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: 28 mei 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **WIJZIGINGSHISTORIE** | | | |
| *Datum* | *Versie* | *Beschrijving* | *Auteur* |
| 27 maart 2020 | 1.0 | opleverVersie | LDTM |
| 18 mei 2020 | 2.0 | opleverVersie | LDTM |

Contents

[1. Voorwoord 1](#_Toc41597139)

[2. Inleiding 2](#_Toc41597140)

[3. Package diagram 3](#_Toc41597141)

[4. Deployment diagram 4](#_Toc41597142)

[5. Verantwoording overige keuzes 5](#_Toc41597143)

[6. Conclusie 6](#_Toc41597144)

[7. Bronnen 7](#_Toc41597145)

# 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,28 mei 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 gedeployd moet worden en tot slot zal ik mijn keuzes gaan verantwoorden.

# Package diagram

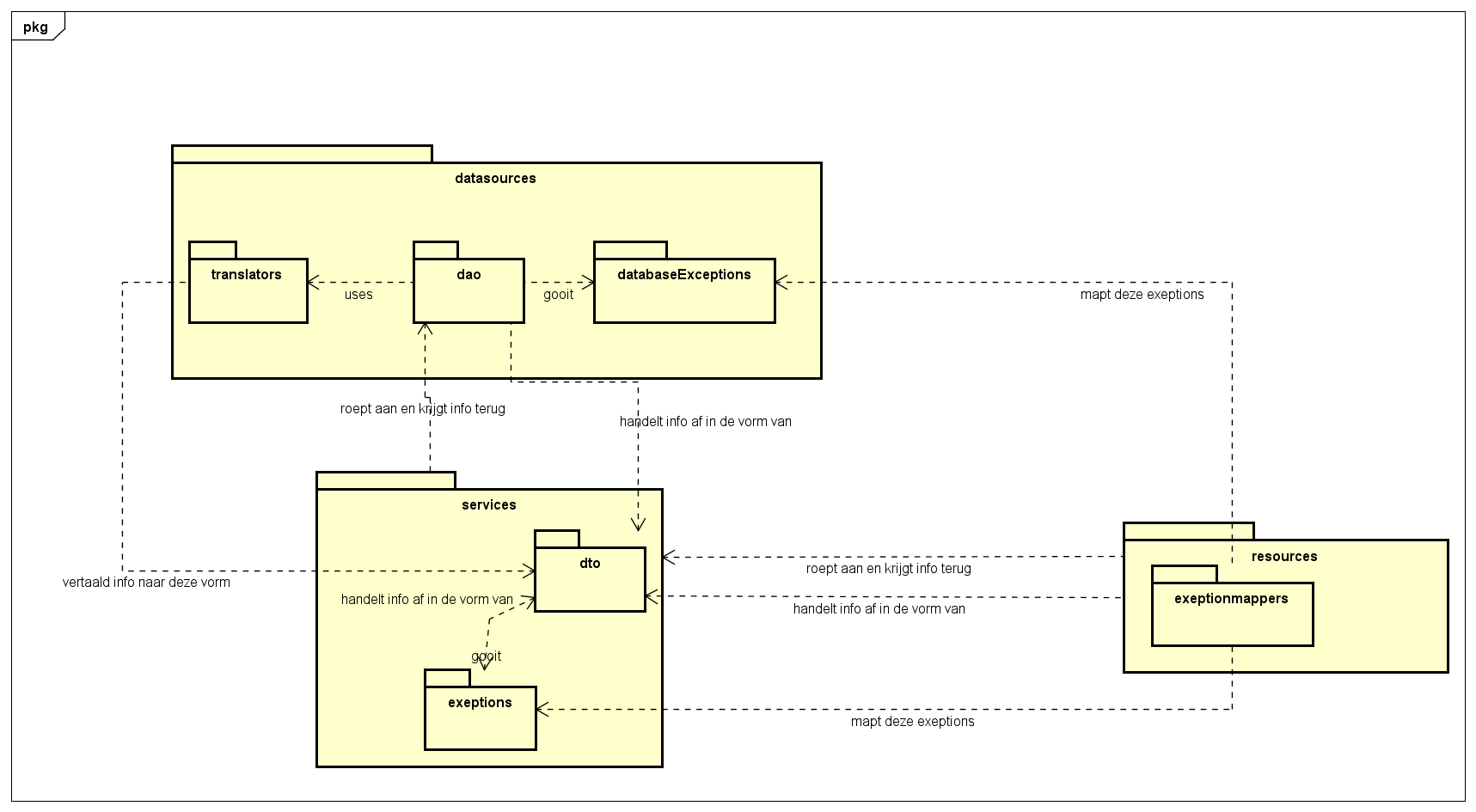


Figure 1: package diagram

In dit diagram zijn de packages van mijn spotitube. Wat hier opvallend is is dat ik veel afhankelijkheden heb. Wel zijn deze afhankelijkheden door imports of dependency injection gedaan. Hierdoor is het makkelijk sommige delen te vervangen zonder dat de code omvalt. Ik kreeg het helaas in astah niet voor elkaar de correcte pijlen te gebruiken dus ik heb besloten om te beschrijven wat de relatie doet. Ik zal nu een paar opvallendheden uitleggen.

* Ik heb 3 hoofdlagen. Dit zijn de resource-, service en datasource laag.
* De informatie wordt over het systeem verspreid door gebruik te maken van data transfer objecten(DTO’s)
* De excepties staan op de lagen waarin ze gegooid worden. Hier heb ik voor gekozen om de excepties makkelijk terug te vinden.
* Mijn mappers heten translators. Dit komt doordat ik in het begin het woord mapper niet goed begreep en translator beter begrijpelijk was voor mij.

# Deployment diagram

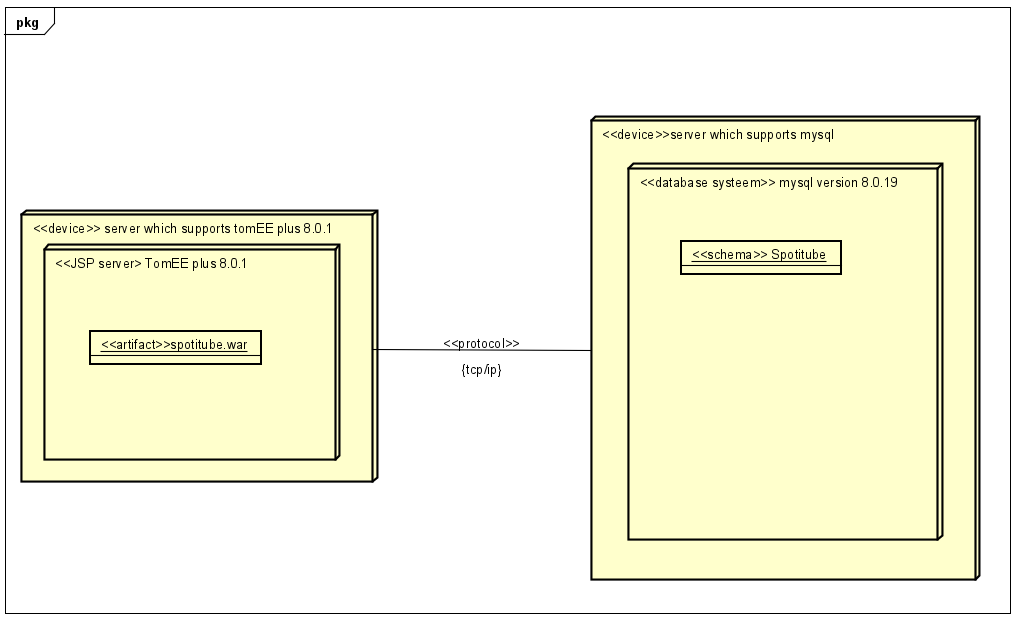


Figure 2: Deployment diagram

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 compleet getest omdat die lastig te testen zijn. 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 en translators

Hierdoor kom ik wel op een lager code coverage. Maar de resources en services zijn wel honderd procent getest. Dus de functionaliteiten en de paden die het programma doorloopt worden allemaal (op DAO na) getest. Dus weet ik toch dat mijn functionaliteiten werken. Ook heb ik met postman en de aangeleverde front end gecontroleerd of de gehele integratie werkt en ik kan gelukkig zeggen dat alle functionaliteiten werken.

1. **Het maken van een interface tussen data- en service laag**

Ik heb er voor gekozen om een interface tussen de data- en service laag te zetten. Dit heb ik besloten met oog op de toekomst. Het is dan namelijk mogelijk om een andere datasource op te zetten met eventueel een ander type database en het dan makkelijk over te zetten door één annotatie te veranderen namelijk de @Default naar @Alternative en andersom.

# Conclusie

Het opgeleverde product is prima maar slecht getest op sommige punten. Hierom zal ik in volgende producten mijn database laag anders moeten schrijven. Ik heb tijdens het OOSE-project gemerkt dat dit beter gaat als ik van tevoren beter nadenk/documenteer. Dit zal ik dus later zeker vaker doen.

# Bronnen

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