Project Management Report

Equipe Projet

Axel BERNARD
Moussa BERTHE
Yao KONAN

SOMMAIRE

I. Rappel des besoins et des éléments principaux de la solution

Les besoins sont de mesurer les données énergétiques dans le développement logiciel afin de sensibiliser et d'aider le développeur à faire attention à son utilisation afin d'adopter des bonnes pratiques.
Éléments principaux de la solution :

- Plugin Jenkins qui mesure la consommation d'énergie d'un build d'une pipeline CI/CD et donne les informations d'énergie et de puissance sous forme de graphiques.
- Linter VS Code qui doit prédire la consommation d'un build en fonction des fichiers de code du projet et qui doit assister le développeur dans l'application de bonnes pratiques.

II. Estimation et description du déroulement du projet

Estimation :

Nous avions estimé lors du sprint 0 que pour avoir le temps de faire le linter VS Code il faudrait avoir fini le plugin Jenkins pour fin mars. Cette estimation ne tenait pas compte du temps qu'il nous faudrait pour le plugin Jenkins -que nous avions du mal à estimer- c'était plutôt pour nous la condition nécessaire pour avoir le temps de réaliser les deux outils.

Déroulement du projet :

Nous nous sommes organisés par sprints selon la méthode Agile et nous avons donc défini des user stories et pour chacune différentes tâches à réaliser, nous avons compilé tout cela dans un backlog afin de suivre le déroulement du projet. Pour le sprint 1 nous nous sommes répartis les tâches en fonction de ce que chacun avait déjà le plus expérimenté lors du sprint 0. Rapidement nous nous sommes heurté à des difficultés pour la liaison de Jenkins et PowerAPI donc nous nous sommes en grande partie consacrés à résoudre ce problème. Voyant le temps passé nous avons opté pour une solution alternative : utiliser le fichier RAPL sans passer par l'intermédiaire de PowerAPI afin d'avoir déjà une base sur laquelle nous pourrions continuer le développement de notre projet. Pour rappel PowerAPI utilise aussi ce fichier (qui contient la consommation d'énergie depuis que la machine est allumé), cependant PowerAPI utilise des cgroup pour pouvoir séparer les processus de la machine. Notre difficulté a été d'utiliser ces cgroup pour Jenkins, le défaut de notre solution alternative et que nous tenons compte de tout ce qui tourne sur l'ordinateur et non seulement du build Jenkins, cependant si le build est la seule chose à tourner cela permet déjà d'avoir une bonne approximation et surtout cela nous permet de poursuivre le projet plutôt que de rester bloqués.

Fort de cette première expérience au sprint 1, nous nous sommes complètement réorganisés à partir du sprint 2. Nous avons décidé de nous séparer en 2 groupes : 1 personne continuera à voir pour lier PowerAPI à Jenkins et les 2 autres développeront le

plugin Jenkins. Cela permet de pouvoir malgré tout avancer sur notre projet tout en se réservant la possibilité de réussir à avoir la mesure que l'on voulait au début (PowerAPI).

Le sprint 3 a été dans la continuité du sprint 2, et nous avions pu avoir une manière d'intégrer PowerAPI par l'utilisation de Docker. Nous avons en parallèle continué le développement des fonctionnalités du plugin, notamment la récupération de données de différentes manières (RunListener, GraphListener, FlowExecutionListener comme expliqué dans le fichier de documentation des sprints) afin d'avoir différentes informations à afficher dans notre onglet selon le choix de l'utilisateur (courbe d'évolution au fil du build, historique des builds, consommation par stages).

III. Description des rôles et des responsabilités

Les rôles à partir du sprint 2 ont été les suivant :
PowerAPI : David.
Développement Jenkins : Axel et Moussa.

IV. Gestion du déroulement du projet

Pour gérer notre projet nous effectuions des réunions après chaque sprint et nous utilisions plusieurs outils :

- notre backlog : nous avons des colonnes pour indiquer à qui une tâche est attribué en début de sprint et une colonne pour compléter les personnes ayant participé à cette tâche, cela nous permet de nous juger en termes de répartition des tâches. Nous avons aussi des colonnes pour la durée passée sur les tâches : une pour indiquer le temps prévu et une pour indiquer le temps passé (où la somme des différentes personnes y ayant passé du temps le cas échéant). Cela nous a permis lors de nos réunions d'ajuster nos prévisions.
- un fichier de rapport journalier : dans ce fichier chacun entre à la fin de chaque séance ce que chacun a fait durant celle-ci. Cela nous permet d'avoir une trace de ce que l'on fait mais ils nous ont aussi servi pour les différents bilans en fin de projet.
- un fichier de documentation des sprints : nous remplissions ce fichier lorsqu'une tâche était finie afin de documenter ce qu'il en découlait, cela nous a aussi été utile pour finaliser le projet.

Ces documents ont été mis en place lors du passage du sprint 1 au sprint 2 (excepté pour le backlog que nous avions déjà dans une version moins aboutie). Après les avoir mis en place nous avons immédiatement constaté des bénéfices sur notre organisation et notre vision des choses à réaliser, cela s'est aussi répercuté dans notre productivité et nous avons pu avancer plus vite à partir de ce moment.

V. Description et gestion des exigences

Une des exigences que nous avons eu fut de nous adapter suite à la perte de 2 membres de notre groupe initial qui furent en Erasmus ce semestre. Cela impliqué un réorganisation qui a été mal gérée au début, ensuite une fois l'adaptation faite nous avons cela a été plus facile à gérer.

Notre principal exigence fut la capture de la consommation d'un build (réussir à l'isoler) et de surcroît la capture par stage. Cela nous a causé des problèmes notamment pour en faire un lien direct avec PowerAPI, nous avons tout de même géré cette exigence en utilisant les temps auxquels démarrent les différents stages grâce à un listener Jenkins.