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* |
| 26 maart 2020 | 1.0 | opleverVersie | LDTM |

Contents

# Voorwoord

# 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.

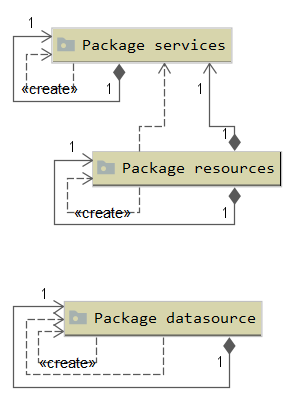


Figure : Package diagram op lagen niveau

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 service laag vraagt namelijk aan interfaces 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.

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.

## Package diagram datasourceLaag

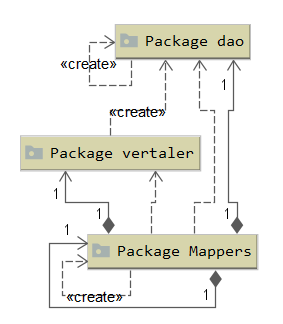


Figure : package diagram Datasource laag

Resources

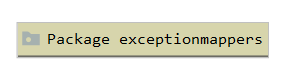


Figure : package diagram resource lag

Hierboven zie je dat mijn Mappers bij alle data kunnen en een vertaler gebruiken om de informatie over te zetten. De naamgeving is 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.

## Package diagram service laag

Services

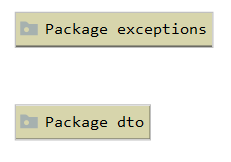


Figure : package diagram service lag

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.

# Conclusie

# Bijlagen

# Bronnen

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