# Project overview

Mijn project bestaat uit een react front-end , een node.js backend in combinatie met express. Aansluitend gebruik ik een mongo database die mijn data bijhoudt

# Containers & Dockerfile

## Voorbereiding

* Instaleer docker
* Start docker
* Installeer VS code

## Backend

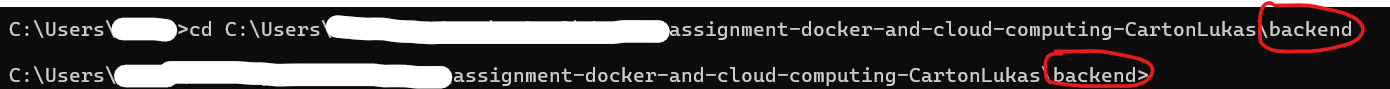
1. In de root folder van de back-end maak je een file aan *Dockerfile*Afbeelding met tekst, schermopname, Lettertype, ontwerp

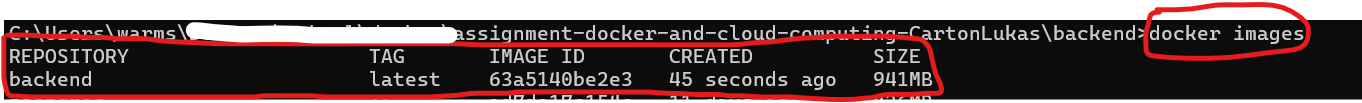
   Automatisch gegenereerde beschrijving
2. In deze file zet je wat er nodig is voor een docker image te kunnen maken van deze backend

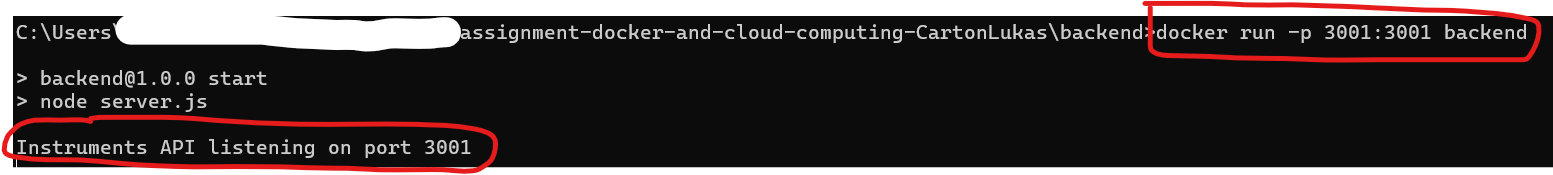
Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

1. Je navigeert naar de folder van de backend



1. In de map voer ik het volgende comando uit : *docker build -t backend .*  Dit zal een image maken van mijn back end na even wachten krijg je de terminal terug en is de image aangemaakt
2. Controleer of de image is aangemaakt door het commando *docker images*  Uit te voeren en controleer of de image backend er bij staat   
   
3. Nu kan je de container runnen door het commando *docker run -p 3001:3001 backend* Uit te voeren



1. De backend is nu aant het runnen in een docker container
2. Deze kan niet verbinden met de mongo database daarvoor heb je een netwerk of een docker-compose file nodig die je vind in het volgende deel

## Front end

1. In de root folder van de frontend maak je een *Dockerfile* aan   
   Afbeelding met tekst, schermopname, Lettertype

   Automatisch gegenereerde beschrijving
2. In deze file zet je wat er nodig is voor een docker image te kunnen maken van deze frontend we zullen nginx gebruiken om dat dit efficent en veilig is

We doen dit in 2 stappen :

* React app builden
* React app plaatsen op nginx en dit hosten

# stap 1: React app builden

# de basis img voor de docker

FROM node:16 AS build

# de plaats waar we zullen de files zetten

WORKDIR /app

# de packages.json file in kopieren naar de docker zodat ze kunnen geinstallerd worden

COPY package.json package-lock.json ./

# De dependencies instaleren

RUN npm install

# De app kopieeren naar de docker

COPY . .

# De app buildend voor productie

RUN npm run build

#----------------------------------------------------------------

# stap 2: React app plaatsen op nginx en dit hosten

#de basis img van nginx

FROM nginx:alpine

# De build files van react kopieren naar de juiste map in nginx

COPY --from=build /app/build /usr/share/nginx/html

# De poort 80 openzetten

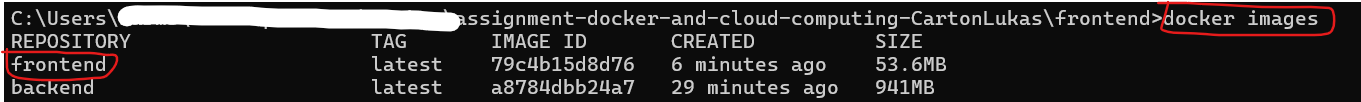
EXPOSE 80

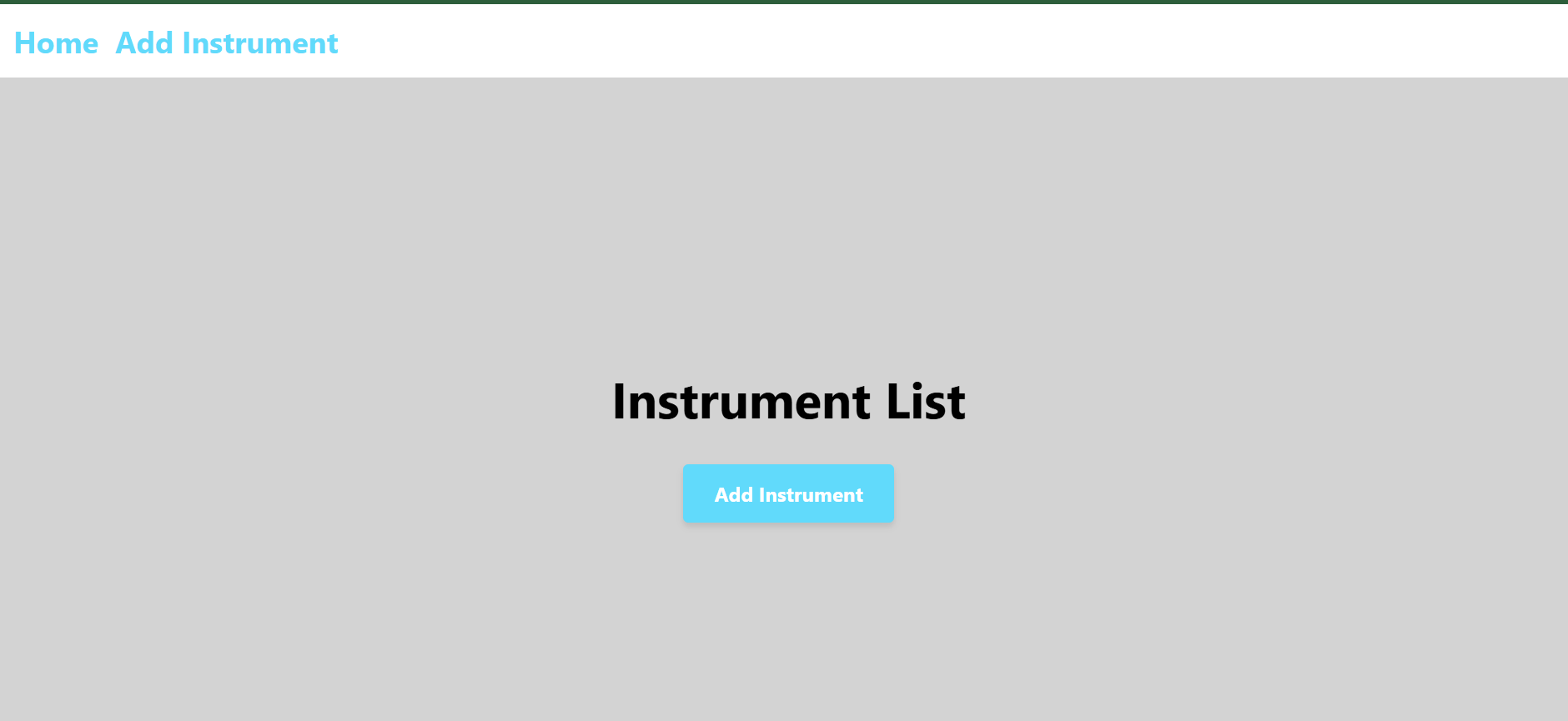
# Nginx server starten

CMD ["nginx", "-g", "daemon off;"]

1. Je navigeert naar de folder van de frontend



1. In de map voer ik het volgende comando uit : *docker build -t frontend .*  Dit zal een image maken van de frontend na even wachten krijg je de terminal terug en is de image aangemaakt
2. Controleer of de image is aangemaakt door het commando *docker images*  Uit te voeren en controleer of de image frontend er bij staat   
   
3. Nu kan je de container runnen door het commando *docker run -p 80:80 frontend* Uit te voeren
4. De frontend is nu aant het runnen in een docker container
5. Je kan deze bekijken op <http://localhost:80>



1. Deze kan niet verbinden met de backend daarvoor heb je een netwerk of een docker-compose file nodig die je vind in het volgende deel

# Docker compose

## opzetten

1. Open de file *docker-compose.yaml* de vind je onder de map *docker*
2. In deze file zet je de configuratie voor de docker container die de 3 services zal bevatten

version: '3.8'

services:

  frontend:

    build:

      context: ./frontend  # locatie van de frontend

    ports:

      - "80:80"  # Poort 80 open zetten zodat je kan verbinden met de frontend

    depends\_on:

      - backend  # Frontend pas opstarten na de backend

  backend:

    build:

      context: ./backend  # locatie van de backend

    ports:

      - "3001:3001"  # Poort 3001 open zetten zodat je kan verbinden met de backend

    environment:

      - MONGO\_URI=mongodb://db:27017/instrumentsdb  # env variable zetten voor de connectie met de database in de backend

    depends\_on:

      - db  # backend pas opstarten na de database

  db:

    image: mongo  # De img voor een mongo database binnenhalen

    ports:

      - "27017:27017"  # De poort 27017 openen zetten zo dat je kan verbinden

    volumes:

      - mongo-data:/data/db  # Data opslaan ook na stoppen van de container

volumes:

  mongo-data:  # data opslaen ook na stoppen van de container

1. Navigeer naar de map waar de *docker-compose.yaml* file staat



1. Voer het volgende command uit om de docker container te starten (ik geef het de naam project ) *docker-compose -p project up -d*
2. Na eventjes ziet u het volgende op de terminal dit betekend dat de container is opgestart

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

1. Je kan nu de website bereiken op <http://localhost:80>

## Testen

### frontend

1. Connecten met de frontend op <http://localhost:80>  
   Afbeelding met tekst, schermopname, Lettertype, software

   Automatisch gegenereerde beschrijving
2. Instument toevoegen aan de lijst   
   Afbeelding met tekst, schermopname, Lettertype, ontwerp

   Automatisch gegenereerde beschrijving Afbeelding met tekst, schermopname, Lettertype, nummer

   Automatisch gegenereerde beschrijving

### Backend

1. Connecten met de backend op <http://localhost:3001/>

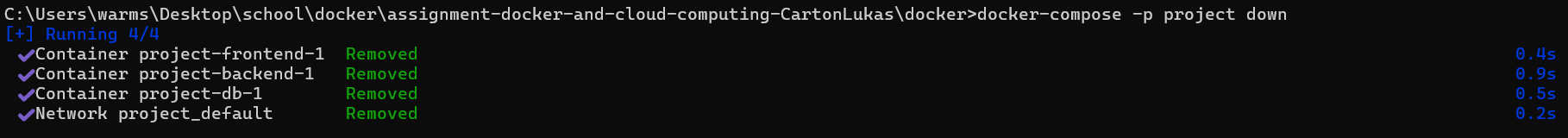
Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving

1. Controleren of dat ons nieuwe instrument er bij staat   
   Afbeelding met tekst, schermopname, grafische vormgeving, Lettertype

   Automatisch gegenereerde beschrijving

## Container stoppen

1. **Doe dit door het commando *docker-compose -p project down* in de terminal uit te voeren
2. Controleer of de container daadwerkelijk is gestopt door het commando *docker ps* in de terminal uit te voeren en te controlen of project er bij staat.

**