliser** le script pour d'autres entités ou services.
- Meilleure **homogénéité des données** transférées.

3.8 Limites et points de vigilance

- Risque de **données mal formatées** en entrée (CSV mal exporté).
- Limitation de l'API (taux de requêtes par minute, champs obligatoires).
- Complexité à gérer les **liens entre objets** (ex. : lier un utilisateur à un matériel déjà migré).
- Nécessité de **valider manuellement une partie des données** après import.

3.9 Planning prévisionnel

Semaine	Tâches prévues
Semaine 1	**Analyse du besoin, export des données locales**
Semaine 2	**Étude de l'API REST de GLPI national**
Semaine 3	**Développement du script Python (v1)**
Semaine 4	**Tests sur un jeu de données restreint**
Semaine 5	**Ajustements, ajout des logs, gestion des erreurs**
Semaine 6	**Import réel sur données globales + vérification manuelle**

3.10 Budget prévisionnel

Aucun budget financier direct n'a été engagé.
Le projet a été réalisé **en interne**, avec les ressources humaines et techniques déjà disponibles.

Coûts indirects :

- Temps de développement estimé à ~2 mois
- Supervision par un technicien de la DSI.

3.11 Critères de réussite

- Taux d'erreur inférieur à 5 % lors de l'import.
- Aucune perte ou altération des données sensibles.
- Création automatisée d'au moins 90 % des entrées.
- Log d'erreurs complet et exploitable.
- Validation du projet par le responsable informatique.

3.12 Glossaire

- **GLPI** : Gestionnaire Libre de Parc Informatique.
- **API REST** : Interface de programmation permettant l'échange automatisé de données via HTTP.
- **CSV** : Format de fichier contenant des données tabulaires (Comma-Separated Values).
- **Jeton (token)** : Clé d'authentification sécurisée pour accéder à une API.
- **POST** : Méthode HTTP utilisée pour envoyer des données à un serveur.

3.13 Étude de faisabilité

- **Technique** : Faisable grâce à la documentation complète de l'API et la simplicité du format CSV.
- **Organisationnelle** : Adapté aux ressources humaines et compétences disponibles.
- **Temporelle** : Réalisable dans le temps imparti pendant le stage.
- **Financière** : Sans coût supplémentaire pour l'organisation.

3.14 Diagramme de cas d'utilisation

Acteurs principaux :

- **Administrateur local GLPI** (technicien de la DSI locale)
- **API GLPI national**

Cas d'utilisation principaux :

1. **Préparer les fichiers d'export**
    - Acteur : Administrateur local GLPI
    - Description : Extrait les données de l'instance locale GLPI sous forme de fichiers CSV (matériels, utilisateurs, tickets…).
2. **Lancer le script Python**
    - Acteur : Administrateur local GLPI
    - Description : Exécute le script en ligne de commande pour démarrer le traitement.

3. **Lire les fichiers CSV**
    - Acteur : Script Python
    - Description : Le script lit les fichiers ligne par ligne, en nettoyant les données si nécessaire.
4. **Générer les requêtes API**
    - Acteur : Script Python
    - Description : Pour chaque entrée du fichier, le script prépare une requête POST avec les données formatées en JSON.
5. **Envoyer les données vers GLPI national**
    - Acteur : Script Python ↔ API GLPI national
    - Description : Le script envoie les requêtes vers les endpoints de l'API REST du GLPI national.
6. **Recevoir les réponses de l'API**
    - Acteur : API GLPI national
    - Description : L'API retourne un code de statut HTTP (succès, erreur, duplication…) que le script interprète.
7. **Enregistrer les résultats dans un fichier de log**
    - Acteur : Script Python
    - Description : Les erreurs ou réussites sont inscrites dans un fichier journal lisible par l'administrateur.

8. **Corriger les erreurs manuellement si nécessaire**

    - Acteur : Administrateur local GLPI

    - Description : Analyse les erreurs enregistrées dans le fichier de log et effectue les corrections ou réexécutions.

Relations :

- Le **script Python** agit comme médiateur entre les fichiers CSV et l'API REST.

- L'**administrateur** supervise l'ensemble du processus, en intervenant au lancement et à la correction éventuelle.

Bilan du projet

4.1 Fonctionnalités réalisées

- **Export complet des données** depuis GLPI local au format CSV (matériels, utilisateurs, tickets).
- **Développement d'un script Python** automatisant l'envoi des données vers l'API REST du GLPI national.
- **Authentification via token** sécurisé à l'API centrale.
- **Création automatisée** de plusieurs types d'objets (ordinateurs, imprimantes, utilisateurs…).
- **Gestion des erreurs** avec journalisation dans un fichier de log clair et structuré.
- **Tests réussis** sur un échantillon représentatif avant déploiement global.
- **Documentation interne** du script pour permettre une réutilisation future.

4.2 Fonctionnalités non implémentées

- Absence d'une **interface graphique** pour faciliter l'usage par des profils non techniques.
- Non-implémentation de la **suppression ou mise à jour automatique** des doublons dans GLPI national (traitée manuellement si nécessaire).
- Le script n'intègre pas encore de **tableau de bord de suivi** en temps réel de la migration.

4.3 Contraintes rencontrées

- **Format hétérogène** de certaines données issues du GLPI local, nécessitant un nettoyage manuel.

- **Limitations de l'API REST**, notamment sur certains champs obligatoires ou restrictions de fréquence.

- **Erreurs de correspondance** entre les ID locaux et les ID nationaux (par exemple pour les entités ou catégories).

- **Temps de traitement rallongé** sur les gros fichiers, nécessitant des optimisations par lot.

- Difficultés initiales à comprendre la **documentation API**, parfois incomplète ou peu détaillée.

4.4 Perspectives d'évolution

Développement d'une **interface utilisateur** (GUI ou web) pour faciliter le lancement du script par des non-développeurs.

Ajout d'un **mode de simulation** (dry-run) pour tester le script sans modifier l'API cible.

Intégration d'un **tableau de bord de suivi** avec indicateurs de progression.

Réutilisation du script pour d'autres préfectures ou entités publiques migrantes vers le GLPI national.

Mise en place d'une **vérification automatique post-migration** (comparaison des volumes ou des identifiants).

Conclusion

Ce projet de migration de données GLPI, réalisé dans le cadre de mon stage à la préfecture de Guyane, m'a permis de mettre en pratique mes compétences en **développement Python**, en **gestion de projet**, ainsi qu'en **exploitation d'une API REST** dans un contexte professionnel concret.

En répondant à un **besoin réel de la DSI**, j'ai conçu un **outil automatisé** fiable et réutilisable, permettant de transférer une grande quantité de données depuis un environnement local vers une infrastructure nationale, tout en respectant les contraintes techniques et organisationnelles.

Cette expérience m'a aussi sensibilisé à l'importance :

- d'un **nettoyage rigoureux des données** avant traitement,
- d'une **documentation API claire** pour une bonne intégration,
- et d'une **gestion des erreurs robuste** pour garantir la qualité des migrations.

Au-delà des aspects techniques, ce projet m'a permis de mieux comprendre le fonctionnement global d'un **Système d'Information public**, ainsi que les enjeux de **standardisation et de centralisation** des outils dans une administration d'envergure.

Enfin, il constitue une **réelle valeur ajoutée** à mon parcours en BTS SIO option SLAM, en m'ouvrant à des problématiques de **déploiement, d'interopérabilité et de sécurité** propres aux grandes organisations publiques.

Annexe

Architecture technique :

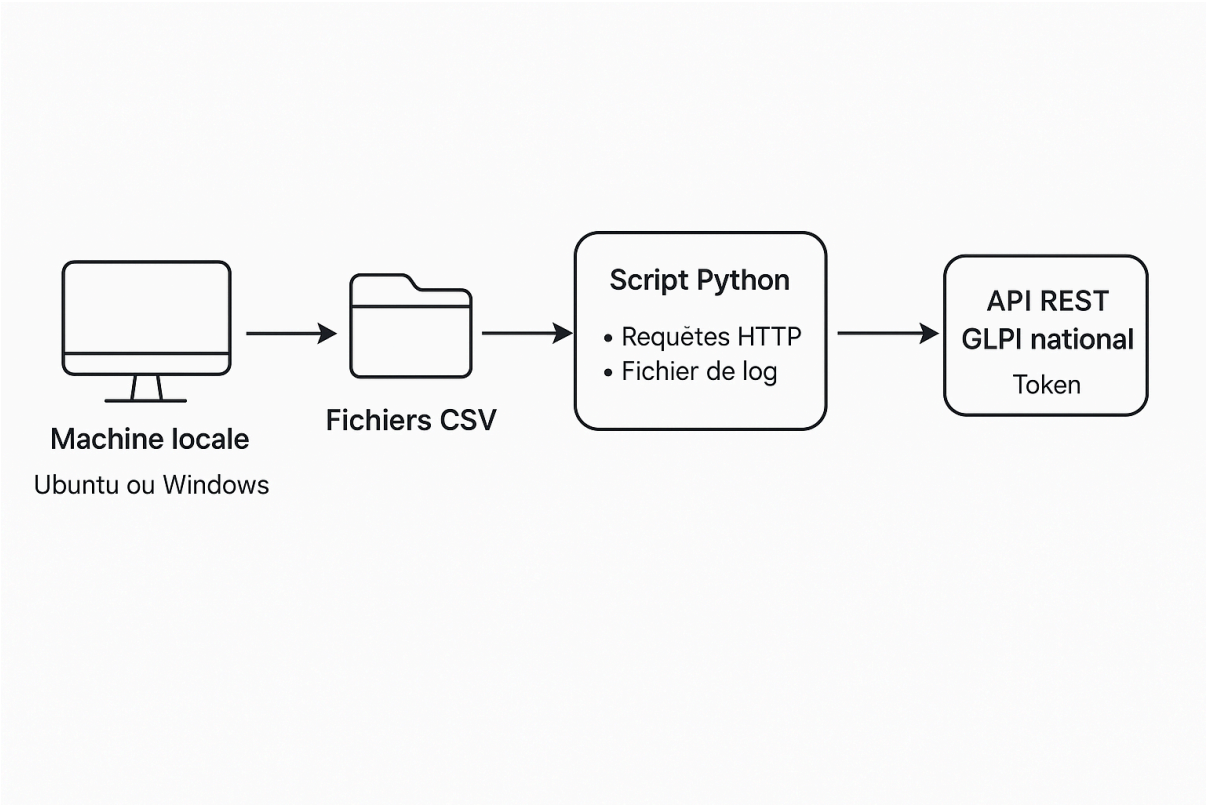
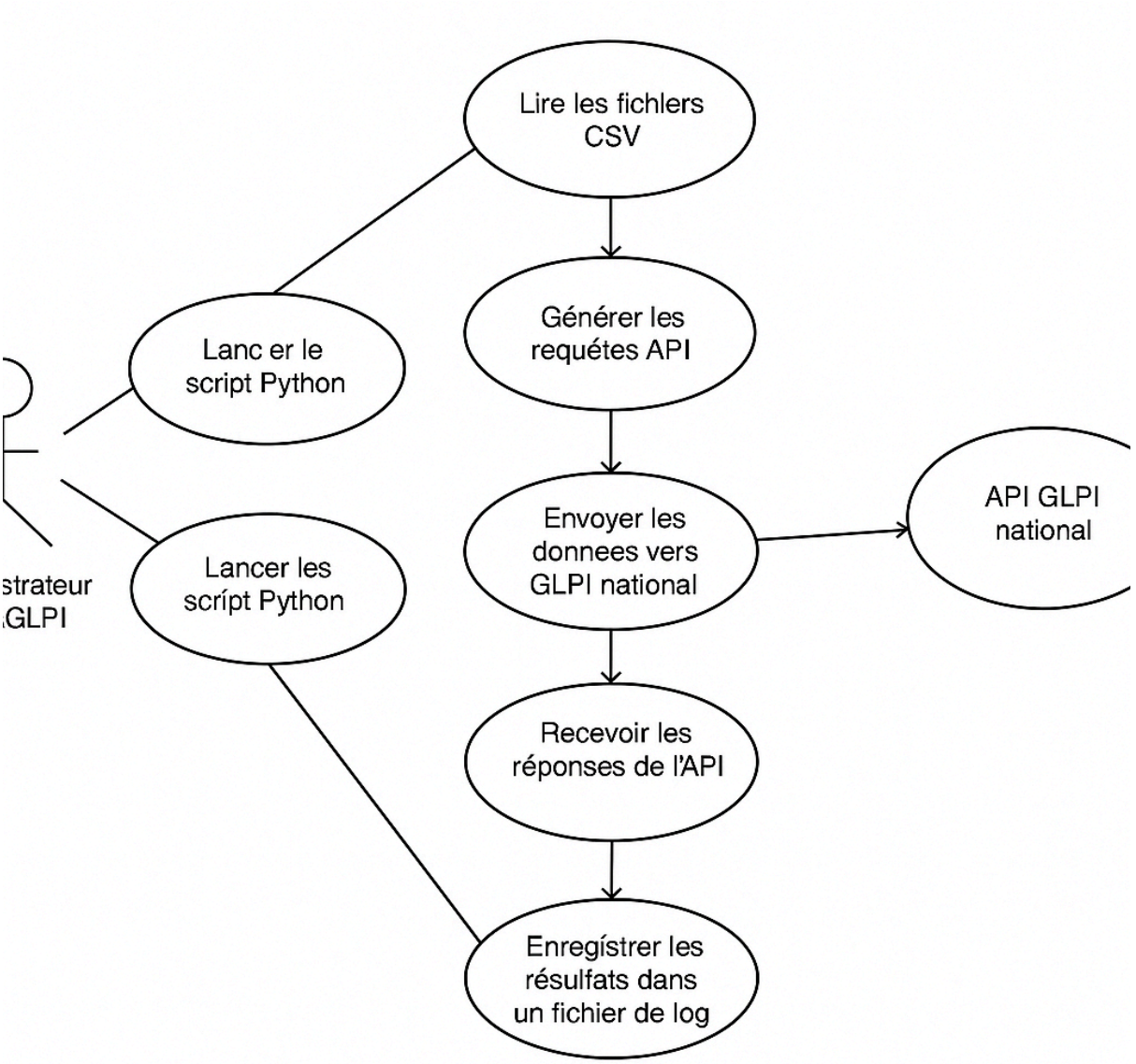


Diagramme de cas d'utilisation :



Script Python :

```python
# === FONCTION D'ENVOI DES DONNÉES ===
def send_data(session_token, data):
    headers = {
        'App-Token': APP_TOKEN,
        'Session-Token': session_token,
        'Content-Type': 'application/json'
    }
    response = requests.post(f"{API_URL}/Computer", headers=headers, data=json.dumps(data))
    if response.status_code == 201:
        logging.info(f"Succès : {data['name']}")
    else:
        logging.error(f"Erreur lors de l'envoi de {data['name']} : {response.text}")

# === LECTURE DU CSV ET TRANSFERT ===
def process_csv(session_token):
    with open(CSV_FILE, newline='', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            data = {
                "name": row["name"],
                "entities_id": row["entities_id"],
                "serial": row["serial"],
                "otherserial": row["otherserial"],
                "comment": row["comment"]
            }
            send_data(session_token, data)
```