Overdrachtsdocument  
Monitor van de stad

Ontdekstation013 & Bibliotheken Midden-Brabant

Fontys Hogeschool | Tilburg

|  |  |
| --- | --- |
| **Auteur**: | Cyrion van dongen, Janno Vos, Janiek van Pelt, Stan Akkermans, Fabiënne Leidekker, Daan Brouwer |
| **Locatie**: | Tilburg |
| **Versie:** | 1.2 |
| **Datum**: | 14 juni 2022 |

# Versiebeheer

|  |  |  |
| --- | --- | --- |
| **Versie** | **Wijziging** | **Datum** |
| 0.1 | Opzet basis | 8 februari 2022 |
| 0.2 | Aanpassingen a.d.h.v. framework keuzes | 10 maart 2022 |
| 0.3 | Invulling hoofdstuk ‘Java’ | 24 maart 2022 |
| 1.0 | Invulling React JS / update database & controllers | 21 & 26 april 2022 |
| 1.1 | Update database | 24 mei 2022 |
| 1.2 | Update voor overdracht | 9 & 14 juni 2022 |

# Inhoud

[1 Versiebeheer 2](#_Toc106093104)

[2 Inhoud 3](#_Toc106093105)

[3 Inleiding 4](#_Toc106093106)

[4 Notities 4](#_Toc106093107)

[5 Aanbevelingen 5](#_Toc106093108)

[6 React JS 6](#_Toc106093109)

[6.1 Keuze 6](#_Toc106093110)

[6.2 Opzet 6](#_Toc106093111)

[6.3 Testen 6](#_Toc106093112)

[7 Java 7](#_Toc106093113)

[7.1 Keuze 7](#_Toc106093114)

[7.2 Opzet 8](#_Toc106093115)

[7.2.1 Database opzet (in-memory) 8](#_Toc106093116)

[7.2.2 IntelliJ IDEA opzet 10](#_Toc106093117)

[7.2.3 Toevoegen aan de code 11](#_Toc106093118)

[Entities en DTO’s 11](#_Toc106093119)

[Controllers 13](#_Toc106093120)

[Repositories en Services 14](#_Toc106093121)

[7.2.4 Cors 15](#_Toc106093122)

[8 Leaflet 16](#_Toc106093123)

[8.1 Wat is leaflet? 16](#_Toc106093124)

[8.2 Hoe gebruik je leaflet? 16](#_Toc106093125)

[9 Reflectie 17](#_Toc106093126)

[9.1 Wat kan er beter? 17](#_Toc106093127)

[9.2 Waar zijn we trots op? 17](#_Toc106093128)

[9.3 Hoe zijn we met feedback omgegaan? 17](#_Toc106093129)

# Inleiding

Ontdekstation 013 is een organisatie in Tilburg waar jongeren hun talenten kunnen ontdekken op gebied van wetenschap en techniek. In samenwerking met Bibliotheek Midden-Brabant hebben zij een project opgezet rondom het meten van luchtkwaliteit in de regio Tilburg. In de regio staan meerdere sensoren om verschillende waardes op te meten. Om deze meetgegevens weer te geven voor burgers en bedrijven zal dus een nieuwe applicatie gemaakt moeten worden. Deze applicatie zal met zijn functionaliteiten kunnen laten zien hoe bepaalde acties van personen invloed hebben op bijvoorbeeld de ventilatie en de luchtkwaliteit zelf.

Gezien dit project door meerdere groepen over een langere tijd uitgewerkt zal worden, is de overdraagbaarheid hiervan belangrijk. Om deze overdraagbaarheid te bevorderen is dit document opgesteld.  
In dit document worden van gebruikte services en omgevingen uitgelegd waarom ze gekozen zijn, hoe ze opgezet kunnen worden door een volgend team en hoe ze beheerd en uitgebreid kunnen worden.

# Notities

Onder dit kopje worden er enkele dingen genoemd voor volgende groepen om op te letten.  
Allereerst is het op te merken dat de opdrachtgever vanuit Ontdekstation013 maar met een enkel persoon contact zou willen hebben. Stel hiervoor dus een enkele contactpersoon aan die dit onderhoudt. Daarnaast is het van belang om op tijd bericht te sturen wanneer er vragen zijn of afspraken te maken zijn, gezien het mogelijk een aantal dagen kan duren voor er een reactie komt.

Daarnaast is het van belang om voldoende met de opdrachtgever afspraken te maken wat er voor een volgende sprint gedaan zal worden. Zo hebben zij ook zicht op wat ze kunnen verwachten en eventueel nog hun eigen inbreng geven. Maak hierbij dan ook de voldoende notities.

De opdrachtgever zou daarnaast zelf ook de notities van gesprekken met hen willen ontvangen. Ook documentatie zelf die is gemaakt, voornamelijk onderzoeken die zijn gedaan voor het project, willen ze ook kunnen inzien. Stuur deze dan ook het liefst zo snel mogelijk door wanneer het toepasselijk is, bijvoorbeeld aan het einde of begin van een sprint.

# Aanbevelingen

Op de [git repositories](https://github.com/orgs/OntdekIT/repositories) staan er per repository issues aangegeven waaraan gewerkt kan worden deze zijn gebaseerd op de verschillende onderdelen die nog niet zijn toegevoegd of dingen die niet werken/gerepareerd moeten worden. Issues kunnen tags hebben, zoals ‘bug’ of ‘good first issue’. Bug spreekt grotendeels voor zich, dit slaat op een probleem binnen de bestaande code of structuur dat opgelost zal moeten worden.   
Good first issue is een taak die voor een nieuwe groep makkelijk zou moeten zijn om uit te voeren vergeleken met andere taken. Deze zullen dan ook kunnen helpen met de structuur van de code door te krijgen.

De documentatie doornemen is daarnaast ook een stevige aanbeveling. Voornamelijk het analysedocument, de mock-ups, het mock-up keuzes document en van nature dit overdrachtsdocument. Deze documenten zijn met name het nuttigst om gemaakte afspraken en structuur te leren begrijpen.

Ook zijn er een aantal opmerkingen gemaakt bij de gebruikerstesten van de mock-ups. Deze verbeteringen staan in het “Mock-up keuzes” document. Deze opmerkingen zouden kunnen verwerkt in de bestaande mock-ups om het proces daarbinnen te verbeteren.

Verder is er vanuit de opdrachtgever aangegeven dat een volgende groep eventueel naar het volgende zal kunnen kijken:

* Kijk verder naar de gebruikersgroepen: de jonge kinderen, de volwassenen en de ouderen; en ook de ‘standaard gebruikers’ en de ‘mensen die veel meer informatie willen vinden’. Onderzoek hierbij hoe het bestaande design eventueel nog verder verbeterd kan worden om het bruikbaarder te maken voor al deze groepen.
* Kijk of het mogelijk is om eventueel meerdere talen te kunnen toevoegen. Natuurlijk is het belangrijkste om het volledig in het Nederlands werkend te hebben, gezien het in regio Tilburg gebruikt zal worden. Mogelijk kunnen er bijvoorbeeld buitenlandse studenten interesse hebben en zou de optie om naar Engels te wisselen gewenst kunnen zijn. Dit is momenteel echter niet al te cruciaal.

Belangrijk om op te letten is dat er momenteel geen autorisatie en authenticatie aanwezig is. Zodra dit gedaan is, zal er ook naar bestaande functies gekeken moeten worden om bij te voegen wie er wel en geen toegang tot hebben. Daarnaast zal dan ook voor dingen zoals eigen station ingevoegd moeten worden dat alleen de stations van de ingelogde gebruiker opgehaald en aangepast kunnen worden.

# React JS

React JS is een front-end framework dat voornamelijk gebruik maakt van Javascript. In dit gedeelte wordt beschreven hoe een React project in elkaar steekt en hoe je het project draaiende krijgt!

## Keuze

De keuze om React JS te gebruiken is gekomen vanuit twee belangrijke redenen. Allereerst was het nodig om veel data in diagrammen en kaarten te laten zien. Beide op browser en op mobiel. Door deze combinatie is React een van de betere opties om op te pakken.   
Daarnaast is het volgens de opdrachtgever belangrijk om zo veel mogelijk open source te gebruiken. Gezien het gebruik van React geen extra kosten vraagt, voor het coderen en de IDE, is dit al een goede optie. Ook voor het zoeken naar API’s en extra dependencies heeft React veel open source opties van goede kwaliteit.

## Opzet

Voor het gebruik van React zul je eerst node.js moeten installeren, deze kan je hier vinden: <https://nodejs.org/en/>

Na het installeren hiervan ga je React installeren voor het project. Open het project in vs code en maak een nieuw terminal aan. Voer hierbij het volgende commando in: ‘npm install’. Na dit commando zou je het project moeten kunnen opstarten!

Het project opstarten doe je via dit commando:



Voor de routing van het project gebruiken we react-router. Dit is een van de populairste routing oplossingen voor React. Deze installeer je via dit command: ‘npm install react-router-dom@6’.

## Testen

Voor end-to-end test wordt er gebruik gemaakt van Cypress. Met Cypress kan er een aparte front-end gerund worden. Hiermee kunnen testbestanden geschreven worden. Cypress zal dan aan de hand van deze tests in real-time de interacties met de front-end uitvoeren. Daarnaast kan Cypress REST-calls afvangen, zodat deze niet naar een back end gestuurd worden. In plaats daarvan kan in Cypress een testrespons teruggegeven worden, om zo nog preciezer resultaten te krijgen.

Cypress kan geïnstalleerd worden met ‘npm install cypress’. Via het commando ‘npx cypress open’ kun je de Cypress test runner openen en zo je code testen.

Alle Cypress testbestanden staan in een aparte Cypress folder. In de integration-folder worden de tests gemaakt. De fixtures-folder zorgt ervoor dat je een api call kan afvangen en een bepaalde json kan teruggeven. Zo ben je niet afhankelijk van de back-end in het testen.

# Java

Als backend wordt een Maven Java project gebruikt, opgezet met Spring.

## Keuze

Voor de backend waren er vrij weinig opties mogelijk om uit te kiezen. Allereerst was het wederom van de opdrachtgever nodig om zo min mogelijk kosten eraan te koppelen, waarbij het liefst ook niet met opties van grote bedrijven te werken, zoals Microsoft. Hierbij viel het gebruik van .NET en C# al weg. Twee van de beste grote open opties die werden gegeven waren Java en PHP.

Vanuit Fontys zijn er ook criteria aan het werk dat geleverd moet worden. Aan de hand hiervan is gekozen voor Java. Enerzijds gezien het gebruik van Java al sterk wordt aangemoedigd, anderzijds omdat er vanuit Fontys resources te krijgen zijn die met Java kunnen helpen.

## Opzet

Dit hoofdstuk zal gaan over het opzetten van het project in IntelliJ IDEA en het opzetten van een database.

### Database opzet (in-memory)

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

Afbeelding met tekst

Automatisch gegenereerde beschrijving

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.

In onze “application properties” file zal aangegeven moeten worden hoe de connectie ligt.

Text

Description automatically generated

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.

Graphical user interface, text, application, email

Description automatically generated

Om de database te kunnen bereiken zal een “Generic H2 (Embedded)” database gekozen moeten worden. Benoem de driver en JDBC URL zoals ze in de “application properties” staan. In dit geval dus ‘org.h2.Driver’ en ‘jdbc:h2:mem:climateCheckerDB’ respectievelijk. Voer hierna ook de Username en Password in die in de “application properties staan”. Username is dus ‘sa’ en er is geen password.

Door “connect” te selecteren zal de console openen en kunnen er met SQL wijzigingen gemaakt worden direct aan de database en kan data worden opgehaald en gevisualiseerd worden met tabellen.

Table

Description automatically generated with medium confidenceVanuit hier kan men ook direct SQl commands runnen om data uit te lezen of aan te passen.

Om bij iedere startup standaard data ingevoerd te krijgen, is het echter wel nodig om hiervoor instellingen te maken. In dit geval kan dat door een ‘data.sql’ file in de resources folder van het java project te stoppen.  
In deze file kan aangegeven worden in SQL queries wat er ingevoerd zal moeten worden. Een template voor de ‘data.sql’ is te vinden in de documentatie git repository.

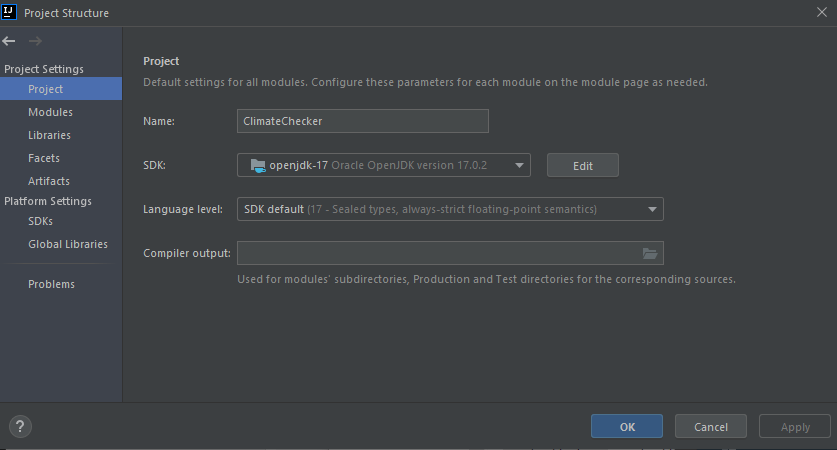
De sql file staat als ignored op de .gitignore, dus iedere gebruiker kan hem naar eigen wil aanpassen zonder dat er conflicts hoeven te ontstaan.

### IntelliJ IDEA opzet

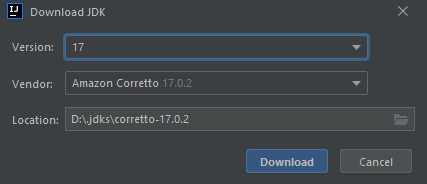
Om aan het bestaande project toe te kunnen voegen, zijn er een aantal stappen nodig om alles eerst werkend te krijgen. Allereerst zal het project via een git clone lokaal geïmporteerd moeten worden.

In dit project is de SDK default 17. Hiervoor zal de lokale SDK als 17 ingesteld moeten worden. Ga hiervoor binnen IntelliJ naar *File > Project Structure*. Kijk binnen de tab *Project* of ‘SDK’ en ‘Language level’ op een versie van 17 staan.

Afbeelding met tekst

Automatisch gegenereerde beschrijving 

Als de SDK niet op 17 staat, dan zal dit eerst gedaan moeten worden. Als er binnen de drop-down niet de optie is om deze te kiezen, selecteer dan ‘Add SDK’ en daarbinnen ‘Download JDK…’.  
Kies hier versie 17 en selecteer een vendor en locatie om het op te slaan. Gewenst is om de Oracle versie 17 te nemen, maar als die optie er niet is, werken andere opties ook. Download de JDK en voeg hem toe aan de Project Structure.



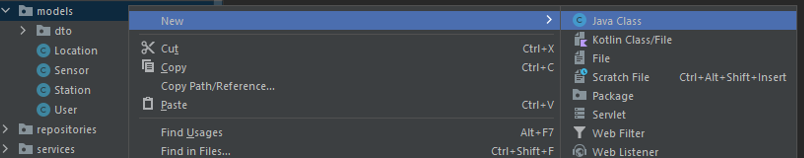
### Toevoegen aan de code

Om binnen het bestaande project uitbreidingen te maken zijn er twee opties. Kijk eerst of de benodigde class al bestaat. Als dit het geval is, kan deze aangevuld worden. Zo niet, dan zal hieronder een uitleg staan over hoe bepaalde onderdelen toe te voegen.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

#### Entities en DTO’s

Onder de package ‘models’ zullen nieuwe Entities en Data Transfer Objects (DTO) toegevoegd worden.  
Om een nieuwe Entity aan te maken, selecteer binnen ‘models’ *New > Java Class* en geef de class de passende naam. 

Voeg boven de class de javax persistence annotation ‘@Entity’ toe. Hierdoor zal de Hibernate ORM weten dat deze class gebruikt zal worden voor database purposes. Daarnaast zijn ook ‘@Getter’, ‘@Setter’ en ‘@NoArgsConstructor’ aan te bevelen. Met deze opties hoef je zelf niet meer voor ieder attribute Get en Set te schrijven. ‘@NoArgsConstructor’ creëert automatisch een *lege* constructor voor de class, dus hoeft deze ook niet neergezet te worden. Afbeelding met tekst

Automatisch gegenereerde beschrijving

Let er bij de attributen op dat hier ook de benodigde annotations staan. Voeg bij de primary key ‘@Id’ en ‘@GeneratedValue’ toe. ‘@Id’ laat weten dat dit de key/het ID is, met ‘@GeneratedValue’ kan met GenerationType.IDENTITY aangegeven worden dat dit een auto-increment heeft.

Bij Entities die aan elkaar zijn gekoppeld moet er gekeken worden wat de relatie tussen de twee is. Aan de hand hiervan zal [‘@ManyToOne’](https://docs.oracle.com/javaee/6/api/javax/persistence/ManyToOne.html)/[’@OneToMany’](https://docs.oracle.com/javaee/6/api/javax/persistence/OneToMany.html), [‘@ManyToMany’](https://docs.oracle.com/javaee/6/api/javax/persistence/ManyToMany.html) of [‘@OneToOne’](https://docs.oracle.com/javaee/6/api/javax/persistence/OneToOne.html) gebruikt gaan worden. Voor een volledige uitleg over hoe dit is op te zetten, ga dan naar de Javax persistence documentatie. Let er hierbij in ieder geval voornamelijk op dat ‘mappedBy’ hierbij de naam van het attribuut in de andere class aanvraagt.

Voor het aanmaken van een DTO, selecteer dan binnen de package ‘DTO’ *New > Java Class* en geef de class de passende naam.  
Voeg daarna bij het DTO ‘extends Dto’ toe, refererend naar de ‘Dto’ Abstract class binnen de package. Ook hier zijn de annotations ‘@Getter’, ‘@Setter’ en ‘@NoArgsConstructor’ aan te bevelen. Met deze opties hoef je zelf niet meer voor ieder attribute Get en Set te schrijven. ‘@NoArgsConstructor’ creëert automatisch een *lege* constructor voor de class, dus hoeft deze ook niet neergezet te worden.

#### 

#### Controllers

Afbeelding met tekst

Automatisch gegenereerde beschrijvingControllers zijn ervoor om de connectie met de front-end werkend te krijgen. Hierbinnen worden de REST-calls ontvangen en doorgestuurd naar de services om te laten verwerken. Deze controllers sturen dan ook weer een reactie terug naar voren aan de hand van de logica die is geschreven. Om een nieuwe controller aan te maken, selecteer de package ‘controller’ en kies *New > Java Class*. Geef deze een passende naam op welke functies erin komen, eindigend op “-Controller”.

Voeg bovenaan de controller ‘@RestController’ en ‘@RequestMapping’ toe. Achter de requestmapping komt het eerste deel van de verwachte url. Standaard proberen wij hierbij ‘/api/[*controllernaam*]’. Voor een ‘UserController’ is het dus ‘@RequestMapping(/api/User)’.  
Binnen de controller zijn er een aantal objecten nodig. Allereest een ‘private final’ van alle services die benodigd zijn binnen de controller. Voeg hierna een ‘@Autowired’ constructor toe waarbij deze services geïnstantieerd worden.

Afbeelding met tekst

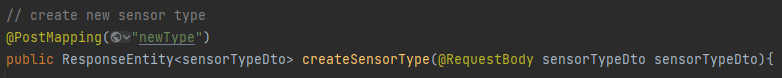
Automatisch gegenereerde beschrijving

Om calls te kunnen ontvangen, zullen er boven de methodes de correcte mapping aangegeven moeten worden. Dit is voor een GET ‘GetMapping([*url*])’, POST ‘PostMapping([*url*])’, PUT ‘PutMapping([*url*])’ en voor DELETE is het ‘DeleteMapping([*url*])’. ‘[*url*]’ishierbij het einde van de http die ontvangen wordt. Deze kan ook leeggelaten worden. Voor ‘GetMapping(page)’ zal de url dus bijvoorbeeld eindigen op “/api/User/page”.

Binnen de url kan ook tussen accolades een stuk data meegegeven worden ‘{userId}’. Dit zal een getal of string zijn binnen de url. Om deze op te vangen, is er een ‘@PathVariable’. Als er een body meegestuurd wordt met de http, zal deze worden opgevangen met ‘@RequestBody’. Dit is over het algemeen een DTO die in beide front- en back end hetzelfde moet zijn opgesteld.

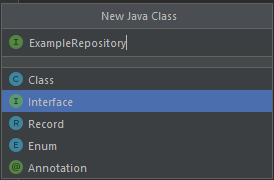
Afbeelding met tekst

Automatisch gegenereerde beschrijving



#### 

#### Repositories en Services

Om connectie te kunnen leggen met een database, zullen een Service en Repository nodig zijn. Services leggen de voornaamste logica en algoritmes af, waar repositories de werkelijke connectie en sql afhandelen. Om een repository aan te maken, selecteer binnen de package ‘repositories’ *New > Java Class*. Geef het een passende naam en **vergeet niet** om te selecteren dat het een Interface is. 

Voeg hierna aan de interface toe ‘Extends JpaRepository<[*Entity*], Long>’. Dit zorgt ervoor dat de [*Entity*](#_Entities_en_DTO’s) die hier ingevoerd word als subject van de database-tabel genomen word. Daarnaast zal de repository automatisch al een aantal queries aanmaken, zoals een GetByID, GetAll, DeleteAll en DeleteByID.



Zodra er queries nodig zijn die niet hierbinnen vallen, kunnen deze handmatig toegevoegd worden aan de repository. Let er hierbij op dat de benaming zo omschrijvend mogelijk is over wat de query doet.

Voor het toevoegen van een service selecteer je in de package ‘services’ *New > Java Class*, waarna je de class een passende naam geeft.  
Voeg hierna boven de class de annotation voor het springframework stereotype ‘@Service’ toe.

Voeg als ‘*private final’* attributen de benodigde repositories toe.  
Maak hierna een constructor aan voor de Service. Voeg hierboven ook de annotation ‘@Autowired’ toe. Geef binnen deze constructor de repositories aan die mee worden gegeven en vul de eigen repositories daarna hiermee in. Afbeelding met tekst

Automatisch gegenereerde beschrijving

Hierna kunnen de functies toegevoegd worden. Maak deze functie het object dat je terug wil krijgen en de data die je ervoor nodig hebt. Refereer hierbij naar de queries uit de repository die nodig zijn om dit uit te voeren. *Return* uiteindelijk het object dat je terug hebt gekregen. Indien er de optie is dat een Exception teruggestuurd kan worden, geef dit dan ook aan met ‘throws Exception’. Afbeelding met tekst

Automatisch gegenereerde beschrijving

### Cors

Cross-Origin Resource Sharing (CORS) is een op HTTP-headers gebaseerd mechanisme waarmee een server elke andere oorsprong (domein, schema of poort) kan aangeven dan zijn eigen oorsprong van waaruit een browser het laden van bronnen moet toestaan. CORS vertrouwt ook op een mechanisme waarmee browsers een "preflight"-verzoek doen aan de server die de cross-origin-bron host, om te controleren of de server het daadwerkelijke verzoek toestaat. In die preflight verzendt de browser headers die de HTTP-methode aangeven en headers die in het daadwerkelijke verzoek worden gebruikt.

Zonder een CORS file in de backend zullen de API calls geblokeerd worden in de browser. De CORS file ziet er als volgt uit:



Hier zie je dat de CORS calls van <http://localhost:3000> accepteert zolang dit "Get","Post","Put" of "Delete" calls zijn

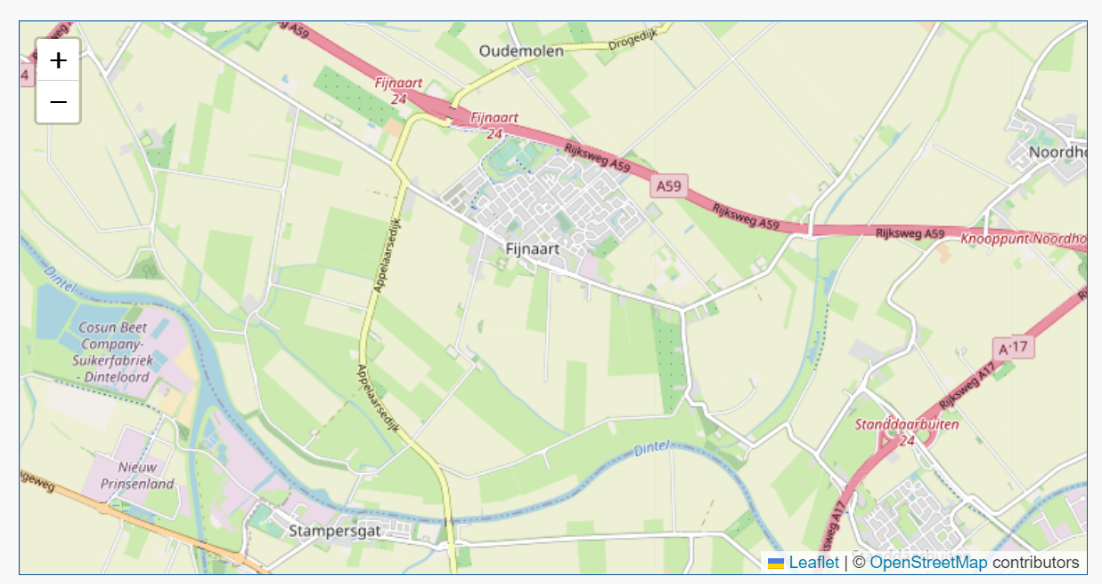
# Leaflet

## Wat is leaflet?

Leaflet is een open source Javascript library dat wordt gebruikt om web mapping applicaties te maken. Een paar bekende programma's dat leaflet gebruikt zijn bijvoorbeeld buienradar.nl en Sensor.community. Leaflet is een geschikt programma voor onze applicatie vanwege de mogelijk om gebieden op te delen in zone's en omdat het open source is natuurlijk!

Leaflet is de library waarmee je kaarten kan bewerken. De kaart zelf is echter een open source programma genaamd OpenStreetMap. OpenStreetMap is een kaart van de wereld gemaakt door een community van vrijwilligers. (Voor meer informatie zie het document 'Kaartonderzoek')

Zowel leaflet en OpenStreetMap zijn open source maar je moet ze nog wel credit geven in je code.



Afbeelding: Leaflet in ons programma, merk op dat rechts onderin een vermelding van Leaflet en OpenStreetmap moet staan.

## Hoe gebruik je leaflet?

Je kunt leaflet op meerdere manieren downloaden en importeren in je code. Voor visual studio Code is het simpel om in de command console het volgende te typen:

Npm install [leaflet@1.6.0](mailto:leaflet@1.6.0)

Npm install react-leaflet@3.2.0

Omdat we react gebruiken moeten we ook nog react-leaflet installeren. React-leaflet is een versie van leaflet dat bindings tussen react en leaflet biedt. De syntax is dan ook iets anders dan voor de standaard leaflet. Daarnaast is het belangrijk om de juiste versies (zoals hierboven vermeld) te hebben aangezien niet elke leaflet, react-leaflet en react versie met elkaar compitabel zijn.

In de onderstaande links staan tutorials en documentatie om meer te leren over leaflet en leaflet-react (voor dit project is het echter aan te raden om alleen je te verdiepen in react-leaflet):

[Leaflet - a JavaScript library for interactive maps (leafletjs.com)](https://leafletjs.com/)

[Introduction | React Leaflet (react-leaflet.js.org)](https://react-leaflet.js.org/docs/start-introduction/#:~:text=React%20Leaflet%20uses%20React%27s%20context,as%20descendants%20of%20a%20MapContainer%20.)

# Reflectie

## Wat kan er beter?

Op sommige plekken in de code doen we vaak hetzelfde bijvoorbeeld bij de API calls op de front-end maar de manier waarop verschilt vaak. Zo halen we altijd data met axios op maar de manier waarop doet iedereen anders. Het zou beter geweest zijn als we met zijn allen één manier kozen en dit aanhouden.

Devops bijhouden kan beter, vaak doe je dingen rondom een taak die niet op de lijst staan. Zo weet je ook niet altijd meer waar je mee bezig was. Om dit beter te doen zou je een checklist kunnen aanmaken op de kaartjes en zo één voor één afstrepen.

We hadden niet echt een rol voor devops/planning waardoor sommige mensen extra taken kregen. De volgende keer zouden we dan beter moeten bespreken wat elke rol inhoudt en waar we misschien een extra rol voor zouden moeten hebben.

## Waar zijn we trots op?

We zijn trots op het opzetten van een nieuw project met een nieuwe taal en nieuwe features die we eerst nog niet kenden. Zo zijn we erg blij met hoe we met leaflet hebben gewerkt om een kaart te maken. Het werken met React en creëren van front-ends. We hebben kennis mogen maken met verscheidene talen (javascript, java en Css) en het gebruik van een devops omgeving om een project te managen. Verder zijn we trots op ons team en de samenwerking. In de eerste weken was het nog een beetje wennen maar we hebben altijd goed onderlinge communicatie gehad.

## Hoe zijn we met feedback omgegaan?

Tijdens dit project hebben we veel aandacht aan feedback besteed. Na elke retrospective hebben we de feedback van docenten en stakeholders toegepast op ons project. Zo hebben we na een retrospective te horen gekregen dat we meer aandacht moeten besteden aan de double-diamond methode en naar wat we nou precies gingen maken in plaats van er al aan te beginnen. We hebben toen uitgebreid de heuristics van Nielsen bestudeerd en toegepast en de onderzoeken naar de doelgroep gedaan.