Silvain Théréné
Lucas Payet Julien Save

API: Spend your time

Introduction:

Le but de ce projet est de réaliser un API (application programming interface) permettant d'interfacer le jeu Spend Your Time en cours de développement par Silvain. Afin d'avoir une architecture fonctionnelle, propre et facilement modifiable, cette API se basera le plus fidèlement possible sur le modèle REST. De plus, l'API sera la plus sécurisée possible avec entre autres l'encryptage des mots de passes, le blocage d'injections SQL…

Le développement de notre API sera réalisé dans le langage Kotlin qui est un langage de programmation orienté objet et fonctionnel très proche du java. Nous utiliserons également Javalin qui est un Framework permettant de prendre en charge les requêtes web sockets, HTTP2 et async, ainsi que Jackson permettant la sérialisation et désérialisation des json, et SLF4J nous offrant la possibilité de logger dans la console des informations.

Structure de l'API et Organisation du travail :

Avant de commencer à développer notre api nous avons commencé par définir les routes que nous voulions une par une et définir ce que chacune d'elle allait renvoyer en termes de données.

Une fois ceci réalisé, nous avons organisé la structure du code de la façon suivante :

- Un dossier data contenant les classes de nos différents objets comme la Map, un Player, un User, une Guilde…
- Un dossier Helper nous permettant de mettre les objets annexes comme la base de données, un vérificateur d'adresse email, la génération de token…
- Dans le dossier racine, un fichier serveur contenant l'ensemble des routes

Nous avons également hiérarchisé nos routes de la façon suivante :

- Les routes possèdent une minuscule au début
- Les chemins possèdent une majuscule au début

Pour se repartir le travail, nous avons chacun pris une partie différente dans l'API. Julien a eu la charge de remplir le fichier serveur avec l'ensemble des routes en positionnant des

vérifications, les bons retours et code retour. Lorsqu'il lui fallait une fonction pour accéder à une liste de guilde, ou enregistrer un user il demandait à Lucas. Lucas lui avait la charge de remplir les différentes fonctions composant les objets Player, User, Guilde… Avec l'aide de Silvain. Néanmoins, Silvain avait la charge de gérer la gestion des tokens pour les connexions des utilisateurs, du hashage des mots de passes, de la gestion de la base de données avec sérialisation et désérialisation.

Difficultés rencontrées et solutions apportées :

L'un des problèmes que nous avons pu avoir a été celui de ne pas définir suffisamment proprement nos routes. De ce fait, au milieu du projet afin d'améliorer la propreté du code et l'harmonisation de nos url de routes. Il nous a ainsi fallut redéfinir certaines ULR de route, les placer aux bons endroits suivant si l'on jugeait qu'elle était pour le user, le player, la guilde ou autre.

Une des plus grosses difficultés que nous avons rencontrées est celle de la documentation de l'API à l'aide de swagger. En effet, de la façon dont nous avions fait notre code, l'intégration de ce Framework nous a obligé à revoir une partie de notre code en ajoute des else if à la place de simple if avec des retour dans ceux-là. Et la complétion de l'ensemble de la documentation dans le code a été longue et fastidieuse.

Malgré ces quelques problèmes rencontrés l'ensemble des objectifs ont été remplis et le projet s'est bien déroulé. Nous avons apprécié travailler sur ce projet et apprendre comment réaliser une API Rest la plus fonctionnelle possible. Pour deux personnes du groupe, ce projet nous a également appris le langage Kotlin.

Quant au cours, il a été très enrichissant et nous n'avons pas spécialement de retour négatif à faire. Le format d'apprentissage entre suivre les tps ou avancer dans son projet permet une liberté de travail.

Le choix du projet libre mais encadré dans la création des routes composants celui-ci a été également une façon de faire très motivante qui, pour nous, il faudrait garder pour les années suivantes.

En conclusion le cours comme le projet a été très intéressant. Il reste bien sur des choses à améliorer dans notre code et nous pourrions également aller plus loin en configurant notre image docker avec kubernetes.