Toegepaste Informatica

Design & Develop with Frameworks

**Paper**

Geïntegreerde Opdracht

Team: 5

Abdelwahid El Mazghari, Iejan Massart, Bram Van Pevenage,Sami Hafiz , Yarne Valck, Mohamed Amine

Kallachi

Tweede bachelor – Toegepaste informatica

Vandenbogaerde Hans

Academiejaar 2020-2021



## Inhoudsopgave

[Inleiding 3](#_Toc74062609)

[Micro-services 4](#_Toc74062610)

[Front 23](#_Toc74062611)

[REST-resource 24](#_Toc74062612)

[API Documentatie 24](#_Toc74062613)

[JAR Files 24](#_Toc74062614)

[Besluit 25](#_Toc74062615)

[Zelfreflectie 26](#_Toc74062616)

### Inleiding

De geïntegreerde opdracht bouwt verder op wat we doorheen dit 2de semester hebben gezien in de lessen Design & Develop with Frameworks. Vooraleer we aan de opdracht zijn begonnen, hebben we snel nog eens de leerstof herhaald om de best practices te bekijken.

Wij bieden een backend aan die het beheren van reservaties & lokalen implementeert. Hiervoor hebben we gebruikt gemaakt van REST Api’s om zo een toegang te bieden naar de front toe. Verder is de backend opgebouwd uit verschillende microservices met elk hun eigen database.

Deze opdracht richt zich naar het toepassen van alle leerstof naar een praktijk voorbeeld. We werken daarvoor samen met Web&Mobile waar Iejan de appliatie Meetup heeft ontwikkeld. Deze applicatie kan nu onze api gebruiken. In deze paper beschrijven hoe we deze opdracht tot een goed einde hebben gebracht.

### Micro-services

Voor deze opdracht hebben we onze backend opgedeeld in drie verschillende microservices. Elk van deze microservices heeft zijn eigen database. Voor de database hebben we gebruik gemaakt van de H2 database. Een zeer eenvoudige en goede database.

Verder beschrijven we hieronder welke decompositie methode hebben gebruikt voor het splitsen van onze microservices.

#### Decompose by Business Capability

Volgens het 3 stappen plan dat ons aangereikt is tijdens de lessen hebben we onze microservices ingedeeld. In dit 3 stappenplan heb je 2 vaarwaters.

Eerste stap van dit plan is de system operations definiëren. Als uitgangspunt nemen we ons domainmodel, requirements and the give when then model. Deze kunnen we jammer genoeg niet weergeven omdat op het laatste moment zijn moeten veranderen van project. Waardoor de analyse van het vorige semester niet over de meetup gaat en dus niet relevant is.

Hierbij hebben wij gekozen om de decompositie toe te passen op case via “Business Capability”.

Hierbij kijken we naar specialiteiten die onze service moet aanbieden. Bij ons heb je de volgende drie:

* Inloggen van gebruikers
* Beheren van lokalen
* Beheren van reservaties

Logischerwijs volgt hier uit dat we drie microservices hebben:

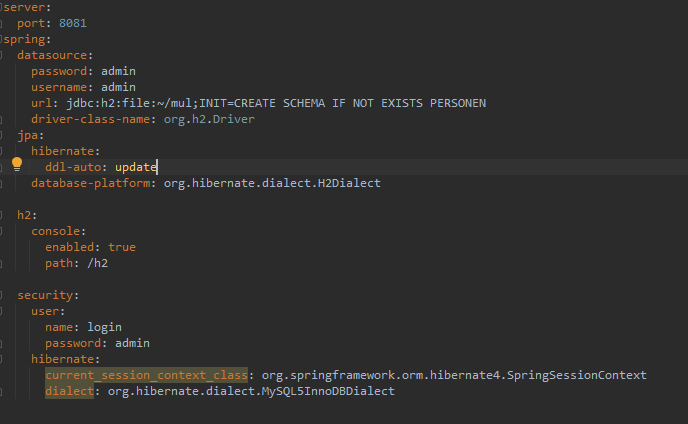
* Login
* Lokalen
* Reservaties

#### Login

De eerste microservice is de login service. Deze maakt het mogelijk voor de frontend om gebruikers te laten inloggen & registreren. Maar ook de gebruikers die als rol “admin” hebben de mogelijkheid bieden om gebruikers te gaan verwijderen om hun rol te veranderen.

##### Application.yml

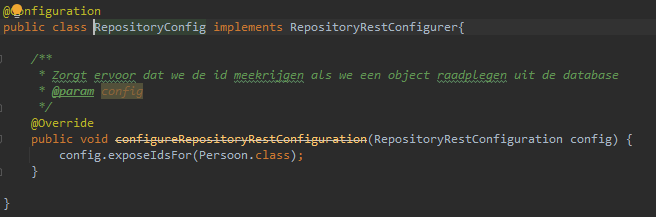
In dit bestand geven we de poort aan waarop onze service gaat draaien. In dit geval is dit poort 8081. Verder geeft het aan waar onze H2 database zich bevind en hoe deze is beveiligd. We geven een passwoord en een username door. Nog een property voor de database is wat er moet gebeuren met de database wanneer de applicatie wordt opgestart. In ons geval is dat update, updates de database schema wanneer ons domain model zou veranderd zijn.Tenslotte geven we aan dat onze service beveiligd word met volgende gebruikersnaam en passwoord. Beter zou zijn dat dit geen statische waarden zijn maar waarden dat je aanmaakt. Maar dit zou ons te ver drijven.

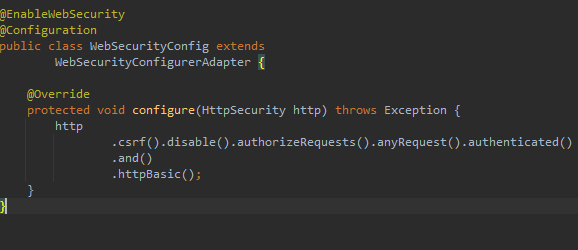


##### Configuration files

Hieronder vindt je twee configuratie bestanden. De eerste is bedoeld wanneer we objecten opvragen in dit geval een Persoon, dat hun id wordt meegeleverd. Dit is nodig voorin de front.

Het tweede bestand is om te zorgen dat we post- en andere niet get-requests kunnen gaan uitvoeren zonder csrf token. Maar dat onze service nog steeds beveiligd is.





###### Main methode

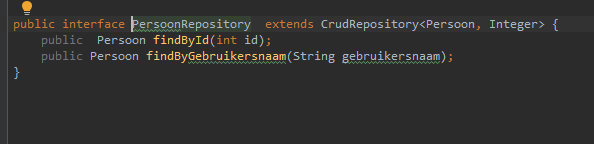
Dit bestand bevat onze main methode dat onze service zal gaan runnen het start embedded tomcat. Voor onze servce op het web toegankelijk te maken. Verder wordt er onze database geseed met 2 default users. Een user met admin rechten en een user zonder admin rechten.

Tenslotte zorgen we voor een paar security opties via Annotations.



##### DAO

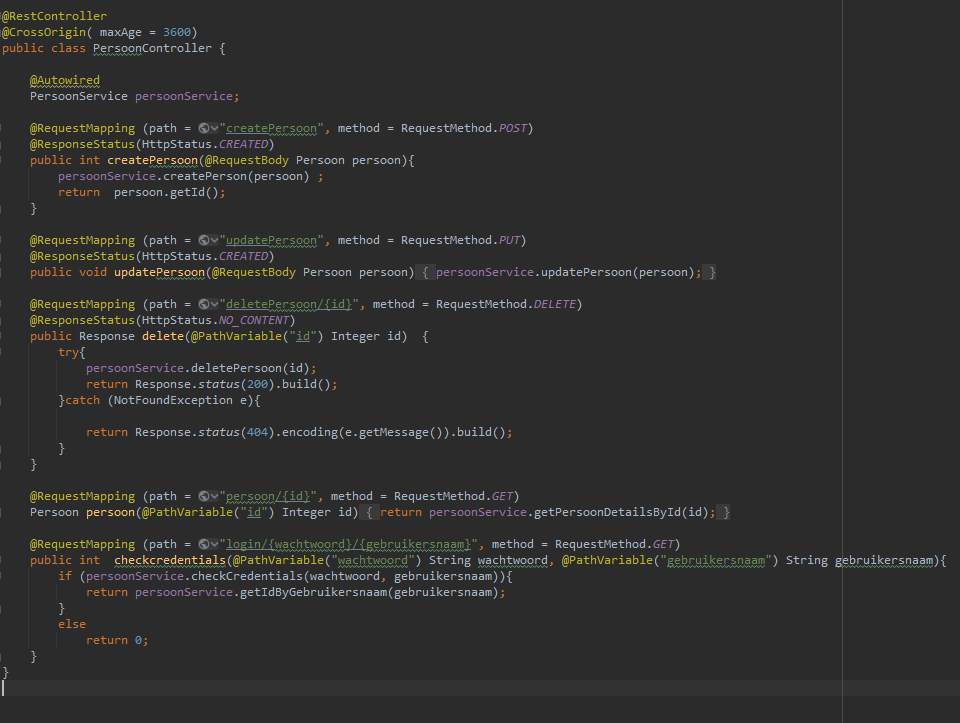
n dit bestand definiëren welke operaties we willen kunnen uitvoeren op de database. De standaard CRUD operaties worden aangeboden door de CrudRepository die extendenden. Verdere zaken die je wilt uitvoeren buiten de crud kan je hierin definiëren. Het gaat hier over de operaties op de Personen entities.



##### Controller

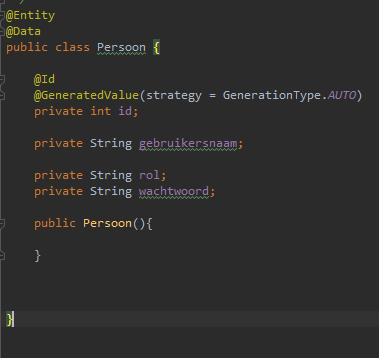
De controller is hetgene dat zal worden aangesproken door de request die worden gemaakt door een frontend. Op basis van het path dat er wordt meegegeven wordt er een andere acte uitgevoerd. Verder gebruiken we een autowired van de PersoonService een service die ons zal helpen op de acties als operaties op de database te gaan uitvoeren. Verder in deze paper gaan we het hebben over de Api documentatie daar zal het uitgebreid overgaan wat de controller allemaal aanbied.

@CrossOrigin hebben we nodig zodat aanvragen van verschillen locatie kunnen komen.



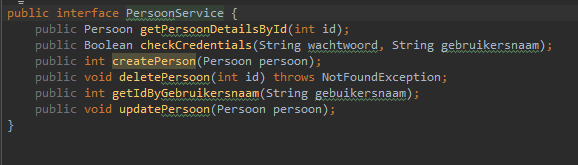
##### Domain

In ons domain package hebben we enkel de Persoon klasse. Enkel deze hebben we nodig om een login functionaliteit te kunnen aanbieden. Dus meerdere klassen hebben we hier niet nodig. Een persoon object bevat een id, username, wachtwoord en een rol.



##### Service

In de onderstaande afbeelding wordt een een bestand getoond met alle methodes die we in de PersoonServivceImpl klasse gaan uiwerken.



In onderstaande afbeelding wordt getoond hoe dat onze service uitgewerkt is om de CRUD operaties te kunnen toepassen op de database.

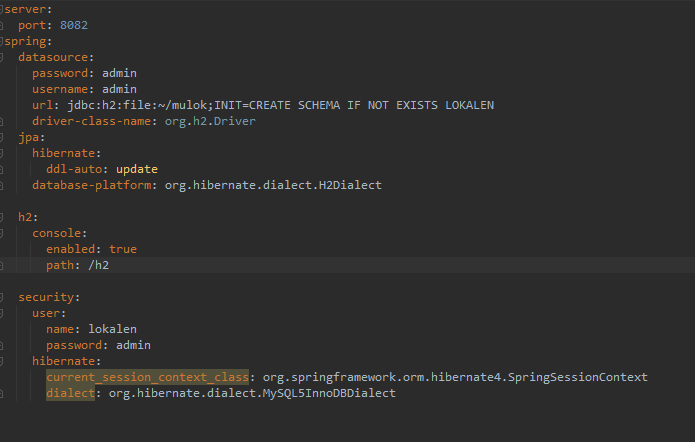


#### Lokalen

De volgende microservice is de lokalen service. Regelt het beheren van de lokalen.

##### Application.yml

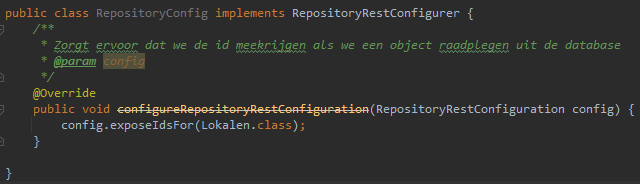
In dit bestand geven we de poort aan waarop onze service gaat draaien. In dit geval is dit poort 8082. Verder geeft het aan waar onze H2 database zich bevind en hoe deze is beveiligd. We geven een passwoord en een username door. Nog een property voor de database is wat er moet gebeuren met de database wanneer de applicatie wordt opgestart. In ons geval is dat update, updates de database schema wanneer ons domain model zou veranderd zijn. Tenslotte geven we aan dat onze service beveiligd word met volgende gebruikersnaam en passwoord. Beter zou zijn dat dit geen statische waarden zijn maar waarden dat je aanmaakt. Maar dit zou ons te ver drijven.



##### Configuration files

Hieronder vindt je twee configuratie bestanden. De eerste is bedoeld wanneer we objecten opvragen in dit geval een Persoon, dat hun id wordt meegeleverd. Dit is nodig voorin de front.

Het tweede bestand is om te zorgen dat we post- en andere niet get-requests kunnen gaan uitvoeren zonder csrf token. Maar dat onze service nog steeds beveiligd is.

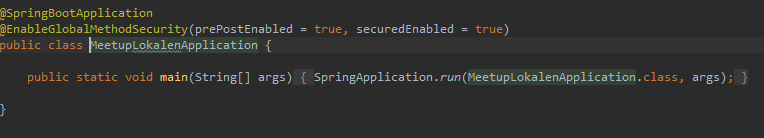




##### Main methode

Dit bestand bevat onze main methode dat onze service zal gaan runnen het start embedded tomcat. Voor onze serivce op het web toegankelijk te maken.

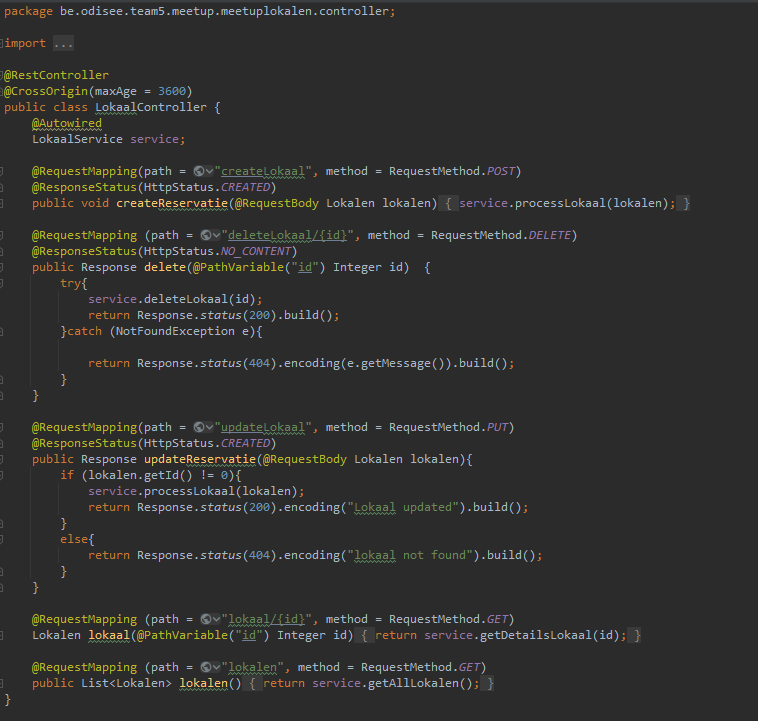
Verder zorgen we voor een paar security opties via annotations.



##### Controller

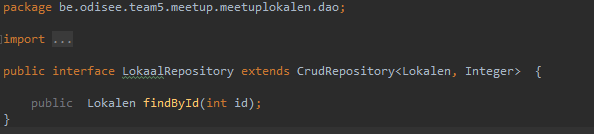
De controller is hetgene dat zal worden aangesproken door de request die worden gemaakt door een frontend. Op basis van het path dat er wordt meegegeven wordt er een andere acte uitgevoerd. Verder gebruiken we een autowired van de LokalenService een service die ons zal helpen op de acties als operaties op de database te gaan uitvoeren. Verder in deze paper gaan we het hebben over de Api documentatie daar zal het uitgebreid overgaan wat de controller allemaal aanbied.

@CrossOrigin hebben we nodig zodat aanvragen van verschillen locatie kunnen komen.



##### DAO

In dit bestand definiëren welke operaties we willen kunnen uitvoeren op de database. De standaard CRUD operaties worden aangeboden door de CrudRepository die extendenden. Verdere zaken die je wilt uitvoeren buiten de crud kan je hierin definiëren. Het gaat hier over de operaties op de lokalen entities.



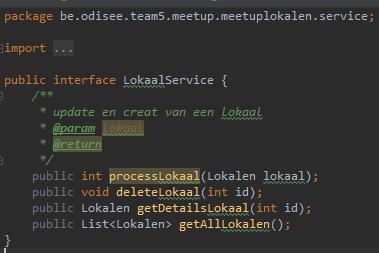
##### Domain

De enige klasse in ons domain model is Lokalen. De enige dat we nodig hebben om de benodigde functionaliteiten te voorzien voor deze service.



##### Service

In de onderstaande afbeelding wordt een een bestand getoond met alle methodes die we in de LokaalServivceImpl klasse gaan uiwerken.



In onderstaande afbeelding wordt getoond hoe dat onze service uitgewerkt is om de CRUD operaties te kunnen toepassen op de database.

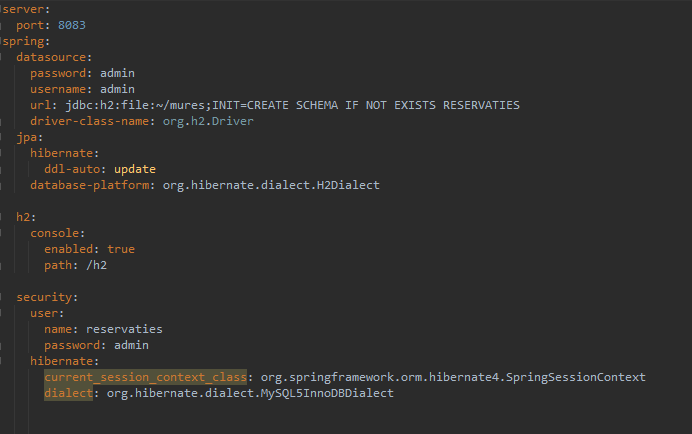


#### Reservaties

Tenslotte hebben de reservaties microservices. Wordt gebruikt wanneer gebruikers via het front lokalen gaan reserveren.

##### Application.yml

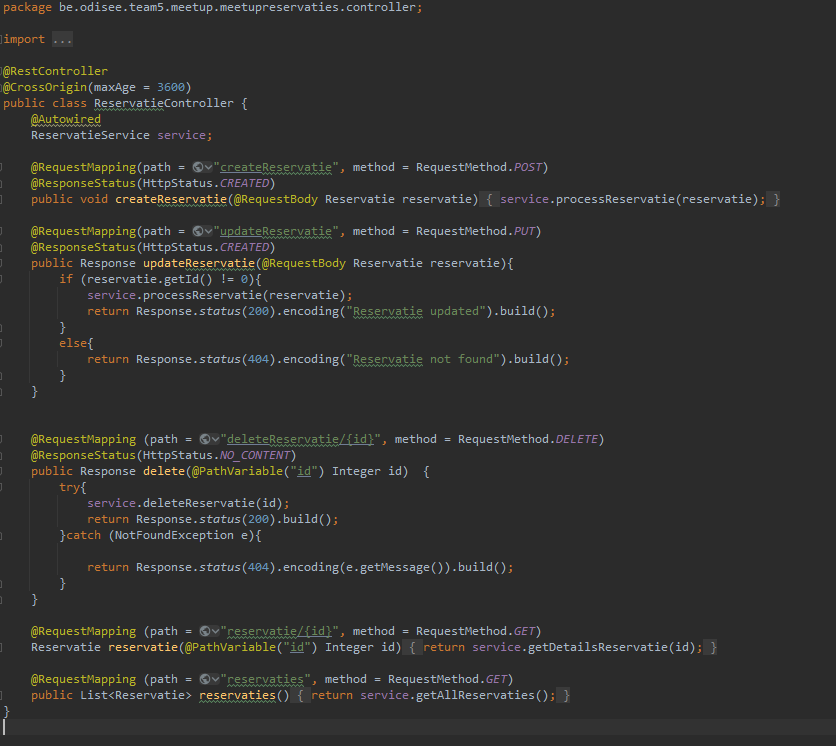
In dit bestand geven we de poort aan waarop onze service gaat draaien. In dit geval is dit poort 8082. Verder geeft het aan waar onze H2 database zich bevind en hoe deze is beveiligd. We geven een passwoord en een username door. Nog een property voor de database is wat er moet gebeuren met de database wanneer de applicatie wordt opgestart. In ons geval is dat update, updates de database schema wanneer ons domain model zou veranderd zijn. Tenslotte geven we aan dat onze service beveiligd word met volgende gebruikersnaam en passwoord. Beter zou zijn dat dit geen statische waarden zijn maar waarden dat je aanmaakt. Maar dit zou ons te ver drijven.



##### Controller

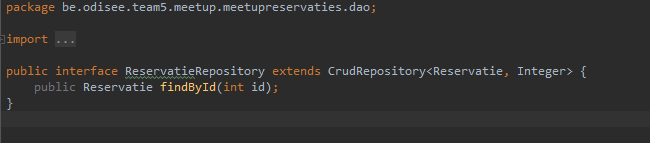
De controller is hetgene dat zal worden aangesproken door de request die worden gemaakt door een frontend. Op basis van het path dat er wordt meegegeven wordt er een andere acte uitgevoerd. Verder gebruiken we een autowired van de ReservatieService een service die ons zal helpen op de acties als operaties op de database te gaan uitvoeren. Verder in deze paper gaan we het hebben over de Api documentatie daar zal het uitgebreid overgaan wat de controller allemaal aanbied.

@CrossOrigin hebben we nodig zodat aanvragen van verschillen locatie kunnen komen.



##### DAO

In dit bestand definiëren welke operaties we willen kunnen uitvoeren op de database. De standaard CRUD operaties worden aangeboden door de CrudRepository die extendenden. Verdere zaken die je wilt uitvoeren buiten de crud kan je hierin definiëren. Het gaat hier over de operaties op de lokalen entities.

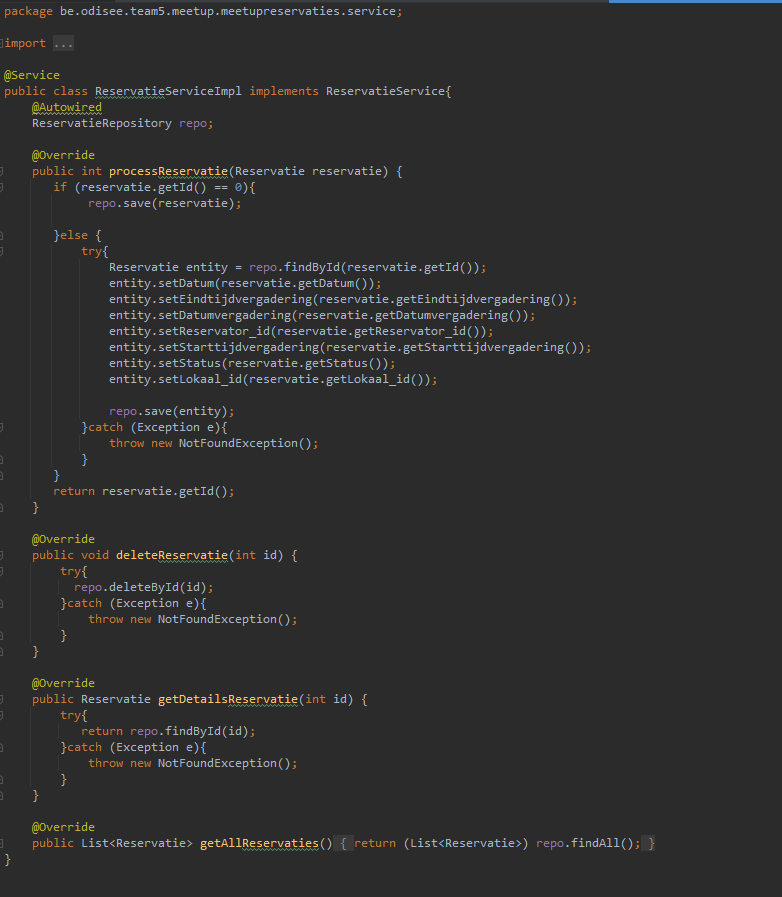


##### Service

In de onderstaande afbeelding wordt een een bestand getoond met alle methodes die we in de ReservatieServivceImpl klasse gaan uiwerken.



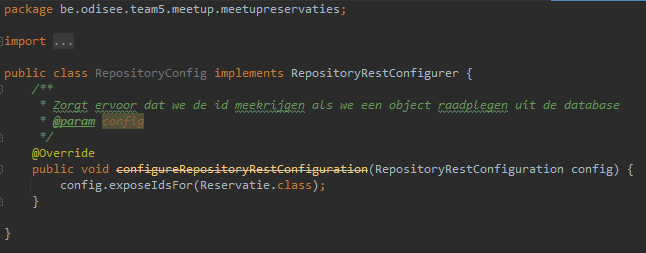
In onderstaande afbeelding wordt getoond hoe dat onze service uitgewerkt is om de CRUD operaties te kunnen toepassen op de database.

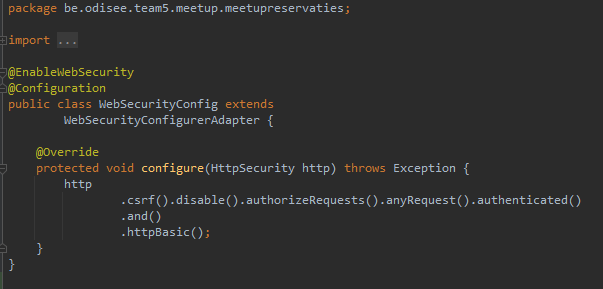


##### Configuratie bestanden

Hieronder vindt je twee configuratie bestanden. De eerste is bedoeld wanneer we objecten opvragen in dit geval een Persoon, dat hun id wordt meegeleverd. Dit is nodig voorin de front.

Het tweede bestand is om te zorgen dat we post- en andere niet get-requests kunnen gaan uitvoeren zonder csrf token. Maar dat onze service nog steeds beveiligd is.

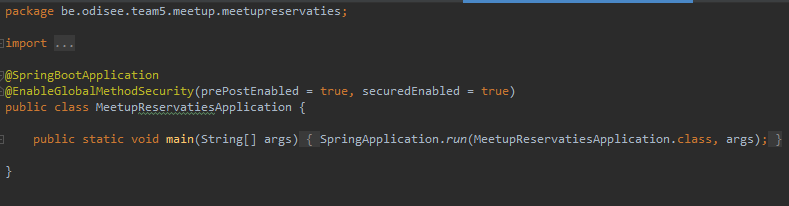




##### Main methode

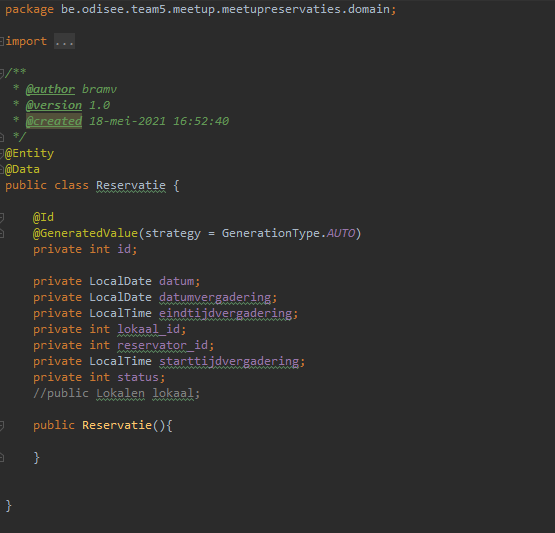
Dit bestand bevat onze main methode dat onze service zal gaan runnen het start embedded tomcat. Voor onze servce op het web toegankelijk te maken.

Verder zorgen we voor een paar security opties via Annotations.



##### Domain

De enige klasse in ons domain model is Reservatie. De enige dat we nodig hebben om de benodigde functionaliteiten te voorzien voor deze service.



### Front

### REST-resource

### API Documentatie

Hieronder vindt de api documentatie die we aanbieden voor diegene dat onze api’s willen gaan gebruiken.

### JAR Files

Aangezien er drie microservices zijn hebben we ook drie JAR files aangemaakt. We hebben dit aangemaakt via maven. Hierdoor komt je JAR file in de target map terecht en kan je die met volgende commando runnen: java -jar [naam jar file].jar

### Besluit

Deze opdracht was een goede afsluiter voor dit vak een totaal plaatje van de geziene leerstof dat we op een opdracht van een ander vak mocten gaan toepassen.

### Zelfreflectie

**Iejan**:

**Amine**:

**Abdelwahid**:

**Bram**:

**Yarne**:

**Sami**: