Design Document

Equipe Projet

Axel BERNARD
Moussa BERTHE
Yao KONAN

SOMMAIRE

I. Principe de mise en œuvre de la solution (comment)

La solution a été mise en œuvre en suivant les étapes principales suivantes :

- Développement des composants de base : Création des listeners (*ListenerExecution*, *ListenerStages*, *ListenerRunEnergyMonitoring*) et actions (*DisplayChart*, VariablesConsumptionsPreviousBuild) nécessaires pour surveiller la consommation énergétique pendant les builds Jenkins.
- Intégration de la mesure énergétique : Utilisation de scripts PowerAPI pour mesurer la consommation énergétique et intégration de ces mesures dans le plugin Jenkins via des classes telles que *ScriptGetEnergeticValues* et *InfluxDBData*.
- Stockage des données énergétiques : Utilisation de InfluxDB pour stocker les données historiques de consommation énergétique via la classe *InfluxDBData*.
- Affichage des données énergétiques : Développement d'une interface utilisateur pour afficher les données énergétiques sous forme de graphiques à l'aide de Chart.js, intégré dans la classe *DisplayChart* et la page *index.jelly*.

II. Règles d'architecture

- Séparation des responsabilités : Chaque classe a une responsabilité spécifique. Les listeners sont responsables de la collecte des données, les actions de Jenkins sont responsables de l'affichage et de l'intégration avec l'interface utilisateur, et les classes de service comme *ScriptGetEnergeticValues* et *InfluxDBData* sont responsables de la lecture et du calcul des données énergétiques.
- Extensibilité : Le plugin est conçu pour être extensible, permettant d'ajouter facilement de nouvelles fonctionnalités ou de nouveaux types de mesures énergétiques.
- Modularité : Les différents composants du plugin sont modulaires, facilitant la maintenance et les tests.

III. Modèle statique : organisation des packages, descriptions des classes et de leurs responsabilités

Organisation des packages :

Toutes nos classes sont dans le même package, nous les avons organisé par la façon de les nommer en commençant par le type de classe dont il s’agit de telle sorte qu’on ait des regroupements logiques du fait que leur ordre d’affichage est établi selon l’ordre alphabétique.

Nous tout de même aussi des fichiers dans le dossier ressouces et dans le dossier test du projet de notre plugin

Descriptions des classes et de leurs responsabilités :

ListenerExecution : Surveille la consommation énergétique globale pendant l'exécution d'un build.
ListenerStages : Surveille la consommation énergétique par étape.
ListenerRunEnergyMonitoring : Surveille la consommation énergétique du début à la fin du build, incluant l'initialisation et la finalisation.
ScriptGetEnergeticValues : Classe utilitaire pour lire et calculer les valeurs énergétiques à partir des données du système.
InfluxDBData : Classe utilitaire pour lire et calculer les valeurs énergétiques à partir des données stockées dans InfluxDB.
ValuesExecutionData : Stocke les données énergétiques pour l'exécution globale.
ValuesStageData : Stocke les données énergétiques par étape.
DisplayChart : Gère l'affichage des données énergétiques sous forme de graphiques dans l'interface utilisateur de Jenkins.
VariablesConsumptionsPreviousBuild : Stocke les données énergétiques historiques des builds précédents.
CompletionLatch : Utilitaire pour synchroniser l'exécution et l'achèvement des tâches.

IV. Modèle dynamique : flux des événements, d’états

Flux des événements, d’états :

1. Démarrage du build : *ListenerRunEnergyMonitoring* initialise la mesure énergétique.
2. Exécution du build : *ListenerExecution* et *ListenerStages* surveillent la consommation énergétique et enregistrent les données.
3. Fin du build : *ListenerRunEnergyMonitoring* finalise la mesure énergétique, stocke les données et déclenche l'affichage des graphiques via *DisplayChart*.
4. Affichage des données : *DisplayChart* récupère les données énergétiques et les affiche dans l'interface utilisateur de Jenkins.

V. Explication de la prise en compte des contraintes d'analyse

- Mesure précise de la consommation énergétique : Utilisation de PowerAPI et d'InfluxDB pour assurer des mesures précises et fiables.
- Intégration avec Jenkins : Utilisation des listeners et des actions de Jenkins pour une intégration transparente avec les projets Jenkins existants.

- Affichage utilisateur : Développement d'une interface utilisateur intuitive avec Chart.js pour visualiser les données énergétiques.

VI. Cadre de production : outils de dev, de configuration et de livraison.

Outils de développement :

Java : Langage principal utilisé pour le développement du plugin.
Maven : Utilisé pour la gestion des dépendances et la construction du projet.
Jenkins : Plateforme de CI/CD utilisée pour tester et déployer le plugin.
InfluxDB : Base de données utilisée pour stocker les données énergétiques historiques.
Chart.js : Bibliothèque JavaScript utilisée pour l'affichage des graphiques.
Outils de configuration et de livraison :

Git : Utilisé pour le contrôle de version du code source.
Jenkins Pipeline : Utilisé pour l'intégration continue et la livraison continue du plugin.

Livraison :

Mise en place d'un serveur permettant d'upload le plugin sur Jenkins directement à partir d'une URL :
https://settled-leopard-flowing.ngrok-free.app/energy_checker.hpi
Malheureusement étant donné que ce serveur est hébergé localement, cette méthode de livraison ne peut pas être utilisée durablement, il faut utiliser le fichier .hpi et upload le plugin grâce à ce fichier