Belegaufgabe 4

songsWS mit Endpoint songLists

Abgabetermin: Ihre Übung in KW 3, am 14./15.1. oder in KW 4, am 21./22.1. (Achtung: Am Sa, 25.01. findet die Klausur statt.)

In diesem Beleg werden Sie einen weiteren Endpoint **/songsWS- TEAMNAME/rest/songLists** implementieren. Dieser Endpoint wird Listen von Songs (playlists) ausliefern, empfangen und löschen. Zusätzlich zu unserer Songs-Collection legen wir eine Sammlung von Songlisten und entsprechende DB-Tabellen an. Eine Songliste hat folgende Eigenschaften:

- Id (Integer, PK)
- OwnerId (User, FK)
- einen Namen (optional)
- ist "private" oder "public"
- eine Liste von Songs

D.h., eine Songliste gehört einer Owner-Person und kann "private" oder "public" sein.

Die Songs gehören allen Users, haben also keine Besitzer_innen. GET, POST und PUT von Songs soll für Clients bestehen bleiben. **Clients dürfen Songs allerdings nicht mehr löschen.** D.h., die Anfrage **DELETE /songsWS-TEAMNAME/rest/songs/6 muss mit HTTP-Code 405 (Method not allowed) beantwortet werden.**

Der neue Endpunkt kann wie folgt erreicht werden:

/songsWS-TEAMNAME/rest/**songLists/<SONGLIST_ID>** &
/songsWS-TEAMNAME/rest/**songLists?userId=SOME_USERID**

Zum Verständnis der Anforderungen nehmen wir an, dass

- **User "mmuster" hat sich bei Ihrem Service erfolgreich angemeldet.**
- **http-Anfragen von mmuster** an Ihren Service sind mit dem "Authorization"-Header versehen. Diesen Header werde ich übersichtshalber im Folgenden weglassen.

Anforderungen an den "songLists"-Endpoint:

a) GET, POST und DELETE implementieren
b) Ein- und Ausgabeformate sind JSON und XML
c) GET /songsWS/rest/songLists?userId=SOME_USERID
Beispiel: '**GET /songsWS/rest/songLists?userId=mmuster'** soll **alle** Songlisten von 'mmuster' an User 'mmuster' zurückschicken.
Beispiel: '**GET /songsWS/rest/songLists?userId=eschuler'** soll nur die '**public'** Songlisten von 'eschuler' an User 'mmuster' zurückschicken.
Beispiel: '**GET /songsWS/rest/songLists?userId=usergibtsnicht'** schickt HTTP-Statuscode 404 zurück.
d) GET /songsWS/rest/songLists/<SONGLIST_ID>
Beispiel: '**GET /songsWS/rest/songLists/22'** soll die Songliste 22 an User 'mmuster' zurückschicken. Eine Songliste 22 muss existieren. Wenn die Liste mmuster gehört, dann kann sie an mmuster zurückgeschickt werden. Wenn die Liste einem anderen User gehört, dann kann die Liste nur zurückgeschickt werden, wenn sie "public" ist, ansonsten einen HTTP-Status 403 (FORBIDDEN) schicken.
e) **POST /songsWS/rest/songLists** mit XML oder JSON-Payload legt eine neue Songliste an. Songlisten, die gepostet werden, enthalten keine SongList-Ids, da diese von Ihrem Service oder der DB vergeben werden. Owner der neuen Songliste soll aus dem Token im Authorization-Header ermittelt werden. Falls erfolgreich, d.h., neue Songlist für User 'mmuster' wurde angelegt, dann schickt Ihr Service die URL mit neuer SongList-Id im Location-Header zurück. Alle Songs in der Payload der Anfrage müssen in der DB existieren. Falls einer der Songs nicht existiert, können Sie der einfachheithalber eine 400 (BAD_REQUEST) zurückschicken. In Moodle werden ich Ihnen Payloads für Songlisten in JSON und XML bereitstellen. Diese Payloads können zum Testen des POST-Endpoints verwendet werden.

f) DELETE /songsWS/rest/songLists/<SONG_LIST_ID>
User können nur eigene Songlisten löschen, nicht die der anderen User, auch keine fremden, public Songlisten.
Beispiel: '**DELETE /songsWS/rest/songLists/22'**. Löscht die Songliste 22, falls diese 'mmuster' gehört.
Beispiel: '**DELETE /songsWS/rest/songLists/33'** von User 'mmuster' requested, aber 33 gehört eschuler, soll von Ihrem Service mit HTTP-StatusCode 403 (FORBIDDEN) abgewiesen werden.

g) PUT-Funktionalität von Songlisten muss **nicht** implementiert werden.

h) mmuster und eschuler sollen in Ihrer DB jeweils mindestens 2 Songlisten besitzen: mmuster besitzt eine private und eine public und eschuler besitzt eine private und eine public Songliste.

i) Unit-Tests für das DB-DAO für Songs mit HSQLDB