École Nationale Supérieur des Sciences Appliquées et de Technologie

JEE
Rapport

Gestion de DJ

Élève :
Clément GONZALES,
Tanguy LEVREY,
Enzo RODRIGUES,
Paul SCHELLER

Enseignant :
Alexandre Chenebaud
Encadrant :
Vincent Barreaud

4 avril 2024

Table des matières

1 Introduction

1.1 Naissance du Projet

Dans le cadre de notre apprentissage des services web, nous avons suivi des cours de programmation Java Entreprise Environment dans le but de créer une api REST très courant dans le monde des services WEB. Afin de mettre en pratique nos nouvelles connaissances un projet est mis en place.

1.2 Présentation du Projet

Le but du projet est de réaliser une API REST ainsi qu'une interface Web permettant de communiquer avec cette dernière. La toile de fond de ce projet est l'élaboration d'un logiciel de gestion des DJs qui permet notamment de planifier des évènements.

1.3 Attentes et Contraintes

Les attentes concernant le site web sont les suivantes :
- Un ensemble de pages permettant de gérer les DJs : affichage de la liste des DJs, création, modification et suppression d'un DJ.
- Une page affichant la liste des lieux pouvant accueillir les événements des DJs.
- Une page permettant de planifier un événement d'un DJ.
- Une page permettant de lister par mois tous les événements à venir.
- Une page listant le top 5 des DJs qui ont le plus d'événements passés.

Quelques précisions sur la nature des objets utilisés sont également fournies de même que des contraintes sur certains de ces objets.

- Un DJ est représenté par :
  - Nom
  - Prénom
  - Nom de scène
  - Date de naissance
  - Lieu de résidence (de type String, indépendant de la classe Lieu)
  - Style musical (parmi la liste : electro, house, hard style, EDM)

- Un événement est caractérisé par :
  - Une date
  - Une heure de début
  - Une heure de fin
  - Un DJ
  - Le lieu de l'événement (instance de la classe Lieu)

- De plus le système intègre 2 règles concernant les événements, en plus des logiques excluant la possibilité de 2 événements en même temps au même endroit ou un DJ à 2 endroits en même temps.
  1. Pour un DJ, il faut respecter une période de 24h minimum entre 2 évènements qui ont lieu sur le même continent.

2. Le délai s'étend à 48h pour 2 évènements sur des continents distinct

1.4 Versionnage

Afin d'assurer la cohérence de notre travail ainsi que de permettre le versionnage de notre projet nous avons utilisé Git lors de la production du projet par l'intermédiaire du Gitlab de l'ENSSAT. Notre dépôt Git est présent à cette adresse : Projet Batman

2 Structure de la base de données

2.1 Tables

Pour la création de la base de données, il a fallu réaliser des choix d'implémentations pour s'adapter à la création de certains triggers par exemple. Nous avons donc créé les 3 tables en prenant en compte certains détails.

2.1.1 DJ

La table DJ comprend les colonnes nom, prenom, nom_scene, date_naissance, lieu_naissance et style_musical. Nous avons utilisé la colonne nom_scene comme primary key car nous sommes parti du principe que chaque DJ à un nom de scène différents.

2.1.2 Evenement

La table Evenement comprend les colonnes date_debut_evenement, date_fin_evenement, DJ, lieu_evenement et ville_evenement. Dans cette table, c'est le couple date_debut_evenement DJ qui sert de primary key, et il y a la colonne DJ qui est une external key de la table DJ sur la colonne nom_scene. Et aussi les colonnes lieu_evenement et ville_evenement qui sont l'external key des colonnes nom_lieu et ville de la table Lieu. Nous avons eu besoin d'ajouter la ville de l'évènement à la table Evenement pour je sais plus quoi.

2.1.3 Lieu

La table Lieu comprend les colonnes nom_lieu, ville, pays et contient. Nous avons utilisé le couple de colonnes nom_lieu et ville comme primary key car nous avons remarqué que dans la base de données de lieux donnée dans l'énoncé, certains lieux possèdent le même nom mais ne sont pas situé dans la même ville.

2.2 Triggers

2.2.1 DJ

Nous avons mis en place un trigger dans table DJ qui permet de supprimer tout les évenements prévus pour un artiste avant de supprimer cette artiste de la base de données.

2.2.2 Evenement

Dans la table Evenement, nous avons du mettre en place plusieurs triggers. Les triggers Evenement_meme_continent et Evenement_diff_contient permettent de vérifier si le DJ choisi a déjà un événement de programmé dans les 24 h si c'est sur le même contient et dans les 48h si c'est sur un contient différent. Le troisième trigger quand à lui vérifie si il n'y a pas déjà un un événement dans programmé dans le même lieu par un autre DJ au moment d'en ajouter un nouveau.

2.2.3 Lieu

Nous avons mis en place un trigger dans table Lieu qui permet de supprimer tout les évenements prévus pour un lieu avant de supprimer ce lieu de la base de données.

3 BackEnd

3.1 Les Endpoints

Pour l'utilisation de notre API Rest, nous avons crée une liste d'EndPoints, qui se caractérise de la manière suivante :

Gestion des DJs

— Afficher les DJs (URI : /djs) (méthode : GET) (corps :)
— Récupérer un DJ (URI : /djs/nom/{nomScene}) (méthode : GET)
— Ajouter un DJ (URI : /djs) (méthode : POST) (corps :) (headers : 'Content-Type' : 'application/x-www-form-urlencoded') (corps : données sous forme de formulaire)
— Éditer un DJ (URI : /djs/nom/{nomScene}) (méthode : PATCH) (headers : 'Content-Type' : 'application/json') (corps : string json)
— Supprimer un DJ (URI : /djs/nom/{nomScene}) (méthode : DELETE)
— Afficher le top 5 DJs (URI : /djs/filtre/top)

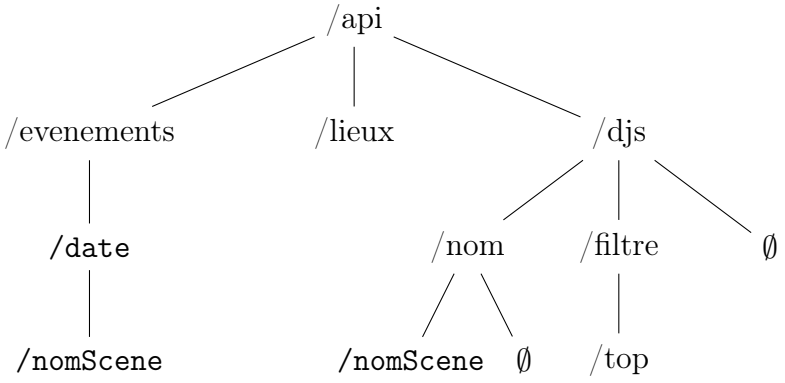
Gestion des lieux

— Afficher les lieux (URI : /lieux) (méthode : GET) (corps :)

Gestion des évènements

— Ajouter un évènement (URI : /evenements) (méthode : POST) (corps :)
— Afficher le calendrier (URI : /evenements) (méthode : GET) (corps :)

On retrouve l'arbre correspondant au chemin d'accès de l'api ci-dessous :



3.2 DJs

3.2.1 Classe DJ

Nous avons créé une classe DJ regroupant les attributs présentés dans la rubrique 1.3 à la page 2. Elle est composée d'un constructeur ainsi que des **getters** et **setters** pour chaque attribut.

3.2.2 DAO

La DAO contribuant à la gestion des DJs permet de faire le lien entre l'application et la base de données. Son fonctionnement est le suivant :

Le code implémente des chaînes de caractères qui correspondent à des requêtes SQL incomplètes du type :

```
1 private static final String SELECT_DJ_BY_NAME = "SELECT * FROM DJ WHERE
    nom_scene = ?";
```

Listing 1 – Exemple de requête SQL incomplète

Dans ces requêtes, on remarque des points d'interrogation à la place de certains attributs, ce sont ces derniers qui doivent être modifiés pour obtenir une vraie requête.

Les fonctions implémentées ci-dessous permettent de compléter ces requêtes à l'aide de la bibliothèque *java.sql*, qui permet notamment de préparer la requête avant de l'exécuter :

- `getAllDJs()` : permet de récupérer la liste des DJs répertoriés dans la base de données.
- `getDJ(nomScene)` : permet, à partir du nom d'artiste du DJ, de récupérer ses données dans la base de données et de les renvoyer.
- `addDj(dj)` : prend en entrée un DJ et l'ajoute à la base de données.
- `editDj(dj)` : permet la modification des informations d'un DJ dans la base de données.
- `deleteDj(nomScene)` : permet de supprimer un DJ de la base de données en utilisant son nom de scène.
- `topDj()` : permet de récupérer dans la base de données les DJs ayant effectué le plus d'événements.

3.2.3 Controller

Le contrôleur permettant la gestion des DJs consiste essentiellement à communiquer avec le "FrontEnd" ou plutôt avec l'utilisateur au travers de requêtes `HTTP` : **POST**, **GET**, **PATCH**, et **DELETE**.

L'objectif principal de ce contrôleur est tout d'abord de récupérer les données à intégrer, modifier ou supprimer et de les structurer. Ensuite, il utilise les méthodes implémentées ci-dessus par le DAO, en utilisant les données précédemment mises en forme. Après l'exécution de la méthode, soit on renvoie un acquittement, soit on renvoie une chaîne JSON.

Les données reçues par le controlleur peuvent être de plusieurs types :

- Paramètres de formulaire : l'ajout d'un DJ se fait, comme son nom l'indique, via un formulaire en ligne pour lequel on récupère les données.

— Paramètres issus du chemin de l'API : il est possible de récupérer le nom de scène d'un artiste s'il est intégré lors de la requête dans le chemin de l'API. Exemple : `api/djs/nom/{nomScene}`.
— Chaîne JSON : pour l'édition d'un DJ, les données sont directement modifiées dans le tableau sur la page. Il est donc plus simple et ergonomique de récupérer une chaîne JSON pour récupérer les données.

3.3 Événements

3.3.1 Classe

La classe Événement décrit des objets de la manière suivante :

```
public class Evenement {
  private String dateDebut;
  private String dateFin;
  private String nomDj;
  private String lieu;
  private String ville;
 ...
 }
```

Listing 2 – Structure de la classe Événement

Où chaque élément correspond donc intuitivement aux index de la table Événement de notre base de données. Elle est également composée de getter et setter pour chacun de ses éléments.

3.3.2 DAO

Tout comme la DAO des DJs la DAO des événements fonctionne sur la base de requêtes pré-complétées auxquelles ont ajoute les éléments issus des appels de fonctions. Les fonctions implémentées dans la DAO sont les suivantes :

— `getAllEvents()`, qui renvoie la liste des événements sous forme `List<Evenement>`
— `getEventById(String dateDebut, String nomDj)` qui renvoie l'événement ayant comme clé primaire le couple `(dateDebut, nomDj)`
— `addEvent(Evenement event)` qui prend un entrée un événement et qui l'ajoute à la base de données en prenant en compte ses attributs.

3.3.3 Controller

Le controller permet le lien entre le site internet et le back-end. Le controller des événements envoie les données des requêtes GET sous forme de JSON qui est ensuite récupéré dans le front-end qui met en forme ces données. Il récupère également les données du formulaire envoyé dans la requête POST pour appeler la requête `addEvent()` de la DAO

3.4 Lieux

3.4.1 Classe

La classe Lieu décrit des objets de la manière suivante :

```java
public class Lieu {
  private int id;
  private String nom_lieu;
  private String ville;
  private String pays;
  private String continent;
 }
```

Listing 3 – Structure de la classe Événement

Où chaque lieu correspond donc intuitivement aux index de la table Lieu de notre base de données. Elle est également composée de getter et setter pour chacun de ses éléments.

3.4.2 DAO

Tout comme la DAO des DJs et des évènements, la DAO des lieux fonctionne sur la base de requêtes pré-complétées auxquelles ont ajoute les éléments issus des appels de fonctions. Les fonctions implémentées dans la DAO sont les suivantes :

- `getAllPlaces()`, qui renvoie la liste des événements sous forme `List<Lieu>`
- `getLieu(String nom_lieu)` qui renvoie l'événement ayant comme clé primaire `nom_lieu` On pourrait implémenter une fonctionnalité permettant de rajouter des Lieux, dans le cas où un évènement à planifier se déroulerait dans un lieu, encore non répertorié.

3.4.3 Controller

Comme pour Evenement et Dj, Le controller des Lieux permet le lien entre le site internet et le back-end. Le controller des événements envoie les données des requêtes GET sous forme de JSON qui est ensuite récupéré dans le front-end qui met en forme ces données(sous forme d'une liste dans l'onglet "Liste des lieux" ou bien dans le menu déroulant de l'ajout d'un nouvel évènement pour que l'utilisateur puisse choisir le lieu parmi la liste existante).

4 FrontEnd

L'interface web réalisée présente 5 onglets permettant de profiter des fonctionnalités implémentées.

Une liste des DJs est consultable dans le premier onglet, il est possible de voir leurs différents attributs dans les différentes colonnes.
Il est possible d'ajouter un DJ à la liste via un formulaire. De plus, ces DJs peuvent aussi être modifiés ou effacés dans cette même page.

Les évènements sont consultable dans l'onglet "évènements à venir". Ils sont représentés dans un calendrier. Un mois est représenté par une page, avec une case par jour, cette dernière listant les évènements survenant le jour en question.

Pour ajouter un évènement un formulaire est disponible dans l'onglet "planifier un événement" ou un formulaire permet d'ajouter le dit événement.

Finalement une page affichant la liste des 5 DJs ayant réalisés le plus d'événements est présente.

Le site web dispose d'une barre de navigation pour faciliter le déplacement entre les différentes sections.

4.1 Librairies utilisées

— La partie CSS du site et tout l'aspect graphique de ce dernier est géré par la librairie **Bootstrap**.
— L'affichage des événements suivant un calendrier utilise **FullCalendar**

5 Conclusion

En conclusion ce projet nous aura permis de mettre en pratique nos connaissances relatives aux services Web par l'intermédiaire d'une API REST tout en se servant d'autres enseignements tels que la BDD et l'enseignement de programmation Web. Le projet fourni correspond au cahier des charges hormis un bonus non développé (la mise en place de la sécurisation du site par le biais d'une page de login). Une meilleure interface graphique pourrait être mise en place ainsi que des fonctionnalités supplémentaires sur la gestion des lieux de production des DJs. En effet on pourrait imaginer une page par lieux recensant les événements y ayant eu lieu. Ces compléments pourrait être envisagés dans le cadre d'une poursuite de ce projet.