YNetflix

Ynov M2 - Micro-services

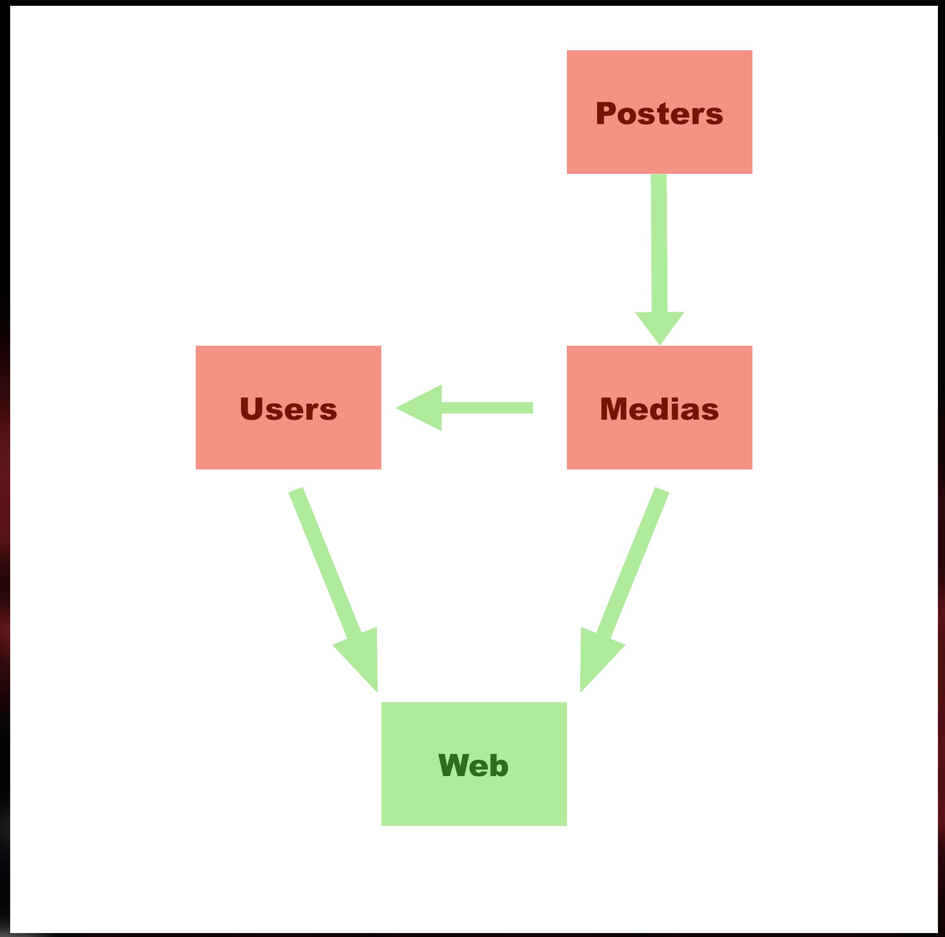
Anthony Khelil

Objectifs du projet

Développer une application de streaming vidéo semblable à Netflix, et intégrant l’architecture micro-service. Chaque élément du projet (Utilisateur, Média, Poster) doit être distinct des autres, notamment en utilisant un serveur à part, une base de données distincte, et communique avec les autres micro-service via des requêtes HTTP.

Architecture applicative

- Le coeur du projet est composé de 3 services.
- Le premier d'entre eux, Média, interagit avec les deux autres. Via Poster, il va récupérer le poster des différents médias, et via Users, il va vérifier que l'utilisateur courant à l'autorisation d'accéder à ces médias.
- Poster n'est jamais appelé directement. Il permet de modifier et récupérer les posters en fonction d'un média donné, et est donc appelé par le service Médias
- Users permet la gestion des différents utilisateurs, et ajoute un système de vérification.
- Enfin, la partie Web est l'interface du site web. Elle va faire appel aux autres services, afin de récupérer les différentes données à afficher aux visiteurs du site.



Stack applicative - Générale

Le projet est composé de deux parties distinctes : Le front et le back. Le front, qui implémente les différentes opérations sur les 3 principaux micro-services, fonctionne sous PHP. Le back, qui va donc présenter visuellement les différents médias, est code en HTML, et utilise le langage JavaScript pour les opérations dynamiques (Modification d'un compte utilisateur, …)

Nous allons les voir plus en détails dans les slides suivantes

Stack applicative - Médias

Un média désigne un programme que l’utilisateur va suivre, qu’il soit un film ou une série. Chacun est accessible via un CRUD, et correspond à une ligne en base de données.

Le champ type désigne le type de média. Ensuite, une jointure sera faite, sur la table Movies ou Episodes, pour retrouver soit l’aperçu du film, soit la liste des épisodes de cette série. En faisant cela, les films et les séries sont donc parfaitement séparés, comme exigé dans le cahier des charges, et l’ont pourrait même s’imaginer créer des sous-micro-services indépendant pour les stocker.

Stack applicative - Posters

Les médias décrits précédemment n’impliquent aucun poster, qui sont administrés dans un micro-service séparé. Celui-ci met à disposition différents endpoints pour manipuler ses posters en fonction de l’heure de la journée, et permettre de récupérer le poster actuel, associé à un média. (Film ou série)

Actuellement, deux posters sont possibles, en fonction de l’heure de la journée, mais on peut facilement imaginer avoir davantage de poster, et les afficher selon d’autres critères, comme les préférences de l’utilisateur, sa région, …

Stack applicative - Users

Chaque utilisateur qui se connecte au site doit impérativement être inscrit, et autorisé à se connecter. Pour cela, le micro-service Users dispose de plusieurs fonctions, dont une qui va vérifier cette autorisation. Chaque utilisateur à en effet un champ “status”, qui va contenir soit “admin” pour un administrateur, soit “user” pour un utilisateur classique. Tout autre valeur indique que l’utilisateur n’est pas autorisé à se connecter, et qu’il a été bloqué par un administrateur.

Axes d’amélioration

Ce projet étant terminé, nous pouvons alors faire un point sur ce qui a pu être développé, ce qui n’a pas pu être ajouté, et les potentielles améliorations que nous pourrions lui apporter. Nous pouvons lister notamment :

- Une dockerisation des différents micro-services.

- L’ajout d’un système d’authentification complet. À l’heure actuelle, et comme demandé dans notre cahier des charges, chaque utilisateur peut se connecter librement. Il pourrait être intéressant de rajouter un mot de passe à l’inscription, sa réinitialisation, …

- Une amélioration du système d’horaire des posters. Chaque média dispose de deux posters distincts, affiché soit le jour soit la nuit. Nous pourrions imaginer rajouter d’autres plages horaires, voir de laisser chaque administrateur de choisir de quelle heure à quelle heure chaque poster doit être affiché.

- Une interface front-end pour les administrateurs. La plate-forme web permet en effet à un utilisateur classique de voir les films, mais les administrateurs ne disposent pas d’une interface dédiée pour leurs opérations, et doivent appeler eux-même les endpoints des actions qu’ils veulent faire (Modifier un film, ajouter un poster, …)

Ce que j’ai appris sur ce projet

En développant ce projet, j’ai pu apprendre à mettre en place un environnement basé sur les micro-services, et constater par moi-même les avantages et inconvénients que cela comporte. Parmi les points qui m’ont le plus marqué, il y a :

- L’apprentissage d’un langage (Java) et d’un framework (Spring) dont j’avais beaucoup entendu parler auparavant, mais dont je n’avais pas eu le temps d’apprendre

- L’utilité d’avoir des services indépendants les uns des autres. Notamment lorsque, pendant mon développement, j’ai involontairement rendu inaccessible un de mes micro-service, et que j’ai vu que le reste du projet fonctionnait parfaitement sans lui, ce qui n’aurait pas été le cas si tous les services du projet étaient liés les uns aux autres.

- L’utilisation de docker, et pourquoi il permet de simplifier et de gagner du temps sur des projets conséquents. Auparavant, j’avais uniquement utilisé docker dans le cas de l’installation de projets déjà existants sur mon poste, notamment chez Ippon, sans en comprendre l’intérêt. Le fait d’essayer de le mettre en place dans ce projet, même s’il n’est pas terminé, m’a fait prendre conscience de l’interet de la conteneurisation d’un projet, afin de gagner du temps lorsqu’il sera réutilisé à l’avenir

- Sur la partie back, l’appel d’une API pour récupérer des données, qui sont ensuite envoyées au back. Je ne pensait pas que cela serait possible, aussi rapide, et complètement transparents pour l’utilisateur final.

Merci !

Par Anthony Khelil