Alex Post

531671

OOSE DEA

27 maart 2020

Spotitube documentatie

Alex Post

Inhoud

[Inleiding 2](#_Toc36043481)

[REST-api 3](#_Toc36043482)

[Database 4](#_Toc36043483)

[Deployment 5](#_Toc36043484)

[Lagen 6](#_Toc36043485)

[Package diagram 6](#_Toc36043486)

# Inleiding

In dit document zal ik de applicatie Spotitube beschrijven. Voor de applicatie is gebruik gemaakt van de door school aangeleverde frontend, te vinden op <http://ci.icaprojecten.nl/spotitube/>.

Ik heb de backend en REST-api geïmplementeerd zodat de frontend naar behoren te gebruiken is.

Dit document zal inzicht geven in de structuur van de backend, de database en de deployment van de backend applicatie. Hierbij zal ik ook toelichten welke keuzes ik heb gemaakt om tot het behaalde resultaat te komen.

De backend van de applicatie is gemaakt met Java EE. Hier bovenop is Apache TomEE Plus gebruikt als Java Application server.

# REST-api

Vanuit school is een lijst met endpoints opgesteld, die moesten geïmplementeerd worden in de backend. Voor deze REST api is gebruik gemaakt van JAX-RS, dat gebruikt wordt om RESTful webservices te maken.

Hieronder volgt een lijst met endpoints die geïmplementeerd zijn:

url: /login

method: POST

url: /playlists

method: GET

query parameter: token

url: /playlists/:id

method: DELETE

query parameter: token

url: /playlists

method: POST

query parameter: token

url: /playlists/:id

method: PUT

query parameter: token

url: /tracks

method: GET

query parameter: forPlaylist

query parameter: token

url: /playlists/:id/tracks

method: GET

query parameter: token

url: /playlists/:id/tracks/:id

method: DELETE

query parameter: token

url: /playlists/:id/tracks

method: POST

query parameter: token

# Database

Spotitube maakt gebruik van een MySQL database, waar de data voor de applicatie in is opgeslagen. De applicatie communiceert met de database door middel van de MySQL JDBC driver. De verbinding met de database is gerealiseerd in de configuratie files van Apache TomEE Plus. Dit voorkomt dat er configuratiezaken van de database in de code terecht komen.

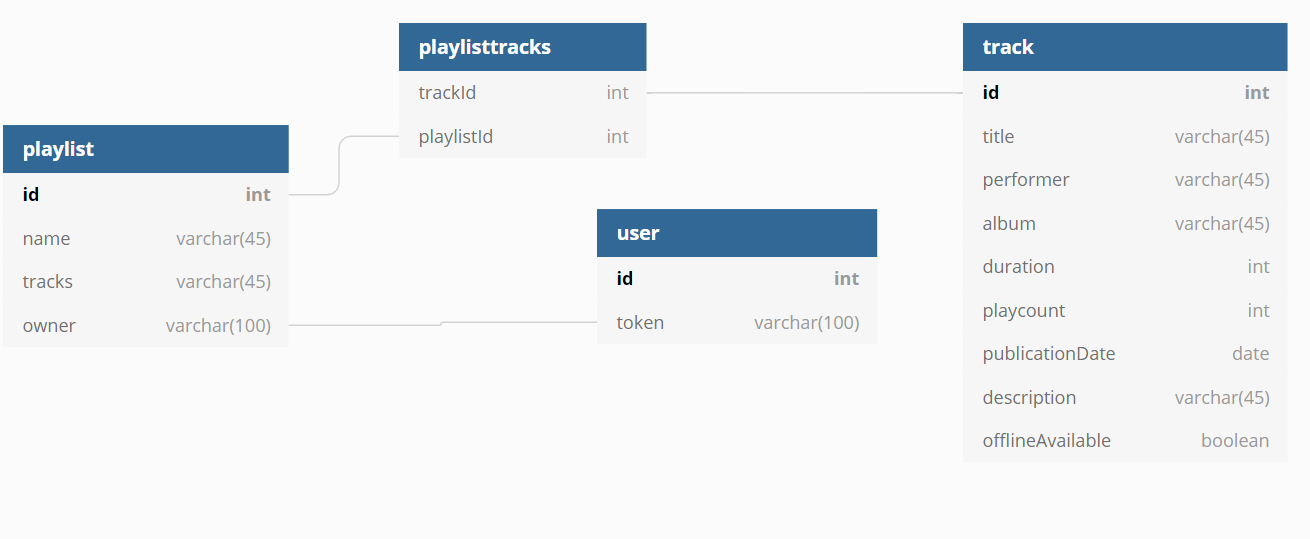
De database bestaat uit vier tabellen: user, playlist, track en playlisttracks.

In de tabel user wordt alles opgeslagen wat met useraccounts te maken heeft, in dit geval is dat een unieke id en de token die gegenereerd wordt bij het aanmaken van een user account.

De tabel ‘playlist’ bestaat uit de id, naam, owner en tracks. Het veld ‘owner’ is gekoppeld aan het veld ‘token’ uit de tabel ‘user’, deze velden zijn aan elkaar gekoppeld door middel van een foreign key.

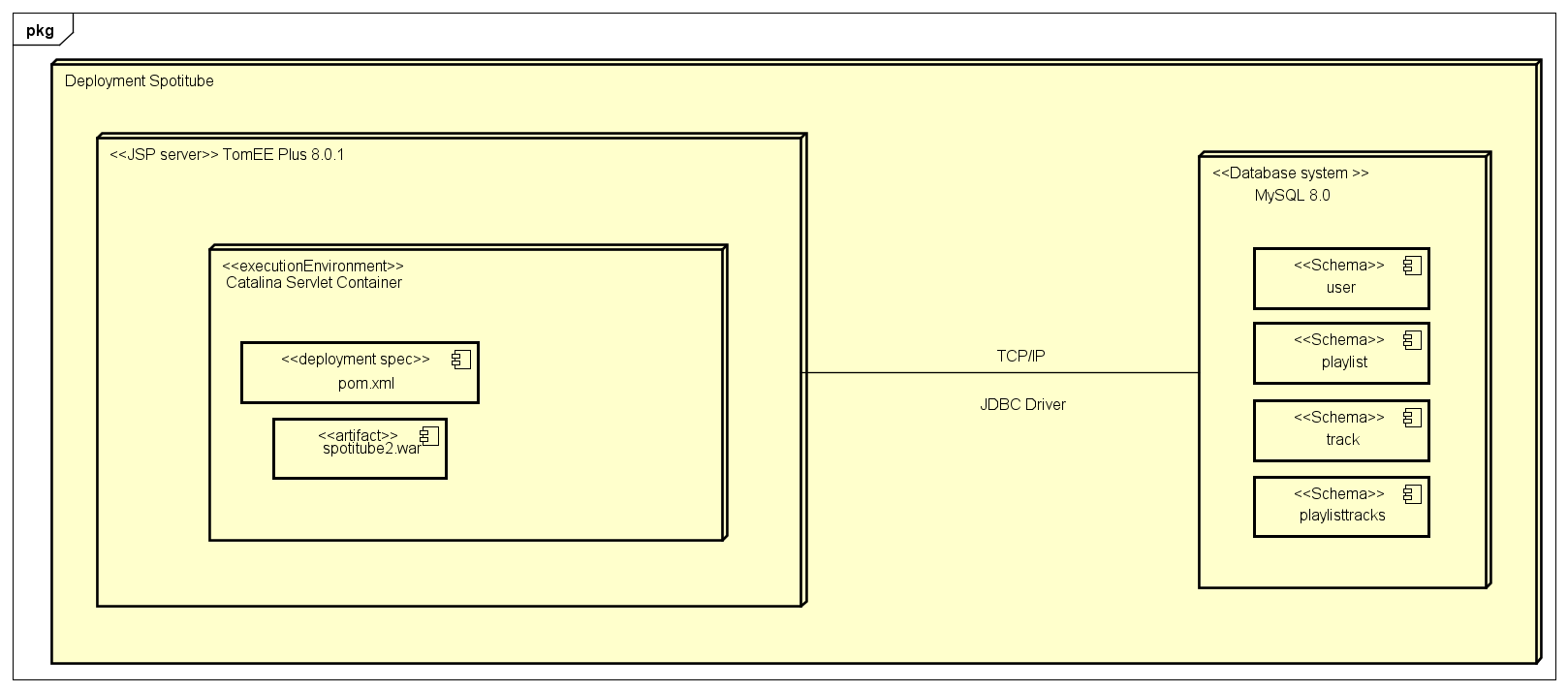
In ‘track’ is alles opgeslagen wat met een specifieke track te maken heeft.

De tabel ‘playlisttracks’ koppelt tracks aan playlists. Dit wordt gedaan door middel van de twee velden, die op hun beurt met foreign keys corresponderen met de id’s van ‘track’ en ‘playlist’. Op deze manier is het mogelijk om bij te houden welke tracks bij welke playlists horen.



Figuur . Database diagram Spotitube

# Deployment

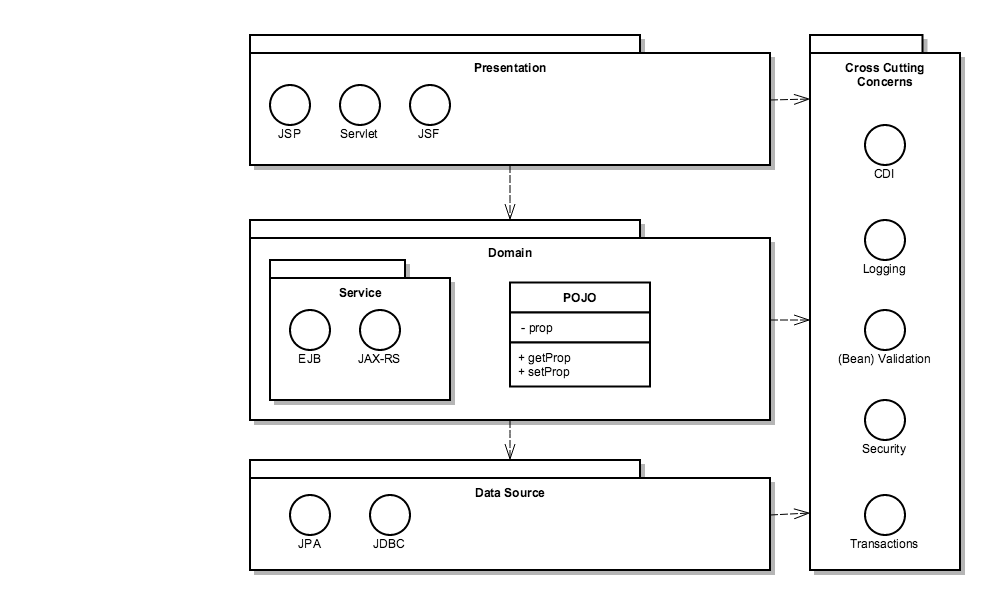


Figuur . Deployment diagram Spotitube

# Lagen

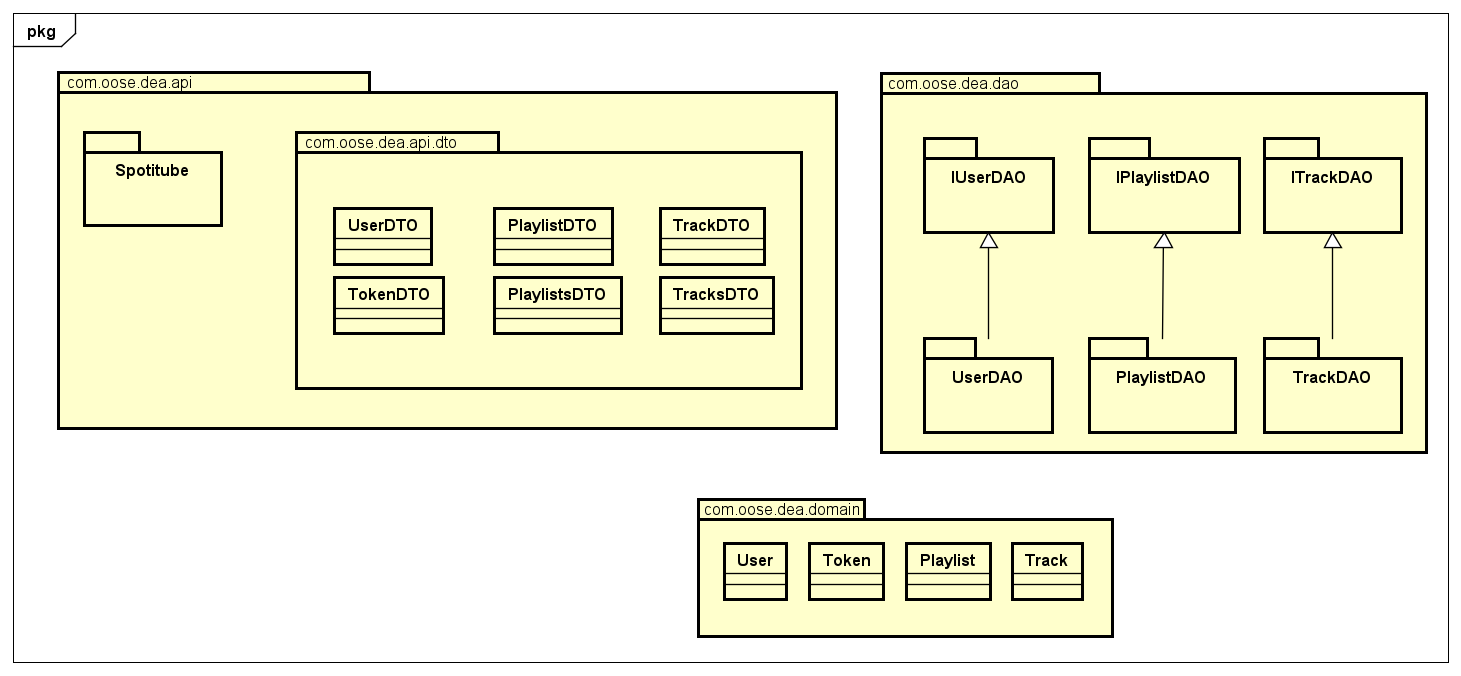
Binnen de Spotitube backend wordt gebruik gemaakt van verschillende lagen die voldoen aan het lagenmodel, zoals we dat geleerd hebben in de course. Een voorbeeld hiervan is de datasourcelaag, waarin alle zaken die met de database te maken hebben afgehandeld worden.

Een ander voorbeeld is de domainlaag, waarin de REST api zich bevindt. Volgens het schema in Figuur 3 mogen lagen alleen communiceren met de laag die zich direct onder de laag bevindt. In het voorbeeld van de domain- en datasourcelaag is dit het geval. De domainlaag(waarin de REST api zich bevindt) is de enige laag die communiceert met de datasourcelaag.



Figuur . Java EE platform architecture

## Package diagram



Figuur . Package diagram Spotitube