Lab 02

Gruppe 4:

Ole Kirchner
Jasmin Furche
Andrey Krasavin
Anas Yunis Altaee

Projektname: Wolke
Repo URL: https://github.com/gelkoh/wolke

Projektvorschlag: Unsere Idee ist es, einen kleinen File-Hosting-Service mit dem Namen “Wolke” zu entwickeln. Bei diesem Service soll man sich registrieren können und nach anschließender Anmeldung Dateien hochladen können. Wir möchten uns dabei erst einmal auf kleine Dateien wie zum Beispiel einfache Textdateien beschränken, da wir noch nicht wissen, wie realistisch es ist Bilder oder gar Videos in unserer Cloud speichern zu können. Es handelt sich bei unserer Idee um eine CRUD-Applikation, da man Dateien hochladen (Create), Dateiinhalte einsehen und lesen (Read), Dateien umbenennen (Update) und natürlich auch Löschen (Delete) kann. Bei der Umsetzung gibt es allerdings gewisse Spielräume, die wir erst in der späteren Entwicklung begrenzen werden. Create könnte ja beispielsweise auch heißen, dass man Textdateien auch direkt "in der Cloud” erstellen und editieren kann. Vermutlich würde das aber den Rahmen sprengen, weshalb wir uns die genaue Umsetzung noch offen halten wollen.

User stories:

Epic 1: User Authentification

- Story 1.1: Als neuer Nutzer möchte ich mich mit meiner E-Mail Adresse, einem selbst festgelegten Nutzernamen und einem selbst festgelegten Passwort registrieren können.
- Story 1.2: Als registrierter Nutzer möchte ich mich einloggen können, damit ich Zugriff auf sonst gesperrte Funktionen habe.
- Story 1.3: Als eingeloggter Nutzer möchte ich mich wieder ausloggen können.
- Story: 1.4: Als Nutzer möchte ich meine hinterlegte E-Mail Adresse, meinen Nutzernamen oder auch mein Passwort im Nachhinein ändern können.
- Als Nutzer möchte ich die Möglichkeit haben, meinen Account wieder zu löschen.

Epic 2: Interaktion mit der Cloud

- Story 2.1: Als eingeloggter Nutzer möchte ich die Möglichkeit haben, unterstützte Textdateien von meinem Computer in meine persönliche Ablage in der Cloud hochzuladen.
- Story 2.2: Als eingeloggter Nutzer möchte ich die Möglichkeit haben, Dateien in meiner Ablage zu löschen.

- Story 2.3: Als eingeloggter Nutzer möchte ich die Möglichkeit haben, Dateien in meiner Ablage umzubenennen.
- Story 2.4: Als eingeloggter Nutzer möchte ich mir Inhalte von Textdateien anzeigen lassen können.
- Story 2.5: Als eingeloggter Nutzer möchte ich eine Übersicht aller Dateien, inklusive Metadaten, meiner Ablage einsehen können.

Das sollten erst einmal die wichtigsten User Stories sein. Auch hier könnte man später noch um viele weitere Stories erweitern. Man könnte zum Beispiel die Möglichkeit entwickeln, dass man weitere Ablagen erstellen kann, die man mit anderen Nutzern teilen kann. Oder auch, dass man sich in der Cloud Ordner erstellen kann. Diese Funktionalitäten sind allerdings nicht zwingend notwendig für eine simple Cloud-Anwendung wie wir sie entwickeln wollen.

Zusammenfassung davon was wir gemacht haben:

- Wir sind zunächst die PDF zum “The Complete Developer”-Buch zum Thema Docker überflogen, um noch mal etwas mehr Verständnis über Docker und “Containerization” zu bekommen
- Anschließend haben wir noch 1-2 YouTube Videos geschaut, um auch mal eine visuellere Darstellung vom Prinzip von Docker zu sehen
- Danach sind wir einfach den Anweisungen der README’s aus Ihren Git-Repositories gefolgt, um ein Express-Projekt und eine Docker-Umgebung zum Laufen zu bekommen und haben dabei Anpassungen gemäß unseres Projektnamens gemacht
- Zum Bauen des Images und zum Starten und Testen eines Containers haben wir zu Beginn die Docker-Commands einzeln aufgerufen. Später haben wir allerdings mehr auf die Commands der Makefile zurückgegriffen

Reflektion zu Problemen:

- Was das Express-Setup anging, so hatten wir keinerlei Probleme
- Bis wir auf die Commands der Makefile umgestiegen sind, haben wir oftmals Typos gemacht oder auch schlichtweg das Setzen bestimmter Flags vergessen
- Nach dem Aufsetzen von Docker gab es kurzzeitig beim Aufruf von “http:localhost:3001/hello” die Fehlermeldung: “This site can’t be reached”. Das lag aber einfach nur an einer Inkonsistenz der verwendeten Ports. Nun verwenden wir überall Port 3000, um es so simpel wie möglich zu halten
- Bei 2b) gab es die Frage, was bei dem Aufruf von http:localhost:3000/ passiert und warum. Man macht eine Anfrage an den Root vom localhost, allerdings wurde nicht festgelegt wie mit der Anfrage umgegangen werden soll. Also haben wir nun festgelegt, dass bei einem Aufruf vom Root man HTML zurückbekommt, welches einen Link zur /hello-Seite beinhaltet
- Ansonsten gab es noch kleinere Probleme beim Hinzufügen des NGINX-Servers mit der compose.yml, die allerdings durch eine kurze Internetsuche oder das Fragen von ChatGPT gelöst werden konnten.