Docentnaam: Uwe van Heesch  
Studentnummer: 621457  
Datum: 27-03-2020  
Klas: OOSE-B-F  
Course: DEA  
Versie: 1.0

opdracht EAI: Spotitube

Bram Verdouw

Inhoud

[1. Introductie 2](#_Toc36198859)

[2. High-level design 3](#_Toc36198860)

[2.1 Package-diagram 3](#_Toc36198861)

[2.2 Ontwerpkeuzes omtrent structuur 4](#_Toc36198862)

[2.2.1 Presentatie-laag 4](#_Toc36198863)

[2.2.2 Domein-laag 4](#_Toc36198864)

[2.2.3 Data-laag 5](#_Toc36198865)

[2.3 Geraakte requirements 6](#_Toc36198866)

[3. Deployment informatie 7](#_Toc36198867)

[3.1 Deployment-diagram 7](#_Toc36198868)

[3.2 Ontwerpkeuzes omtrent deployment 8](#_Toc36198869)

[3.3 Geraakte requirements 8](#_Toc36198870)

[4. Andere ontwerpkeuzes 9](#_Toc36198871)

[4.1 Dependency injection 9](#_Toc36198872)

[5. Conclusie 10](#_Toc36198873)

# 1. Introductie

**Inleiding**

Dit document bevat alle technische informatie over de API die gemaakt is in opdracht van Spotify en Youtube. De API is gemaakt om klanten te voorzien van afspeellijsten met daarin audio en video -streams. Voordat deze API was gerealiseerd, bestond er al een webapplicatie die gemaakt was met Angular. Het doel van deze API is om een deel van de back-end te ontwikkelen en deze te testen met de bestaande webapplicatie.

**Inhoud**

In dit document wordt de high-level design van de API beschreven door middel van een package-diagram. Ook wordt uitgelegd op welke manier de API kan worden gedeployed en hoe het deployment-diagram in elkaar zit. Tot slot worden de overige ontwerpkeuzes benoemd en wordt er een conclusie getrokken uit dit project.

# 2. High-level design

In dit hoofdstuk wordt het package-diagram gepresenteerd. Ook wordt aangegeven welke packages bij welke lagen horen en waarvoor de lagen verantwoordelijk zijn. Daarnaast worden een aantal ontwerpkeuzes omtrent de structuur van de applicatie toegelicht. Tot slot worden de geraakte requirements benoemd.

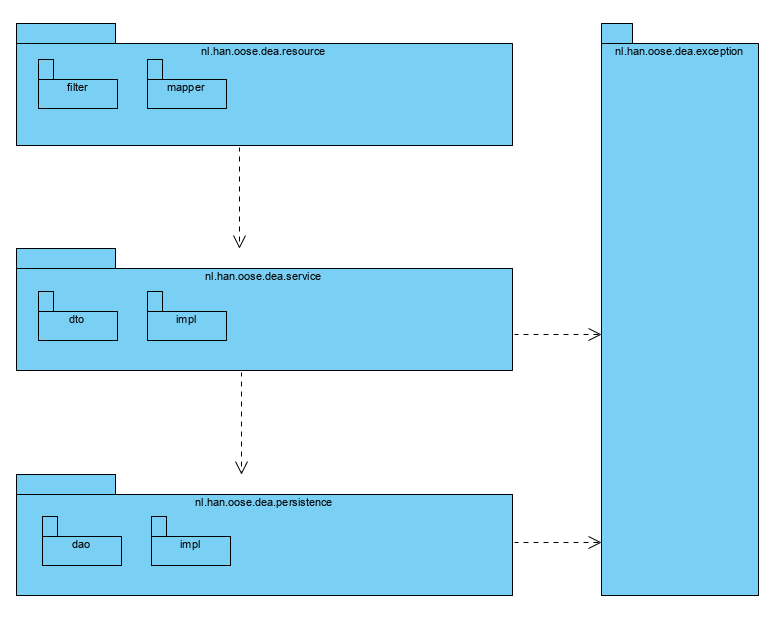
## 2.1 Package-diagram

De API bestaat uit lagen die zijn opgedeeld in verschillende packages. Welke laag bij welke package hoort en de hoofdverantwoordelijkheden per laag staan vermeld in (tabel 2).

|  |  |  |
| --- | --- | --- |
| Laag | Package | Verantwoordelijkheden |
| Presentation | nl.han.oose.dea.resources | * Requests ontvangen; * Responses terugsturen; * Exceptions afhandelen; * Tokens verifiëren; |
| Domain | nl.han.oose.dea.service | * Handelingen verrichten op data; * Exceptions gooien; |
| Data source | nl.han.oose.dea.persistence | * Verbinden met de database; * Data opvragen vanuit de database; |

Tabel 2: package per laag

In (figuur 1) staat het package-diagram. De exceptions vallen buiten de lagenstructuur omdat ze worden gebruikt in de domein-laag en de data-laag.



Figuur 1: package-diagram

## 2.2 Ontwerpkeuzes omtrent structuur

### 2.2.1 Presentatie-laag

**Beschrijving**

Er is gekozen voor een presentatie-laag omdat de API in staat moet zijn om meerdere requests te verwerken en daarbij verschillende responses terug moet kunnen geven. Bij deze laag hoort de package: nl.han.oose.dea.resource.

**Token verificatie**

Per request wordt gecontroleerd of het meegestuurde token correct is. Hiervoor is een class aangemaakt in de filter package. Het alternatief is om per request een methode aan te roepen die het meegestuurde token verifieert. Er is gekozen voor een filter omdat het eenvoudiger is om een filter aan te wijzigen.

**Foutafhandeling**

Deze laag is verantwoordelijk voor het geven van een juiste status code als een fout optreedt tijdens het verwerken van een request. Hiervoor zijn verschillende exception mappers gemaakt. Deze mappers zijn terug te vinden in de mappers package.

### 2.2.2 Domein-laag

De domein-laag verwerkt opdrachten die zijn gegeven vanuit de presentatie-laag. Het is onoverzichtelijk om alle handelingen te verrichten in de presentatie-laag voordat een response wordt teruggestuurd. De domein-laag zorgt een scheiding tussen de presentatie en de data -laag. Het voordeel hiervan is dat handelingen kunnen worden verricht voordat de data wordt opgevraagd en nadat de data beschikbaar is. Bij deze laag hoort de package: nl.han.oose.dea.service.

**Data transfer objects**

De domein-laag gebruikt de data-laag om data op te vragen. Vervolgens verricht de domein-laag handelingen op deze data. Om deze handelingen uit te voeren worden DTO’s gebruikt. De DTO’s zijn terug te vinden in de DTO-package.

**Foutmeldingen**

Als er een fout optreedt tijdens het verrichten van handelingen op de data, is de domein-laag verantwoordelijk voor het gooien van een exception.

### 2.2.3 Data-laag

Er is gekozen voor een data-laag omdat daardoor een scheiding tussen de applicatie en de database ontstaat. Deze laag is verantwoordelijk voor het teruggeven van de juiste data. Bij deze laag hoort de package: nl.han.oose.dea.persistence.

**Data acces objects**

Deze laag is verantwoordelijk voor het terugsturen van de juiste data. Deze data wordt teruggestuurd met behulp van DAO’s (data transfer objects). Deze DAO’s zijn terug te vinden in de DAO-package. Implementaties van deze DAO’s zijn terug te vinden in de IMPL-package.

**Factory pattern**

Om met de database te verbinden is er gebruik gemaakt van het factory pattern. Hierdoor is het mogelijk om in de toekomst een alternatieve implementatie te maken als er bijvoorbeeld een ander type database wordt gebruikt.

**Database authenticatie**

In het bestand src/main/resources/database.properties staan de gegevens waarmee kan worden verbonden met de database. Hiervoor is gekozen omdat het eenvoudig is om de authenticatiegegevens te wijzigen in een los bestand.

**Foutmeldingen**

Deze laag is niet verantwoordelijk voor het gooien van exceptions. Deze laag mag alleen een exception gooien er in een request een verkeerd token wordt meegestuurd.

**Stored procedures**

De applicatie gebruikt stored procedures die van tevoren aangemaakt zijn met SQL. Het voordeel hiervan is dat de applicatie niet aangepast hoeft te worden om een wijziging te maken in de SQL-queries.

**Extra controles met SQL**

Voor veiligheidsredenen wordt er door middel van SQL gecontroleerd of de gebruiker wel toegang heeft tot de opgevraagde data. Dit is een extra veiligheidsmaatregel en de domein-laag blijft hoofdverantwoordelijk voor de beveiliging van de applicatie.

## 2.3 Geraakte requirements

**REST, JSON en JAX-RS**

Clients kunnen requests sturen naar de API via nette URL’s. De API accepteert POST, GET en DELETE -requests met een JSON request body. De API gebruikt JSON om een body mee te geven in de response. De API gebruikt JAX-RS om deze requests te verwerken.

**CDI (Context & Dependency injection)**

Om de lagen van elkaar te scheiden wordt dependency injection gebruikt. JAX-RS is verantwoordelijk voor het injecteren van de juiste dependencies in alle lagen.

**Angular front-end**

De angular front-end is in staat om met de API te communiceren volgens de REST API specificatie.

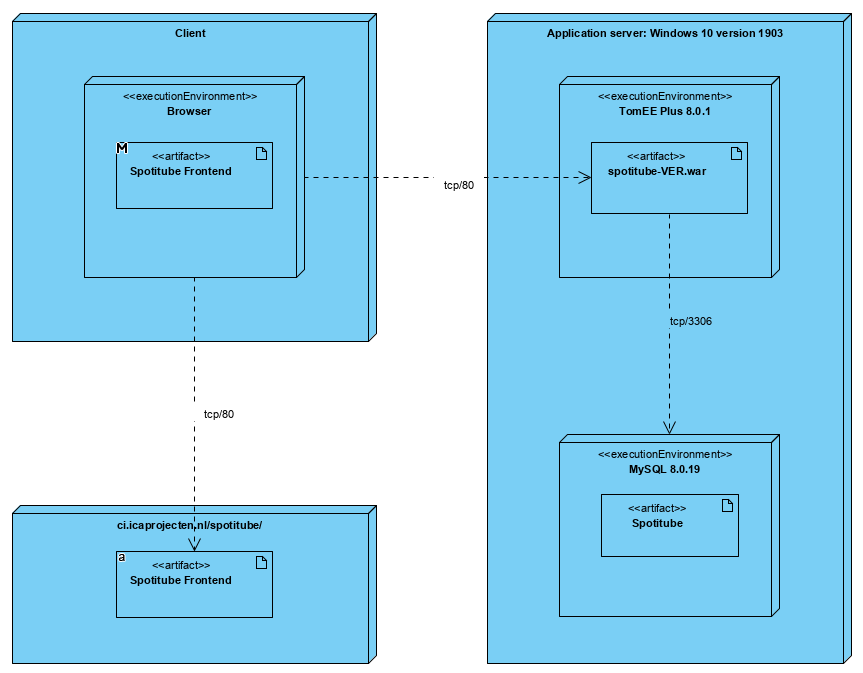
# 3. Deployment informatie

In dit hoofdstuk wordt het deployment-diagram gepresenteerd en beschreven. Vervolgens worden de keuzes omtrent deployment benoemd en toegelicht. Tot slot worden alle geraakte requirements benoemd.

## 3.1 Deployment-diagram

In (figuur 2) staat het deployment-diagram. In dit diagram wordt aangegeven welke servers, protocollen, clients en software wordt gebruikt om de API te deployen. Belangrijke informatie om dit diagram goed te begrijpen:

* Bij artifact ‘spotitube-VER.war’ staat ‘VER’ voor het versienummer dat gebruikt is;
* De browser is een execution environment omdat de Angular-applicatie daarmee wordt uitgevoerd;
* De browser haalt de Angular applicatie op via ci.icaprojecten.nl/spotitube/;
* De artifact ‘Spotitube’ is een database in de MySQL execution environment;



Figuur 2: deployment-diagram

## 3.2 Ontwerpkeuzes omtrent deployment

**Windows 10 version 1903**

Er is gekozen voor het operating system Windows 10, versie 1903. De rede hiervoor is dat de API getest is met dit operating system.

**TomEE Plus 8.0.1**

Er is gekozen voor het TomEE Plus 8.0.1. De rede hiervoor is dat de API getest is met deze versie van TomEE Plus.

**MySQL 8.0.19**

Er is gekozen voor MySQL omdat er een betrouwbare connector aanwezig is voor MySQL waarmee de API kan verbinden met de database.

**De database en API op dezelfde server**

Hiervoor is gekozen omdat de API in deze omgeving ontwikkeld is. Het ook een optie om de MySQL omgeving op een andere server te plaatsen.

## 3.3 Geraakte requirements

**JDBC API**

Een MySQL connector is gebruikt om een verbinding te maken tussen de API en de MySQL-database.

**Apache TomEE Plus**

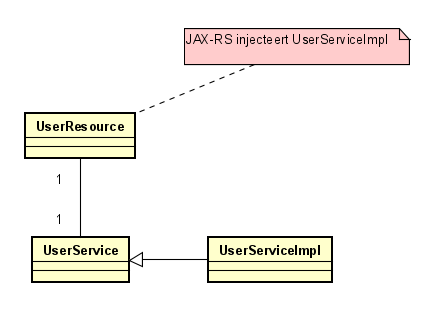
De API kan worden gedeployed op TomEE Plus 8.0.1.

# 4. Andere ontwerpkeuzes

Alle overige ontwerpkeuzes worden in de hoofdstuk behandeld.

## 4.1 Dependency injection

Resources in de presentatie-laag maken gebruik van services uit de domein-laag. Om deze lagen te scheiden, is gebruik gemaakt van dependency injection. Een resource gebruikt een interface om aan te geven welke methodes worden uitgevoerd. Vervolgens wordt door middel van JAX-RS een implementatie geïnjecteerd van deze interface. Daardoor is een resource niet verantwoordelijk voor het aanmaken van een nieuwe implementatie en is de resource niet in staat om de service te wijzigen. Het voordeel hiervan is dat meerdere implementaties kunnen worden gemaakt, zonder de resource aan te passen. Ook is het mogelijk om van implementatie te wisselen zonder de resource aan te passen. In (figuur 3) staat een concreet voorbeeld van dependency injection zoals het is toegepast in de API.



Figuur 3: voorbeeld dependency injection

# 5. Conclusie

**Doorontwikkeling**

De belangrijkste ontwerpkeuzes zijn benoemd in dit document. Ook is, aan de hand van het package-diagram, uitgelegd voor welke functionaliteit een package verantwoordelijk is. Hierdoor is het eenvoudig om wijzigingen op de API door te voeren. Ontwikkelaars kunnen dit document uitbreiden als wijzigingen worden gemaakt aan de API.

**Productie**

Met behulp van het deployment-diagram kan de applicatie worden gedeployed op een server. Daardoor kan de API kan worden getest met de bestaande webapplicatie. Dat de belangrijkste requirement van Youtube en Spotify.