Spotitube

By Dani Botland (617442)

datum: 26-02-2024

vak: OOSE-DEA

docent: Bart van der Wal

versie: 1.2

# Inhoud

**Table of Contents**

[Inhoud 2](#__RefHeading___Toc105_810035096)

[Inleiding 3](#__RefHeading___Toc107_810035096)

[Package diagram 4](#__RefHeading___Toc109_810035096)

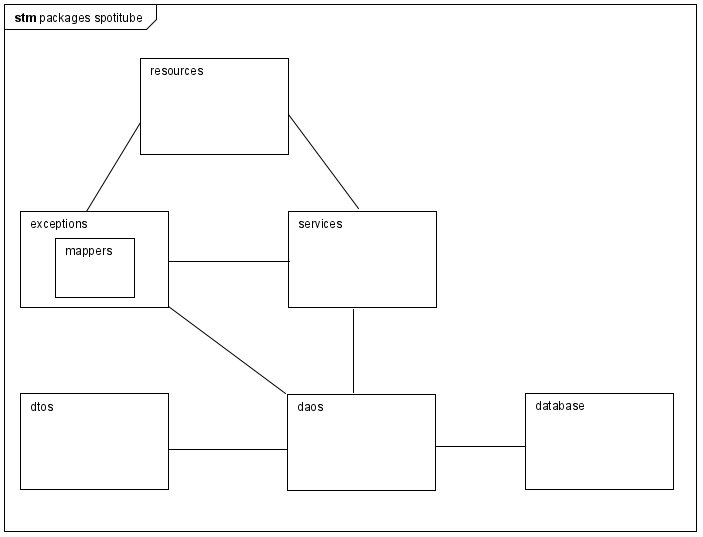
[Deployment diagram 4](#__RefHeading___Toc111_810035096)

[Ontwerpkeuzes 4](#__RefHeading___Toc113_810035096)

# Inleiding

Op de opleiding HBO-ICT hebben wij tijdens het OOSE semester de spotitube opdracht gekregen. Hierbij gaat het om een casus waarbij youtube en spotify de handen in een hebben geslagen en een gezamelijke site hebben opgericht waar je naar muziek kunt luisteren en naar video kunt kijken. In dit document beschrijf ik mijn uitwerking van de casus en geef ik eventuele toelichting waar nodig.

# Package diagram



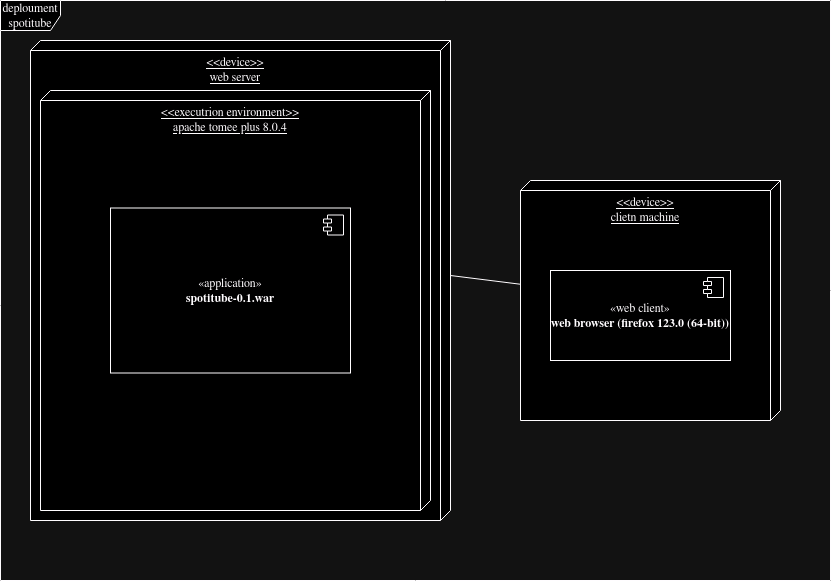
Zoals te zien is in het package diagram hierboven heb ik de applicatie opgedeeld in verschillende packages. Deze packages zijn ook mooi in 3 lagen opgedeeld, je hebt de frontend, de logica en de data laag. De data laag houd zich alleen maar bezig met het bijhouden van data, dit zijn dus de dto’s, dao’s en de database. De dao’s communiceren met de database en zetten de informatie weer om naar dto’s zodat de rest van de applicatie er makkelijker mee kan werken.

De logica laag bestaat uit de services, zij sturen de dao’s aan om de juist data op te halen, in dit geval is er weinig logica nodig en spelen ze deze data dus vooral door naar de resources. Als er iets fout gaat is deze laag er ook veranwoordelijk voor dat de juist exception wordt opgeroepen.

De laatste laag is de voorkant ook wel de resources. Dit is de daadwerkelijke API waarmee de gebruiker in aanraking komt. Alles dat het doet is de request van een gebruiker ontvangen en naar de juiste service doorverwijzen.

Als laatste zijn er nog de exceptions en de exception mappers. Deze hebben als taak om ervoor te zorgen dat niet de hele applicatie crasht wanneer er iets mis gaat. De mappers zorgen er dan voor daat de juist response wordt teruggestuurd.

# Deployment diagram

Figure 1: deployment diagram

Zoals te zien is in het diagram hierboven zijn er twee devices die mee doen aan deze applicatie. Dit zijn de webserver en de client, er is bewust gekozen om geen aparte database server te gebruiken omdat ik deze opdracht meer als POC heb gedeveloped. Om dezelfde reden zit de database dus ook in de .war bij in begrepen.

Verder draaien in dit geval beide devices een arch based linux OS met kernel 6.7.x, maar de client zou ook een ander OS kunnen draaien mits het toegang heeft tot een web browser. De war draait op een apache tomEE plus server versie 8.0.4.

# Ontwerpkeuzes

In dit deel ga ik een aantal ontwerpkeuzes bespreken waarvan ik denk dat ze wat meer aandacht vereisen

## Datamappers

Een keuze die misschien wel opvalt is het gebruik vann datamappers, ik gebruik ze wel voor de exceptions, maar niet voor het retourneren van daadwerkelijke data. Dit komt omdat een DTO voor elke request anders wordt gebruikt en de response vaak ook een andere HTTP code bevat, waardoor een enkele mapper niet voldoende zou zijn, waardoor het in mijn ogen voor te veel boiler plate zou leiden. Exceptions zijn een ander verhaal, want wanneer een exception wordt gegooid, geeft die exception altijd dezelfde HTTP code terug, alleen de message is anders. Dit was makkelijk om te zetten in een enkele mapper voor elke exception waardoor ik niet in de resource classes gedrag moest uitwerken voor de verschillende exceptions.

## Setter Injection

Dit is een opmerkelijke keuze als je door mijn code kijkt, al was de keuze vrij simpel. Voor een aantal classes moest het op deze manier, ik koos voor uniformiteit en heb er dus voor gekozen dit vrijwel overal te doen.

## CORS Filter

Er zijn een aantal opties om de CORS Filter toe te passen, ik heb gekozen voor de variant waar het te regelen via Javax. Hiermee hoefde ik maar een functie te overschrijven uit de ContainerResponseFilter class, waardoor alles op een plek zit. Wanneer ik het gedrag wil veranderen kan ik dit dan in deze class doen te hoeven zoeken.

## Performance consideration

Zoals eerder in het document is benoemd heb ik deze opdracht uitgewerkt als POC, hierom gebruik ik een H2 database als SQL database. Dit betekent dat alle data al binnen de applicatie zit en het uitvoeren van queries in een lus een vrij kleine performance hit geeft. Hierom heb ik op een aantal plaatsen een Query in een lus staan, bij een volledige database zou ik dit waarscheinlijk oplossen met een join of een stored procedure.

# Bronnen