In dit document

In dit document bevat het package diagram en deployment diagram inclusief beschrijvingen bij de diagrammen

Opleverdocument

Opdracht Spotitube

Naam: Joris Huinink

Studentnummer: 597240

Datum: 29 oktober 2018

Klas: OOSE-B

Docent: M. Brouwer

Course: DEA

Inhoudsopgave

[1. Packagediagram 3](#_Toc528572679)

[1.1 Requirements 4](#_Toc528572680)

[1.2 Patterns en ontwerpprincipes 4](#_Toc528572681)

[1.3 Alternatieve oplossing 4](#_Toc528572682)

[1.4 Huidige oplossing 4](#_Toc528572683)

[2. Deployment diagram 5](#_Toc528572684)

[2.1 Requirement 5](#_Toc528572685)

[2.2 Patterns en ontwerpprincipes 5](#_Toc528572686)

[2.3 Alternatieve oplossing 5](#_Toc528572687)

[2.4 Huidige oplossing 5](#_Toc528572688)

# Packagediagram

Hieronder staat het packagediagram corresponderend met de code van de applicatie.

Afbeelding met schermafbeelding

Beschrijving is gegenereerd met hoge betrouwbaarheid

Figuur 1 Package diagram Spotitube

## Requirements

Het package diagram hierboven laat zien dat gebruikt wordt gemaakt van een “Data access layer” in de package “dao”, een “Data Mapper” die alle data mapt op objecten van de package “objects”, een “Service Layer” in de package “services”. Ook laat het package diagram zien de interactie tussen de verschillende lagen doormiddel van dependency pijlen.

## Patterns en ontwerpprincipes

Bij het maken van de applicatie is gebruikt van verschillende lagen. Deze lagen in de applicatie zijn: “Presentation Layer”, “Service Layer pattern” en “Data access Layer pattern”. De “Presentation Layer” vraagt data op uit de “Service layer” en de “Service layer” vraagt de data op uit de “Data access Layer”.

In de Data access layer wordt gebruikt gemaakt van het pattern “Data Mapper” om de opgehaalde data op te slaan in een object. Daarna wordt dit object toegevoegd aan een List van het type van dit object.

Wanneer er geen data meer beschikbaar is uit de database wordt de List teruggegeven aan de Service Layer. Waarna de Service Layer laag klaar is met het ophalen van alle data, wordt de return waarde teruggegeven aan de “Presentation Layer” die de data op het scherm toont via de REST API.

## Alternatieve oplossing

Een alternatieve oplossing voor de indeling van de packages zou zijn om alle RequestDTO’s en ResponseDTO’s apart te houden en de package “track” (die nu een dependency heeft met de package “dao” voor het ophalen van alle tracks) te verplaatsen naar een package die alle klassen bevat die een “@Path” hebben.

## Huidige oplossing

In de huidige situatie heb ik per onderdeel van de applicatie alle bij elkaar horende klassen per laag in één package gehouden zodat de code voor mij overzichtelijker wordt. Bijvoorbeeld alle RequestDTO’s en ResponseDTO’s staan in een package met de bijbehorende naam bijvoorbeeld “Playlist”. Vervolgens word de service layer aangeroepen die als naam heeft een combinatie van de klasse die de service aanroept en het woord “Service” bijvoorbeeld “PlaylistService” en voor de DAO wordt de package naam bijvoorbeeld “PlaylistDAO”. Deze naam conventie is gebruikt om de gehele applicatie te bouwen en is voor mij overzichtelijk.

# Deployment diagram

Hieronder staat het deployment diagram van de gebouwde applicatie.

Afbeelding met schermafbeelding

Beschrijving is gegenereerd met zeer hoge betrouwbaarheid

Figuur 2 Deployment diagram Spotitube

## Alternatieve oplossing

Als alternatief voor de Application container zou er gebruikt kunnen worden gemaakt van een andere server dan JEE 7 Full profile bijvoorbeeld Spring (gebaseerd op J2EE).

Voor de database kan er gebruik worden gemaakt van een andere database Transact-SQL, NoSQL, PostgreSQL, enzovoorts.

Als client kan er gebruik worden gemaakt van het HTTPS protocol. Dit is veiliger dan het nu gebruikte HTTP maar, HTTPS accepteert geen onveilige verbinding en kan er dus ook geen onveilige informatie via HTTPS verstuurd worden.

## Huidige oplossing

In de huidige oplossing is gebruik gemaakt van JEE 7 Full profile omdat dit werd aangeleverd door school. Voor een nieuw project kan ik gaan onderzoeken welke server het beste past bij mijn applicatie.

Voor de database heb ik gekozen om mezelf MySQL aan te leren in plaats van de aangeleerde database protocol, Transact-SQL. Dit heb ik gedaan omdat MySQL een User Interface aanbiedt via XAMPP. Hierdoor is alle opgeslagen data overzichtelijk in te zien.

Er is nu gebruik gemaakt van HTTP boven HTTPS omdat de applicatie is gebouwd in een test omgeving en geen live omgeving. In een test omgeving is HTTP voldoende.