Lecture Sheet 3

General procedure: Each student / team can earn 100 points across the whole course. 30 points for active participation during the course (be present, ask questions, take part in discussions, et cetera) and 70 points for the final oral presentation (see introductory presentation held in the first lecture). Each lecture is accompanied by an exercise, which enables students to apply their knowledge in practice. In the last lecture, each student / team presents all exercises in one large oral presentation of 15 minutes presentation time and 5 minutes discussion time. The result of each exercise needs to be submitted until one day before the next lecture is planned. Each student / team is incurred one point of the 30 active participation points for each day the submission is delayed.

Use Case Extension:

RentMe Ltd. is now running autonomously in a Docker-containerized environment. Program interactions are possible by HTTP requests containing and replying JSON.

The next step consists of an extension to make interactions with the application more comfortable. For this purpose, add a HTML/JavaScript-based (JavaScript is used for requests to the backend after the webpage is loaded) frontend and put the focus on showing the current state of the cars and customers. Therefore, this extension translates the output of the endpoints, which were implemented in the lecture sheet 2 to show the states of cars and customers, to a human-friendly and -readable format. To fulfill this requirement, it is sufficient to embed the content in a HTML-based table with the most important columns.

Since another technology is introduced, also another Docker image is recommended to use. This Docker image contains a HTML static webpage showing the response in terms of a HTML-based table with the most important columns. It is important that the state of the cars and customers can be seen. To serve this HTML, extend the NGINX server[4] and copy & paste the static web page in. Choose an appropriate port to serve the HTML on the host system and explain your decisions in the documentation.

Because two Docker images are in use (the Python program as backend and the HTML output as frontend), configure both services in a Docker Compose file[5]. The Docker Compose file bundles different Docker services together and orchestrates these. It is used to start a bundle of Docker images in a pre-configured way. After executing a bundle of services with Docker Compose, multiple containers are running as configured in the Docker Compose file "docker-compose.yml". Make sure that Docker Compose is installed on the system (it is not necessarily installed once just Docker is installed).

[4] https://hub.docker.com/_/nginx
[5] https://docs.docker.com/compose/