Bedjaoui Khalil – Moguet Martin

Rapport Projet

Technologies du web

Université Polytechnique HAUTS-DE-FRANCE

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE

Table des matières

I. Remise en contexte :

Rappel du sujet :

Le but du TP est de réaliser un mini site web de quelques pages qui réutilise les technologies étudiées en CM/TD :

- Application autonome avec Spring Boot
- Services REST avec Jersey (consommés depuis le javascript)
- Un framework JavaScript (jQuery, Angular, React, Vue.js …)
- Persistance de donnée avec spring-data et hsqldb

Le TP peut être réalisé seul ou en binôme.
Votre API doit permettre de faire du CRUD sur vos ressources et respecter au maximum les normes REST.
La base de données doit être composée de plusieurs entités avec au moins une relation entre 2 entités.

II. Le Projet

Bienvenue dans WikiClassic ! Le site qui répertorie les musiques classiques à travers le monde (ou presque), explique leur histoire et ce que chaque morceau cache derrière sa mélodie. Le but est d'expliquer à un écouteur de musique lambda, ce que le compositeur d'un morceau a voulu transmettre avec sa musique et que chaque visiteur puisse exprimer son ressenti pour chaque morceau.

Fonctionnalités :

Un visiteur peut accéder aux détails d'un morceau en cliquant sur son image ou en le recherchant dans la barre de recherche. Il y trouvera les différentes informations concernant les morceaux ainsi que les commentaires sur chaque morceau. Il peut consulter les commentaires ou en ajouter s'il décide de s'inscrire/se connecter. Il peut aussi consulter les morceaux de chaque mouvement classique s'il le souhaite grâce à la barre de recherche. Un utilisateur est déconnecté dés qu'il entre dans la page pour s'inscrire ou se connecter.

Le choix du front :

Nous avons décidé d'utiliser la librairie JQuery. Pourquoi ? Eh bien, après discussions avec des proches, collègues ou camarades de classe, du fait que nous avions aucune expérience en web que ce soit en html css ou javascript, nous avons décidé d'utiliser JQuery.

Au fur et à mesure du projet, ce choix s'est avéré être une erreur. En effet, la logique dont le site devait fonctionner de base se rapproche plus de celle du React que du JQuery.

En effet, nous voulions afficher plusieurs pages qui sont quasiment identiques. De ce fait, il aurait été plus optimisé de tout afficher sur la même page html que de créer une page html spécifique à chaque fois. Cela aurait rendu notre programme plus générique, plus facilement modifiable et beaucoup plus optimisé.

Une alternative en JQuery était pourtant possible si nous voulions utiliser qu'une seule page html. En théorie, il aurait fallu définir une variable globale qui détermine qu'un élément est actif et de définir que tous les autres sont cachés. En appuyant sur un élément, au lieu d'accéder à la page html en question, on aurait pu dire que la variable globale devient maintenant la "page" sélectionnée et que toutes les autres "pages" deviennent cachées. Après s'être rendu compte des possibilités de ReactJS, nous avons décidé de rester sur la création distincte de pages html car nous étions trop avancés dans le projet pour faire un tel changement.

La répartition du travail :

Khalil	Martin
- Création des classes - Création des pages HTML + éléments affichés - Création de tous les Controllers - Création de tous les Repositories - Création du système d'inscription/login - Création du système d'ajout de commentaire - Création des requête AJAX - Création du CSS - Implémentation des animations JQuery - Ajout de la fonctionnalité de recherche - Rapport	- Création des classes - Création des pages HTML - Rapport - Diagrammes

Les classes de notre projet :

Pour ce projet, nous avons 4 classes **: Utilisateur**, **Compositeur, Morceau** et enfin une classe **Commentaire**

Diagramme de classe :

Voici le diagramme de classe de notre projet généré par via IntelliJ

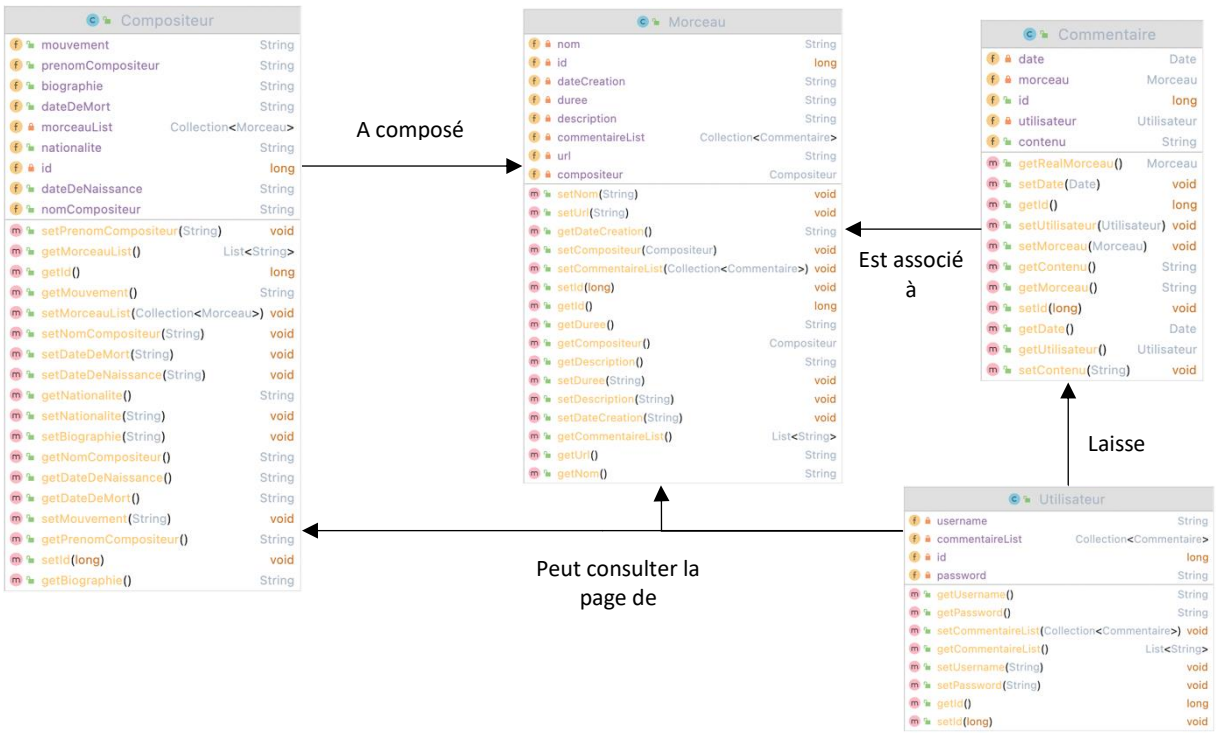
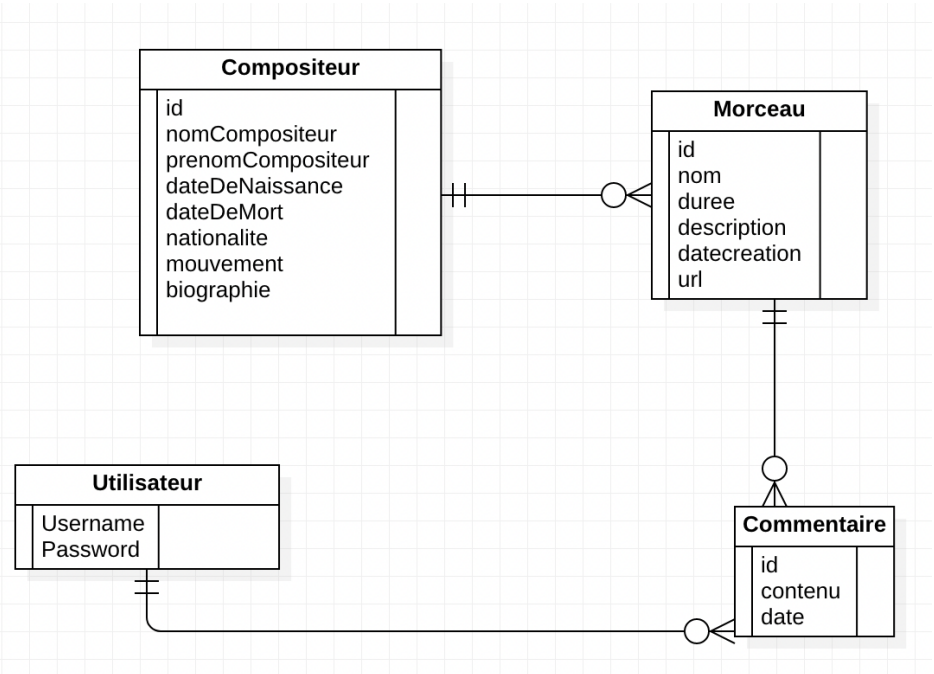


Diagramme entité relation :



III. La base de données :

Pour notre jeu de données, nous avons 8 morceaux différents :

- Ballade No. 1 in G minor, Op. 23
- Canon in d
- Cello Suite No. 1 in G Major
- Clair de Lune
- Experience
- Nocturnes, Op. 27, No. 2 in D-Flat Major
- Piano Concerto No. 2, Op. 18 in C Major, 1st Movement
- The Sixth Station

Chacun de ses morceaux possède : un **id**, un **nom**, une **durée**, une **description**, une **date de création**, un **compositeur** et une **url**.
Nous avons également 7 compositeurs :

- Frédéric Chopin
- Claude Debussy
- Joe Hisaishi
- Johann Pachelbel
- Jean-Sébastien Bach
- Sergei Rachmaninoff
- Ludovico Einaudi

Chaque compositeur possède : un **id**, un **nom**, un **prénom**, une **date de naissance**, une **date de décès**, une **nationalité**, un **mouvement**, une biographie et est associé à une liste de morceau.

Enfin il y a les utilisateurs qui sont créés lors de l'inscription, ils possèdent : un **id**, un **nom d'utilisateur** et un **mot de passe** qui sont stocké dans le LocalStorage lors de l'inscription. Il possède également une liste de **commentaires**.

Pour finir, il y a dans la base de données, les commentaires, ceux-ci sont postés par les utilisateurs et possèdent : un **id**, un **contenu**, une **date**, un **utilisateur** et un **morceau**.

IV. Difficultés rencontrées :

Ce fut pour nous deux la première fois que nous faisions du HTML, du CSS, et du JavaScript. Cela nous a en partie compliqué la tâche car nous avons dû apprendre toutes les bases pour pouvoir avancer dans le projet, il y a également eu quelques soucis avec le projet et IntelliJ. Il était impossible pour Martin de lancer le projet, il a dû supprimer tout le projet de son pc puis le recloner.

Il y a eu une multitude de problèmes rencontrés lors de l'évolution du projet. Ces problèmes sont dus, la plupart du temps, à un manque de connaissances.

Le premier gros problème rencontré était l'affichage de l'intitulé d'un morceau suivi du nom du compositeur. Nous ne savions pas que l'on pouvait manipuler les fichiers JSON avec des requêtes AJAX. De ce fait, des méthodes ont été testées pour contourner le problème, certaines fonctionnaient plus ou moins mais elles n'étaient pas aussi bien optimisées que d'utiliser la méthode actuelle. Comme faire une requête GET dans une autre, qui fonctionnait une fois sur trois car les requêtes ne se faisaient pas dans l'ordre souhaité.

Le deuxième gros problème était l'affichage des carrousels. En effet, au début du projet, lorsque l'on ouvre la page index.html, le carrousel ne s'affiche pas. Il s'affiche seulement si la fenêtre change de taille (F11, F12, changer manuellement la taille …). En analysant le problème, il s'avérait que le carrousel n'attendait pas que les images se chargent pour s'afficher (en regardant la console, la taille des images étaient de 0px). Plusieurs méthodes ont été testées : importer un script JS qui permettait d'attendre que toutes les images soient chargées pour effectuer une action. Cette méthode fonctionnait à 70% du temps, il restait tout de même des fois où le carrousel ne se chargeait pas totalement. La seconde méthode et celle qui a été retenue, est d'attendre 0,3s après que le document soit prêt pour afficher le carrousel. Cette méthode fonctionne 99% du temps. Le 1% restant correspond au fait que lorsqu'on lance l'index.html, il faut rafraichir la page pour que le carrousel s'affiche.

Un troisième problème était l'affichage d'un fichier JSON lorsque l'on récupérait un objet qui dépendait d'un autre. En fait, la requête GET fonctionne mais crée une boucle infinie : par exemple, un morceau possède un compositeur sauf qu'un compositeur possède une liste de morceaux. Donc le morceau affichait le compositeur qui lui affichait une liste de morceaux dans laquelle on trouvait le compositeur et ainsi de suite. Il a fallu changer les méthodes dans les classes correspondantes pour ne pas retourner le compositeur ou le morceau en entier mais simplement un de ses attributs comme, par exemple, le nom du morceau ou le nom prénom du compositeur. En plus de cela, lorsqu'on ajoutait un morceau, celui-ci ne s'ajoutait pas à la liste de morceaux du compositeur. Au contraire, il en créait un nouveau. Il a fallu ajouter le morceau au compositeur existant et supprimer le nouveau compositeur.

Le système de Login ne posa pas tant de problème que ça, du fait que Khalil code en C# sur Unity et que la notion de localStorage en JS est égale à un concept sur Unity qui s'appelle PlayerPrefs.

D'autres problèmes mineurs se sont ajoutés au fur et à mesure du temps, comme notamment le fait que deux utilisateurs identiques pouvaient s'inscrire, cela causait un problème dans les requêtes GET pour les utilisateurs car on récupère un seul utilisateur pourtant on en trouvait 2. Celui-ci a été réglé en imposant à un visiteur de créer un utilisateur unique. Un autre exemple est la création de la date lors de la création d'un commentaire. Les affichages YouTube, le css….

V. Améliorations possibles :

Sécurité : notre site n'est pas du tout sécurisé. En effet, il suffit de consulter le localStorage afin de trouver le mot de passe d'un utilisateur connecté. Il pourrait être intéressant de crypter le mot de passe lorsqu'on veut le consulter en inspectant la page.

Fonctionnalité : notamment sur la recherche d'un morceau, il faut écrire exactement l'intitulé du morceau sinon on ne peut pas le trouver. On pourrait alors créer une page de recherche qui affiche les premiers résultats et pouvoir après sélectionner le morceau voulu.
Il serait intéressant aussi de voir quel utilisateur est actuellement connecté

Pour aller plus loin : Pour les morceaux dont l'interprète est différent du compositeur, notamment pour les morceaux de longues dates comme pour Clair de Lune ou la première Ballade de Chopin, il serait plus poussé de créer une nouvelle classe qui s'appellerait "interprète" qui possède les mêmes attributs qu'un compositeur.

On aurait pu alors créer aussi une page pour chaque compositeur et interprète pour expliquer plus en détails leur biographie et autres.

Les morceaux possèdent, pour la plupart, une partition. Ajouter un attribut "partition" à la classe Morceau aurait été un plus.

Conclusion :

Ce projet a été bénéfique pour l'ensemble du groupe, à différentes échelles :

Khalil : “J’ai vraiment aimé réaliser ce projet, à vrai dire, je suis un fan de musique classique, donc le sujet que nous avions choisi était particulièrement motivant pour moi. Au point qu'il eut des jours où je ne pensais qu’à l’avancement du projet. J’avais du mal à passer à autre chose lorsque le code ne fonctionnait pas ou même lorsque j’avais décidé d’arrêter de travailler dessus pour le reste de la journée. J’avais aussi du mal à dormir du fait que je pensais constamment à des améliorations possibles ou à des correctifs de bugs. Je suis fier à chaque fois de montrer notre projet à des proches car j’ai vraiment investi énormément de temps dedans. Vous pourrez voir dans le commit certains intitulés peut-être un peu… comment dire… vous verrez de vous-même ;). J’ai l’impression d’avoir appris énormément de choses en si peu de temps.”

Martin : “ J’ai personnellement eu beaucoup plus de mal que Khalil, peut-être que cela est dû au fait que je ne porte pas un grand intérêt à la musique classique contrairement à Khalil qui adore cela. N’ayant jamais fait de développement web auparavant, j’ai pu acquérir des bases en HTML et en CSS, mais j’ai également pu acquérir des bases pour le JavaScript grâce à Khalil. Je pense qu’il serait intéressant que je refasse un projet de ce type personnel pour vérifier si j’en suis capable.”

Enfin, nous tenions à remercier Théo Barbary pour l’aide en JQuery, Martin Thibaut qui nous a aidé à démarrer notre projet en nous réexpliquant les bases d’une API REST. Enfin le frère de Khalil pour des explications sur le css.