Duaal Belastingdienst Software Engineering

Leerlijn Backend Programming

INTEGRALE EINDOPDRACHT

Verantwoordingsdocument

Vincent Marcin Seremak

Hoofddocent: Robin Bakels

INLEIDING

In dit document worden de keuzes die ik heb gemaakt bij het ontwikkelen van mijn webapplicatie.
Ik ga ze een voor een uitleggen, met de afwegingen die bij mij opkwamen, hopelijk om een beter beeld te geven in mijn gedachten gang. Ook komt er gedeelte waar ik uitbreidingen zal bespreken en ook limitaties blootstel.

1. TECHNISCHE KEUZES

1.1 MySQL database

De reden waarom ik een MySQL database gebruikt, is omdat ik al eerder ervaring heb opgedaan met de omgeving, hierdoor kon ik dus makkelijker van start, Verder vind ik PostgreSQL zelf niet fijn werken, en MariaDB is wellicht wel sneller (op bepaalde punten) en relatief hetzelfde, aangezien het een fork van MySQL is). Ik vond zelf de grafische omgeving van MySQL Workbench zeer fijn.

1.2 Thymeleaf

Aangezien ik een front-end heb gebouwd had ik een template renderer nodig, aangezien zodra het op een server draait, de pad locaties en database objecten variëren laat ik thymeleaf dit regelen.
Waarom ik Thymeleaf koos over JSP, is omdat Thymeleaf code veel op HTML lijkt, it vond ik fijner om in te werken, ook is Thymeleaf, zover ik weet, meer gebouwd voor springboot dan JSP die meer op Java EE is gefocused.

1.3 Rollen tabel in plaats van ENUM

Ik heb meerdere malen met Enums gewerkt in vorige jaren en vond dit niet al te fijn, om dat gepaard met nieuwe technologieën te oefenen vond ik iets te risicovol. Ook vind ik het zelf overzichtelijker om het via de database te doen.

1.4 Geen aparte DELETE and PUT requests, maar samen in POST

Niet elke framework en browsers ondersteunen DELETE en PUT, verder kan ik via HTML forms alleen POST en GET actions doen, vandaar dat deze keuze eigenlijk al gemaakt is voor mij.

1.5 Waarom de scheiding tussen student/app/admin controller

Ik hou de taken van de controllers gescheiden, dit zorgt voor overzichtelijker code, als er iets mis gaat aan de studenten kant, dan weet ik snel waar ik moet zijn, ook heb ik ze verder in regions gedaan door comment regions aan te maken, hierdoor kan je snel oriënteren op waar je moet zijn, voor een kleinere applicatie is dit een kleiner probleem, maar geen excuus om het niet te doen.

1.6 CommandLineRunner

Aangezien het handig zou zijn als de database al gevuld is wanneer we het applicatie gaan testen, gebruik ik een CommandLineRunner om deze te vullen, hierdoor kan er snel worden getest en hoeft er niet elke keer een nieuwe gebruiker handmatig worden aangemaakt, ook zorgt dit ervoor dat als de laatste Admin account per ongeluk wordt verwijdert, dat die weer terugkomt.

1.7 Waarom een builder in student en niet in user

Ik vind een builder wel fijn, maar op sommige plekken heb ik liever het meer uitgebreide manier van typen, omdat ik het duidelijker vind lezen, wellicht omdat ik dan zeker weet dat ik het aan het goede object meegeef, maar het kan ook komen omdat ik het hele builder principe nog nieuw vind en ik ben best koppig. Ik heb het in de student verwerkt, om te tonen dat ik het wel kan.

1.8 Waarom geen lombok

In de jaren dat ik heb geprogrammeerd heb ik dit niet gehad, dus wederom beetje koppigheid, ik weet wel hoe het werkt, maar ik zou het in dit geval graag zelf willen typen, dan weet ik precies dat het klopt en dat het bestaat, verder is het gemak tussen een code regio maken en weg klikken, nadat je de getters/setters door intellij laat maken ook niet groot. Als een team lombok wilt gebruiken sluit ik mij daar gewoon bij aan.

1.9 Waarom een front-end

Ik vind de uitdaging leuk om een front-end en back-end te combineren in een, dit is ook iets wat later vaker voorkomt, dus wou ik mijzelf alvast verdiepen, verder vind ik het ook veel leuker om naar te kijken dan naar bijvoorbeeld Postman, overigens maakt dit het testen naar mijn mening om makkelijker, want ik kan visueel zien dat het klopt of niet. Ook kan ik deels van mijn fouthandeling negeren, omdat ik het via HTML kan doen. Dit scheelt weer wat werk.

1.10 waarom een int voor mijn primary keys ipv een long

Het formaat van de applicatie is zodanig klein, dat ik geen extra geheugen wil innemen door het een long te maken, Het zal denk niet voorkomen dat er opeens meer als 2147483647 gebruikers of lessen komen.

2. LIMITATIES

Mijn wonderbaarlijke applicatie is natuurlijk niet een alles kunner, er zijn vanzelfsprekend gebreken deels door scope creep en deels door gebrek aan kennis. De huidige gebreken zal ik een lijst hieronder zetten met een prioriteit die ik er bij vind passen.

Gebrek	Prioriteit
Er is geen mogelijkheid om van gebruikersnaam te veranderen.	**Laag**
Een docent kan niet zien hoeveel leerlingen er naar zijn klas willen komen.	**Laag**
Het doosje in de midden is niet altijd groot genoeg voor de inhoud, ook kan deze niet goed aanpassen aan verschillende resoluties.	**Middel**
Er wordt weinig gecontroleerd op de string inputs, hier zou meer controle op moeten, zodat wat er staat ook daadwerkelijk klopt.	**Middel**
Een docent heeft niet zijn eigen rol, hierdoor moet hij of als admin of student worden weggezet, dit geeft diegene te veel of te weinig rechten en is dus niet geschikt	**Hoog**
Er is geen beveiliging tegen brute-force aanvallen.	**Hoog**

3. UITBREIDINGEN

Er zijn altijd manieren om iets uit te breiden, hieronder zal ik er een paar vertellen:

- Een docent rol, zodat die ook een klas kan aanmaken of verwijderen.
- Een student zou zijn eigen gegevens kunnen veranderen.
- Een student zou zich kunnen aanmelden bij een les.
- Er kan een limiet komen op het aantal inlog pogingen.
- Een klaslokaal kan een maximaal aantal leerlingen hebben.
- Een docent zou huiswerk kunnen opgeven.
- Er zou een email kunnen worden gekoppeld.

Dit zijn naar mijn mening wel de eerste en belangrijkste aanpassingen die ik erin zou willen verwerken.