Compte rendu TP

CREATION API

Gauvin Alexandre
Delcroix Florian

Technologie web | 02/12/2022

Lancement

Pour lancer le logiciel il suffit de se placer dans le répertoire du projet et d'exécuter les scripts gradle.

Build : .\gradlew build

Lancer : .\gradlew bootrun

Tests : .\gradlew test

Architecture

Notre architecture se base sur le modèle Contrôleur-Service-Modèle-Repository.

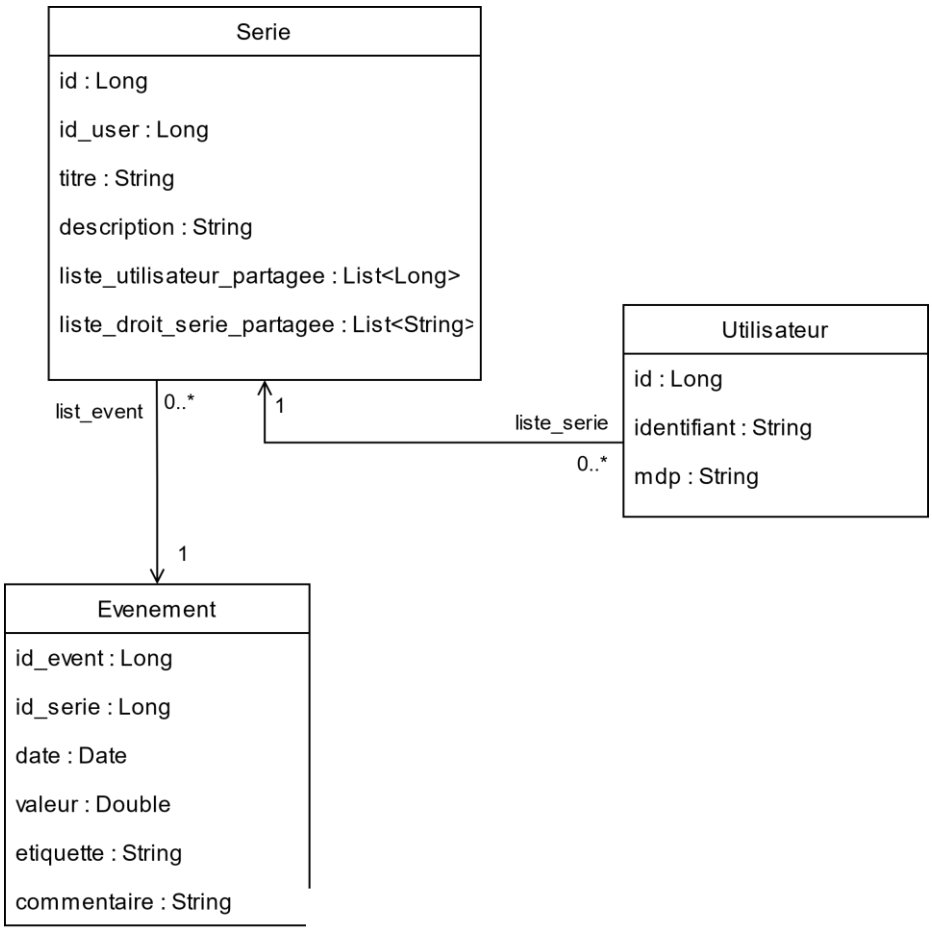
Ce modèle est très largement utilisé dans les projets SpringBoot car c'est un modèle qui sépare proprement et de façon distincte toutes les parties de notre projet. L'autre raison est que SpringBoot donne accès à beaucoup d'outils qui facilitent le développement de projet sous ce modèle.

Pour rentrer un peu plus dans le détail :

- Dans un premier temps, nous avons le Controller, cette couche est seulement responsable d'exposer les fonctionnalités de l'API.
- Ensuite nous avons le Service, cette couche s'occupe de toute la logique de traitement des informations.
- Le Repository quant à lui s'occupe de faire le lien avec la base de données (quelque que soit l'implémentation de base de données choisie).
- Enfin nous avons le Modèle qui sert de base pour les données manipulées.

Modèle de données

Ci-dessous, le schéma de notre modèle de données :



Nous avons choisi de travailler avec trois entités :

- Les utilisateurs qui possèdent et gèrent des séries.

- Les séries qui se composent d'évènements et contiennent les informations permettant leurs accès (créateur et utilisateur partagé).

- Les évènements.

Notre approche

Pour travailler sur ce projet nous avons utilisées l'approche TDD, vu l'an dernier dans l'option ingénierie logiciel.

TDD pour Test Driven Développement est une approche qui vise à créer des jeux de test pendant le développement afin de toujours s'assurer du bon fonctionnement des fonctionnalités implémentées.

En outre, nous avons dans un premier lieu travaillé sur le modèle de données, qui est la base de ce projet, une fois notre modèle de donnée complet nous avons développé les 3 autres couches, à savoir les services, les contrôleurs et les repository.

Les ressources

Dans cette section nous allons détailler les différentes urls ainsi que les méthodes qui permettent de gérer notre API. Les urls liées aux séries et aux évènements contiennent toutes un premier paramètre avant l'url indiquant quel utilisateur est à l'origine de la requête. Le but de ce paramètre et vérifier les autorisations d'accès à l'action demandée sans passer par une phase d'authentification.

Utilisateurs

/utilisateurs

Méthode GET, retourne tous les utilisateurs.

Retourne 200 en cas de succès.

/utilisateur

Méthode POST, créé un utilisateur, les données pour créer un utilisateur sont dans le corps de la requête (Les champs nécessaire sont *identifiant* et *mdp*).

Retourne 201 en cas de succès.

/utilisateur/{userid}

Méthode GET, retourne l'utilisateur qui a pour id : userid.

Retourne 200 en cas de succès.

/utilisateur/{userid}

Méthode DELETE, supprime l'utilisateur qui a pour id : userid.

Retourne 200 en cas de succès.

/utilisateur

Méthode PATCH, modifie l'utilisateur donné dans le contenu de la requête.

Retourne 200 en cas de succès.

/utilisateur/connect

Méthode GET, permet de récupérer l'id d'un utilisateur après vérification du couple identifiant – mot de passe.

Evènements

/{id_user}/evenement

Méthode POST, créé un événement contenu dans le corps de la requête.

Exemple de corps au format JSON :

{ "id_event":1,
"id_serie": 1,
"date": "2022-10-27",
"valeur": 150.0,
"etiquette" : "Escrime ludique",
"commentaire" : "Trop Fun" }

Retourne 201 en cas de succès.

/{id_user}/evenement /{id}

Méthode DELETE, supprime l'évènement dont l'id correspond à id.

Retourne 200 en cas de succès.

/{id_user}/evenement

Méthode PUT, modifie l'évènement passé dans le corp de la requête.

Retourne 200 en cas de succès.

{id_user}/evenement/{id_event}

Méthode GET, retourne l'évènement dont l'id correspond à id_event.

Retourne 200 en cas de succès.

Séries

/serie

Méthode POST, créé une série contenue dans le corp de la requête.

Exemple de corps en JSON :

{ "id_serie": 1,
"date": "2022-10-29",
"valeur": 250.0,
"etiquette" : "Sieste",
"commentaire" : "" }

Retourne 201 en cas de succès.

/{id_user}/serie/{id}

Méthode GET, retourne la série dont l'id correspond à id.

Retourne 200 en cas de succès.

/{id_user}/serie/{id}

Méthode DELETE, supprime la série dont l'id correspond à id.

Retourne 200 en cas de succès.

/{id_user}/serie/{id_serie}/events

Méthode GET, retourne tous les évènements dont l'id de la série correspond à id_serie.

Retourne 200 en cas de succès.

/{id_user_init}/serie/{id_serie}/partage_consultation/{id_user}

Méthode PATCH, modifie la liste des droits de consultation.

Retourne 200 en cas de succès.

/{id_user_init}/serie/{id_serie}/partage_modification/{id_user}

Méthode PATCH, modifie la liste des droits de modification.

Retourne 200 en cas de succès.

/{id_user_init}/serie/{id_serie}/modifier_type_partage_en_modification/{id_user}

Méthode PATCH, modifie une série, et ses droits de modification. (Passage des droits en modification).

Retourne 200 en cas de succès.

/{id_user_init}/serie/{id_serie}/modifier_type_partage_en_consultation/{id_user}

Méthode PATCH, modifie une série, et ses droits de modification. (Passage des droits en consultation).

Retourne 200 en cas de succès.

/{id_user_init}/serie/{id_serie}/id_user}

Méthode PATCH, retire le partage de la série à l'utilisateur dont l'id correspond à id_user.

Retourne 200 en cas de succès.