OPLEVERDOCUMENT SPOTITUBE

Naam: Sjaak Kok

Studentnummer: 620581

Klas: ITA-OOSE-A-f

Docent: Michel Portier

Datum: 26-03-2021

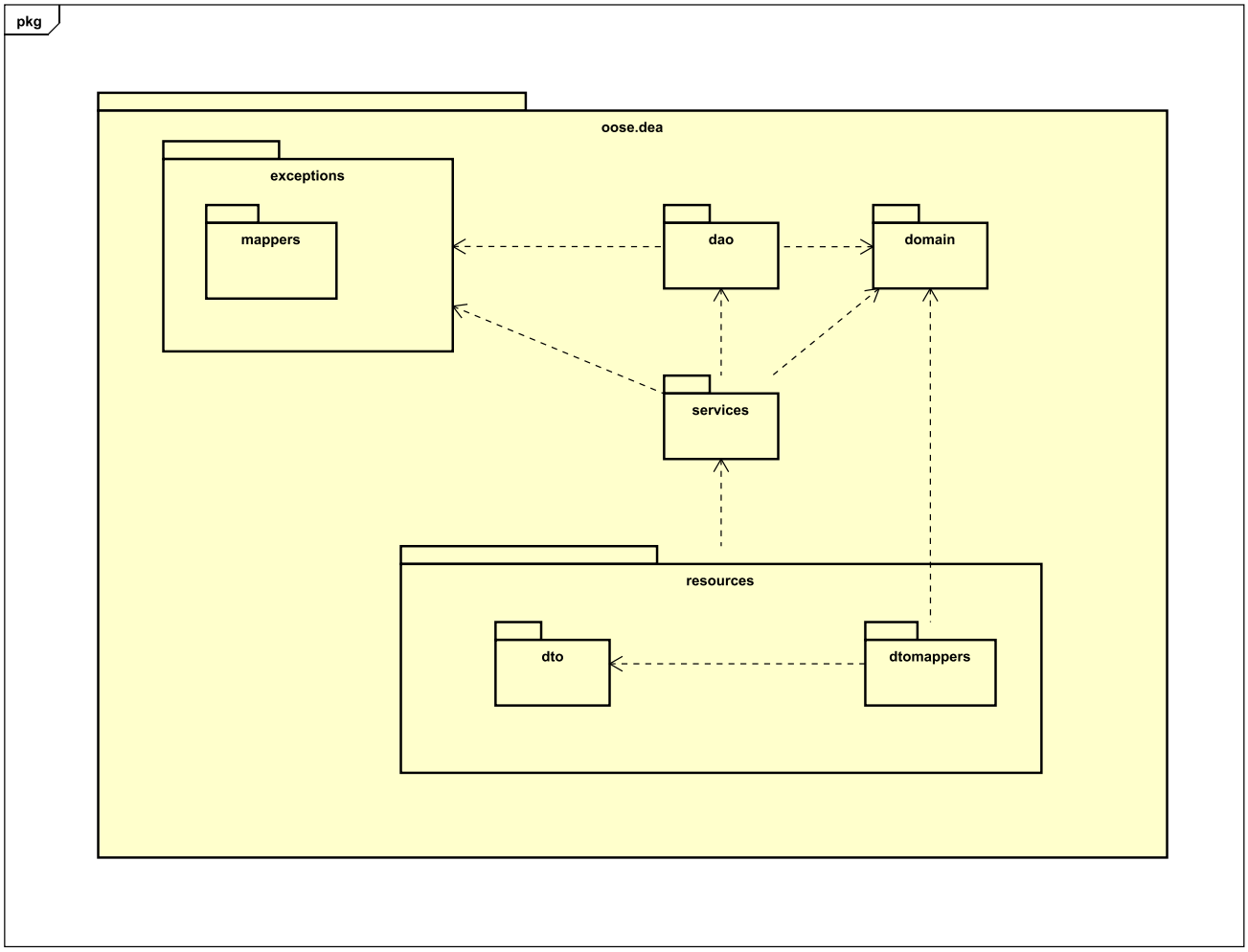
Course: DEA

INHOUDSOPGAVE

INLEIDING

Dit document is het opleverdocument van de Spotitube opdracht. De opdracht is om een back-end te ontwikkelen en deze te testen via een bestaande webapplicatie. De front-end en back-end moeten Restful kunnen communiceren volgens de REST API specificatie (HAN, sd).

PACKAGE DIAGRAM

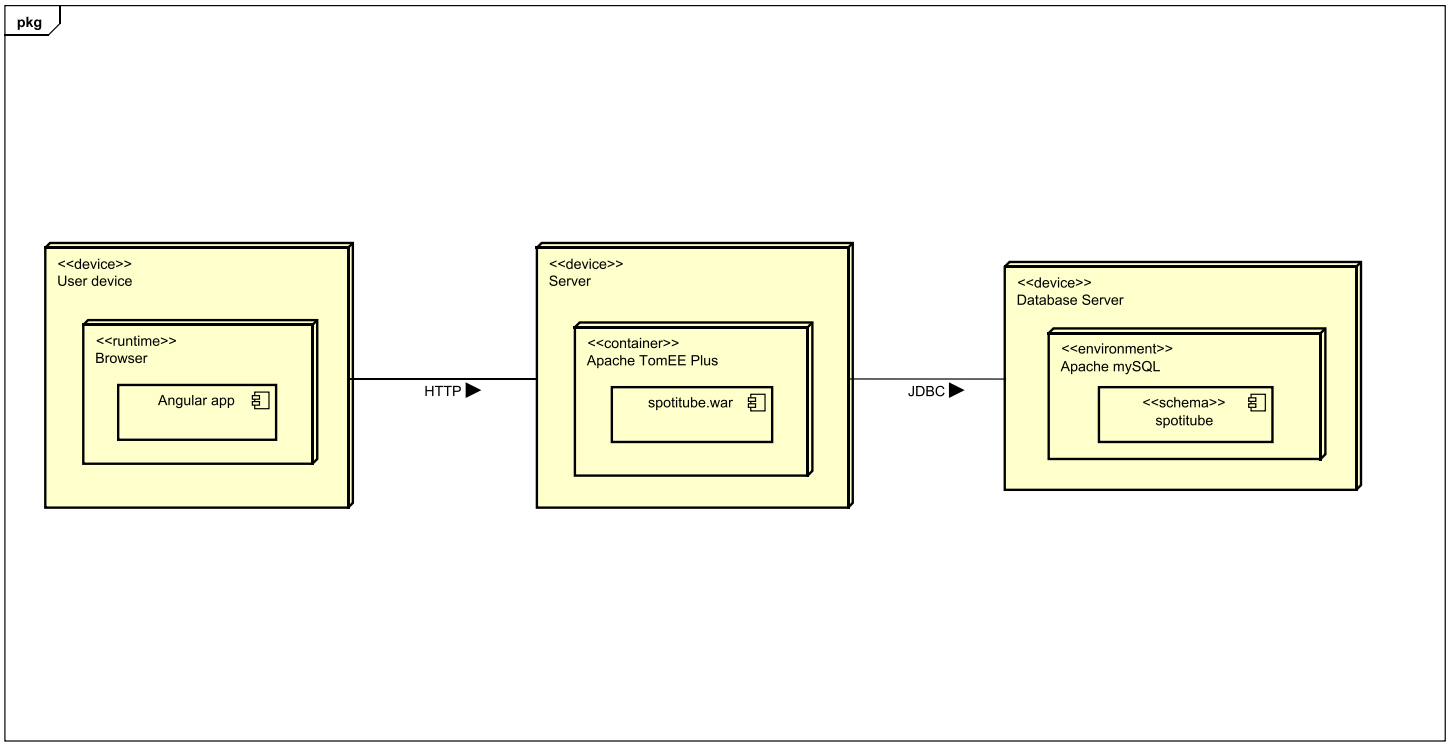


Figuur 1: Package Diagram

Hierboven staat het package diagram. Hierin komen de afhankelijkheden tussen packages naar voren.

In de afbeelding komen de Remote Facade Pattern, Service Layer Pattern en de Data Access Layer goed naar voren. In de resources package komen de requests binnen. De resource interacteert vervolgens met de service laag. De service laag spreekt vervolgens de DAO laag aan. Een DAO kan verschillende implementaties van een interface hebben, dus daar is ook voldoen aan de Seperated Interface Pattern. De DAO haalt vervolgens informatie op uit de database. Zo krijgt de resource laag een domain object terug. In de resources laag wordt het domain object vervolgens omgezet naar een Data Transfer Object. Hiermee wordt voldaan aan de eis om verschillende lagen te implementeren.

DEPLOYMENT DIAGRAM



Figuur 2: Deployment Diagram

Hierboven staat het deployment diagram. Hierin komt duidelijk naar voren dat er drie devices zijn: de user device, de server en de database server.

De spotitube client, die gemaakt is in Angular, draait in een browser. De client communiceert door middel van het HTTP-protocol met de server waarin spotitube.war draait. De server communiceert vervolgens via JDBC met de MySQL database. Het communiceren met de database via JDBC was één van de eisen van de opdracht.

De MySQL database had ook een andere database kunnen zijn, maar ik heb voor een MySQL database gekozen, omdat ik het werken hiermee iets makkelijker vind.

ONTWERPKEUZES

DTOMAPPER

In de resource laag heb ik een klasse gemaakt genaamd MapToDTO die methodes heeft om objecten uit het domein om te zetten naar een DTO. Dit heb ik gedaan, omdat de DAO laag en de services laag geen kennis hoeven te hebben van DTO's aangezien deze alleen maar worden gebruikt om JSON van de client te krijgen en JSON naar de client terug te sturen. Op deze manier worden de afhankelijkheden tussen lagen verkleind.

SERVICE LAAG

Ik heb er voor gekozen om een services laag te implementeren, omdat er anders dingen gebeuren in de resource laag die ik niet bij de resource laag vond passen. In de klasse LoginService wordt nu bijvoorbeeld gecontroleerd of een wachtwoord correct is en in de klasse TokenService is er een methode om een token te genereren. Ook interacteert elke service klasse met maar één DAO. Op deze manier worden afhankelijkheden gescheiden. Als deze laag er niet was moest bijvoorbeeld de LoginController met de UserDAO én de TokenDAO interacteren.

VERWIJZINGEN

HAN. (sd). *spotitube*. Opgehaald van GitHub: https://github.com/HANICA-DEA/spotitube