Cahier des charges

1- Contexte

La récolte de données d'un parc informatique peut s'avéré studieux, bien que des outils tels que Ansible ou Asset Vie existent. En effet, ces derniers peuvent être complexes à utiliser, prennent du temps à être installés et surtout, ce sont des logiciels payant.

HAL quand à lui, permet de faire très facilement de la remonté de données avec son système de plugin, et d'ajouter, mettre à jour ou bien supprimer ces derniers tout aussi facilement. Le logiciel est totalement libre et se compose d'un client et d'un serveur totalement customisable aux besoins des utilisateurs.

2- Objectifs

HAL est un projet de supervision destiné à récupérer différentes donnés d'un parc informatique et à les envoyer sur un serveur, dans le but d'agréger les données en vue de les traiters.

Il utilise un système de plugins, qui sont chargés automatiquement au démarrage du client. Plusieurs langages pour écrire les plugins sont supportés:

```
- C/C++/Go (.dll / .so)
- Python (.py)
- Ruby (.rb)
- Shell (.sh)
- PowerShell (.ps1)
```

Et d'autres peuvent être ajoutés manuellement si besoin.

Les plugins sont déposés dans le dossier "plugins", qui est un dossier spéciale scanné permettant de charger automatiquement tous les plugins qui s'y trouvent.

HAL est destiné à tout utilisateur voulant superviser les ordinateurs sur un réseau.

2- Besoins

Besoins: Récolter des données de façon générique
Contraintes: donnéées obligatoirement récoltées au format JSON pouvant être stockées de manière générique, sur base de données, en locale...

Besoins:
Contraintes:

Besoin: Créer des plugins
Contraintes: Une lise de langages doit être supportées, que ce soit des langages de script

```
- Python
- Ruby
- Bash
- Powershell
```

ou des langages bas/haut niveau

```
- C
- C++
- C#
- Go
```

Chaque script doit, sur sa sortie standard, envoyer a format JSON tous les éléments devant être remonté au client.

Chaque .dll, .so doit quand à eux retourner via le point d'entrée un string au format JSON.

Besoin: Configurer les plugins
Contraintes: Créer un fichier de configuration, permettant de renseigner diverses informations sur les plugins:

```
* activated (booléen):
    permet de savoir si le plugin doit être executé
* heartbeat (double):
    heure/heartbeat execution
* os (array de string):
    * optionnel, lance le plugin sur une ou plusieurs famille d'os
    * si rien de spécifié, le plugin sera disponible sur tous les os
* si l'OS cible est linux:
    * admin_rights (booléen):
        si le script doit être exécuté en administrateur,
            * si non spécifié, alors false par défaut
    * si admin_rights est vrai et que l'os cible est sous linux:
        * un username doit être renseigné
        * userame (string): l'username passer en argument de "sudo"
```

Besoins: Configurer les intépréteurs pour l'exécution des scripts
Contraintes: Un intépréteur par défaut est défini pour tous les langages de script supportés pour le logiciel, mais peut être changé si besoin.

Besoins: Ajout / suppression / mise à jour des plugins
Contraintes: un dossier scanné spécifique doit être lu permettant de charger les plugins et de voir si le serveur à une nouvelle version de ce dernier, un nouveau plugin, ou une suppression et faire les opérations correspondantes

Besoins: API REST permettant de metre à jour, récupérer ou supprimer des données
Contraintes: doit être accessible depuis un serveur et faire des requêtes sur une base de données

Besoins: Inteface WEB adminstrateur
Contraintes: utilisation simple et permet de très facilement visualiser les différentes informations sur les différents ordinateurs

Besoins: Interface mobile de visualisation
Contraintes:

Besoins: doit avoir une couche d'abstraction permettant l'implémentation d'une BDD
Contraintes: prévoir l'abstraction dans le model

Besoins:
Contraintes:

Besoins materiels

Besoins: doit être compatible Windows/Linux
Contraintes: implémenter une gestion de plugins permettant l'utilisation soit sur Windows, soit sur Linux, soit sur les deux.

Besoins: doit posséder un client et un serveur
Contraintes: la multitude de clients doit se connecter automatiquement au serveur et communiquer de manière asynchrone

Besoins: ne doit pas consommer trop de ressources
Contraintes: éviter d'allouer trop de threads par lancement de plugin

Besoin de performances

Besoins: doit tourner sous forme de daemon
Contraintes: l'application ne doit pas consommer beaucoup de ressources et se doit d'être rapide

Besoins: les plugins doivent être rapidement téléchargés
Contraintes: envoit en asynchrones à tous les clients

3- Contraintes

Délais

Un premier livrable est prévu en décembre, qui sera uniquement le cient/serveur. Un second, début avril, implémentera une interface web permettantde visualiser

les données des cients en temps réel. un dernier, début juillet, vec une application mobile permettant de visualiser les données des clients depuis n'importe quel endroit.

4- Déroulement du projet

Planification

Fin aout - Début Décembre

Création du client/serveur en daemon, les plugins pourront être éxécutés avec un fichier config et un possible déploiement sur le campus sera effectué pour avoir une remontée de données et faire les modifications nécessaires.

Décembre - Début Avril

Création du site web permettant de gérer l'administration et d voir en temps réel les données remontées par les plugins.

Avril - Debut Juillet

Création d'une application mobile affichant aussi les données en temps réel remontées par les plugins.

Juillet

Maintenance et ajouts de fonctionnalitées si besoin

Plan d'assurance qualité

Pour s'assurer de la qualité, chaque livrable à la fin de sa conception aura une phase de test en condition réel, sur le campus de l'université.

De pus, le client est très souvent mit à jour des avancées du projet, ce qui permet d'avoir des retours rapides si des fonctionnalitées doivent changer.