hochschule mannheim

Virtualisierte Arbeitsumgebung für den Test verteilter Systeme

Jan-Philipp Willem

Studienarbeit
Studiengang Informatik

Fakultät für Informatik
Hochschule Mannheim

23.06.2017

Betreuer: Prof. Dr. Sandro Leuchter
Zweitkorrektor: Prof. ...

Zusammenfassung

Ein verteiltes Softwaresystem zu entwickeln und dessen Auswirkungen zu verstehen, kann eine Hürde für Studierende im Grundstudium der Informatik darstellen. Oft ist es schon anspruchsvoll genug, in Übungen eine geforderte Transferleistung für Grundlagen umzusetzen. Um dies zu erleichtern, soll eine Umgebung geschaffen werden, welche es ermöglicht, eine verteilte Anwendung unabhängig des eigenen Computers zu testen.

Eine Programmieraufgabe soll von einem Dozenten als Experiment definiert werden können und aus einer gegebenen Anzahl an Instanzen eines ebenso definierbaren Serversystems bestehen. Anschließend soll das Experiment von Studierenden durchgeführt und die eigenen Lösungen hochgeladen werden können. Ferner soll die Möglichkeit gegeben sein, die aus voneinander getrennten Rechner-Instanzen resultierenden Ausgaben darzustellen und die Ergebnisse zu analysieren.

Abstract

As a computer science undergraduate it might be challenging to develop a distributed server-system and yet fully understand its ramifications. Frequently it is demanding enough to apply new principles in tutorial class. To facilitate this, an environment should be build, whereby it is possible to test a distributed program independently of one's own computer.

A lecturer should be able to define an experiment as a programming exercise, which consists of a given amount of instances of a definable server system. Following this, a student is supposed to conduct this exercise and upload his/her own solutions, whereas resulting outputs by independent computer instances are displayed, which should be analyzed.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Problemstellung

Diese Arbeit unterteilt sich in vier Kapitel. Das erste Kapitel soll eine Einleitung in die Problematik geben, welche erforscht werden soll. Als nächstes folgen Grundlagen, welche die Techniken und Technologien des VAR-Tool's erklären. Im dritten Kapitel wird der Planungsprozess und mit dessen Hilfe der praktische Teil der Studienarbeit erläutert. Weiterhin folgt ein Fazit der geleisteten Vorgänge und ein Ausblick, wie das Projekt weitergeführt werden könnte.

1.1 Bisheriger Ablauf

Der typische Ablauf bei Programmieraufgaben in einer Laborübung von Verteilte Architekturen (VAR) ist folgendermaßen: Ein Student soll verteilte Aufgaben programmieren, welche es nötig machen, zunächst eine passende lokale Entwicklungsumgebung zu installieren. Dazu wird die Nutzung von schwergewichtigen Integrated-Development-Environments (IDE's) wie Eclipse oder Netbeans vorgeschlagen. Ein Grund dafür sind die schon vorhandenen IDE-Integrationen von Servern und Technologien, welche eingesetzt werden sollen. Dennoch führt schon dieser Prozess bei einigen Studenten zu Problemen.

Gelingt es die Integrationen einzurichten, so kann die Funktionsweise der eigenen Programme getestet werden. Allerdings führt diese Herangehensweise zu keiner echten Verteilung, da alle Services auf der gleichen Maschine ausgeführt werden. Das Ergebnis kann sich bei einer Trennung der Instanzen auf Netzwerkebene erheblich unterscheiden, oder ist unter Umständen nicht lauffähig.

1.2 Anforderungen

Es soll ein Prototyp entwickelt werden, mit dessen Hilfe ein Dozent so genannte Experimente im Kontext von verteilten Systemen definieren kann.

Diese sollen jeweils eine gegebene Anzahl an Instanzen besitzen können, auf denen Programmieraufgaben eines Studierenden und die dafür benötigten Technologien ausgeführt werden sollen. Eine jeweilige Instanz wird in einer Konfigurationsdatei durch die Angabe eines Namens, dessen geöffneten Netzwerk-Ports und eine Argumentenliste beschrieben. Das Deployment und die Konfiguration der Systeme soll mithilfe von Docker realisiert werden. Auf Netzwerkebene sollen sich die Instanzen gegenseitig erreichen können, jedoch soll der Zugriff auf Instanzen anderer Nutzer und dem Internet unterbunden werden.

Die Durchführung eines Experiments geschieht durch die Lösung von gewissen Aufgaben innerhalb des Instanzkontextes. Dies geschieht durch einzelnes Hochladen der Programmpakete (Jar, War,..) pro Instanz und der anschließenden Angabe einer Main-Class und einer Argumentenliste. Anschließend kann eine Instanz gestartet und dessen Ausgaben beobachtet werden.

Der Grund für das getrennte Hochladen ist in der Hoffnung begründet, dass die echte Verteilung der Instanzen so von den Studenten besser verstanden werden kann.

Kapitel 2

Grundlagen

Dieses Kapitel möchte einige Grundlagen definieren, welche im Kapitel 3 vorausgesetzt werden. Es handelt von der Definition des funktionalen Programmier-Paradigma, welches in den Sprachen Clojure und Elm zum Einsatz kommt. Weiterhin werden die Webtechnologien Single-Page-Applications (SPA's) und Websockets erklärt, welche das Frontend des VAR-Tool's nutzt. Die Architektur der ganzen Applikation wird maßgebend durch die Trennung in unabhängige Microservices beeinflusst, welche mit Docker realisiert sind.

2.1 Funktionale Programmierung

Die funktionale Programmierung gehört zu den Deklarativen Programmierparadigmen und stellt neben der imperativen und objektorientierten Programmierung eines der am Häufigsten genutzten Paradigmen dar.

Anders als bei imperativen Sprachen, unterscheiden sich funktionale Sprachen vor Allem an ihrer Unveränderbarkeit (engl. Immutability) von Variablen und ihrer Datenstrukturen. Es wird dabei angestrebt, bei einer Veränderung des Zustandes (engl. State) einer Variablen nicht den Inhalt ihrer Referenz zu verändern, sondern stattdessen eine Kopie der Daten zurückzuliefern. In der Praxis**(TODO: quote)** hat sich vielfach herausgestellt, dass sich die häufig verwendete Veränderbarkeit (engl. Mutability) von Zuständen gerade auch in der objektorientierten Programmierung zu häufigen Fehlerquellen führen kann. Es wurde jedoch gezeigt, dass das Zurückgeben von Kopien ebenso sehr performant funktionieren kann **(TODO: add quote)**.

Wie der Name schon vermuten lässt basiert die funktionale Programmierung zu größten Teilen aus Funktionen. Diese sind meist als Module gekapselt, welche nach einer bestimmten Aufgabe gruppiert sind. Anders als bei Objekten möchte man offen mit den an die Funktionen übergebenen Daten umgehen und sie nicht in der Implementierung verstecken (engl. Information-Hiding). Dabei sollte eine Funktion nur diese eine Aufgabe erledigen, welche mithilfe ihres Namens beschrieben wird. Um dieses deter-

ministische Verhalten zu erreichen, werden oft sehr granulare Funktionen erstellt, welche anschließend durch eine Komposition miteinander vereint werden[1]. Jegliche Seiteneffekte sollten vermieden werden, was man auch als reine (engl. pure) Funktionen bezeichnet. So sollten Funktionen als Daten-Transformationen verstanden werden: Man erhält Daten, verarbeitet diese und gibt sie anschließend transformiert zurück, welches als Eingabe, Verarbeitung, Ausgabe (EVA)-Prinzip bekannt ist.

Da Funktionen selbst auch Typen darstellen, können sie ebenso als Parameter anderer Funktionen dienen, welche man dann auch als Funktionen Höherer Ordnung (engl. Higher Order Funktions) bezeichnet. Mit dieser Voraussetzung können Verhalten aus einer aufzurufenden Funktion herausgelöst werden, womit das jeweilige Verhalten dynamisch bestimmt werden kann. Diese Art von funktionen werden auch als Callbacks bezeichnet. Hilfreich sind bei dieser Vorgehensweise auch so genannte Lamdas oder Closures, welche Funktionen darstellen, die inline definiert werden können und somit anonym beziehungsweise (bzw.) Unbenannt sind.

Grundsätzlich werden funktionale Sprachen nochmals zwischen solchen Sprachen unterschieden, welche neben anderen Paradigmen auch das Funktionale Paradigma unterstützen und solchen, welche ausschließlich auf den Prinzipien der funktionalen Programmierung beruhen. Die sogenannten reinen funktionalen Sprachen haben nicht einmal beispielsweise (bspw.) die Möglichkeit Seiteneffekte auszuführen und sind sehr oft strikt typisiert. In Multiparadigmensprachen beruhen viele funktionale Prinzipien und deren Erfolg, auf deren Einsatz und der generellen Bedachtheit des jeweiligen Programmierers bzw. des einsetzenden Teams. Man mag diese Prinzipien auch (Programming-)Patterns nennen, welche es zu befolgen, oder missverstehen gilt.

2.2 Verwendete Programmiersprachen

Das Projekt verwendet für den Server die Sprache Clojure, wohingegen der Client mithilfe von Elm umgesetzt ist.

2.2.1 Clojure

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene

[1] `f(x) = g(x) • h(x) <=> f(x) = h(g(x))`

Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2.2.2 Elm

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2.3 Eingesetzte Webtechnologien

2.3.1 Single-Page-Applications

Einzelseiten-Webanwendungen (engl. Single-Page-Applications) stellen eine spezielle Art einer Webanwendung dar. Sie unterscheiden sich zu einer klassischen Webseite dahingehend, dass sie nur aus einem einzelnen Html-Dokument bestehen und ihren Inhalt dynamisch nachladen. Innerhalb dieser Einzelseite wird ebenso Java-Script-Code geladen, welcher die eigentliche Funktionalität enthält. Oft weisen SPA's außerdem Merkmale von Rich-Internet-Applications (RIA's) auf, da ihre Oberflächen typischerweise eher einer Desktop-Anwendung gleichen, als einer statischen Webseite. In diesem Kontext spricht man deswegen auch von so genannten Web-Apps.

Da eine Verlagerung von sonst in einem Server enthaltene Präsentations-Logik stattfindet, resultiert diese Web-Architektur in einem Fat-Client. Es wird dadurch die Serverlast reduziert und gleichzeitig die Grundlage für einen flüssigeren Übergang der Darstellung der Oberflächen geschaffen. Alternativ dazu endet bei einer normalen Webseite der Präsentationsvorgang bei einem neuen Abfragen einer weiteren Unterseite.

Um die zu nutzenden Daten zur Anzeige der Applikation zu erhalten, bedarf es einer gewissen Client-Architektur, welche einzelne **AJAX!** (**AJAX!**)-Requests an eine (**REST!** (**REST!**)-)Schnittstelle schickt, oder Nachrichten mit Hilfe einer beidseitigen Verbindung über Websockets sendet. Seit es in der Java-Script-Welt versucht wurde, das Konzept einer SPA umzusetzen, war es schwierig, ein vernünftig wartbares System zu entwickeln, welches die anfallenden und zu nutzenden Daten auch vernünftig verarbeiten können. Bekannte Vertreter von Libraries und Frameworks, welche die anfallenden Probleme versucht zu lösen, sind Google Angular und Angular 2, beziehungsweise Facebook React oder eine gänzlich neue Sprache wie Elm oder Clojure-Script.

2.3.2 Websockets

Websocket-Verbindungen beruhen auf einem gleichnamigen Netzwerkprotokoll, welche eine beidseitige Client-Server-Kommunikation über **TCP!** (**TCP!**) ermöglichen. Sie haben jedoch, wie ihr Name es vermuten lässt, wenig mit klassischen Unix-Sockets gemeinsam.

2.4 Container-Virtualisierung mit Docker

Docker[2] ist eine Software zum Deployment von Applikationen innerhalb von Containern. Im Vergleich zu Virtuelle Maschinen (VM's) ähnelt sich deren Vorgehensweise, jedoch laufen die Prozesse der Container direkt auf dem

[2] `https://docs.docker.com/`

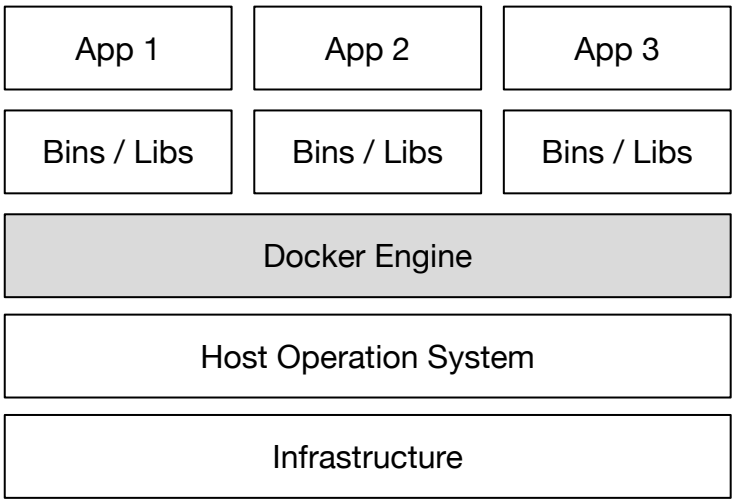

Abbildung 2.1: Schichten einer Virtualisierung mit Docker

Host-Betriebssystem. Grundlage dafür sind Bibliotheken, welche im Kernel oder sehr nah an diesem arbeiten, wohingegen klassische VM's durch ihre potenziell abstrakteren Zwischenschicht, weniger performant arbeiten können. Trotz dieser Tatsache sind die Prozesse mithilfe von unter Anderem (u.A.) Control-Groups und Kernel Namespaces voneinander isoliert. Weiterhin hat man die Möglichkeit mit gängigen Mandatory-Access-Control-Frameworks wie SE-Linux oder App-Armor die Rechte innerhalb der Container zu beschränken. Als Anforderung besteht keine spezifische Hardware-Infrastruktur wie bei beispielsweise VMWare ESXi. Ebenso ist Docker mittlerweile auf allen gängigen Betriebssystemen lauffähig, wobei Linux-Derivate die beste Unterstützung erhalten. Dies ist damit begründet, dass die Docker-Engine (Abb. 2.1) auf den einzelnen Betriebssystemen verschieden aussieht.

Ein Container wird mithilfe eines Dockerfiles beschrieben. Darin werden deskriptive Instruktionen definiert um Abhängigkeiten zu installieren, Konfigurationen vorzunehmen und andere Build-Steps auszuführen. Ein spezifischer Container wird als Image bezeichnet und setzt sich aus granularen Sub-Images zusammen. Beim Erzeugen von Images eigener Dockerfiles müssen im Vergleich zu VM's keine großen Dateien transferiert werden, da sie aus ihrem Rezept reproduzierbar sind. Es besteht auch die Möglichkeit, von anderen Dockerfiles zu erben und diese über das Netzwerk verteilt bereitzustellen. Falls die über eine Docker-Registry angebotenen Schichten eines Containers noch nicht auf dem eigenen Rechner vorhanden sind, so werden diese heruntergeladen.

Weiterhin gibt es einen entscheidenden Vorteil gegenüber einer traditionellen VM: Ein Dockerfile trägt implizit zur Dokumentation bei, da jede Änderung an einem Container in seinem Rezept ergänzt werden muss, um

diese bei einem erneuten Start bzw. erneuten Build zu erhalten.

Wird eine Applikation in einzelnen Services aufgeteilt, so bietet sich das Tool **Docker-Compose**[3] an. Es ermöglicht in einer einzelnen Datei (`docker-compose.yml`) alle Konfigurations-Parameter der einzelnen Container zu definieren. Das können bspw. Volume-Mounts, Ports, Environment-Variables oder Netzwerke sein, welche sonst bei jedem `docker exec` Command angegeben werden müssten. Alternativ dazu stünden eigene und oft fehlerintensive Shell-Scripts um ähnliches zu erreichen.

Es bietet sich auch an, getrennte Environments für Development, Test und Production als zentrale Dokumentation der Applikation zu nutzen (`docker-compose.[env].yml`).

[3] `https://docs.docker.com/compose/overview/`

Kapitel 3

VAR-Tool

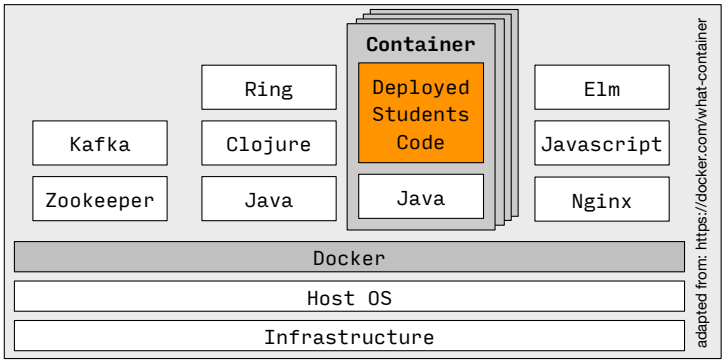

Abbildung 3.1: Container-Übersicht

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.1 UI-Mockup

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

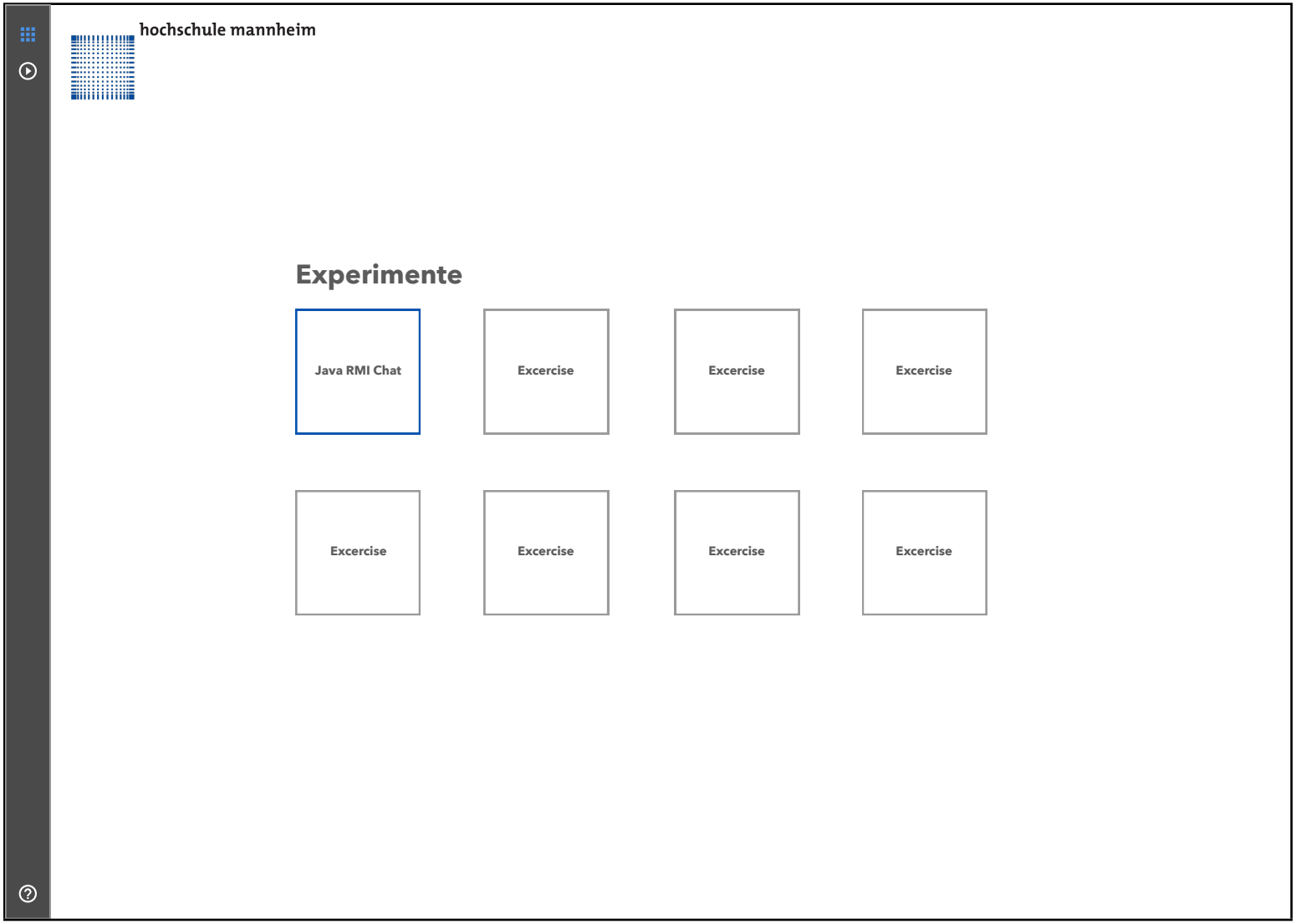

Abbildung 3.2: UI-Mockup: Übersicht der Experimente

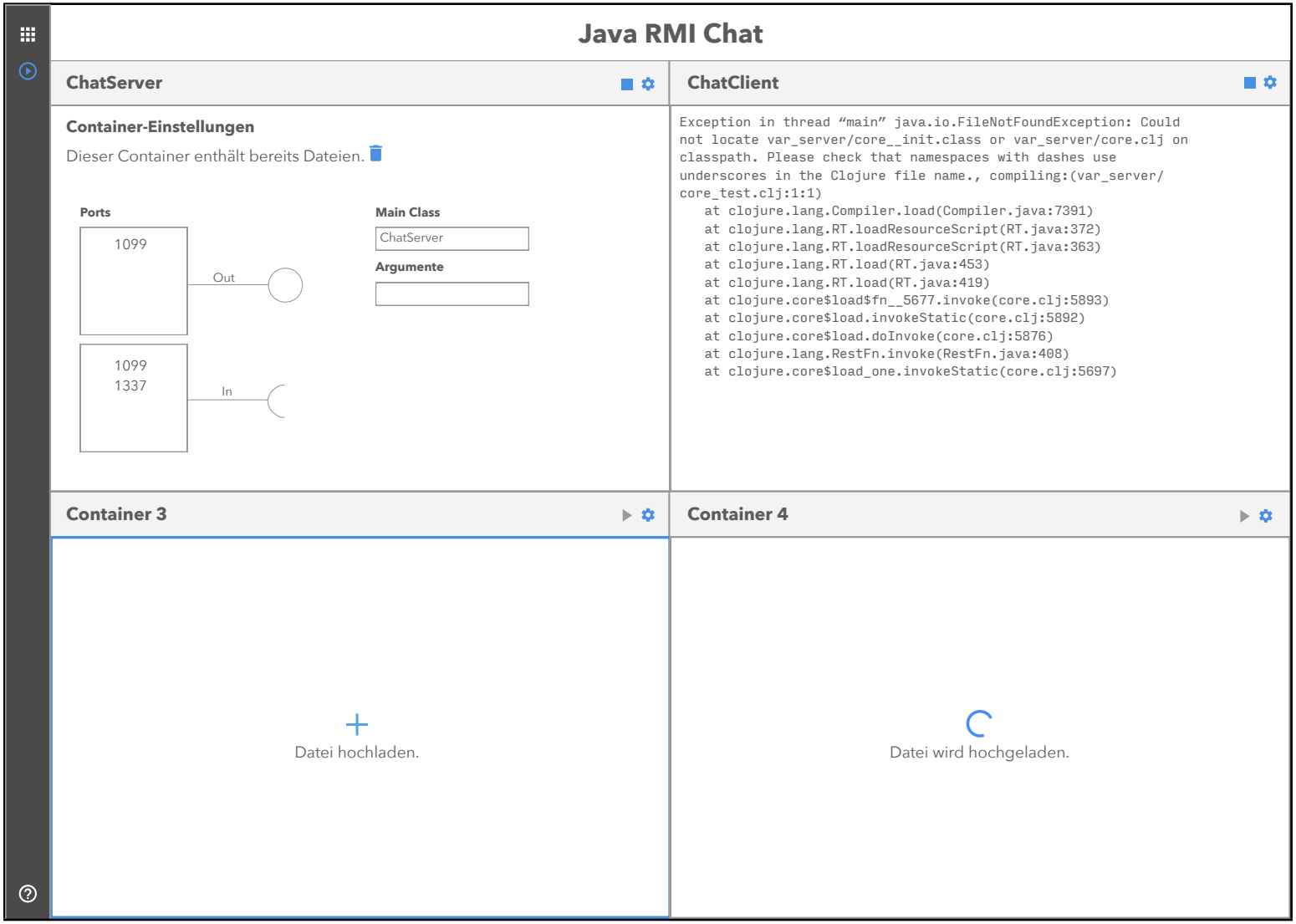

Abbildung 3.3: UI-Mockup: Experimentenansicht mit verschiedenen Zuständen pro Instanz

3.2 Architektur

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.2.1 Geschäftsprozesse

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

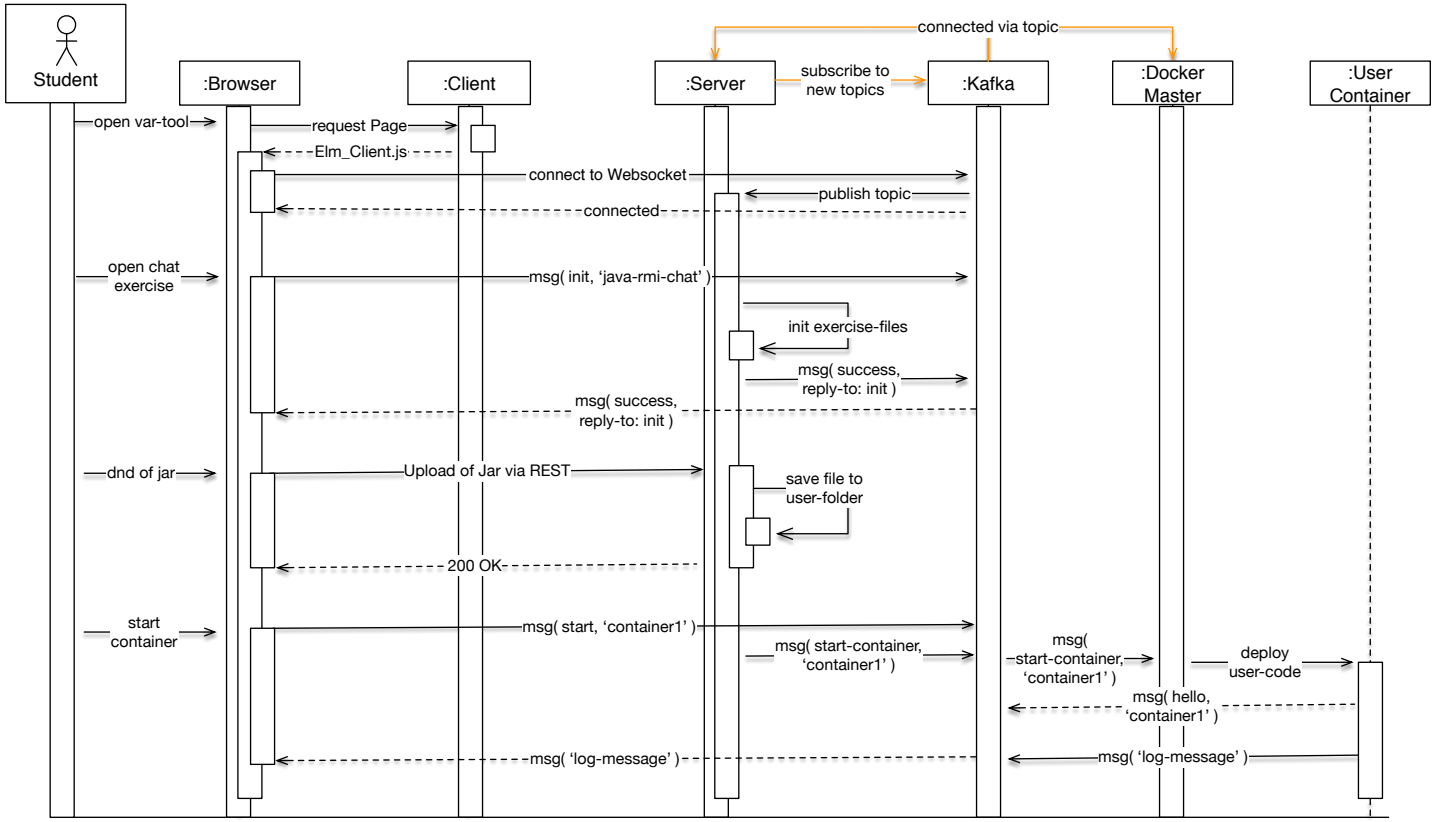

Abbildung 3.4: Sequenzdiagramm

3.2.2 Netzwerkgrenzen

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

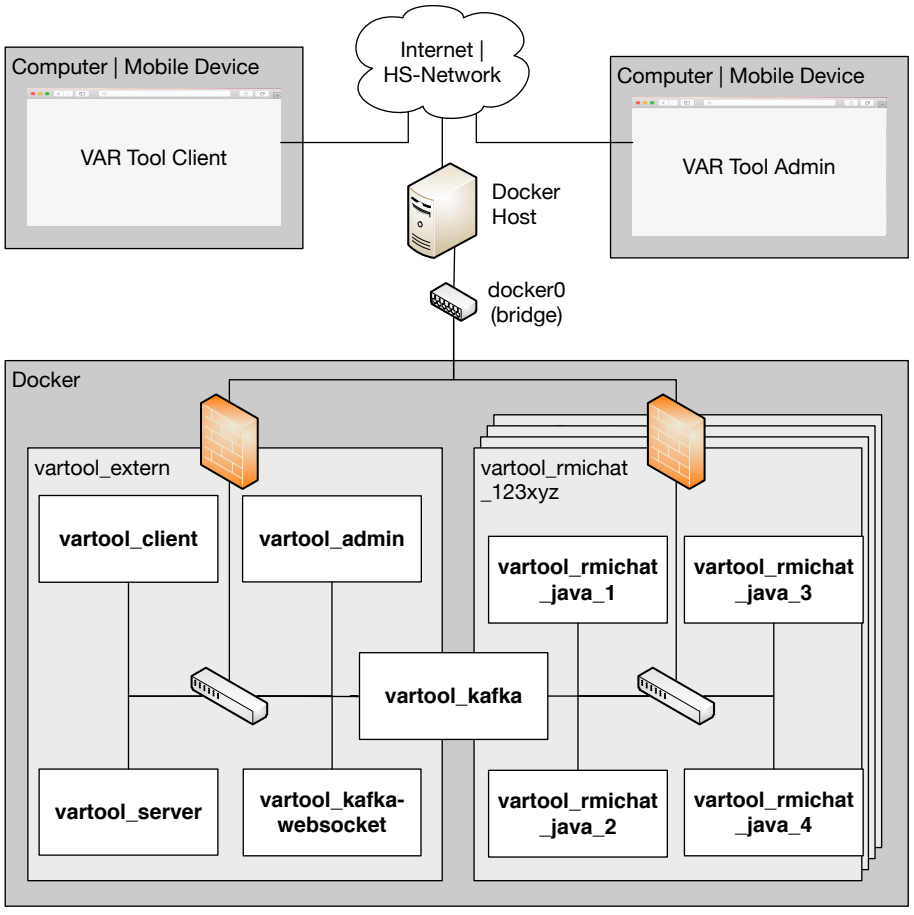

Abbildung 3.5: Netzwerkdiagramm

3.2.3 Deployment

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

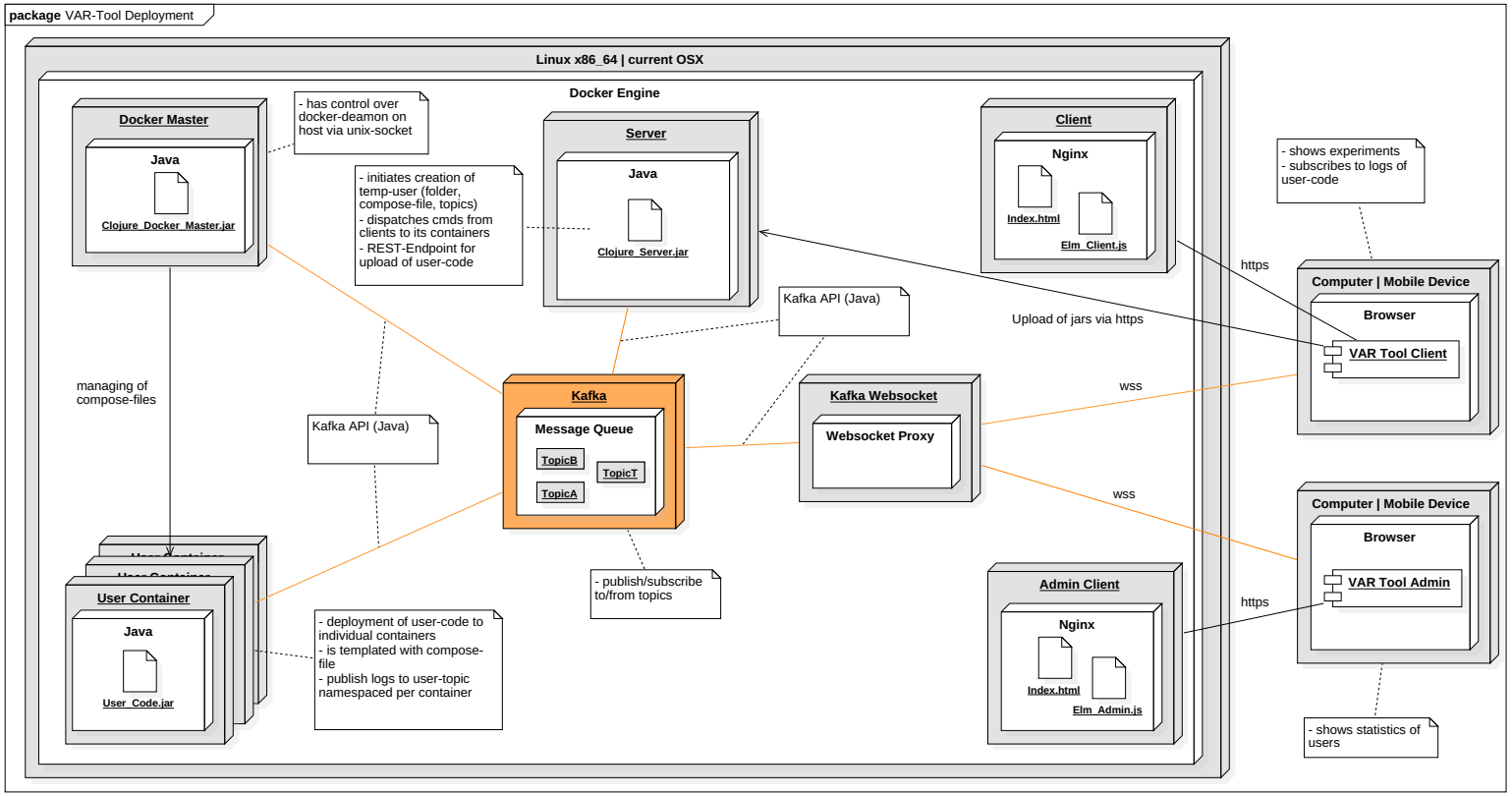

Abbildung 3.6: Deploymentdiagramm

3.3 Umsetzung

3.3.1 Kommunikation über Nachrichten

Im Vorfeld wurde ein Nachrichten-Protokoll definiert, welches die Kommunikation zwischen Server und Client standardisiert. Dabei heißen Client-Nachrichten *Commands* und Server-Nachrichten *Messages*. So können beide Nachrichten-Typen durch ein triviales Merkmal unterschieden werden. Jeweilig unterscheiden sich diese weiter in Kinder-Typen und deren *Payloads*. Dabei werden die Daten über eine Websocket-Verbindung in einem JSON-Envelope als String übertragen.

Commands:

- request-experiments:
- add-input:
- start-instance:
- stop-instance:

Messages:

- log:
- reply:

3.3.2 Server

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele

verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.3.3 Client

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte mög-

lichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

3.4 Erwartungen an die Performance

3.5 Installation

Als erstes wird das Projekt mithilfe von `git clone https://github.com/jwillem/var-tool.git` heruntergeladen. Um die Applikation zu starten, werden zunächst die Installationen von Docker[1] und Docker-Compose[2] benötigt. Nach einem Ausführen von `docker-compose build` im Projektverzeichnis kann die App mit `docker-compose up` gestartet werden.

[1] `https://www.docker.com/community-edition`
[2] `https://docs.docker.com/compose/install/`

Kapitel 4

Fazit

- Umsetzung mit Kafka leider nicht möglich.
- mehraufwand eigenen kafka proxy zu schreiben
- grundüberlegung von kafka kann dennoch in folgenden arbeiten zu thema kommen
- kafka kann log daten empfangen und auch dauerhaft vorhalten
- microservice gedanke kann weiter ausgebaut werden
- Ergebnisse (eigenes Kapitel?)
- Performance!
- websocket sollte über tls erfolgen
- Im Experiment könnte noch bei geringem Aufand ein dnd-container zum einsatz kommen.
- Admin-Client muss noch in weiterer Arbeit entwickelt werden
- Abschottung der Container muss noch mit bspw iptables geschehen

Quellenverzeichnis

Abkürzungsverzeichnis

Abb. Abbildung

bspw. beispielsweise

bzw. beziehungsweise

EVA Eingabe, Verarbeitung, Ausgabe

VAR Verteilte Architekturen

VM Virtuelle Maschine

IDE Integrated-Development-Environment

SPA Single-Page-Application

RIA Rich-Internet-Application

u.A. unter Anderem