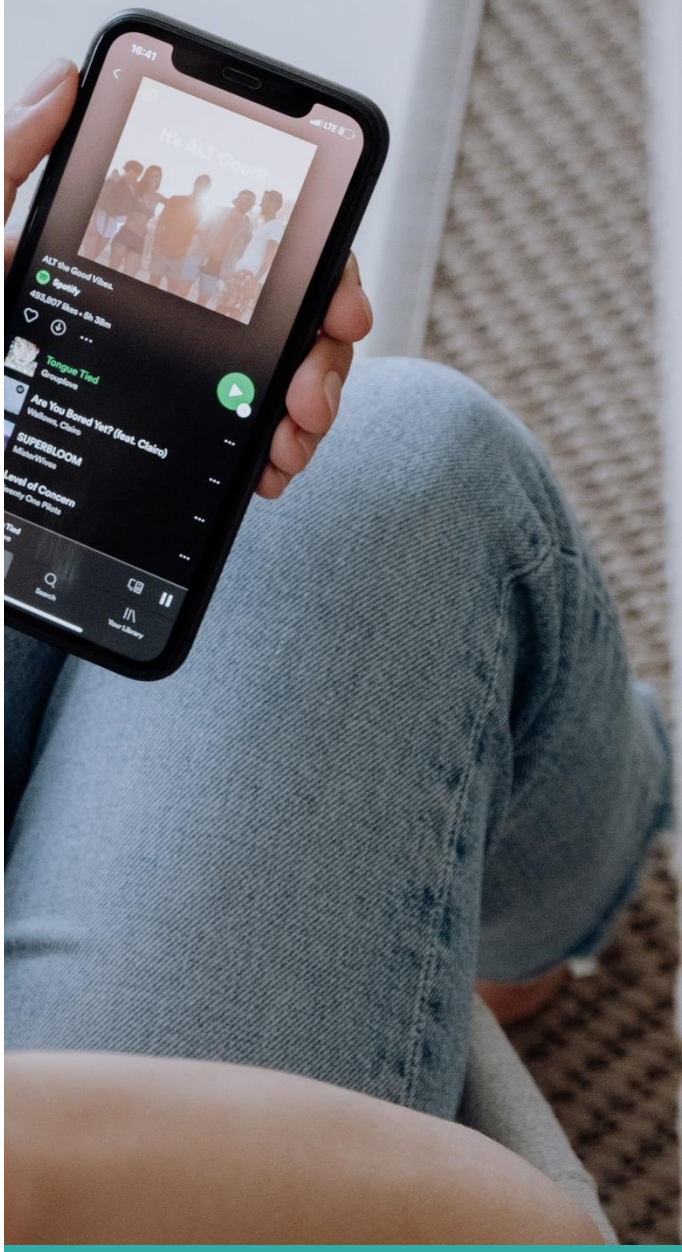
Spotitube



26 MAART

Klas: ITA OOSE-A-f DEA
Versie: 1.0 2020/03/06
Autheur: Robert Boudewijn
Studentnr: 631286
Docent: Dhr. M. Portier

HAN_UNIVERSITY OF APPLIED SCIENCES

Inhoudsopgave

Inleiding

Spotitube is een opdracht van het van DEA in semester OOSE. De bedoeling van de opdracht is om met de vergaarde kennis van de lessen van het vak een JAX-RS javaEE webserver te bouwen. Dit is in thema van een casus. In dit document wordt de struktuur van de applicatie die gebouwd is beschreven en de keuzes die gemaakt zijn tijdens het bouwen van de webserver.

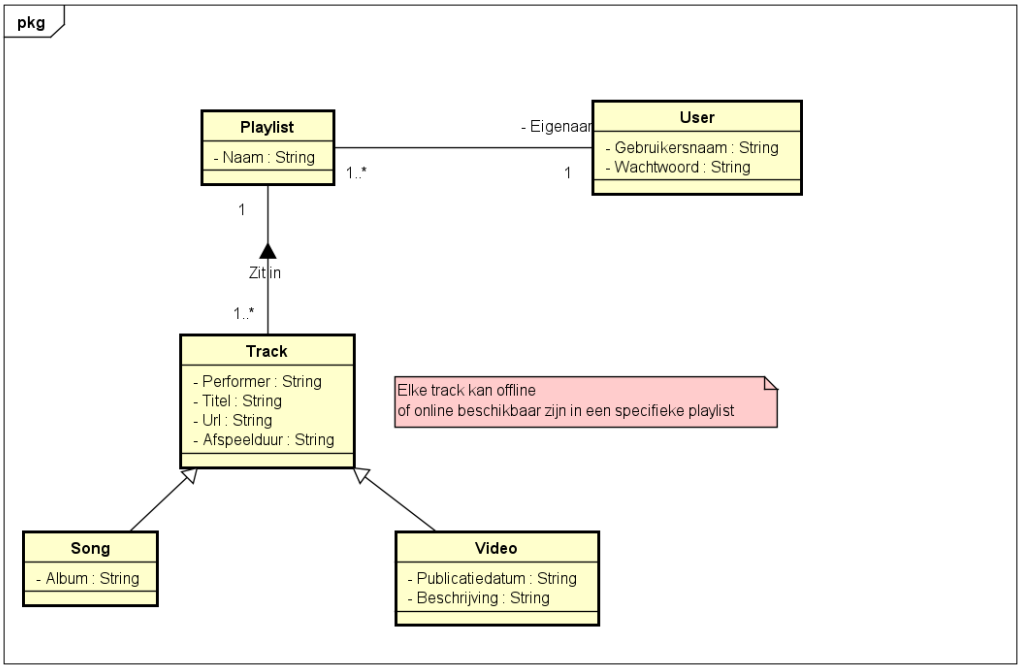
Casusbeschrijving

Spotify en Youtube hebben de handen ineengeslagen en werken gezamenlijk aan een app (Spotitube) waarmee een klant een overzicht kan krijgen van afspeellijsten met daarin audio- en videostreams. Ze willen eerst een deel van de back-end ontwikkelen en deze testen via een bestaande webapplicatie alvorens over te gaan tot de ontwikkeling van de app. (HAN - AIM - OOSE - DEA, 2020)

Een Playlist heeft een naam en een eigenaar, een eigenaar heeft een gebruikersnaam en wachtwoord. Er kunnen 2 soorten tracks in een playlist worden opgeslagen namelijk liedjes (song) en filmpjes (video). Een track heeft een performer, titel, url en afspeelduur. Een song heeft naast alle eigenschappen van een track een album, een video heeft naast alle eigenschappen van een track een publicatiedatum en een beschrijving. Elke track kan offline of online beschikbaar zijn in een specifieke playlist. De applicatie moet meerdere verschillende relationele databases ondersteunen. Het wisselen van database moet mogelijk zijn zonder de applicatie opnieuw te moeten compileren. (HAN - AIM - OOSE - DEA, 2020)

Domein

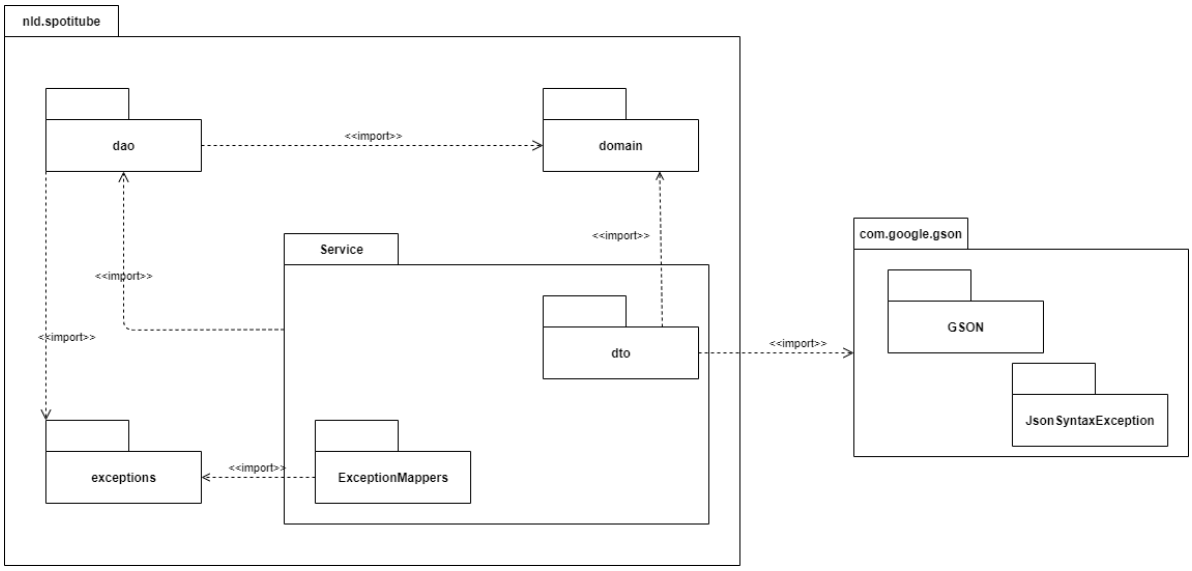
Om het domein gemakkelijker te maken om te begrijpen is er een domeinmodel uitgewerkt van de casus. Zie figuur 1.



Figuur 1; Domeinmodel voor spotitube

1. Package structuur

Van de applicatie is een package diagram gemaakt om zo de verschillende componenten met elkaar te kunnen koppelen. In figuur 2 zie je goed dat dat nld.spotitube.service gebruik maakt van alle packages. Dit komt doordat de service package de plek is waar de http requests binnen komen en deze worden verwerkt en terug worden gestuurd. Daardoor moet deze package weet hebben van andere packages.

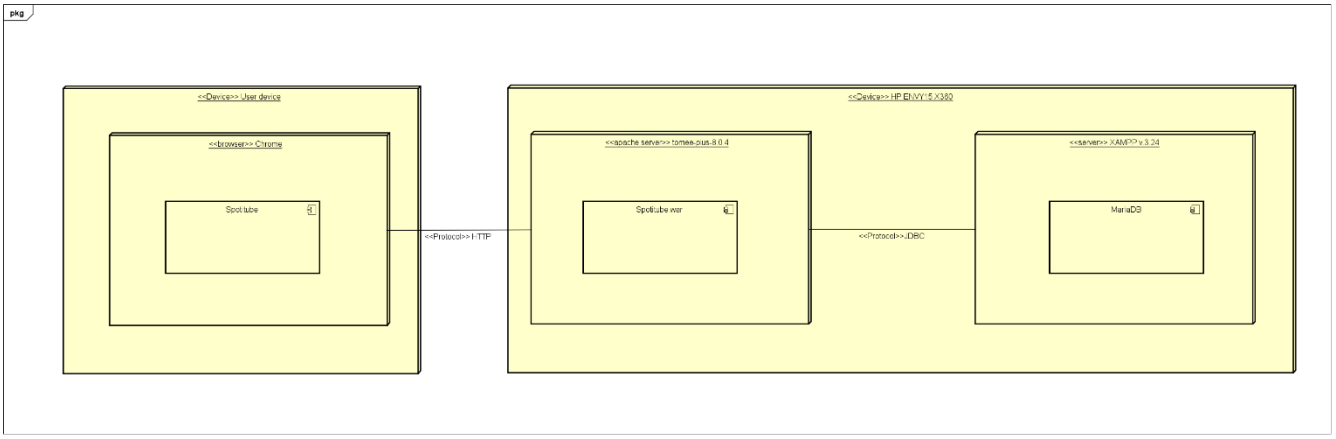


Figuur 2; Package diagram van Spotitube

Echter kan je in figuur 2ook goed zien dat nld.spotitube.dao alleen weet heeft van de packages domain en exceptions. En niet van de service package. Dit toont actief een multitierarchitectuur(ook wel een layered architecture) aan. Http -> Service -> domain -> dao (zie hoofdstuk '2. Deployment omgeving' voor meer informatie). Waarbij in dit geval de domain package gebruikt wordt als communicatie objecten tussen de dao en service package.

2. Deployment omgeving

Er is om de deployment omgeving in kaart te krijgen een UML deployment diagram opgesteld(figuur 3). Zoals je kan zien wordt op dit moment de webserver en de database nog uitgevoerd vanaf de laptop van één van de ontwikkelaars. Dit is voor korte duur prima echter is het zeer verstandig om dit om te zetten naar een andere omgeving. Denk hierbij aan Microsoft's Azure, Google's Firebase of Amazons AWS.



Figuur 3; Deployment diagram van de huidige situatie

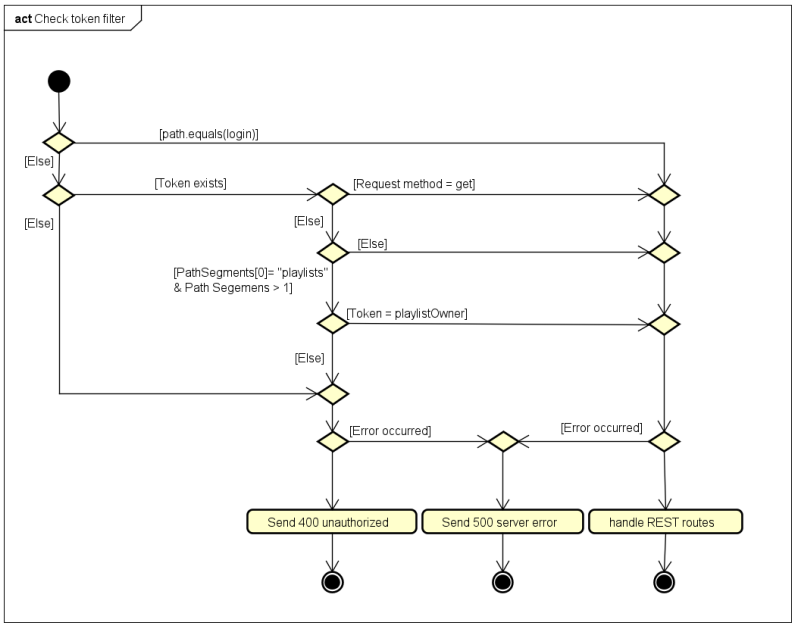
Het is op dit moment duidelijk dat er nog geen extra backup database is toegevoegd. Dit wordt wel aanbevolen en is door het gebruik van een adapter pattern erg simpel. In de package nld.spotitube.dao zijn alle uitgewerkte data transfer objecten vanaf een interface geïmplementeerd. Hierdoor hoef je alleen maar de interface te implementeren en kan je dan snel een database verandering maken door middel van een dependency injection in nld.spotitube.service.

3. Afwegingen

Tijdens het bouwen van de applicatie zijn er een aantal keuzes gemaakt voor het slimmer uitwerken van deze applicatie. Deze worden in dit hoofdstuk toegelicht.

3.1 Gebruik van Filters

Voor het afhandelen van tokens heb ik een filter gebruikt. Dit filter is nogal ingewikkeld te lezen daarom is er een activity diagram opgesteld om semi snel te lezen wat er allemaal gebeurd.



Figuur 4; check token filter

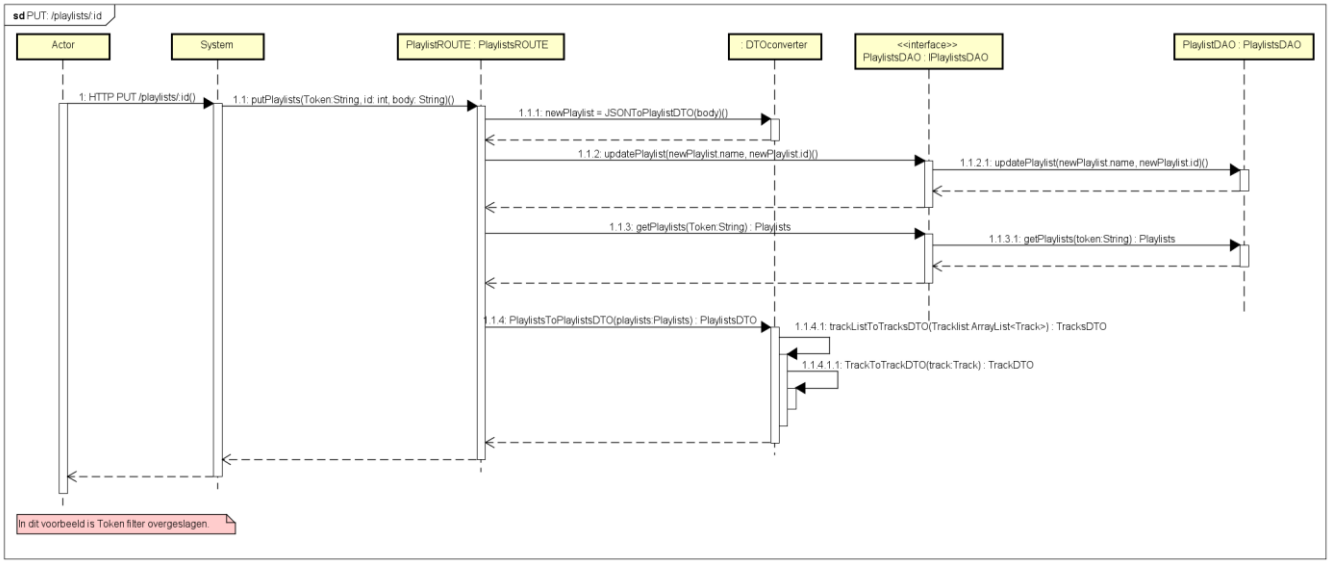
Het handige aan het filter is dat het automatisch checkt of de gebruiker wel mag bepaalde routes mag gebruiken. Daardoor zijn de methodes in package nld.spotitube.service klein gebleven.

3.2 Exception mappers

In de applicatie is gebruik gemaakt van exception mappers. Dit zorgt er voor dat elke exceptions naar boven worden gegeven. Het moment dat er een exception wordt gethrowed zal er automatisch een functie wordt uitgevoerd die een error response terugstuurt. Doordat er allerlei verschillende versies van exceptions zijn gemaakt is er precies geregeld welke response status op welk moment moet worden terug gestuurd.

3.3 Algemene flow van een http request

Er is voor het volgende URI een sequence diagram opgesteld, om zo precies te zien wat de algemene flow van de applicatie is. Zie hiervoor figuur 5.
URI: PUT: /playlists/:id



Figuur 5; sequence diagram van PUT /playlist/:id

In figuur 5 is besloten het token filter niet op te nemen. Voor een gedetailleerdere uitleg van het tokenfilter zie hoofdstuk ‘3.1 Gebruik van Filters’.

3.4 Integratie testen

Er is besloten niet alles van het systeem te unittesten. Het unittesten van een Data Acces Object is namelijk niet acuraat genoeg. Bij een DAO gaat het namelijk voornamelijk om het ophalen van data en het verwerken daarvan. Met een unit test test je dan alleen maar de sql query. De rest mock je. Hierdoor is er besloten om voor de DAO intergratie testen te maken met docker. Hiermee test je direct de connectie met je database, de sql query en het verwerken van de data. Dit zijn dus veel meer coverende tests dan bij een unit test.

Echter heeft intergratie testen ook een nadeel. Het duurt namelijk een stuk langer en je hebt externe dependency’s nodig. Bij Spotitube is er gekozen om een docker container te gebruiken om een test database te verkrijgen. Deze moet je voor de testen starten doormiddel van ‘docker-compose up’ te gebruiken in je terminal in de main directory van het project. Wanneer de test database compleet is opgestart kan je direct beginnen

met testen. Na het testen kan je de container weer afsluiten met ‘docker-compose down’ of met ctrl+C.

Bibliografie

HAN - AIM - OOSE - DEA. (2020). *Programmeer opdracht EAI: Spotitube.*

Gebruikte figuren

HAN_UNIVERSITY
OF APPLIED SCIENCES