Tim Bastiaans

588928

Versie 3

29-05-2020

Programmeer Opdracht

I-OOSE DEA

Inhoud

[1. Inleiding 2](#_Toc41657611)

[2. Package diagram 3](#_Toc41657612)

[2.1 Toelichting 4](#_Toc41657613)

[2.2 Requirements 5](#_Toc41657614)

[2.3 Design pattern 5](#_Toc41657615)

[3. Deployment diagram 6](#_Toc41657616)

[3.1 Requirements 7](#_Toc41657617)

[3.2 Toelichting 7](#_Toc41657618)

[4. Ontwerp-/Craftsmanship –keuzes 8](#_Toc41657619)

[4.1 Interfaces voor DAO’s en services. 8](#_Toc41657620)

[4.2 Voor iedere end-point een DAO. 8](#_Toc41657621)

[4.3 UnauthorizedUserException 8](#_Toc41657622)

[4.4 DaoException 8](#_Toc41657623)

[4.5 DaoExceptionMapper 8](#_Toc41657624)

[4.6 DatabaseConnectionException 8](#_Toc41657625)

[DatabaseConnectionExceptionMapper 8](#_Toc41657626)

[4.7 GeneralExceptionMapper 8](#_Toc41657627)

[5. Referenties 9](#_Toc41657628)

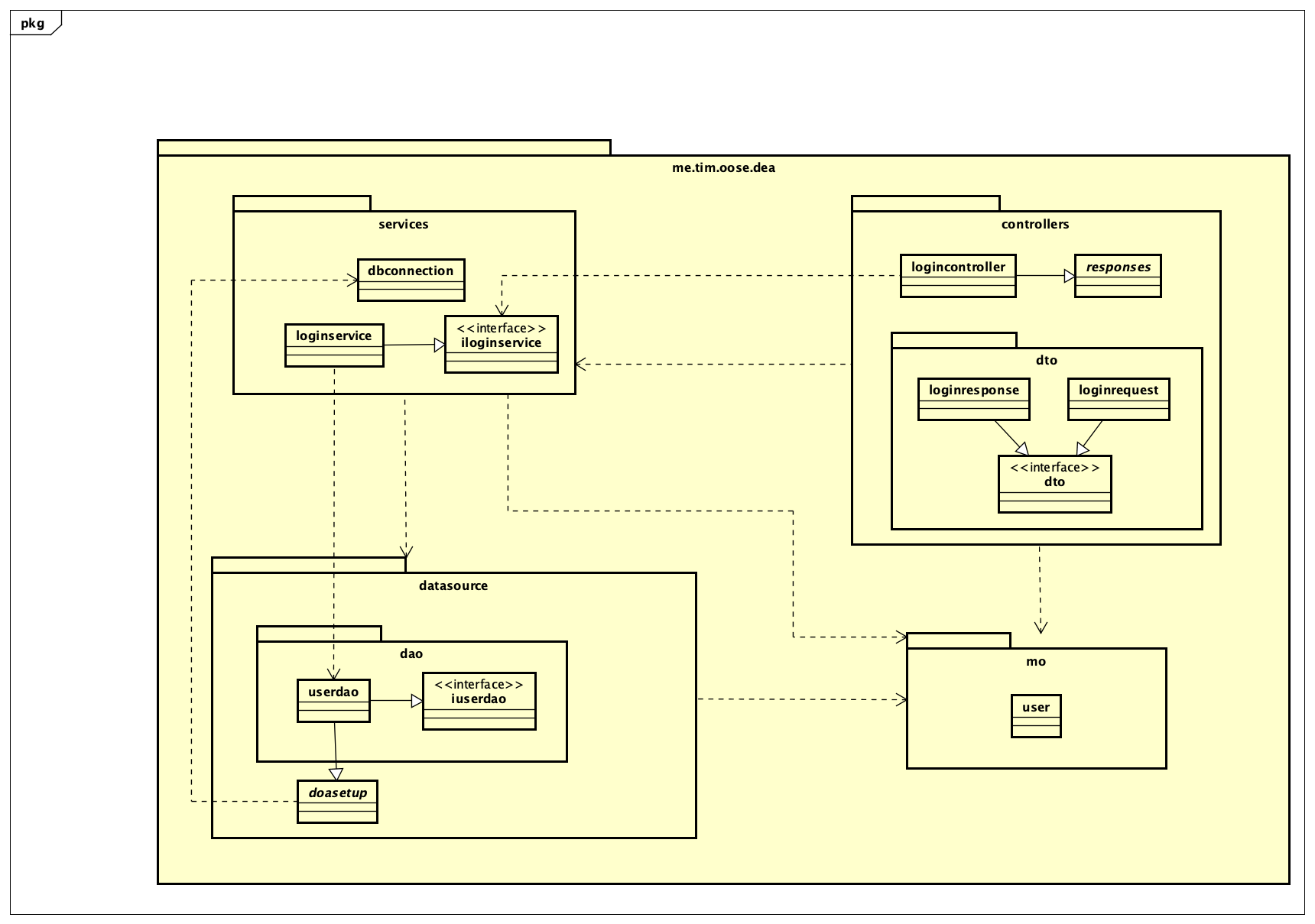
# Inleiding

In dit document worden de nodige onderwerpen behandeld. Er worden twee diagrammen weergegeven en beschreven: Het Package diagram en het Deployment diagram van EAI Spotitube.

Per diagram worden de verwachte requirements en relevante ontwerpbeslissingen uitgelegd. Om mijn denkproces voor deze opdracht beter te begrijpen en te snappen. Als afkortingen niet duidelijk zijn neem de tijd om in mijn toelichtingen tabel te kijken voor verdere uitleg.

# Package diagram

In de afbeelding hieronder is het Package diagram te zien. Om het diagram overzichtelijk te houden is ervoor gekozen om alleen de login end-points erin te verwerken. De end-points playlist en track hebben dezelfde package structuur en worden dus niet weergeven.



Figuur 1 package diagram

## 2.1 Toelichting

|  |  |
| --- | --- |
| Package | Toelichting |
| Services | In de services package staan alle services. De services checken vooral of de goede token is meegegeven en vergelijkt dit soort waardes met de database. |
| Controllers | Hierin worden alle request opgepakt en verwerkt op basis van het Path en de HTTP requests. |
| DTO | In het DTO worden per end-point de requests en responses uitgewerkt zodat deze gebruikt kunnen worden door andere methodes. |
| MO | In het model object staan alle relevante variabele met bijbehorende getters en setters. |
| Datasource | Hierin staat de setup voor de data access objecten. Om verbinding met de database te kunnen maken. |
| DAO | In de data access objecten wordt gecommuniceerd met de database op basis van functies die query’s aanroepen. |

Tabel 1 Toelichting packagediagram

Hierboven een tabel met alle toelichtingen van termen die te vinden zijn in het ‘package diagram’

## 2.2 Requirements

In de onderstaande tabel worden de relevante requirements beschreven.

|  |  |
| --- | --- |
| Requirement | Toelichting |
| As a Developer, I want to present Release Documentation. | Het opgestelde Package diagram zorgt voor een betere en uitgebreidere documentatie van de code. |
| As a Developer, I want to present a correct and complete Package Diagram, so I can give insight in the Application Structure | Het Package diagram is een correcte en complete weergave van mijn implementatie van de Spotitube opdracht. |
| As a Developer I want to add a second layer to my Application, so I can easily decouple and test my code. | In het Package diagram is te zien dat ik gebruik heb gemaakt van het Service Layer pattern. Dit maakt het mogelijk om de applicatie te ontkoppelen en te testen. |

Tabel 2 requirements package diagram

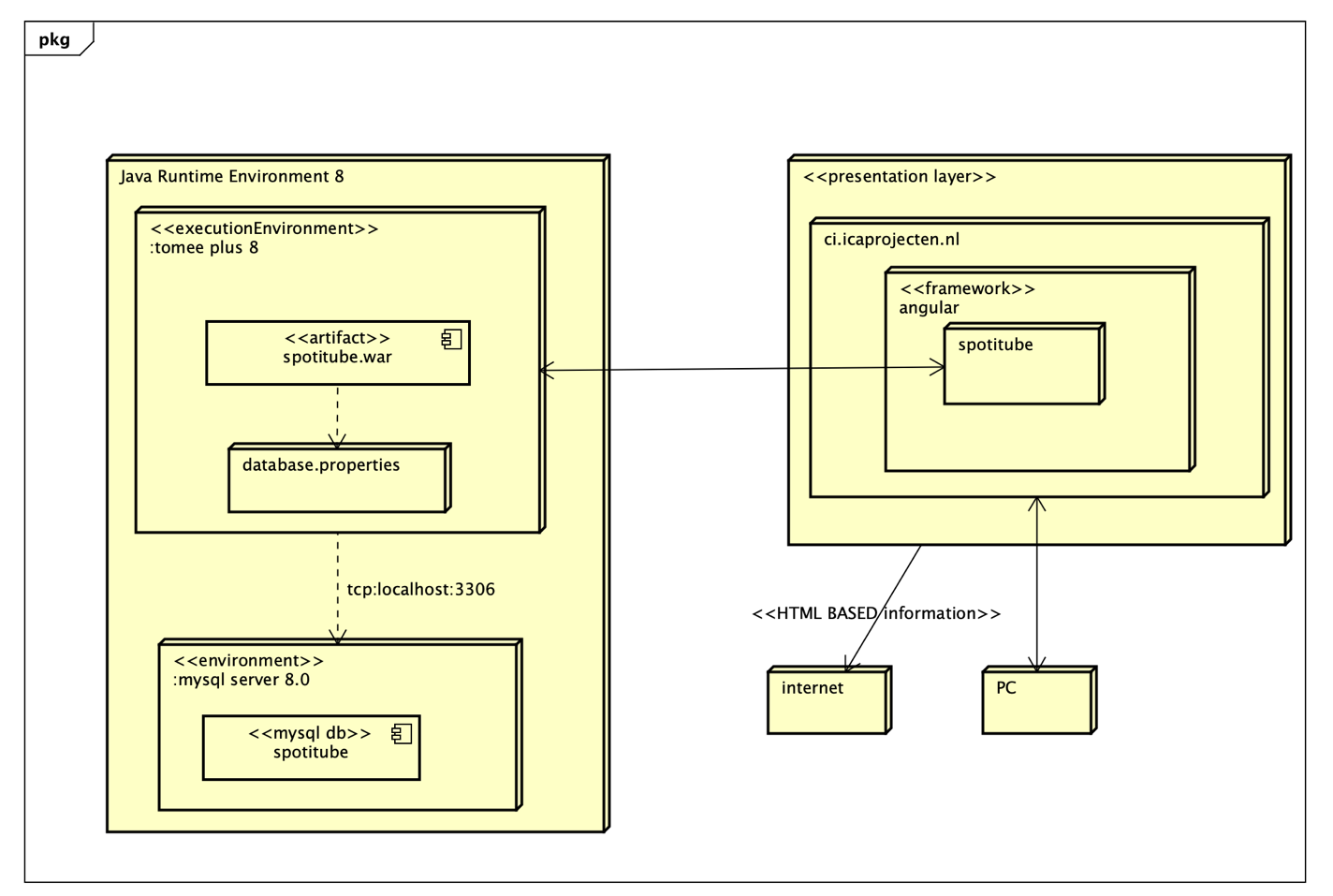
## 2.3 Design pattern

Ik heb ervoor gekozen om de code op te bouwen volgens het service layer pattern. Dit zorgt voor een duidelijke opdeling van de logica over alle services. Ik heb de vijf verschillende lagen verdeelt over packages in de architectuur, maar over het algemeen kunnen de lagen worden geselecteerd op basis van de behoeften van de applicatie. De vijf lagen: de presentatie laag, service laag, bedrijfslogica-laag, persistentie laag en de vijfde laag is de database laag. In deze gelaagde architectuur vervult elke laag een unieke rol en zelfs de componenten binnen deze lagen voeren ook enkele unieke taken uit voor de applicatie.

Hoe communiceren de lagen met elkaar? Het package diagram figuur 1 ‘*package diagram*’ dat in het voorbeeld wordt getoond is een gesloten architectuur. Gesloten geeft aan dat elk verzoek van de applicatielagen komt en dan door de rest van de lagen moet gaan. Ook zorgen deze lagen van de architectuur voor een scheiding van de applicatie. Op basis van het bedrijfsscenario kunnen sommige lagen worden geopend maar er moet wel zorgvuldig mee om worden gegaan. Aangezien dit anders zware prestatie- of beveiligingsrisico's kan opleveren. Ook zitten er voordelen in het programmeren voor deze lagen. Het ontwikkelingen en testen van de lagen is gemakkelijker en ook de implementatie en prestaties versimpelen. Dit maakt het ook een flexibele applicatie binnen de ontwikkelomgeving. Een alternatief hiervoor zou zijn het niet toepassen van een layered pattern wat de snelheid van ’t proces kan versnellen. Dit versnelt het proces alleen als je precies weet wat je doet. Dit is voor mij alleen niet het geval en doe het liever op deze manier.

# Deployment diagram

In de afbeelding hieronder is het figuur 2 ‘*deployment diagram’* te zien. Hoe je de applicatie opgezet wordt is en welke lagen of components met elkaar communiceren en onder zich hebben.



Figuur 2 deployment diagram

## 3.1 Requirements

In de onderstaande tabel worden de relevante requirements beschreven.

|  |  |
| --- | --- |
| Requirement | Toelichting |
| As a Developer, I want to present Release Documentation. | Het opgestelde Deployment diagram zorgt voor betere en uitgebreidere documentatie van de code. |
| As a Developer, I want to present a correct and complete Deployment Diagram, so I can give insight in how the Application should be deployed. | Het onderstaand Deployment Diagram is een correcte en complete weergave van hoe mijn applicatie deployed. |

Tabel 3 requirements deployment diagram

## 3.2 Toelichting

Ik heb gekozen om gebruik te maken van een File Property Pattern. Dit door de properties van de database op te slaan in een file zodat de klassen hierbij kunnen wanneer nodig. Ik had er ook voor kunnen kiezen om de properties hard te coderen. Het probleem is dan dat de properties moeilijker aanpas worden.

# Ontwerp-/Craftsmanship –keuzes

In dit hoofdstuk worden de ontwerpkeuzes beschreven die niet zijn behandeld bij één van de bovenstaande diagrammen.

### 4.1 Interfaces voor DAO’s en services.

Deze interfaces zorgen voor flexibiliteit binnen de klassen van playlist, track en user. De interfaces zorgen ervoor dat de benodigde methodes daadwerkelijk gemaakt worden binnen de klasse. Er kan nu dus bij het aanmaken van de interface al nagedacht worden over de implementatie. Voor elke dao is een daosetup classe die de connectie tussen de database en de dao verzorgd. Ook kun u de kwaliteit van het ontwerp verbeteren door de koppeling tussen de systeemonderdelen te verminderen. Een goede manier om dit te doen, is door de klassen in pakketten te groeperen en de onderlinge afhankelijkheden te beheren. Oftewel Separated interfaces design pattern is toegepast.

### 4.2 Voor iedere end-point een DAO.

Om ervoor te zorgen dat afhankelijkheden goed opgesplitst worden is ervoor gekozen om meerdere data access layers (DAO’s) te maken. Dit zodat niet alles in een DAO terecht komt, want dan zouden alle klassen naar één DAO verwijzen waar ze eigenlijk maar een gedeelte van nodig hebben.

### 4.3 UnauthorizedUserException

Aangezien er een authenticatie systeem aanwezig is wordt er gebruik gemaakt van authenticatie excepties. Ik heb daarom gekozen om zelf een exceptie hiervoor te maken, de UnauthorizedUserException. Dus als er iets mis zou gaan in deze situaties zou het bijna alleen de authorisatie zijn, want anders wordt er gewoon null gereturnt en is er niks aan de hand. Indien dit mis zou gaan heb ik met de UnauthororizedUserExceptionMapper een response gemaakt die een html response stuurt met de code 405. Method not allowed, want als je niet geauthoriseerd bent mag je deze functies niet gebruiken was mijn beredenering.

### 4.4 DaoException

Als er iets fout gaat in de dao’s wil ik niet een algemene SQLExceptie gooien, maar iets waar ik aan kan zien of het nu misgaat in welk stuk van de applicatie en dit terug geven aan het front-end. Ik had dit nog nooit iedereen gedaan, maar ik heb geprobeerd gebruik te maken van een JaxRs exceptiemapper. Die onderschept de gegooide exceptie en geeft aan de hand van de exceptie een html response terug met de juiste response code. Voor als er iets mis zou gaan in de dao dacht ik dat dit met een DaoExceptionMapper een 404 html response moest zijn.

### 4.5 DatabaseConnectionException

Deze exceptie is voor als de database niet connect of de properties niet functioneel zijn in de   
DbConnection service. Hierbij check ik of het bestand aanwezig is en niet leeg is anders deze Exceptie. Met de DatabaseConnectionExceptionMapper dacht ik dat voor deze exceptie een 404 html response aan verbonden zat.

### 4.6 GeneralExceptionMapper

In de bron Huang, M. (2018, 3 december). *Exception Handling in JAX-RS*. Waren twee voorbeelden gebruikt om een hiërarchie te maken voor het afhandelen van de excepties. Ik weet niet of ik dit zou toepassen dus ik heb er één van de twee gebruikt. Deze generalExceptionMapper mapt alle exceptions die geen exceptionmapper hebben en die throwable zijn naar een html response van 404. Ook had er nog één die alle andere foute excepties afhandelde, maar vond dit voor mij niet nodig.

# 5. Referenties

Alexander, A. (z.d.). *A Java JDBC exception example showing try, catch, and finally | alvinalexander.com*. Geraadpleegd op 23 mei 2020, van https://alvinalexander.com/blog/post/jdbc/-decent-example-of-using-try-catch-finally-with-jdbc/

B. (z.d.). *DAO tutorial - the data layer*. Geraadpleegd op 23 maart 2020, van https://balusc.omnifaces.org/2008/07/dao-tutorial-data-layer.html

Huang, M. (2018, 3 december). *Exception Handling in JAX-RS*. Geraadpleegd op 28 mei 2020, van https://mincong.io/2018/12/03/exception-handling-in-jax-rs/

TutorialPoint. (z.d.). *JDBC - Exceptions Handling - Tutorialspoint*. Geraadpleegd op 23 mei 2020, van https://www.tutorialspoint.com/jdbc/jdbc-exceptions.htm