Service Commentaires

Joanna Gueroult

Groupe 3

Déploiement

Avant tout déploiement, il faut renseigner les informations du serveur Kafka et de la base de données dans application.properties et DatabaseController.

Le schéma de données de la base de données est disponible dans commentaires.sql.

APIs

Le service commentaire expose 3 endpoints APIs par type de commentaires (music, playlist, event): GET /comments/[Type] qui renvoie un set de commentaires d'un(e) musique/playlist/évènement spécifique, POST /comment/[Type] qui permet l'ajout d'un commentaire, et DELETE /deletecomment/[Type] qui permet de supprimer un commentaire. En plus de ça, un endpoint API de plus est disponible pour les musiques pour permettre de savoir si un utilisateur a droit de noter une musique (car on ne peut la noter qu'une fois), GET /ratability/music.

GET /comments/[Type]

Ces endpoints prennent 4 arguments mais 3 possèdent des valeurs par défaut et n'ont donc pas à être donnés.

- id: l'ID de la/du musique/playlist/évènement dont on veut récupérer des commentaires.
- sort: Comment trier les commentaires (valeurs parmi dtasc et dtdsc (pour "date ascending", "date descending"), et rtgasc et rtgdsc (pour "rating ascending", "rating descending"), valeur par défaut dtasc.
- amnt: Pour amount, la quantité de commentaires à récupérer, valeur par défaut 10.
- strtAt: Pour startAt, où commencer dans la liste (ce qui permet par exemple de ne récupérer qu'une fraction des commentaires, puis d'en récupérer plus à la demande), valeur par défaut 1 (donc début de la liste).

La liste est retournée au format XML.

POST /comment/[Type]

Ces endpoints prennent un flux XML en entrée et renvoient simplement un int en retour, -1 si l'ajout a échoué, l'ID du commentaire sinon.

Ici les structures des flux XML d'entrée:

```
<playlistcomment playlistid="P" author="A" [replyto="R"]>
    <content>Test comment</content>
</playlistcomment>

<eventcomment eventid="E" author="A" [replyto="R"]>
    <content>Test comment</content>
</eventcomment>

<musiccomment musicid="M" author="A" [replyto="R"]>
    <content>Test comment</content>
```

```
    [<rating>5</rating>]
</musiccomment>
```

DELETE /deletecomment/[Type]

Ces endpoints permettent la suppression de commentaires (attention: La suppression efface le contenu du commentaire mais pas le commentaire lui-même, qui continuera de faire partie des listes de commentaires). Elles prennent en unique paramètre l'ID du commentaire à supprimer.

En retour, 1 pour une suppression réussie, 0 pour un commentaire qui n'existe pas, -1 pour une erreur interne.

GET ratability/music

Ce endpoint permet de connaître la capacité d'un utilisateur à noter une musique. Les paramètres sont id (de la musique) et user (l'ID de l'utilisateur en question).

En retour, 1 pour la capacité à noter, 0 pour l'incapacité, et -1 pour une erreur interne.

Topics

Le service commentaire envoie sur un topic, userComments, et consume sur un autre, userDeletion.

Lorsqu'il lit une ID sur userDeletion, il supprime tous les messages de l'utilisateur (encore une fois, le contenu des commentaires est effacé, mais pas les commentaires eux-mêmes; Le message est différent qu'avec la suppression par un utilisateur).

Lorsqu'un utilisateur commente, un message est envoyé sur userComments pour signaler à certains autres services (comme messagerie, pour envoyer un message à la personne à laquelle il répond, par exemple) qu'un utilisateur a laissé un commentaire. Le contenu du commentaire ne fait pas partie du message (ni son ID), seulement l'ID du commenteur, si il y en a un, l'ID du commentaire auquel il a répondu, le type de contenu commenté, et l'ID du contenu commenté, car le commentaire lui-même n'est à priori pas utile pour updater les statistiques ou prévenir la personne à laquelle il a été répondu.

Le message a donc la structure suivante:

```
userID="U" replyto="R" commented="(music/event/playlist)" commentedID="C"
```