

Rapport de TP de Technologie Web Serveur

Loïc CASTILLO Loris CROCE

1 Conception

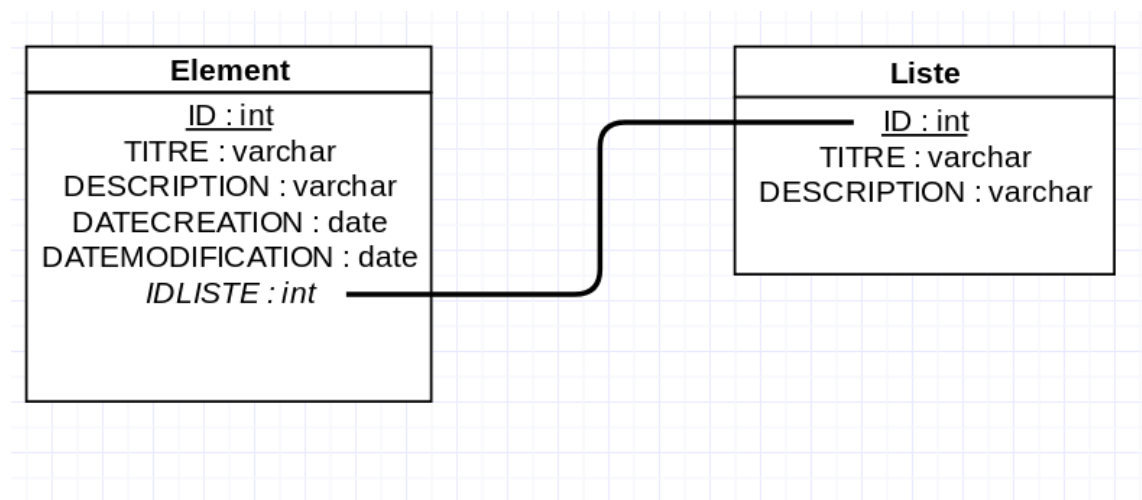


FIGURE 1 – Modèle de données

Ressources On interagira avec des listes et des éléments qui pourront être affichés (un élément affiché dans une liste), ajoutés, supprimés et modifiés.

Espaces de noms Pour l’affichage seules les listes auront une URL de la forme : */liste/nomDeLaListe*

2 Modèle et accès au données

Nous avons donc deux classes métiers pour le modèle : **Element** et **Liste** définissant les deux objets manipulés.

Étant donné que le nombre d’appels à la base de données ne sera pas très élevé, nous avons décidé que notre DAO serait une classe statique. Les informations pour se connecter à la base de données sont stockées dans les variables de la classe. Chaque

fonction retourne des objets métiers et est utilisée pour une tâche précise (ex : récupérer toutes les listes, les éléments, un element particulier, les elements en fonction de leur liste...). Chaque appel de fonction doit recréer une connexion à la base de données.

3 Représentations

Via FreeMarker, nous avons créé 3 templates :

- `complet.ftl`, qui affiche tous les éléments de toutes les listes.
- `listeDetail.ftl`, qui affiche une liste donnée et tous ses éléments. Elle permet de supprimer ou de modifier des éléments.
- `listes.ftl`, qui fait office de page “d’accueil” en affichant un tableau des listes. Elle permet également de d’ajouter, modifier ou supprimer des listes, ajouter des éléments.

4 Interactions

Pour la gestion des interactions par le contrôleur avec Spark nous n’avons utilisé que deux méthodes HTTP : GET et POST. En effet, la gestion de méthodes telles que DELETE ou PATCH ne s’est pas révélée assez pertinente en raison de leur plus complexe mise en place. On a donc un ensemble de *routes* de types `get` pour récupérer du contenu ou supprimer des éléments (on affiche la liste en enlevant l’élément voulu). Ainsi qu’un ensemble de *routes* `post` pour créer ou modifier du contenu.

5 Ajout de fonctionnalités

Statut optionnel Il suffit d’ajouter une table `Etat` dans la base de données, contenant un id et un varchar, qui contiendrait 3 tuples, dont la valeur des varchar serait : soit *à faire*, *fait* ou *supprimé*.

Sous-listes Dans la BDD, la table `Liste`, aurait une colonne `Upper`, qui contiendrait l’ID d’une autre liste, ou serait NULL. En Java, cela se matérialiserait par l’ajout d’une List de Liste dans la classe Liste. Celle-ci serait vide si l’attribut est NULL dans la BDD.

Élément appartenant à plusieurs listes Dans la BDD, il faudrait créer une table de liaison, qui lirait deux listes. On pourrait l’appeler `ListJunction`, et serait composé de deux id de Liste. L’ajout de ces deux id donneraient la clé primaire de l’élément.

Étiquette En Java : `List<String>` dans chaque élément correspond aux tags. En BDD : Ajout d’une table `Tag` [`id` : long (PRIMARY KEY), `text` : varchar]. Ajout d’une table pour unir un Tag à un Element. Cette table contiendrait en attribut l’id d’un tag et d’un element, et sa clé primaire serait la concaténation des deux.