This page contains a guide on how to build and deploy the Quick Check software.

Build & Deployment Process

Prerequisites

- Docker is required to build and deploy our application.

Build

To build and deploy our application, use command:

> ```
> docker-compose up --build
> ```

This command builds and starts the following three containers:

database

- Description: SQL database to store and persist application data.
- Base image: `mysql:8`

service

- Description: Backend providing an interface to the data stored in the database through a web API.
- Base image: `openjdk:11`
- Access URL: `http://localhost:8080`

frontend

- Description: Webserver that hosts the quickcheck website.
- Base image: `nginx:alpine`
- Access URL: `http://localhost:3000`

Development Environment

Frontend

For setting the development server up Node.js and Yarn is required:

1. Install Node.js

`yarn install` first. Afterwards change back to the project root directory by using `cd ..`

1. Run the backend containers by executing `docker-compose up service`
2. Run the frontend container by executing `cd frontend && yarn start`

Now, a browser window navigating to `http://localhost:3000` should pop up. If not navigate to this site manually to access the webserver.

If additional packages are required navigate to the `/frontend` folder and run `yarn add [package name]`.

Backend

Extending the service

To be able to deploy your changes to the service you need to fire a build command within your IDE.

To fire a build command in Intellij press `command + F9` on Mac or from `Menu Build --> Build Project`

The build command causes springboot to restart within the container and apply your changes.

Testing Backend

1. Launch development container
2. Open another tab in your terminal
3. Run the following commands
   i. `docker exec -it service bash` (enter docker container service with a bash script)
   ii. `cd app` (directory app contains pom.xml, needed to run mvn command)
   iii. choose one of the following:
      a. `mvn test` to run all tests
      b. `mvn test -Dtest=classname` to run all tests within classname
      c. `mvn test -Dtest=classname#method` to run a single method within classname