Frémeaux Maxime / Abbas Massar

Rapport du projet WEB Spring

Fonctionnement de l'application :

- Cette application écrite en Java, utilise une base de données HSQLDB ainsi que spring-data pour la persistance des données.
  Elle implémente des services REST avec Jersey.
  Le back-end a été réalisé grâce au framework Spring, et en particulier Spring boot, qui la rend autonome. Le front-end est une combinaison du framework JS JQuery et de Bootstrap 5.
- Elle utilise les principes du CRUD (create, read, update, delete).
- Les commandes d'initialisation et de lancement du programme sont indiquées dans le README du projet source.

Différentes ressources :

Chaque table explicite dispose de sa ressource (JoueurResource et JeuResource). Ces ressources possèdent leurs propres méthodes, qui sont en fait des définitions de webservices (nous allons les utiliser au sein de la même application). Ce sont grâce à elles que nous effectuons les traitements en base de données (ajout, suppression, mise à jour…).

Pour chaque requête, nous avons l'ensemble des possibilités CRUD.
En effet, JeuResource propose l'ajout d'un jeu (POST), sa modification (PUT), ses détails (GET), sa suppression (DELETE).
Par ailleurs, JoueurResource propose l'ajout d'un joueur (POST), sa modification (PUT), ses détails (GET), sa suppression (DELETE), l'ajout (PUT) ou la suppression (PUT) d'un jeu à/de sa bibliothèque personnelle, l'ajout d'un jeu parmi la liste à un joueur choisi (PUT).

La Base de Données :

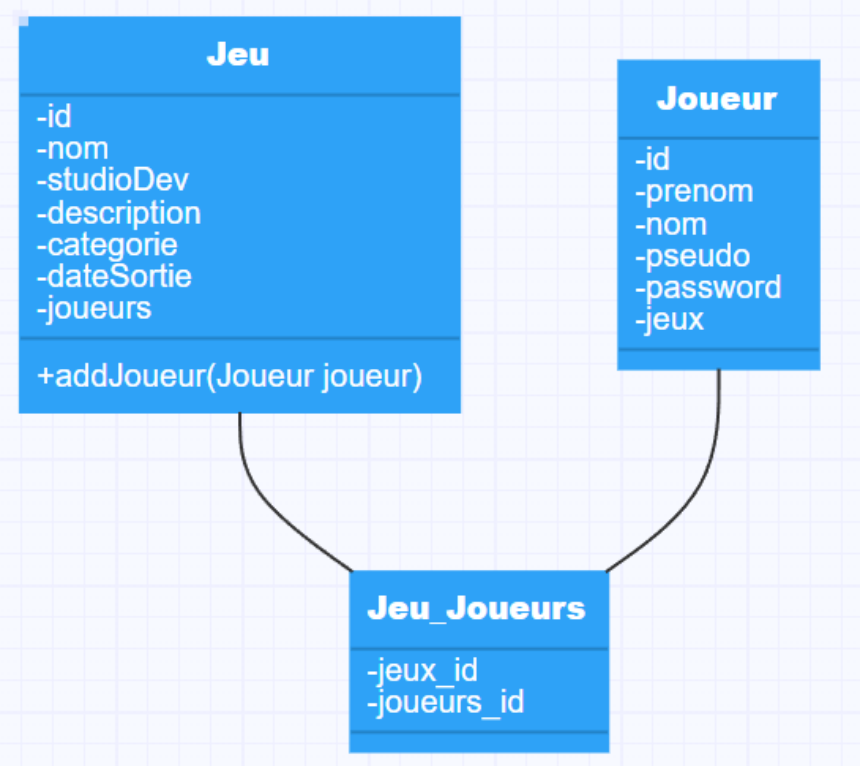

Diagramme de classes

La base de données est composée de 2 tables, ainsi qu'une table implicite, générée par Hibernate, qui réalise la relation entre ces 2 tables.
La première table est Joueur, qui définit donc un joueur de jeux vidéo. La seconde table est la table Jeu, qui contient l'ensemble des jeux référencés sur le site. Ces tables sont en relation multiple, étant donné qu'un joueur peut posséder plusieurs jeux, et qu'un jeu peut avoir plusieurs joueurs.

Répartition du travail :

- Initialisation du projet sous Spring Initializr : Massar
- Initialisation du dépôt git : Massar
- Conception de la classe Jeu et de sa ressource : Maxime
    - Ajout d'un jeu
    - Détails d'un jeu
    - Modification d'un jeu
    - Suppression d'un jeu
- Conception de la classe Joueur et de sa ressource : Massar
    - Ajout d'un joueur
    - Ajout d'un jeu à un joueur (+ Maxime)
    - Suppression d'un jeu à un joueur
    - Détails d'un joueur
    - Modification d'un joueur
    - Modification du mot de passe d'un joueur
    - Suppression d'un joueur
- Implémentation des classes JeuDTO et JoueurDTO : Maxime
- Mise en place des frameworks client (Bootstrap 5) : Maxime
- Header et footer et association aux pages : Maxime
- Association d'un jeu à un joueur
    - back-end + appel AJAX : Maxime
    - front-end (formulaire) : Massar
- Correction de bugs : Maxime
- Rédaction du README : Maxime & Massar
- Ajout des jeux de test utilisant JUnit4 (JeuTest et JoueurTest) : Maxime

Difficultés rencontrées :

- D'abord, le peu de connaissances que nous avions initialement sur la technologie Spring nous a causé beaucoup de difficultés et de temps de recherche.
- D'une manière générale, la relation entre les 2 tables s'est avérée assez complexe dès le départ, étant donné que les 2 tables composaient l'une et l'autre (ce qui posait un problème de récursivité infinie). Cela a été solutionné par l'utilisation des classes DTO.
- La relation ManyToMany nous a globalement posé des problèmes de persistance de données. En effet, lorsque nous avons souhaité effectuer une fonctionnalité plus avancée, qui est l'association d'un jeu à un joueur (et l'ajout du joueur au jeu), nous avions ce même souci de boucle imbriquée, que nous avons solutionné en passant directement l'objet DTO en paramètre.
- L'utilisation de HSQLDB, bien que permettant la saisie de données SQL, est dans son ensemble peu intuitive.
  Rafraîchir (supprimer puis régénérer) la base de données est une opération relativement fastidieuse, car il n'y a pas d'autocomplétion par exemple. De plus, les sauvegardes d'enregistrement sont très difficilement trouvables et réutilisables comme telles qu'elles sont fournies.