Depuis quelques années, les employés du service des technologies de l’information de l’UQAM font de la veille technologique en présentant différents produits et preuve de concept. Un des objectifs de ces rencontres est de tenter de répondre à la problématique de la modernisation du savoir-faire des services TI.

J'ai eu l'opportunité d'être témoin de cette évolution et d'avoir pu partager la vision de mm Simoneau et Martin. J'ai joint l'équipe de madame Coppola pour faire le partage des connaissances et pour offrir de l’aider à l'établissement d'une architecture de développement.

Je vais vous présenter une architecture inspirée des consignes et conseils de monsieur Simoneau. Cette solution est en phase avec les principes architecturaux de la direction. Lors de son élaboration, elle devait surtout :

- cadrer avec le domaine d'affaire de l'UQAM;
- utiliser les meilleures composantes;
- avoir une cohésion forte et un couplage faible;
- avoir un niveau d'abstraction adéquat.

Je vais présenter l'architecture dans son ensemble et détailler les composantes importantes. La solution n'est pas que théorique. Elle répond déjà aux besoins de deux projets « Stage en éducation » et « Mise à jour des coordonnées personnelles de la Fondation UQAM ».

Diagramme des composantes du domaine d'affaires

Ce diagramme présente les composantes en tant qu'unité d'affaires. C’est le plus haut niveau d'abstraction qui permet d'organiser les microservices de façon cohésive.

Chaque composante expose ou requiert des interfaces publiques. En les liant par leurs interfaces l'on réduit le couplage et facilite la mise à l'échelle de l'architecture.

Il y a trois catégories de composante :

- celles du domaine fonctionnel (Stage, dossier étudiant, horaire);
- celles des services transversaux communes aux domaines d'affaires;
- celles qui sont extérieures à l'UQAM et qui interagissent avec nos interfaces publiées.

Les liaisons entre les composantes sont synchrones ou asynchrones. Dans le premier cas, l'on privilégie l'architecture REST et dans le second le protocole AMQP est utilisé.

Diagramme d'une unité d'affaire

L'approche par microservices est appliquée concrètement dans la réalisation des interfaces de composantes. On priorise la chorégraphie entre les services plutôt que l'orchestration. Le système sera moins couplé et plus flexible.

Nous sommes à un niveau d'abstraction plus fonctionnel. On peut voir l'utilisation du patron « Backends for Frontends ». Avec l'explosion des différents appareils électroniques, il faut une solution qui peut s'adapter à la multitude d'interfaces graphiques. Ce patron permet de gérer les requêtes des appareils et d'éviter d'avoir un api monolithique qui serait ardu à maintenir.
Une question s'impose. Comment l'on interagit avec les microservices? En appliquant le style d'architecture REST. Ce sont les liens hypermédias qui vont nous permettre les interactions.

Voici une proposition :

http://<domaine-affaire>.uqam.ca/api/<version>/<interface>/<ressource>/<params?>

Supposons que je veux accéder à mes coordonnées de mon profil étudiant :

http://dossier-etudiant.quam.ca/api/v1/profils/coordonnees/com_g

Dans l'URL on garde la logique du couplage vers une interface d'unité d'affaires. On n'y expose pas de logique d'affaires. Les actions sont sous-entendues par les VERBS HTTP.

Trois niveaux de validation

Nous sommes dans l'exécution du code et couplée à un choix technologique. Le cadriciel « Express » dans ce cas.

Le traitement d'une requête se fait en trois étapes. Premièrement l'application (express) intercepte une requête, elle la soumet à une fonction qui ne fait que valider les entêtes et la requête HTTP. Si pour une raison, elle n'est pas valide, elle est automatiquement fermée. Deuxièmement, la requête est dirigée vers le contrôleur du service. Il a la responsabilité de valider le corps de la requête. Il sait quelle ressource est appelée et il doit s'assurer que le corps contient les données nécessaires pour le traitement de la requête. Troisièmement, les différentes librairies sont appelées pour effectuer le traitement.

Cette approche permet d'augmenter la cohésion. Chaque étape a sa responsabilité dans le traitement de la requête. Il est aussi plus facile d'étendre les fonctionnalités.

Déploiement

Le domaine d'affaires est connu, mais ses limites (exécution, développement, employés) sont inconnues. Il va y avoir beaucoup de changements. Il faut être flexible et laisser l'architecture gagner en maturité.

Dans un premier temps, l'approche classique serveur de développement et production sera utilisée. Nginx servira de «revers-proxy» pour gérer les appels des différents services.

Cependant rapidement, il faudra mettre en place une approche de déploiement continu en utilisant des outils comme Docker et Jenkins-CI.