Maarten Lindt

m.lindt@student.han.nl

spotitube

DEA beroepsproduct

Titel : Spotitube

Ondertitel : beroeps product DEA

Auteur : Maarten Lindt 616195

Course: DEA, leerjaar 2019/2020

Docent(en): Meron Brouwer

Plaats: Nijmegen

School: Hogeschool van Arnhem en Nijmegen

Datum: 27 maart 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **WIJZIGINGSHISTORIE** | | | |
| *Datum* | *Versie* | *Beschrijving* | *Auteur* |
| 27 maart 2020 | 1.0 | opleverVersie | LDTM |

Contents

**No table of contents entries found.**

# Voorwoord

Voor u ligt het verslag over Spotitube. Deze casus was erg nuttig en ik heb er veel van geleerd. Maar ik zat ook vaak vast daarom wil ik Meron Brouwer bedanken voor de 1 op 1 hulp via teams. Verder wil ik Meron ook bedanken voor de lessen over hoe je een API moet maken. Zonder deze lessen en hulp was het mij waarschijnlijk niet gelukt. Tenslotte wil ik ook Ruben Eppink en Niels Borkes bedanken voor het helpen als ik weer eens vast zat. Lang leve puntkomma’s en hoofdletters.

Ik wens u veel leesplezier met het lezen van dit verslag over Spotitube.

Maarten Lindt

Markelo,27 maart 2020

# Inleiding

Mijn naam is Maarten Lindt, ik ben een student aan de HAN en kreeg de opdracht om de API en database voor Spotitube te maken. Spotitube lijkt ergens op Spotify. Je kunt playlists aanmaken, deze playlists vullen met tracks en wat je verder allemaal nog wil doen om je muziek op orde te krijgen. In dit document zal ik laten zien hoe ik de APi uitgewerkt heb en dan ook waarom. Als eerst zal verteld worden hoe de structuur van de API eruitziet. Vervolgens zal ik vermelden hoe de API gedeployed moet worden en tot slot zal ik mijn keuzes gaan verantwoorden.

# Package diagram

In dit hoofdstuk zal de structuur van de ontwikkelde API van boven naar beneden uitgelegd worden.

## Package diagram lagen Niveau.

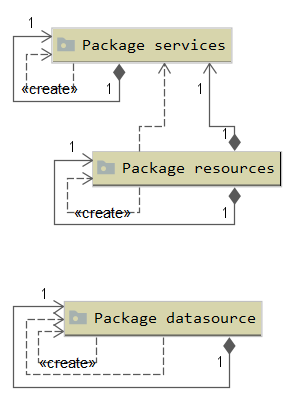


Figure 1: Package diagram op lagen niveau

Zoals hierboven te zien is bestaat de Api uit 3 lagen namelijk:

1. **De resource laag:**  
   Deze laag communiceert met de user interface die door de HAN aangeleverd is.
2. **De services laag:**Deze laag stuurt de juiste opdrachten door naar de juiste ontvanger en handelt enigszins business logica af.
3. **De datasource laag:**Deze laag legt de connectie met de database.

opvallendheden

Hier zullen wat opvallendheden behandelt worden.

Je ziet dat de datasource laag niet verbonden is met de andere lagen. Dit komt doordat er geen harde verbinding tussen zit. De resource laag vraagt namelijk aan interfaces van services om data en hier zit een implementatie in die het doorstuurt naar de datasource laag. Maar zou je een andere database/logbestand zou willen zou dat kunnen als je de interface maar implementeert.

Ook zie je dat ik een service laag heb een geen domein laag. Dit komt doordat het een redelijk domein arme opdracht is hierdoor was het makkelijker om te kiezen voor een servicelaag.

Verder zie je ook dat er tussen lagen geen create statements zitten. Dit komt door dependency injection van javax. Hierdoor zijn er minder afhankelijkheden tussen classes. De injected dependencies kun je zien door de compositie verbindingen.

Tot slot wil ik aandacht richten aan waarom ik gebruik heb gemaakt van deze laag structuur boven een ander(bijvoorbeeld geen servicelaag) is dat ik hierdoor compleet in controle ben. Zie hieronder wat ik bedoel.

Informatie komt binnen 🡪 ik kan het bewerken waar nodig 🡪 verstuur naar Datasourcelaag

Dit zorgt voor een lagere coupling en op zijn beurt zorgt dat weer voor minder aanpassingen als 1 laag aangepast moet worden.

Ook zorgt het voor een hogere bewerkbaarheid. Als ik iets meer logica wil implementeren kan ik dit makkelijk toevoegen aan de service laag.

Als kers op de taart zorgt het er ook voor dat ik een andere interface implementatie kan toevoegen en het door kan sturen naar bijvoorbeeld een logbestand ipv database.

## Package diagram datasourceLaag

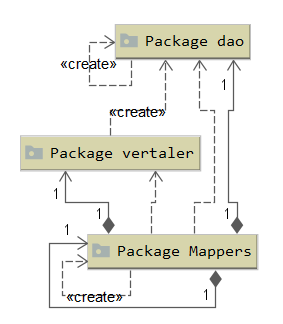


Figure 2: package diagram Datasource laag

Hierboven zie je dat mijn mapper via een vertaler connectie heeft met de DAO. Helaas heb ik dit niet overal zo gedaan vandaar zie je ook een directe dependency naar de DAO. Dit zit tussen de playlistMapper en de playlistDAO. Verder is de naamgeving ook niet het best. De vertaler heet namelijk eigenlijk een Mapper. En Mapper zal ook een andere naam moeten hebben want dit is alleen een tussenlaag tussen de service laag en de DAO waarin nog enkele dingen worden afgehandeld.

Verder voldoet mijn DAO aan een gatewayPattern. Ik heb er namelijk voor gekozen voor iedere call een nieuwe query te schrijven en het niet dynamisch te maken. Dit komt omdat ik veel gebruik maak van subqueries etc.

## Package diagram resource laag

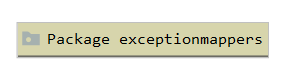


Figure : package diagram resource laag

Hierboven zie je dat de resourcelaag maar 1 package bevat. Dit is de exeptionMapper package. Hierin worden alle exception opgevangen en afgehandeld. Verder bevat de resource laag natuurlijk resources maar ik heb er voor gekozen deze niet in een package te zetten omdat het dan resources.resourses zou worden.

## Package diagram service laag

Services

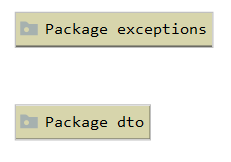


Figure 4: package diagram service lag

Hier zie je een package diagram van de servicelaag. Deze packages hebben geen verbinding. In de service laag zitten namelijk services die de excepties kunnen gooien. En deze services maken gebruik van de dto’s.

# Deployment diagram

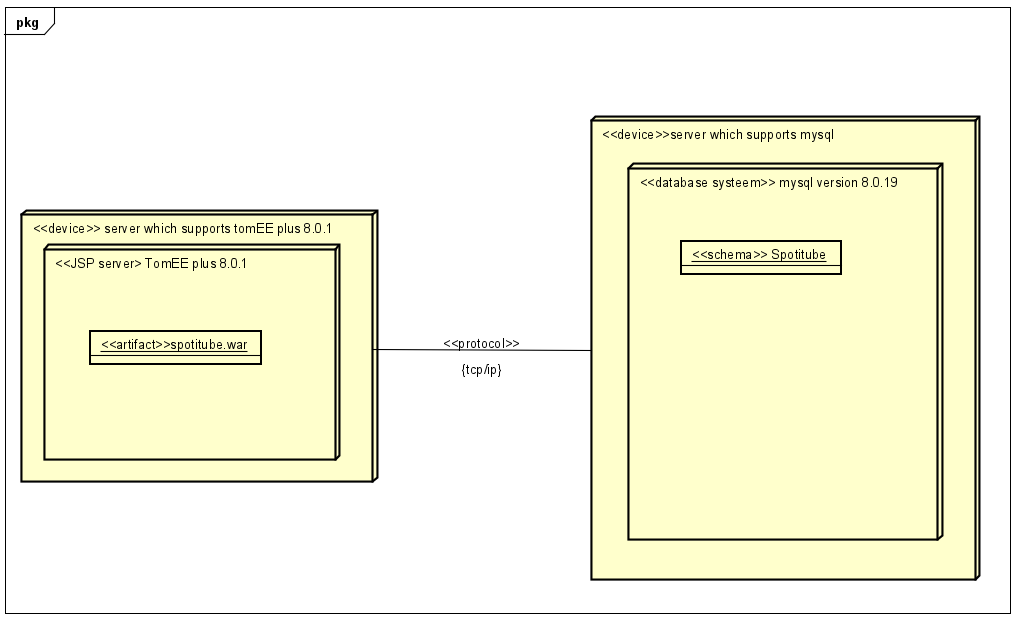


Figure 5: Deployment diagram

In dit diagram zie je dat het systeem gedeployed wordt op 2 servers namelijk een server voor de APi en een server voor de Mysql.

Verder zie je dat in de execution environment van de API server het .war bestand wordt gedeployed. Hier zit de gehele API met benodigde beans.xml en alles in.

In de execution environment van de database server zie je het schema spotitube. Hierin staat de benodigde informatie waar de API bij wilt.

Tot slot zit er een relatie tussen de API en de database. Deze communiceert over het tcp/ip protocol.

# Verantwoording overige keuzes

In dit hoofdstuk geef ik mijn overige keuzes met daarbij de reden. De keuzes zijn:

1. **Niet alles testen.**  
   Ik heb mijn DTO’s niet getest omdat het geen nuttige tests bevat.

Ik heb mijn DAO’s niet getest omdat die lastig te testen is. Hiervoor zul je een integratie test moeten doen, hier moet ik mij nog in verdiepen. Dit is mij helaas niet gelukt binnen dit project.

Hetzelfde geld voor mijn exceptionMappers.

Hierdoor kom ik wel op een lager code coverage. Maar de rest van de lagen wordt wel getest. Dus de functionaliteiten en de paths die het programma doorloopt worden allemaal (op DAO na) getest. Dus weet ik toch dat mijn functionaliteiten werken.

1. **Het niet uitwerken van verschillende functionaliteiten.**ik heb inloggen niet uitgewerkt omdat op de dag voor de opleverdatum mijn hele programma stopte met werken. Ik heb al mijn tests en delen van mijn code opnieuw moeten schrijven. Toen heb ik besloten dat inloggen niet hoog genoeg prioriteit was voor het opleveren omdat ik met de andere functionaliteiten beter mijn competenties kan laten zien. Ditzelfde geld voor het bepalen van de lengte van een playlist en het bepalen van ownership.  
     
   ondanks dat ik niet al deze functionaliteiten heb uitgewerkt vind ik dat ik toch mijn competenties heb bewezen. Ik heb namelijk gewerkt met de vereisten(jaxrs, mockito, etc.). Ook implementeert de aangereikte front-end van de HAN mijn code en werken de meeste functionaliteiten hierop. Alleen ownership zit niet helemaal goed.
2. **Het niet maken van een vertaler voor playlists.**Zoals eerder al verteld ging er vlak voor de opleverdatum iets fout. Hierdoor ben ik ook niet toegekomen aan het extracten van de vertaalslag uit playlistsDAO.
3. **Het maken van een interface tussen data layer en servicelayer**

Ik heb er voor gekozen om geen interface tussen de data layer en service layer te zetten. Dit zodat er in de toekomst ook een ander soort database aan gehangen kan worden.

# Conclusie

Het is jammer dat mijn programma crashte vlak voor de oplevering. Echter heb ik door prioriteren nog steeds een goed eindproduct op kunnen leveren. Verder kun je de API deployen als je je aan het deployment diagram houdt.

# Bronnen

AIM\_ HAN. (z.d.). *OpdrachSpotitube*. Geraadpleegd op 27 maart 2020, van https://onderwijsonline.han.nl/elearning/lesson/XyrGRdmD