Back-End in Java Spring Boot

## Inhoud

* Spring Boot
* Libraries
* Basis uitleg Spring Boot
* Basis uitleg Maven

# Spring Boot

* (REST) API Server
* Regelt de Logic & Communiceert met Database

De back-end van het project is verantwoordelijk voor alle logic die uit gevoerd moet worden om bepaalde acties te voltooien. Dit zijn dus de gedeeltes waarbij de gebruiker in theorie niet veel baat heeft. Denk aan: hoe inlog gegevens in de database opgehaald/opgeslagen worden, en password hashes. De gebruiker wil simpelweg dat het werkt, maar wat in de achtergrond precies gebeurt maakt niet uit. De back-end ontvangt een request van de front-end (meestal op basis van een user interactie) en gaat hiermee aan de slag, en geeft vervolgens een response.

# Libraries

Onderstaand de uitleg van de gebruikte libraries en waarvoor ze functioneren:

**H2Database**:

*De library die gebruikt wordt om bij het opstarten van het programma de embedded database te kunnen gebruiken.*

**LOMBOK:**

*Een library die heel veel Annotations meegeeft als mogelijkheden. (Meer over Annotation bij uitleg Spring Boot) Voorbeeld van een aantal Annotations zijn: @Getter @Setter @NoArgsContructor*

**ModelMapper:**

*Een library die heel simpel maakt om DTO’s en Classes te ‘mappen’ (Converting)*

**Jakarta Validation:**

*Geeft extra annotations om te validate dat een field een bepaald type is: @Email @NotNull*

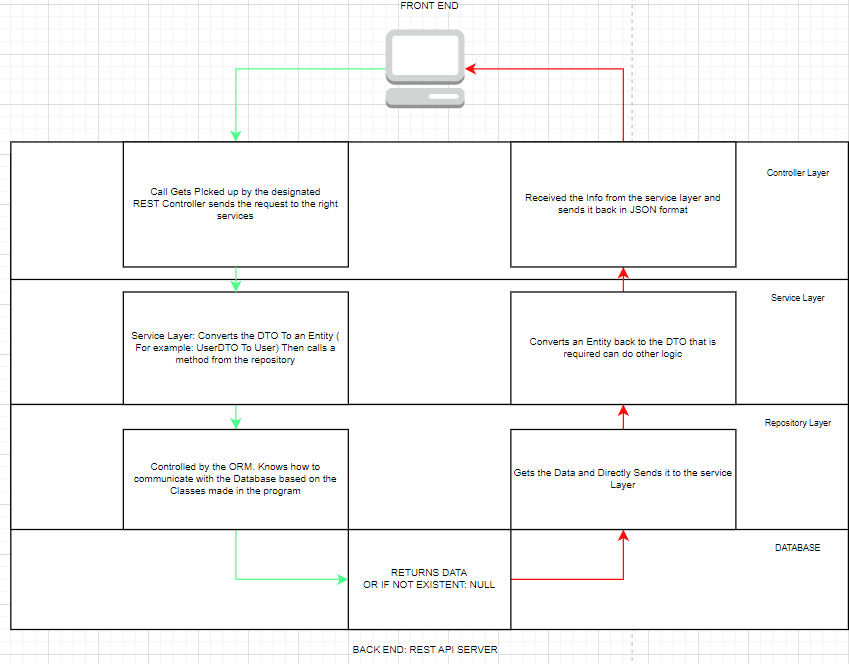
**IO.JSONWEBTOKEN:**

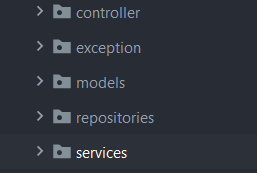
*Library die functionaliteiten biedt om met JWT Tokens en JWS Tokens aan de slag te gaan.*

# Basis uitleg Spring Boot

Spring boot is een Java framework die we gebruikt hebben voor het maken van de backend (REST API)

De front-end maakt een call naar de backend om data te krijgen, en stuurt regelmatig zelf nog data mee.  
  
Onderstaand is een afbeelding om duidelijk te maken hoe de flow werkt wanneer zo’n call gemaakt wordt:





De backend is opgedeeld in bovenstaande structuur.

***Controller:***

De laag die verantwoordelijk is voor communicatie ontvangen en terug turen naar de front-end.  
Wanneer het een aanvraag krijgt stuurt het dit door naar de juiste services

***Exception:***

Hier staan alle exceptions in, je kan zelf exceptions maken mochten bepaalde situaties verwacht worden. (Of juist een onverwachte situatie)

***Models:***

Models bestaat uit 2 onderdelen:  
*DTO: De objecten die naar de front end gestuurd worden.*  
*Classes ( Entities ): De objecten die naar de DB gestuurd worden en waar logic mee wordt uitgevoerd.*

***Service:***

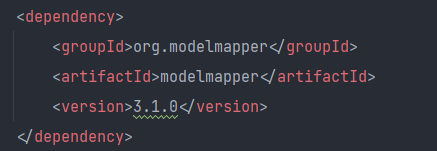
De logic wordt hierin uitgevoerd, ook weet het welke repositories methods uitgevoerd moeten worden.  
***Repositories:***

Interfaces die weten hoe met de database gecommuniceerd wordt door middel van een ORM.  
Spring boot genereert automatisch SQL methodes als je aan deze interfaces methodes toevoegt:

VB: bij userRepo: maak een methode: findByMail(string mail) en Spring boot vult het aan voor je.

# Basis uitleg Maven

Met maven kan je externe libraries gebruiken door ze als een dependancy in de POM File te zetten.

Een voorbeeld van een zo’n dependency is:

Wanneer je op internet zoekt naar Java libraries staat hier eigenlijk altijd ook een Maven dependency bij. Deze moet je toevoegen aan de pom file tussen de <Dependencies> parameters.

Vergeet nadat je de POM file bijwerkt niet op rechts bovenstaande knop te klikken:  
(Intelij)

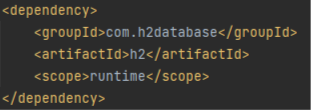


*Knop die Maven POM file bijwerkt.*

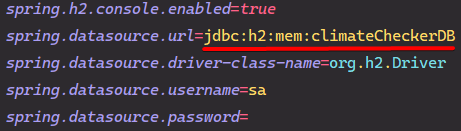
# Basis uitleg H2-console

Gezien een remote database mogelijk offline kan gaan, of een maximaal aantal gebruikers kan bereiken, is er een lokale database opzet voor testen. Deze kan voor ieder persoon aangepaste data hebben die relevant is op dat moment.

Voor de H2 database is er allereerst een dependency in de .pom-file. Deze dependency zal ervoor zorgen dat h2 gerund kan worden en ook kan worden uitgelezen. Zoals te zien hieronder:



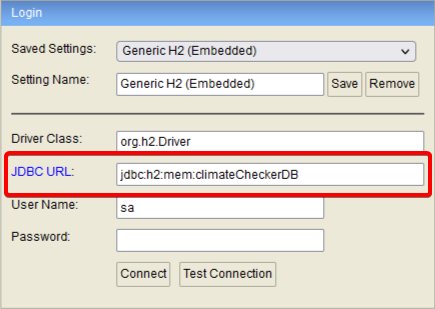
In onze “application properties” file zal aangegeven moeten worden hoe de connectie ligt. (De rode markering is later nodig in de uitleg)



Als er nog geen lokale database is aangemaakt, zal er automatisch een worden opgezet in een nieuwe folder genaamd Database waar de ‘spring.datasource.url’ naar verwijst. In dit geval zal de applicatie een in-memory database aanmaken alleen als de applicatie runt.

Als het project runt kan de gebruiker in hun browser de h2 console bereiken. Dit is standaard via de url <http://localhost:8082/h2-console/> . Als in de “application properties” de port veranderd is, zal de port in deze link anders zijn. Zodra er geen andere port aangegeven staat, zal het op de default van 8080 zijn.

Op het inlog scherm is het van belang dat de ‘url’ ingevuld wordt om zo in de database te komen. Dit is eerder in het ‘application properties’ bestand vastgelegd en gemarkeerd met een rode lijn in de afbeelding hierboven.



# Basis uitleg seeder

Tijdens het developen is er behoefte aan test data. Voor het genereren van die testdata gebruiken we iets wat we noemen een ‘seeder’. Dit komt er op neer dat eenmaal de back-end op start de database wordt gevuld met zelf ingestelde (Hardcoded) test data. Dit doen we omdat elke keer als het project afgesloten wordt de vorige data verwijderd wordt). De data die wordt ge-insert staat in “ClimateCheckerApplication.java” oftewel het begin bestand van de applicatie. Hieronder staat een afbeelding van hoe een stukje van de ‘seeder’ eruit ziet:

