Verantwoording Noviaal - NOVI Backend 2020

Instelling	NOVI HBO Software Development
Leerlijn	BackEnd
Document	Functioneel en Technisch Ontwerp voor de Eindopdracht
Docent	Nick Stuivenberg
Datum	20 november 2020
Auteur	Jurjen Vorhauer
Email	J.Vorhauer@novi-education.nl
Studentnummer	800009793
GitHub	https://github.com/jvorhauer/noviaal
Noviaal	Een Twitterig platform, de naam is een samentrekking van NOVI en Sociaal

Inhoudsopgave

1. Inleiding

In dit document wordt een antwoord gegeven op de vragen * Welke technieken, producten en bibliotheken zijn er gebruikt? * Is aan alle voorwaarden uit de opdracht voldaan?

Conclusie: aan alle voorwaarden gesteld in de opdracht, `Integrale eindopdracht Backend 1.02.pdf` is voldaan. Er zijn alleen ingredienten gebruikt, die werden gevraagd.

2. Functioneel Ontwerp

Zie `novi-backend-foto-jurjenvorhauer.pdf` in deze directory.

3. Tecnhisch Ontwerp

Zie `novi-backend-foto-jurjenvorhauer.pdf` in deze directory.

4. Database

Voor de database is gekozen voor PostgreSQL, een steeds populairdere oplossing voor Relationeel data opslag en beheer van die data.

Door de ingebouwde ondersteuning van Spring Boot voor vele database producten via JDBC drivers, is het niet moeilijk om aan criterium "De backend en de database zijn gescheiden en kunnen los van elkaar op verschillende systemen draaien" te voldoen. Daarom zijn er in de directory/folder `src/main/resources` drie verschillende `application.properties` meegeleverd:

1. default, zonder profiel naam: `application.properties`, waarin de configuratie voor het gebruik met docker-compose en daarmee met een Docker-container met PostgreSQL beschiklbaar is
2. `application-local.properties` waarin de configuratie voor een H2 database die als bestand in de project folder wordt opgeslagen
3. `application-postgresql.properties` waarin een lokale, eerder geinstalleerde óf een extern gehoste PostgreSQL instantie geconfigureerd kan worden.

Overigens zou in principe iedere Relationele Database Management System (RDBMS) gebruikt kunnen worden, maar er is alleen met H2 en PostgreSQL getest.

Deze alternatieve configuratie bestanden kunnen geactiveerd worden door de Noviaal applicatie als volgt te starten:

`./mvnw spring-boot:run -Dspring-boot.run.profiles=local`.

of

`./mvnw spring-boot:run -Dspring-boot.run.profiles=postgresql`.

Vergeet niet om bij deze laatste optie ook de instellingen voor de database aan te passen aan de

aanwezige situatie in het bestand `application-postgresql.properties`.

5. OOP, Java en Spring (Boot)

Alle source code is standaard Java 11, met gebruik van de laatste versie van Spring Boot.

5.1. OOP

Op WikiPedia wordt Object-oriented Programming (OOP) omschreven als

´

> Is een programmeer paradigma gebaseerd op het concept "Object", die zowel code als data kunnen bevatten: data in de vorm van velden (ook wel attribuut of property genoemd) en code, in de vorm van methoden.
>
> — https://en.wikipedia.org/wiki/Object-oriented_programming

In de packages `domain`, sub-packages van `model`, `service`, `controller` is dit alles duidelijk te zien. Java noemt de definitie waarop Objecten gebaseerd worden Classes.

5.2. Java

Er zijn geen andere programmeertalen dan Java gebruikt.

Er is wel gebruik gemaakt van markup-talen als YAML (Spring en Docker Compose) en XML (logback), maar dit zijn geen programmeertalen.

5.3. Spring Boot

Er is van het Spring Boot Framework gebruik gemaakt, en daarvan zijn de volgende modulen gebruikt:

- Web
- JPA
- Security
- Validation
- Test

5.4. Gebruikte hulpmiddelen

Voor het ontwikkelen is gebruik gemaakt van

- IntelliJ IDEA 2020.3.2 (Ultimate Edition) met plugins:
  - AsciiDoc (in plaats van MarkDown dat nogal beperkt is)

    - PlantUML integration (voor sequence, use case en class diagrams)
- Maven 3.6.3
- Postman 7.36.5 en, vanaf 26 februari 2021, 8.0.6

Voor het draaien van de applicatie en de bijbehorende database wordt gebruik maakt van:

- Docker 20.10.2
- Docker Compose 1.27.4

6. Relationele Database

Noviaal is gebouwd tegen een RDBMS, met name H2 en PostgreSQL. Het domein model dat gebruikt wordt is relationeel: tussen de entiteiten zijn relaties gedefinieerd.

7. Git voor versiebeheer

Zie https://github.com/jvorhauer/noviaal en de `.git` directory/folder van het project.

8. Maven voor builds

- `pom.xml`
- Voer bijvoorbeeld `./mvnw verify` (mac/linux) of `mvnw.cmd verify` (windows) uit in de project directory/folder.

9. Unit-testen

Zie de directories/folders onder `src/test/java`.

10. Beveiliging

Middels de door Nick aangeleverde code voor het implementeren van authenticatie met JSON Web Token (JWT) was het mogelijk om de endpoints te beveiligen met authenticate en authorisaties. Het verwijderen van een bestaande gebruiker en het promoveren van een gewone gebruiker (met alleen USER rol) naar een admin (met ook ADMIN rol) zijn beveiligd, zodat alleen gebruikers met de ADMIN rol deze acties kunnen uitvoeren.

11. Wat is er niet gedaan

1. Geen performance tests. Het is dus niet zeker wat er gebeurd als er grotere aantallen gebruikers grote hoeveelheden data gaan invoeren. Waarbij 'groter' dan nog gedefinieerd moet worden (niet erg meetbaar zo). Meer data heeft impact op het schijfruimtegebruik van de database en op de snelheid waarmee tabellen bevraagd worden.

2. Niet veel @MockBean gebruikt. Tijdens het ontwikkelen merkte ik dat ik een mock database, H2 in-memory, genoeg mock vindt en dat tests, met name complexe, met @MockBean er niet overzichtelijker op worden.
3. Het aantal rollen is nogal beperkt: USER en ADMIN, waarbij alle geregistreerde gebruikers altijd USER zijn. Overigens kun je als anonieme gebruiker wel registreren en, na registratie, ook inloggen. Dus er is eigenlijk wel een ANONYMOUS rol, maar die is niet expliciet gemaakt.
4. Het is niet mogelijk om alle mogelijke configuraties van Operating System, Java versies, geinstalleerde programma's, etc. te testen. Het kan dus zijn dat Noviaal niet uit te voeren is met `docker-compose`, zoals in de 'handleiding' beschreven is.
5. De gestelde kwaliteitseis van een test-coverage van 80% is niet gehaald.
6. meer Lombok gebruikt voor reductie van bolierplate code, zoals Setters/Getters, Builders, Constructors en mogelijk meer. Dat had ik graag gedaan, maar ik kreeg tijdens de lessen dat dat wellicht niet zo gewaardeerd zou gaan worden. In de test-sources is Lombok wel wat meer gebruikt alsmede ook het Java keyword `var` om ook daarmee overhead te voorkomen. Mocht dit inderdaad niet gewenst zijn, dan had ik dat graag iets explicieter willen terugvinden in de voorwaarden. Misschien een idee voor een volgend semester? Lombok kan veel extra code voorkomen, waardoor leesbaarheid en onderhoudbaarheid verbeterd worden.

Er zijn een aantal goedbedoelde, maar technisch weinig waarde toevoegende, elementen uit het domein model gehaald:

- Reminders: hebben niet veel zin in een REST service, want niet makkelijk aan te tonen zonder frontend
- Likes: vergelijkbaar met Tags, maar dan zonder naam.
- Een anonieme versie van de tijdlijn: als je niemand volgt, is een tijdlijn niets anders dan een lijst van alle Notes en Media. Alleen geregistreerde gebruikers kunnen andere gebruikers volgen.

12. Opmerkingen

1. Deze app is nog niet af. Ik heb nog plannen voor twee iteraties, waarin de functionaliteit verder verfijnd en enigszins uitgebreid wordt.
2. Een belangrijke verbetering zou het gebruik van de Webflux, de reactive variant van Spring Web, zijn, waardoor de annotaties niet meer nodig zijn. Tijdens het ontwikkelen van Noviaal viel me op dat annotaties
    a. Niet te debuggen zijn
    b. Geen handige foutmeldingen geven waarom ze niet werken zoals gepland
3. Het bedenken en ontwikkelen van Noviaal heeft me een hoop leerzaam plezier geleverd. De lessen van Nick, Peter en de andere docenten waren nuttig en prettig, een zeldzame combinatie.
4. Hoewel ik al vele jaren met Java (vanaf 1.0) en Spring (vanaf 2.5.x) werk, heb ik veel geleerd. Ik heb opzettelijk geen vrijstelling gevraagd om te leren.