Alea Data Est

En parcourant les différentes API à notre disposition, nous avons pu remarquer qu'elles rencontraient à quelques exceptions près, les mêmes problèmes, qui peuvent être plus ou moins dérangeant selon le trafic et les requêtes. En effet, nous avons pu voir que pour la plupart des API, la configuration avait un « Acces-Control-Allow-Origin » ayant pour valeur « * ». Cela signifie simplement que n'importe qui peut accéder à la ressource, ça peut être un problème lorsque les routes peuvent être accéder à d'importants trafics. Nous avons ensuite pu constater que de nombreuses API avaient comme problème le « Content-Type, le problème qui peut-être rencontré ici est que le navigateur peut tenter de lire le contenu en l'inspectant et donc en interprétant celui-ci afin de détecter quel type de contenu est injecté dans cette route, or, il est important que le navigateur comprenne cela à l'aide des headers.

Pour notre part, le début de la séance n'a pas été très fructueux dans la mesure où le serveur ne tournait pas encore suite à un problème de compilation du programme. Une fois ce problème résolu, tout a pu se dérouler correctement.

Voici ce que l'on a pu constater suite aux divers crash test :

Crash Test N°1 : Ce crash test n'a pour nous, pas pu être pertinent dans la mesure où lorsque celui-ci a pris part, notre serveur été déconnecté (PuTTy n'ayant pas vu la moindre activité sur le terminal pendant x minutes, s'est déconnecté). C'est donc suite à cela que nous avons pu demander aux intervenants comment remédier à cela. Nous avons donc découvert la commande « screen » qui permet d'ouvrir un nouveau terminal qui effectuera les tâches en fond.

Crash Test suivants : L'API répond plutôt bien, elle est rapide et répond à la plupart des requêtes. Cependant, cela n'évoque pas que du positif. En effet, en nous basant sur ce que le graphique projeté au tableau pouvait montrer, nous avons pu constater que l'API réagissait de manière normale à un grand nombre de requête provenant d'une même adresse IP. Le comportement le plus adapté pour cela aurait été de bannir l'adresse IP pour une période temporaire afin d'éviter le spam ou le DDOS.

Concernant les améliorations de notre côté, il est primordial de limiter le nombre de requête possible par IP durant un laps de temps, que ça soit à la seconde ou à la minute. Comme nous avons pu le constater durant les crash test, l'API réagissait normalement, elle a pu tenir la charge mais ne se protégeait pas du nombre de requête importante qui a pu arriver en quelques secondes. Il aurait été préférable que le serveur se mettent en stand-by afin d'éviter la surcharge à ce niveau.

En ce qui concerne la gestion du cache, notre API étant basée sur la génération de données aléatoires, il est un peu compliqué de gérer ce cas, tout du moins pour la route principale (/generate?ressource_id=5&nombre=2) par exemple. Nous pouvons par ailleurs, gérer le cache pour d'autres routes comme la récupération des ressources (donc, sans les données aléatoires) ainsi que de ses champs, règles et paramètres. Comme de nombreux groupes, nous avons également eu le souci concernant « Acces-Control-Allow-Origin » qui permet à tout le monde d'accéder aux ressources et donc, de profiter à sa guise des générations aléatoires, et donc potentiellement de surcharger le serveur de requêtes.

D’autres modifications pourraient permettre d’optimiser notre API, celles si se passent du côté applicatif. En effet, en revoyant un peu la partie algorithmique des principales fonctions et requêtes, nous pourrions gagner en efficacité.

C’est d’autant plus le cas dans la mesure où nous avons pris la décision de développer cette API en GoLang, langage qui nous était alors totalement inconnu au début du projet. Cependant, nous concernant, c’eût était une expérience positive et pleine d’apprentissage car il est toujours intéressant de se pencher vers des technologies grandissantes et qui nous sont peu familières. GoLang étant un langage très fortement typé et ne laissant pas le droit à l’erreur, cela nous a permis d’être plus rigoureux dans la manière d’aborder le sujet et de développer les différentes fonctions. Cela n’a malheureusement pas toujours été simple dans la mesure où la communauté GoLang n’est pas aussi présente et active que pourrait l’être celle de PHP ou encore JavaScript, il n’était donc pas toujours évident de trouver une solution à nos problèmes, il a parfois fallu se débrouiller afin de comprendre nous même les erreurs (qui n'étaient pas toujours explicites) et de les corriger.

Pour finir, nous tenons à remercier nos intervenants pour l’aide apporter sur le projet, et en particulier à Anthony Baillard qui nous a permis de déployer notre API sur son serveur

Jordan

Florian