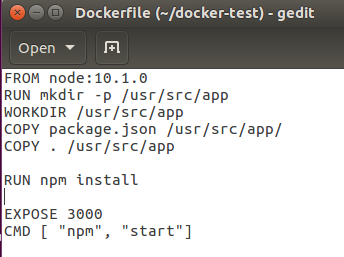
**Installations- und Benutzerbeschreibung**

Before we start, I built a Node.js application.Using the interface below, we can add new users and delete old users with connection to MongoDB.

Next we have to do these steps:

1. 1. written a simple **<Dockerfile>**
2. 2. used Docker Compose to define our services
3. 3. and successfully dockerised our application

To run this application in a [Docker](https://www.docker.com/community-edition) container, we will write a **<Dockerfile>** using the official node image from the Docker Hub registry. We will then use [Docker Compose](https://docs.docker.com/compose/install/), a tool for running multi-container applications, to spin up our containers and run our app.



We are essentially using a bunch of instructions to build our own <**node**> image.

**<FROM>** lets us specify which base image from Docker Hub we want to build from. In our case, we are using the latest version of the official **node** image.

**<RUN>** lets us execute a command, which in our case is to create a new directory.

**<WORKDIR>** sets this newly created directory as the working directory for any <**COPY>**, <**RUN>** and <**CMD>** instructions that follow in the **Dockerfile.**

**<COPY>** is pretty straightforward and lets us copy files or a whole directory from a source to a destination. We are going to <**COPY>** the <**package.json>** file over to our working directory.

**<RUN>** lets us execute the <**npm install>** command which will download all the dependencies defined in <**package.json>**.

**<COPY>** lets us copy our entire local directory into our working directory to bundle our application source code.

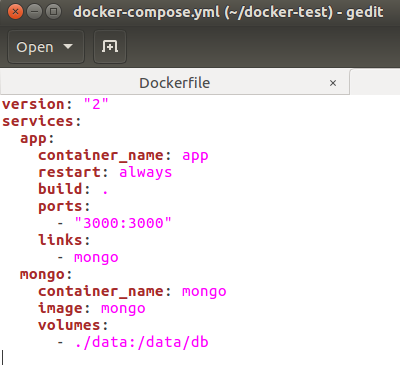
**<EXPOSE>** exposes a port which the container will listen on.

And finally, <**CMD>** sets the default command to execute our container.

**That’s the Node part done, now for MongoDB.**

We could build our own Mongo image but wherever possible, we should look to use official images.This is beneficial as it saves us a fair amount of time and effort — we don’t have to spend time creating our own images or worry about the latest releases or applying updates. All of that is taken care of by the publisher of the image.

**Then we will add a <docker-compose.yml> file to define the services in our application.**

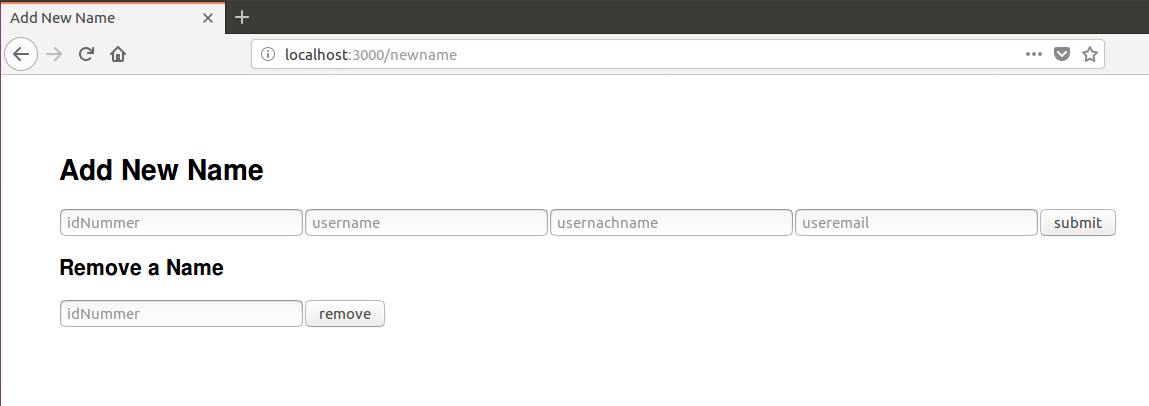


Here is defining a service called **<docker-test>**,adding a container name for the **<docker-test>** service as giving the container a memorable name makes it easier to work with and we can avoid randomly generated container names instructing Docker to <restart> the container automatically if it fails,

building the **<docker-test>** image using the **Dockerfile** in the current directory and mapping the host port to the container port.

We then add another service called **<mongo>** but this time instead of building our own **<mongo>** image, we simply pull down the standard **<mongo>** image from the Docker Hub registry.For persistent storage, we mount the host directory **</data >**(this is where the dummy data I added when I was running the app locally lives) to the container directory **</data/db>**, which was identified as a potential mount point in the **<mongo> <Dockerfile>** we saw earlier.

We can now navigate to the project directory, open up a terminal window and run **<docker-compose build>** then **<docker-compose up>** which will spin up two containers and aggregate the logs of both containers.



Now if we point our browser to the application URL **<localhost:3000/newname>**,add a new user and hit the Submit button or delete a user and hit the Remove button,then we should now be able to see the newly change in **<localhost:3000/namelist>**.

