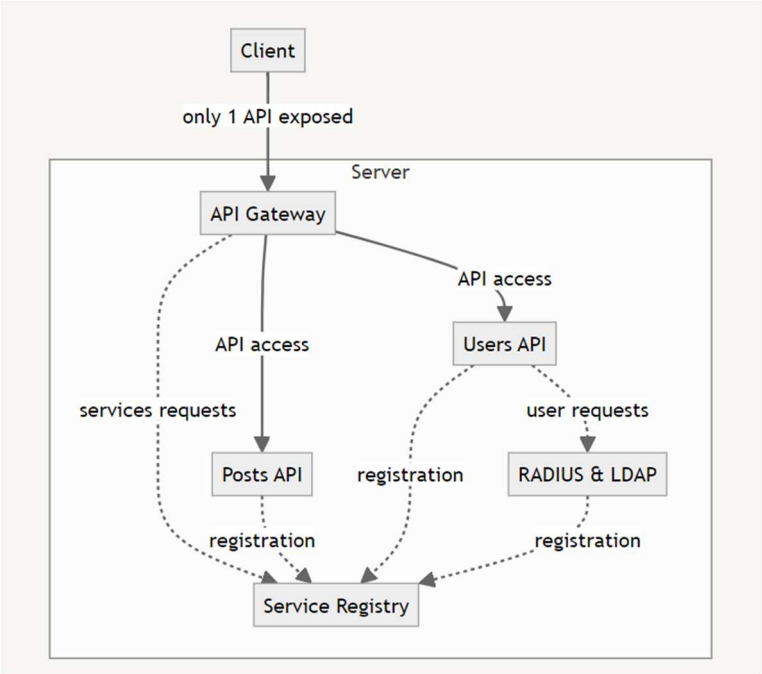
Projet Spring

Architecture

L'architecture se compose de deux services (API) permettant de gérer des utilisateurs (Users) et des articles (Posts), d'une API Gateway ainsi que d'un Service Registry. Les services Users et Posts possèdent respectivement un annuaire LDAP et une base de données MySQL.



Pour faciliter le debug et la correction, tous les services ont un port exposé sur l'hôte local :

Service	**Port**
API Gateway	8080 (HTTP)
Service Registry	8761 (HTTP)
Posts Service	8081 (HTTP)
Posts Database (MySQL)	3306 (SQL)
PhpMyAdmin	8085 (HTTP)
Users Service	8082 (HTTP)
Users Directory (LDAP)	389 ou 636 (LDAP)
PhpLDAPAdmin	8086 (HTTP)

Docker compose

Le Docker compose permet de démarrer tous les services nécessaires au fonctionnement de l'application. Pour lancer aussi les conteneurs PhpLDAPAdmin et PhpMyAdmin et se connecter manuellement respectivement sur l'annuaire LDAP (Users) et la base de données SQL (Posts), le profil « debug » doit être lancé (voir https://docs.docker.com/compose/profiles/#start-specific-profiles).

API Gateway

L'API Gateway était fonctionnelle jusqu'à la tentative d'ajout d'un controlleur REST sur celle-ci, dans l'idée d'implémenter le pattern API Composition. Son fonctionnement n'a pas pu être rétabli, même en supprimant complètement le projet et en le recréant en suivant scrupuleusement le tutoriel du cours.

Dans sa meilleure version, elle permettait l'accès aux services users et posts depuis un seul point d'accès, avec un routage efficace.

Le pattern API Composition est implémenté mais n'a pas pu être testé en raison des problèmes survenus lors de son développement sur la Gateway. Le principe de composition imaginé ici était de faire deux appels HTTP (connexions synchrones) successifs vers les services Posts et Users de manière à concaténer les informations des utilisateurs dans celles des articles. Les appels sont réalisés au moyen d'un REST Client HTTP.

L'API Gateway s'inscrit automatiquement sur le Service Registry et l'utilise pour appeler les différents services.

Connexion asynchrone

La connexion asynchrone entre des services n'a pas pu être réalisée par manque de temps. Elle aurait nécessité une adaptation de l'architecture conséquente qui n'a pas pu être effectuée.

Posts service

Ce service permet de réaliser les opérations standards (CRUD) sur des articles. Des plus élémentaires, il se connecte à une base de données MySQL lancée dans un conteneur différent. Chaque Post possède un publisherId, qui représente le nom d'utilisateur du créateur.

Un service PhpMyAdmin permet de se connecter sur la base de données.

Le service s'annonce automatiquement au Service Registry.

Users service

Ce service gère les utilisateurs, grâce à un annuaire LDAP. Il permet de récupérer les informations d'un utilisateur ou de tous les utilisateurs, de modifier ou supprimer un utilisateur. Il est aussi possible d'y soumettre une demande d'authentification (nom d'utilisateur et mot-de-passe) qui va être vérifiée et validée ou déclinée.

L'authentification était initialement prévue par un serveur RADIUS plutôt qu'un accès direct à l'annuaire LDAP, mais cette architecture n'apportait rien dans le cadre du projet et a donc été jugée superflue.

La création d'un utilisateur et la mise à jour du nom d'utilisateur n'est pas fonctionnelle, car la propriété « cn » qui représente le nom donné à un utilisateur n'a pas pu être ajoutée de manière à permettre ces opérations.

L'annuaire LDAP tourne dans un conteneur séparé et est configuré pour accueillir des utilisateurs. Le conteneur PhpLDAPAdmin permet d'accéder à l'annuaire et d'y ajouter des utilisateurs. Les mots de passe ajoutés par PhpLDAPAdmin peuvent ne pas correspondre à celui demandé dans le User Service à cause des algorithmes de hachage différents.

Username	Password
`cn=admin,dc=he-arc,dc=ch`	passw0rd

Il est donc conseillé de créer l'utilisateur depuis PhpLDAPAdmin, puis d'y ajouter son mot-de-passe par la route PUT (sans modifier le nom d'utilisateur). Lorsque l'ajout est fait, les autres fonctionnalités peuvent être utilisées sans problème.

Le service s'annonce automatiquement au Service Registry.

Conclusion

Une grande partie du temps de travail a été allouée à la résolution des problèmes liés à l'API Gateway, composant important de l'architecture. Le temps perdu sur cette partie du projet n'a pas pu être rattrapé.

La mise en place d'un service d'authentification avec LDAP a pu être réalisée. L'authentification n'a pas été utilisée dans le projet, mais elle est accessible au travers du Users Service.

Le projet n'atteint pas tous les objectifs fixés et ne donne pas entière satisfaction, mais il a permis de mettre en pratique des concepts vu en cours ainsi que d'autres paradigmes intéressants.

Le projet a été rendu dans les temps, bien que pas complétement fonctionnel. Le rapport, quant à lui, a été rendu avec env. 1h de retard.