Spotitube Opleverdocument

ALEX CHENG (634967)
ITA-OOSE-A-F

26-03-2021 | DEA (Michel Portier)

Inhoud

1. Casus Beschrijving

Tijdens de course DEA, zullen we een programmeeropdracht maken genaamd Spotitube.

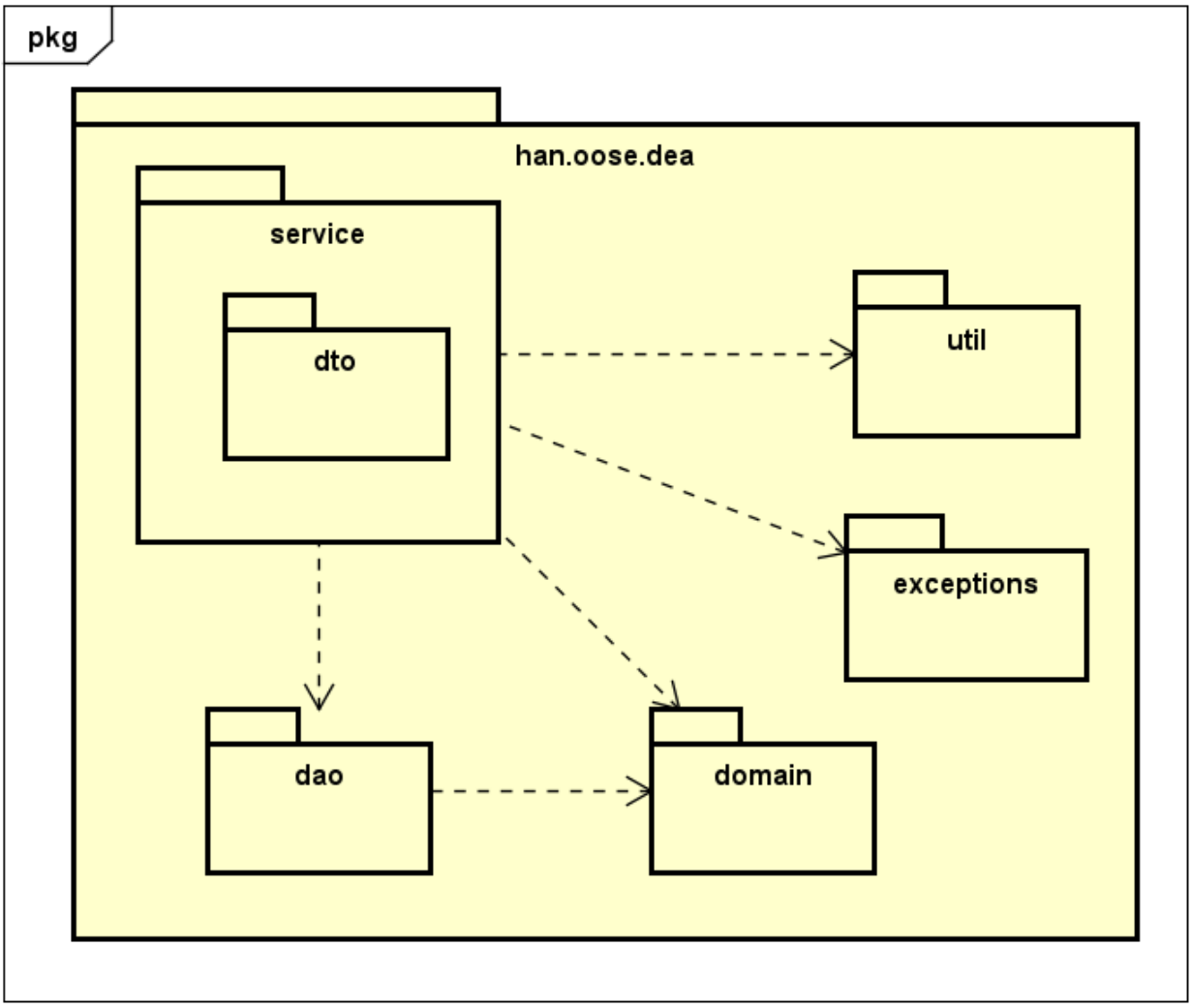
Spotitube is een app dat gedeeltelijk bestaat uit Spotify en YouTube, vandaar de naam Spotitube. Deze app heeft de mogelijkheid voor de klant om een duidelijk overzicht te krijgen van zijn eigen afspeellijsten met daarin audio- en videostreams. Ze willen eerst de back-end ontwikkelen om daarna deze te testen met een bestaande webapplicatie. Daarna kunnen ze kijken na uitgebreidere ontwikkeling van de app.

De Spotitube applicatie maakt gebruikt van de volgende APIs en frameworks:

- JAX-RS v2.0 (REST, JSON)
- CDI (Context & Dependency injection)
- JDBC API

Daarnaast moet de app gedeployed kunnen worden op Apache TomEE plus.
Ten slotte moet de front- en back-end beide Restful kunnen communiceren.

2. Package Diagram



Doormiddel van een package diagram laat ik zien hoe de lagen in mijn applicatie zijn opgebouwd en hoe deze communiceren met elkaar. Deze communicatie wordt mogelijk gemaakt doormiddel van dependency injection. Mijn applicatie bevat 3 lagen: de service laag, dao laag en domain laag.

Service laag:

De service laag bevat alle routes van de applicatie, deze routes zijn allemaal RESTful. Naast de routes bevat deze laag ook alle DTO's. In principe is dit de laag die communiceert tussen de API en client.

DAO laag:

De DAO laag bevat de connectie met de SQL database en stuurt deze terug naar de service laag. Ook communiceert deze met de domain laag om methoden uit deze laag te gebruiken.

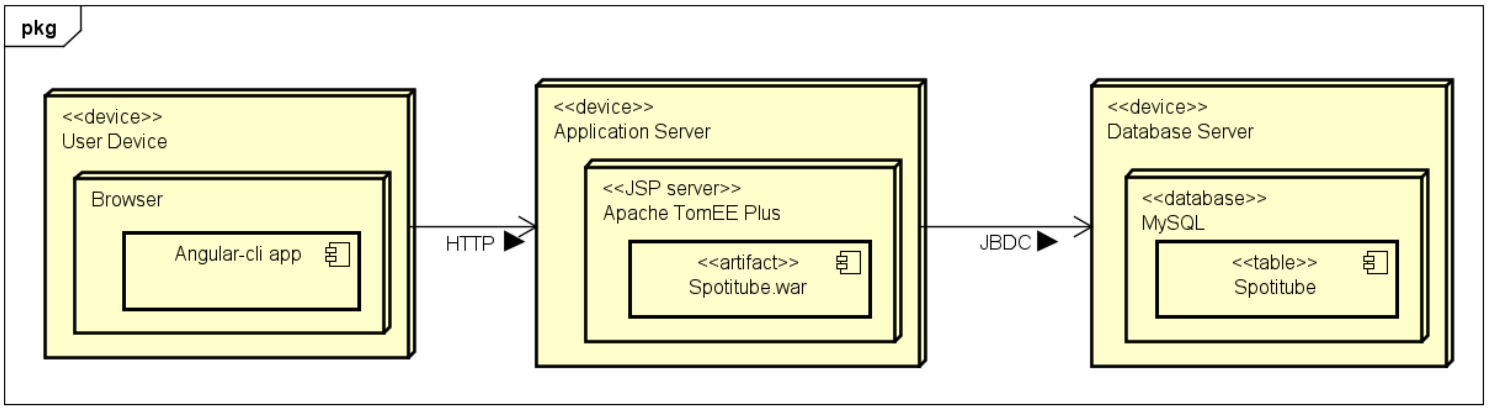
Domain laag:

De domain laag is de plek in de applicatie waar alle data wordt opgeslagen en opgehaald. Dit wordt gedaan doormiddel van getters en setters. Deze methoden worden voornamelijk gebruikt in de DAO's.

Util & Exceptions:

Ten slotte zijn er nog twee overige packages, deze twee packages zijn kleine toevoegingen aan de app. Hier zal meer ingegaan worden bij het kopje 'Ontwerpkeuzes'.

3. Deployment Diagram



Met deze deployment diagram wordt getoond hoe de gehele runtimeomgeving, protocollen, componenten en database in elkaar zit.

De applicatie bestaat namelijk uit 3 devices: user device, application server en database server.

User Device:

Dit device kan gezien worden als de client, deze client zal via zijn of haar browser de Angular applicatie draaien. Oftewel dit is de plek waar de gebruiker de front-end benaderd en HTTP protocollen verstuurt naar de Application Server.

Application Server:

De application server is het belangrijkste device van de drie. Dit komt omdat deze application server eigenlijk de communicatie regelt tussen de user device & database server. Doormiddel van de JSP server, Apache TomEE Plus, wordt de Spotitube.war artifact aangeroepen en hiermee kan deze artifact samen met JBDC communiceren met de database.

Database Server:

De database server bevat de MySQL database met de Spotitube table. Zoals hierboven al vermeld is, wordt de database communicatie geregeld door JDBC en TomEE. Hierdoor hoeft de applicatie alleen nog eenmaal de datasource te injecteren.

Een alternatieve manier van deze oplossing zou geweest kunnen zijn om deze gehele database connectie niet met TomEE te doen, maar direct in de applicatie. De reden dat ik heb besloten om toch TomEE te gebruiken was voornamelijk dat het stukken code scheelt en dus gelijk ook testen. Daarnaast is dit de manier hoe wij het in de course geleerd hebben en vandaar is deze manier voor mij betrouwbaarder.

4. Ontwerpkeuzes

Hierin zal ik kort beschrijven waarom ik bepaalde ontwerpkeuzes heb gemaakt.

Exceptions:

De applicatie verstuurt verschillende status codes bij het teruggeven van de Response. Twee van deze status code zijn 401 & 403. Hiervoor heb ik besloten om twee zelfgemaakte exceptions te maken, dit is om de op de front-end deze HTTP status codes te verduidelijken.

Foutmelding 401 geeft nu terug: “Unauthorized: Invalid username/password”
Foutmelding 403 geeft nu terug: “Forbidden: Invalid token “

Deze zelfgemaakte exceptions zijn niet nodig, maar ik vind ze toch erg gepast voor deze applicatie en daarom heb ik deze geïmplementeerd.

Util:

Util bevat een simpele token generator, dit wordt mogelijk gemaakt met de UUID class. Hierdoor kun je een unieke token genereren en deze gemaakte token wordt daarna toegevoegd in de database bij de juiste gebruiker.

Ook deze UUID class was niet per se nodig, maar doormiddel van deze class was het genereren van een willekeurige unieke waarde gewoon efficiënter en dus makkelijker. Daarnaast werd deze class sterk aangeraden door de Spotitube cases zelf.

5. Conclusie

Met behulp van dit opleverdocument is er hopelijk een beter beeld gecreëerd hoe de Spotitube applicatie in elkaar zit en met elkaar communiceert. Ook is hiermee duidelijk gemaakt hoe deze applicatie ingesteld kan worden op een eigen omgeving.

Dit was het opleverdocument voor de Spotitube programmeeropdracht voor de course DEA (2021).