Rapport de conception - Projet M2 semestre 2

Interopérabilité et innovation dans les SI

Théo MAESEN - Thomas BONVIN - Thomas MILLET - Pierre-Yves REFOUBELET

Rappel du contexte

L’objectif de ce projet est le suivant :

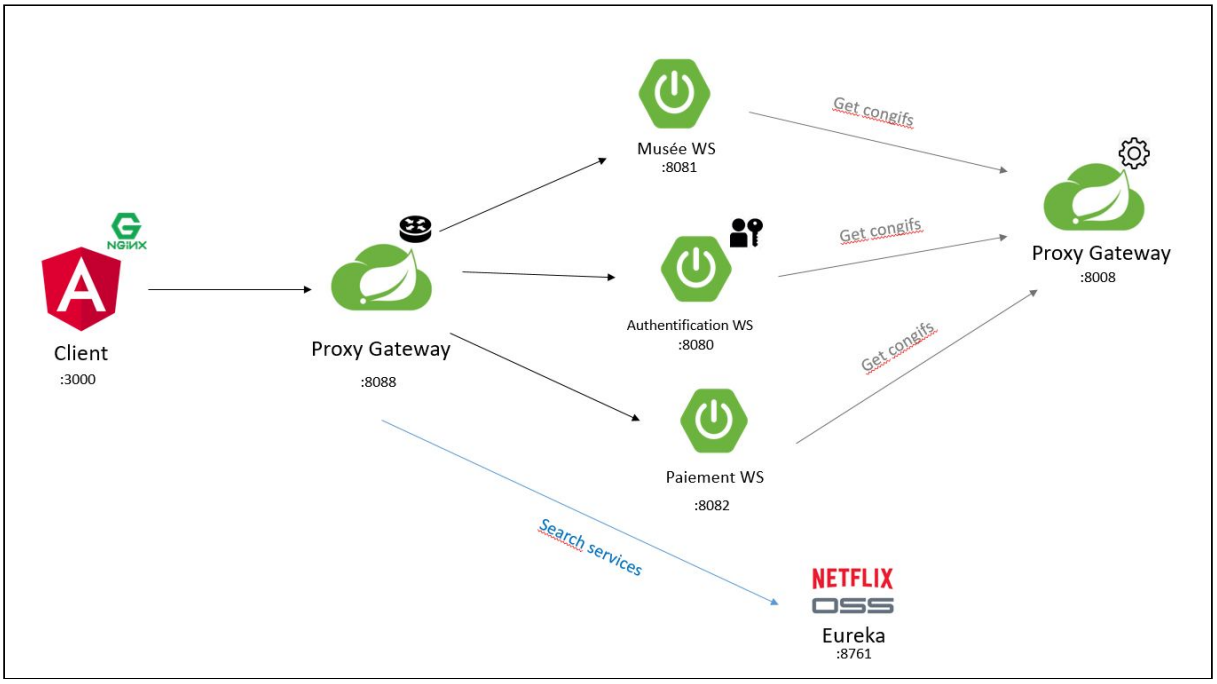
Vous devez modéliser le SI du nouveau muséum d'histoire naturelle d'Orléans, le MOBE. Ce musée possède des collections d'objets, qu'il enrichit chaque année en achetant de nouvelles oeuvres. Ces oeuvres peuvent être exposées dans l'une des 20 salles du musée pendant une certaine période, sinon elles sont remisées dans la réserve jusqu'à la prochaine exposition. Un exemple : le musée a acheté le 15 septembre 2015 un tyrannosaure, nommé Rex, stocké à la livraison dans la réserve ; le 1er janvier 2016, il a été exposé dans le grand hall du musée ; le 1er septembre 2016, Rex a été déplacé dans la grande salle 1, pour l'exposition "les dinosaures". le 15 mars 2017, il est revenu dans le grand hall. Du 20 décembre 2018 au 1 avril 2019, il est retourné en réserve pour être restauré. Il est à nouveau exposé dans la grande salle 1 depuis. Des expositions thématiques sont organisées régulièrement (eg "les dinosaures") mais pas en permanence. Dans ce cas, certaines oeuvres peuvent être prêtées par d'autres musée pour être exposées. Le musée met également en place la vente de billets électroniques en ligne pour les expositions. Après le paiement, les utilisateurs récupèrent un QR code sur leur mobile ; en arrivant dans le hall d'entrée du musée, des tourniquets automatiques avec des scanners de QR code leur permettent de rentrer directement sans faire la queue à la caisse. Chaque QR code ne peut évidemment être utilisé qu'une seule fois.

Ce projet a pour but de nous faire utiliser toutes les technologies rencontrées lors de nos 3 années de MIAGE afin de les faire collaborer et ainsi créer un système complet de gestion d’un musée.

Le délais est de 3 semaines avec un rendu prévu le 11 avril 2020

Choix conceptuels

Voici les différents diagrammes permettant d'avoir un aperçu global du système.



Choix technologiques

Comme donné dans le sujet, certaines parties nous sont imposés vis-à-vis du choix technologique notamment pour la partie JFX ou du Java sera nécessaire.

Côté interface utilisateur, nous avons fait le choix de prendre du Angular, framework Javascript, et maîtrisé par l'un de nous.

L'intégralité des webservices nécessaire au bon fonctionnement du système seront développé à l'aide de Spring permettant d'avoir des systèmes autonomes divisés en micro-services.

JFX

Cette partie concerne la gestion administrateur de l'application. Elle permet une gestion des oeuvres, des expositions et des salles au sein du musée.

Il est possible dans cette partie, d'administrer les oeuvres du musée, c'est-à-dire les créer, les modifier et les supprimer, ainsi que les déplacer dans les différentes salles du musée.
Les oeuvres ont un nom, une date d'achat et un prix d'achat, et sont aussi placées dans une salle du musée.
Il est également possible de gérer les expositions, soit les ajouter et les supprimer.
Les expositions ont un nom ainsi qu'une date de début et une date de fin.

L'utilisateur se connecte avec des accès administrateurs, au moyen d'un login et mot de passe, pour accéder à l'interface d'administration.

Angular

N'ayant jamais vraiment développé de frontend en particulier en javascript nous avons choisi Angular qui est probablement le framework frontend le plus accessible et facile à prendre en main. De plus la documentation est riche et les tutoriels en ligne sont nombreux.
L'architecture est basique : les vues sont reliées à des fichiers typescript qui utilisent des services pour requêter le backend en REST.
Le déploiement est effectué avec NGINX.

Spring

La partie Spring compte quant à elle 3 microservices pour les fonctionnalités métier et 3 microservices répondant aux besoin de l'architecture.

Concernant la BDD, cette dernière est commune car plusieurs WS peuvent accéder à un même information. La mise en place de cette dernière a été compliquée. Nous avons fini par opter pour une db en SQLite avec un dialect custom trouvé sur le web mais faisant très bien le travail et répondant à toutes nos attentes.

AuthentificationWS :
Cette partie est dédiés à la gestion complète des utilisateurs. Aussi bien la connexion/déconnexion que l'ajout/suppression des attributs le concernant.
Elle gère par ailleurs la sécurité des autres micro-services en fournissant un token une fois la personne connectée qui permet d'accéder aux autres parties de l'application notamment la réservation par le paiementws, ou le museews pour la partie administration.

La partie de sécurisation est réalisée par jsonwebtoken. La génération de ce dernier est automatique et unique lors de la connexion.
Les autres webservices métiers (musée et paiement) possèdent un SécurityHandler qui requête sur le webservice authentification pour vérifier l'authenticité du token.
Si un utilisateur perd ce token, il sera alors nécessaire de se déconnecter/reconnecter pour en avoir un nouveau. Un point d'amélioration a été trouvé sur ce sujet, en effet, ce mécanisme oblige à n'avoir qu'une seule session. Si l'on cherche a terme la possibilité d'avoir plusieurs session (web + smartphone par ex.), cela ne sera pas possible.

MuseeWS :

Cette partie gère l'intégralité des composants présent dans un musée : les salles, les oeuvres et les expositions. Il permet d'avoir une gestion complète des objets présent mais aussi de gérer leurs déplacements en fonction des besoins. D'un manière globale toute ses fonctionnalités sont accessibles à travers la partie JFX, partie qui concerne uniquement d'administration.

PaiementWS

Par gain de temps, nous avons implémenté une partie paiement nous même. Nous avons dû créer les entités correspondantes ainsi que les méthodes nécessaires à la réservation des billets mais aussi à la consultation de ces derniers. Encore une fois, ce WS accède directement à la BDD afin de récupérer les différentes informations nécessaire pour traiter les demandes mais aussi pour mettre à jour la situation des billets si ils sont utilisés à l'entrée du musée.
Les billets sont identifiés par un qr-code unique généré lors de la réservation. Ce qrcode n'est pas stocké en image mais uniquement en code à 16 caractères (aaaa-bbbb-cccc-dddd). C'est dans le client web que l'utilisateur peut le générer en image et le télécharger afin de pouvoir le présenter une fois arrivé au musée.

Eureka et Gateway

Nous avons implémenté un discovery serveur qui répertorie nos différents webservices métier à la façon d'un annuaire.
Pour simplifier les appels REST toutes les requêtes passent par le gateway (port 8888). C'est un Spring Cloud Gateway qui a tout simplement le rôle de routeur : il redirige chaque requête vers le webservice correspondant enregistré dans Eureka.

ServerConfig

Pour clarifier tous les aspects de configuration dans le back-end nous avons implémenter un serveur Spring Cloud Config qui met à disposition toutes les configurations des webservices. Ces configurations sont dans un fichier ("appconfigs") à la racine du projet.

Base de données

Comme énoncé plus haut dans ce rapport, l'implémentation de la BDD a été une des parties compliqués. Nous nous sommes beaucoup inspirés de ce que nous avions vu en cours mais sans arriver à le reproduire à 100%. C'est pour cela que nous avons opté pour pour une version SQLite. Le principe de base reste cependant le même, une configuration supplémentaire est nécessaire pour l'utilisation du nouveau Dialect. Nous utilisons donc les dépendances JPA et Hibernate, et l'architecture globale est la même : Entity/DTO/DAO.

Cela nous permet d’avoir des requêtes rapides pour toutes les entitées créées sur le projet et de ne pas avoir à écrire toute les requêtes à la main. Ainsi toutes les entitées sont bien créées en base ainsi que leurs dépendances ce qui nous permet de faire les jointures quand cela est nécessaire pour récupérer certaines informations. Il paraît raisonnable de dire que les méthodes employées ne sont pas les plus optimisées mais cela reste fonctionnel et effectue les tâches nécessaire au bon fonctionnement de toute les services.