

# APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER



## INSTALLATIEHANDLEIDING 'CLOSETTE'

4 februari 2022  
door Jiro Ghianni

### Inhoud

1	Lijst van benodigdheden back-end . . . . .	pag. 3
2	Stappenplan back-end . . . . .	pag. 5
3	Lijst van alle rest endpoints . . . . .	pag. 9
4	Lijst van benodigdheden front-end . . . . .	pag.
5	Stappenplan front-end . . . . .	pag.
6	Aandachtspunten front-end . . . . .	pag.

# APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

## Introductie

Houd deze app bijelkaar binnen 1 directory. Dat wil zeggen: sleep de front-end map *niet* naar een andere plek ten opzichte van de back-end. Dit in verband met de paden van een aantal standaard-afbeeldingen die als voorbeeld zijn gebruikt om de database te vullen.

Let op: wanneer je dit project in 1 keer binnen een zogenaamde 'IDE' importeert van [Github](#), dan moet je er rekening mee houden dat de start scripts in de **sub directories** staan, dus 1 map lager dan de ROOT! Dit heeft invloed op het opstarten van de front-end. Het is daarom eenvoudiger om dit project te DOWNLOADEN als ZIP, deze uit te pakken, en daarna de individuele mappen als 'root' te openen in een IDE naar keuze.

De app kan worden gedownload via <https://github.com/JirosWorld/fullstack-closette-app>

## Complete installatie kort samengevat

- 1 Download de gehele ZIP van [Github](#) (dus: *niet* uitchecken als versioncontrolled project) en pak deze uit op je lokale machine.
- 2 Open de backend-closette map in een Java-ready IDE of console.
- 3 Verander de database gegevens zoals hieronder aangegeven.
- 4 Verander het upload pad naar de Public/uploads map in de front-end directory én vul de database met uploads via Postman.
- 5 Run eventueel Maven vanuit de backend map, en start de Java applicatie '**ClosetteApp**' in de SRC/main map.
- 6 Open de frontend-closette map in een React-ready IDE of console.
- 7 Run \$ npm install en \$ npm start **vanuit** de front-end folder zelf.
- 8 Bekijk de front-end in een browser.
- 9 Zie voor verdere uitleg hieronder.

## APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

### Back-end Applicatie starten ~ uitgebreid

- 1 Lijst van benodigdheden Back-End
- 2 Stappenplan back-end
- 3 Lijst van alle rest endpoints

#### 1. Lijst van benodigdheden Back-End

Voor de installatie is een werkende internet verbinding vereist.

Wanneer we een brief of essay schrijven doen we dit meestal in de editor Microsoft Word. Wanneer we programmeren, gebruiken we ook editors om onze code te schrijven. Code editors noemen we IDE's: een Integrated Development Environment.

Voor de Closette back-end is een IDE nodig die Java code kan uitvoeren en om kan gaan met Spring Boot en Maven en een Tomcat server op kan starten, bij voorkeur IntelliJ Idea. Er moet tevens een PostgreSQL database omgeving zijn, waarmee database tabellen kunnen worden gemaakt, die eventueel beheerd kunnen worden via een 'visueel' programma zoals PgAdmin. Tevens is er voor het uploaden van de vooraf-in-te-laden foto's het programma Postman nodig.

Het is essentieel dat de machine, waarop de Closette app draait, Java als taal aankan. Dit wordt mogelijk wanneer je een zogenaamde JDK installeert. Dit kan o.a. door op de Mac het 'console' programma 'Terminal' te gebruiken, dat zich in de *Applications/Utilities* map bevindt. In deze Terminal kun je commando's typen zoals bijvoorbeeld:

```
$ /usr/libexec/java_home -V
```

Hiermee kun je alvast bekijken welke Java versie je hebt.

Eenvoudiger: Via het nog te installeren programma IntelliJ kun je nieuwere versies van Java downloaden.

Alle onderstaande, benodigde programma's kunnen op een Mac in de Applicatie map worden geïnstalleerd, en zullen meteen 'out of the box' werken. Installeer deze eerst:

# APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

## Download links (gratis)

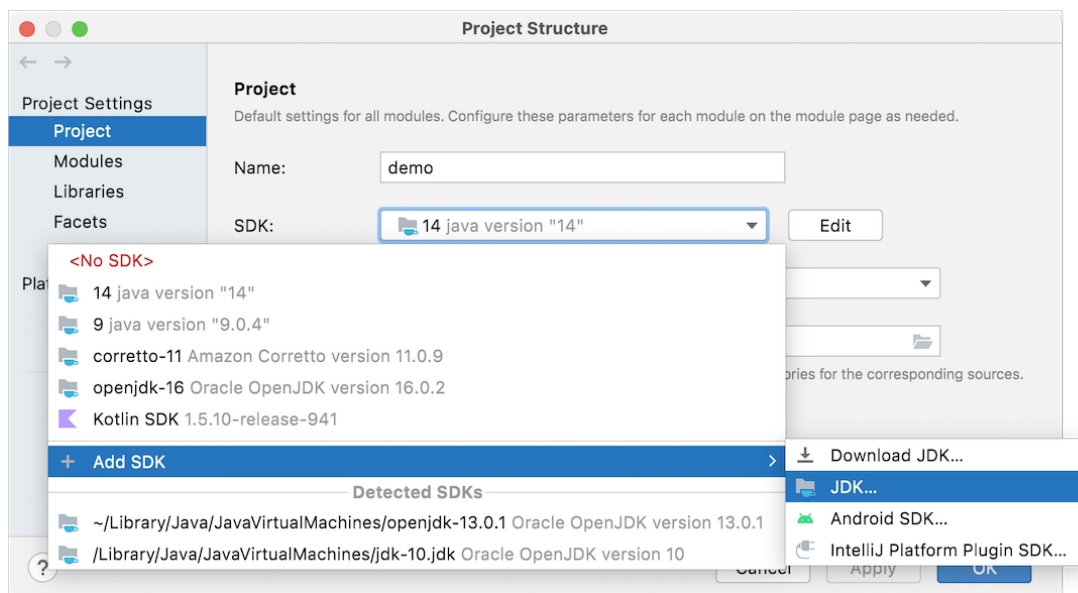
- PostgreSQL <https://www.postgresql.org>
- pgAdmin <https://www.pgadmin.org>
- IntelliJ <https://www.jetbrains.com/idea/>
- Postman <https://www.postman.com/downloads/>

Wanneer je al deze applicaties hebt geïnstalleerd, kun je IntelliJ openen, en via >> **'File' >> new >> Project** – een nieuw project maken.

Kijk bovenin het pop-up scherm dat verschijnt wanneer je een nieuw project probeert aan te maken: daar staat een drop-down menu met de tekst "...JDK..." of misschien met een nummer zoals "...14..." of "...17..." of misschien staat er helemaal niets.

Klik in ieder geval op het dropdown menu en kies de optie "Download JDK" en kies voor versie 17 – van OpenJDK.

Het maakt niet uit welke 'open' versie je kiest. Deze zijn allemaal opensource en dus gratis te gebruiken. Wanneer je alles geïnstalleerd hebt, en dit nieuwe project afgesloten hebt, ben je klaar voor de installatie van de Closette app zelf.



# APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

## 2. Stappenplan back-end

### Back-end aanpassingen database

1. Open PgAdmin en maak een nieuwe database user aan.

Belangrijk: verander deze lokale instellingen!

De database settings, in (bijvoorbeeld) pgAdmin:

- \* postgresql database op localhost: 5432 (port 5432)
- \* databasenaam: closette
- \* user/owner: springboot
- \* password: springboot

Maak dus eerst een user aan met naam/wachtwoord springboot.

En daarna een lege database met de naam closette, gekoppeld aan deze user.

### Standaard users voor het gebruik van de app zelf

(dit zijn de users die kunnen inloggen en nieuwe toiletten kunnen toevoegen etc.)

- admin - password
- user - password
- tester - password

### Back-end pad voor uploads

Belangrijk: Pas in de `main/resources/application.properties` het Upload-pad voor afbeeldingen aan: voor Mac gebruikers zal dit vlekkeloos verlopen omdat daar nooit Backslashes gebruikt worden, maar Windowsgebruikers moeten opletten dat er wellicht een Backslash in hun code nodig is daar waar nu een Slash staat. Het lokale pad hier dienen alle gebruikers in ieder geval aan te passen vanaf het 'Users' path naar de locatie van de front-end public/uploads directory op jouw eigen machine:

`app.uploads= /Users/jolarti/webdevelopment/closette/frontend-closette/public/uploads`

Let op: dit lokale pad MOET verwijzen naar de **uploads** directory die in de **public** Directory staat van de **Frontend**-closette map:

## APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER



### *upload pad directory*

Het mag dus niet een willekeurige map zijn, maar moet specifiek naar deze map verwijzen. Dit in verband met een aantal 'prefilled' images = vooraf gevulde afbeeldingen die in die upload staan en door de database worden ingelezen waardoor de app er representatief uitziet.

Deze app is gebouwd op een **Mac**, dus het is mogelijk dat Windows/Linux gebruikers in de code nog 1 extra aanpassing moeten doen; zie het commentaar in het bestand in de backend-closette directory in regel 61 in het bestand in /main ... /service/FileStorageService

**Windows** users moeten hier mogelijk een BACKSLASH invoeren in plaats van de SLASH die daar staat, resultaat:

```
Path filePath = Paths.get(fileStoragePath + "\\\" + fileName);
```

## De vooraf ingeladen foto's

De database heeft 15 foto's die vooraf ingeladen moeten worden, omdat deze gekoppeld zijn aan de id's van de toilet- entries. Helaas is het niet voldoende om alléén de *data.sql* in te laden, omdat de foto's zijn omgezet naar een zogenaamde 'byte array' die voor elke lokale machine een uniek cijfer geeft. Dit unieke lokale gedrag schijnt iets met de *data access object* (DAO) te maken te hebben.

Gelukkig is er een oplossing: via Postman – of zelfs via de front-end – kunnen de foto's die reeds in de **Uploads** map staan, allemaal opnieuw naar jouw lokale database geüpload worden, zodat deze lokaal op jouw eigen machine een nieuw uniek cijfer krijgen. Dit cijfer heet 'doc\_file' en dient veranderd te worden in het **data.sql** bestand dat in de Resources map staat, op deze manier:

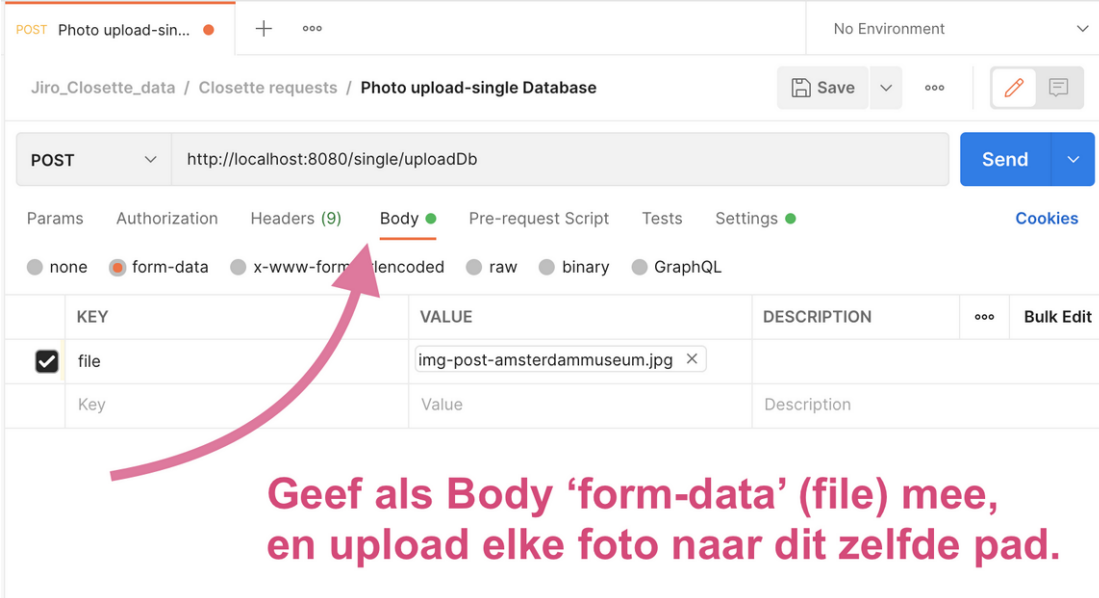
Open in Postman het request dat een POST doet naar het pad

## APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

`http://localhost:8080/single/uploadDb`

en upload daar 1-voor-1 alle images uit de **Public/Uploads** directory die in de **frontend-closette** map staan:



Geef als Body 'form-data' (file) mee,  
en upload elke foto naar dit zelfde pad.

Bekijk daarna welke cijfers er nu zijn meegegeven in jouw eigen PostgreSQL database:

	id [PK] bigint	doc_file oid	file_name character varying (255)
1	199	54101	no-image1.png
2	200	54559	img-post-amsterdammuseum.jpg
3	250	54560	img-post-artistic-toilet.jpg
4	300	54561	img-post-back-to-wall-toilet.jpg
5	400	54562	img-post-black-white-toilet.jpg
6	500	54565	img-post-deco-toilet.jpg
7	600	40649	img-post-krakers-toilet.jpg
8	700	40706	img-post-urinoirs.jpg
9	800	40766	img-post-French_Squatter_Toilet.jpg
10	900	40767	img-post-Japans-Toilet.jpg
11	1000	40769	img-post-victorian-toilet.jpg
12	1100	54558	img-news-UNISEX.jpg
13	1200	54566	img-post-outhouse-toilet.jpg
14	1300	54557	img-news-Unisex-Toilet.png
15	1400	54556	img-news-unisex-sign.png
16	1450	54101	no-image2.png

Deze doc\_file nummers  
zijn uniek voor elke lokale  
installatie.

Maak zelf nieuwe  
doc\_file nummers  
door al deze afbeeldingen  
eerst naar de database  
te uploaden.  
Noteer die nummers, en  
plaats ze bij de juiste ID.



## APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

Nu kun je het **data.sql** bestand aanpassen en bewaren, vóórdat je de server start (of herstart):



The screenshot shows a SQL script for inserting data into a 'photos' table. The script includes comments in Dutch explaining the process: inserting base64 images, changing ByteArray numbers, uploading photos from the 'Uploads' directory, and providing new doc\_file numbers. The script uses an INSERT INTO statement with columns (id, file\_name, doc\_file) and a VALUES list of 15 rows. Red annotations highlight the 'doc\_file' column and provide instructions to update these values in the 'main/resources/data.sql' file.

```
-- insert ALL base64 images in prefilled data
-- Dit is waar de ByteArray nummers veranderd kunnen worden:
-- Upload eerst zelf de foto's uit de Uploads directory
-- en geef dan de nieuwe doc_file nummers per bestand mee
INSERT INTO photos (id, file_name, doc_file)
VALUES ('199', 'no-image1.png', '54101'),
('200', 'img-post-amsterdammuseum.jpg', '54559'),
('250', 'img-post-artistic-toilet.jpg', '54560'),
('300', 'img-post-back-to-wall-toilet.jpg', '54561'),
('400', 'img-post-black-white-toilet.jpg', '54562'),
('500', 'img-post-deco-toilet.jpg', '54565'),
('600', 'img-post-krakers-toilet.jpg', '40649'),
('700', 'img-post-urinoirs.jpg', '40706'),
('800', 'img-post-French-Squatter-Toilet.jpg', '40766'),
('900', 'img-post-Japans-Toilet.jpg', '40767'),
('1000', 'img-post-victorian-toilet.jpg', '40769'),
('1100', 'img-news-UNISEX.jpg', '54558'),
('1200', 'img-post-outhouse-toilet.jpg', '54566'),
('1300', 'img-news-Unisex-Toilet.png', '54557'),
('1400', 'img-news-unisex-sign.png', '54556'),
('1450', 'no-image2.png', '54101');
```

Zet op deze plekken in de backend-closette dir, in het `main/resources/data.sql` bestand je nieuwe lokale `doc_file` nummers, behorend bij het juiste ID nummer/file\_name.

### JAVA versie

Zoals eerder genoemd: dit project werkt alleen wanneer je JDK versie 17 of hoger hebt geïnstalleerd op je computer. Wanneer je deze niet hebt, kun je deze downloaden via <https://jdk.java.net>

### Installatie en opstarten back-end

Als je het project gedownload hebt naar jouw locale machine, en de aanpassingen hierboven hebt gedaan, installeer je eerst de back-end. Liefst via een Java-ready/Maven-ready IDE zoals IntelliJ maar het kan ook door de backend-closette folder te openen in elke terminal van jouw keuze.

Je IDE kan als notificatie de vraag stellen of Maven dit project mag vertrouwen, zeg dan 'yes' op de trust vraag. En accepteer JPA buddy en de andere plugins wanneer daar akkoord op wordt gevraagd. Indien IntelliJ vraagt om een DataSource of SQL dialect te kiezen, doe dit dan NIET, laat het zoals het is. En klik OK op de JPA Buddy default configuratie.

In de meeste gevallen volstaat het om de `backend-closette` map te openen in een IDE, die Java kan compileren, en het pom-bestand van Maven te laten installeren, waarna de Main klasse `ClosetteApp` gedraaid (run) kan worden. (Het back-end



# APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

startscript staat in `bckend-closette/pom.xml`). Maar installatie en runnen vanuit de `backend-closette` directory kan ook in 1 keer via een terminal naar keuze met het commando:

```
$ mvn spring-boot:run.
```

In de terminal kan de back-end app gestopt worden met de toetscombinatie `ctrl + 'C'`.

Bij back-end problemen: `* install pom.xml * of type ./mvnw verify (mac/linux) of mvnw.cmd verify (windows)` uit in de terminal van de project directory/folder.

Laatste stap voor de back-end:

Start de Main klasse `ClosetteApp` op (= run).

## 3. Lijst van alle rest endpoints

Onderstaande endpoints heb ik tevens gepubliceerd op deze documentatie site (inclusief beschrijvingen) die het veel geakkelijker maakt om al deze endpoints te kopiëren en zelf te gebruiken:

<https://documenter.getpostman.com/view/17991980/UVeCR95T> - en in de 'LEESMIJ' map staat een makkelijk te importeren verzameling van alle Postman requests (met JSON).

Hieronder volgt de volledige lijst met endpoints vanaf **http://localhost:8080**

- alléén wanneer een endpoint JSON bevat, dan staat dit er bij, als het geen JSON bevat dan staat er niets naast.

Ook wanneer het punt openbaar is, staat dit (meestal) niet vermeld.

- {PUT [/users/{username}]} = beveiligd met JWT token.

JSON:

```
{
  "username": "gebruikerette",
  "password": "12345",
  "email": "mail@mail.com"
}
```

- {PUT [/toilets/{id}]} = beveiligd met JWT token.

```
{
  "title": "Saaie locatie",
  "latitude": "50.2121211321"
}
```

- {PUT [/news/{id}]} = beveiligd met JWT token.

```
{
  "title": "Titeltje",
  "description": "Vervang door deze nieuwe tekst",
}
```

# APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

```
"paragraph": "Bijwerken met deze nieuwe tekst vervang door deze nieuwe tekst vervang door
deze nieuwe tekst"
}
```

- {POST [/users/register]} = openbaar te benaderen.

```
{
  "username": "gebruikerette",
  "password": "12345",
  "email": "mail@mail.com"
}
```

- {POST [/users/{username}/authorities]} ← **BEVEILIGD, alleen toegang voor admins.**
- {POST [/toilets/{id}/ratings]}= beveiligd met JWT token.
- {POST [/toilets/{id}/photos]}= beveiligd met JWT token.
- {POST [/toilets]}= beveiligd met JWT token.
- {POST [/single/uploadDb]}= beveiligd met JWT token.
- {POST [/single/upload]}= beveiligd met JWT token.
- {POST [/ratings]}= beveiligd met JWT token.
- {POST [/news]}= beveiligd met JWT token.
- {POST [/authenticate]} ← **Log hiermee in, dit levert de JWT token.**
- {PATCH [/users/{username}/password]}= beveiligd met JWT token.
- {PATCH [/toilets/{id}]}= beveiligd met JWT token.
- {PATCH [/toilets/{id}/ratings]}= beveiligd met JWT token.
- {PATCH [/single/uploadDb/{id}]}= beveiligd met JWT token.
- {PATCH [/news/{id}]}= beveiligd met JWT token.
- {GET [/users/{username}/authorities]}
- {GET [/users/{username}]}
- {GET [/users]} ← **BEVEILIGD, alleen toegang voor admins.**
- {GET [/toilets/{id}/ratings]}
- {GET [/toilets/{id}]}
- {GET [/toilets]}
- {GET [/ratings/{id}]}
- {GET [/ratings]}
- {GET [/photos/{id}]}
- {GET [/photos]}
- {GET [/news/{id}]}
- {GET [/news]}
- {GET [/downloadFromDB/{fileName}]}
- {GET [/download/{fileName}]}

## APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

- {GET [/alluploads]}
- {DELETE [/users/{username}/authorities/{authority}]}= beveiligd met JWT token.
- {DELETE [/users/{username}]}= beveiligd met JWT token.
- {DELETE [/toilets/{id}]}= beveiligd met JWT token.
- {DELETE [/news/{id}]}= beveiligd met JWT token.
- { [/error]}
- { [/error], produces [text/html]}

### Server requests

De back-end wordt aangesproken door de front-end, maar als je alleen via Postman requests wil doen, dan kan dat ook: Open <http://localhost:8080> voor requests.

De Postman export staat ook nog eens dubbelop in de 'back-end documentation' map: deze kun je ook importeren in Postman en daarin gemakkelijk meteen uitvoeren.

---

## Front-end Applicatie starten ~ uitgebreid

- 4 Lijst van benodigdheden front-End
- 5 Stappenplan front-end
- 6 Aandachtspunten front-end

### 4. Lijst van benodigdheden front-end

Voor de installatie is een werkende internet verbinding vereist.

Wanneer we programmeren, gebruiken we speciale editors om onze code te schrijven. Code editors noemen we IDE's: een Integrated Development Environment. Bij deze Closette app raad ik aan te werken met WebStorm. WebStorm is een betaald product, in tegenstelling tot andere bekende gratis IDE's (zoals Visual Studio Code met React plug-ins, Atom of Sublime Text). Voor de front-end is een IDE nodig die javascript en JSX/React code kan uitvoeren. Ook dient Nodejs en NPM (15.4.0 of hoger) geïnstalleerd te zijn.

#### Download links:

- Webstorm (betaald) <https://www.jetbrains.com/webstorm/>
- Nodejs <https://nodejs.org/en/download/>
- (NPM zit al in NodeJs)
- een moderne browser <https://www.google.com/chrome/>

## APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

Al deze applicaties werken direct 'out of the box'.

### 5. Stappenplan front-end

Als je het project gedownload hebt naar jouw locale machine, en je de back-end hebt opgestart, is het tijd om de front-end te installeren. De front-end zorgt ervoor dat je de Closette App daadwerkelijk kunt gaan gebruiken, in een webbrowser.

Om deze app te kunnen laten 'afspelen' heb je de zogenaamde 'NPM node bestanden' nodig. Deze dient je zelf te installeren. Je kunt dit doen door je IDE programma (Webstorm) te openen en vandaar uit de `frontend-closette` map te openen! Die front-end map wordt dan automatisch de 'hoogste' directory die je binnen je IDE programma kunt zien. De allerhoogste stap in een hiërarchie noemen we de 'root' = het startpunt. Dit startpunt is essentieel om de front-end te kunnen installeren en te kunnen afspelen.

Dus: installeer nu eerst de `node_modules` door het volgende commando in de terminal van Webstorm te runnen:

```
$ npm install
```

Dit betekent dat je, wanneer je dit project importeert vanuit Github (version control) je het start-script dan niet meteen kunt runnen (het front-end startscript staat namelijk 1 map lager, in `frontend-closette/package.json`). Wanneer je een IDE als Webstorm wilt gebruiken, open het project dan *niet* als 'version-control' project maar: ga in een browser naar Github, download de repository als ZIP, pak dit uit waar je maar wilt, en open dan alleen de `frontend-closette` map als 'root' in een IDE naar keuze (bijvoorbeeld Webstorm, of Visual Studio met een React plug-in). Daar kun je wederom je '`npm install`' commando tikken en wachten tot de NPM installatie klaar is.

Wanneer dit klaar is, kun je (wederom vanuit de terminal bij de `frontend-closette` map) de applicatie starten met behulp van:

```
$ npm start
```

Als je dit project opent in Webstorm kun je hiervoor ook het NPM START afspreekknopje gebruiken.

Je IDE kan als notificatie de vraag stellen of je browser (bijvoorbeeld Chrome) deze app mag vertrouwen; beantwoordt bevestigend op elke trust vraag. Open <http://localhost:3000> om de web-app in een browser te bekijken en te bedienen.

Axios, ESLint, React Router 5.2, React-Hook-Form, JWT-decoder, emailJS etc. zijn reeds gesaved in JSON builder en worden automatisch mee geïnstalleerd.

In de terminal kan de front-end app gestopt worden met `ctrl + 'C'`.

## APPLICATIE JIRO GHIANNI

INTEGRALE EINDOPDRACHT FULLSTACK DEVELOPER

### 6. Aandachtspunten front-end

#### Nodejs / npm versie

Dit project werkt alleen wanneer je Nodejs versie Node.js 15.4.0 of hoger hebt geïnstalleerd op je computer. Wanneer je deze niet hebt, kun je deze downloaden via <https://nodejs.org/en/download/releases/>

#### Voorkeur browsers

NB: Gebruik bij voorkeur **Chrome**, **Edge** of **Opera**. Natuurlijk werkt alles ook in Firefox en Safari maar sommige fonts worden daarin niet mooi 'bold' gerendered. Ook wordt op sommige afbeeldingen het 'filter' CSS attribuut gebruikt, deze werkt momenteel nog niet altijd goed in Edge (en al helemaal niet in Explorer).

~ Jiro Ghianni  
2021 / 2022