3. Praxistransferbericht

Containervirtualisierung der Webinfrastruktur

für das VR-Klassenzimmer

Name, Vorname:	Zhang, Yüxi
Matrikelnummer:	684163
Ausbildungsbetrieb:	Universität Potsdam
Studienjahrgang:	(2020)
Fachbereich:	Duales Studium Wirtschaft • Technik
Studiengang:	Informatik
Modul:	3-2121-TIT20A – 3. Praxistransferbericht
Betreuer Hochschule:	Laura Haase
Betreuer Unternehmen:	Axel Wiepke
Anzahl der Wörter:	2384

Vom Ausbildungsleiter zur Kenntnis genommen:

07.01.2022 F. Zule

(Datum/Unterschrift)

Werder (Havel), den 06.01.2022

(Datum/Unterschrift der/des Studierenden)

Zusammenfassung

Das Ziel des vorliegenden Praxistransferberichts ist die Containervirtualisierung der Webinfrastruktur des VR-Klassenzimmer Projekts der Universität Potsdam. Für die Verständlichkeit der Arbeit wird die Containervirtualisierung mit der herkömmlichen Hardware-Virtualisierung und einer Nomenklatur erklärt.
Im Anschluss findet eine kurze Beschreibung des VR-Klassenzimmers und der Webinfrastruktur statt, gefolgt von einer Anforderungsanalyse. Darin werden die Voraussetzungen und Ansprüche des zu erstellenden Programmes beschrieben. In der Konzeption wird sich mit dem Aufbau und den Anfragen der Webinfrastruktur befasst. Die Implementierung erfolgt in mehreren Abschnitten, wobei ein Teil davon nicht umgesetzt werden konnte. Die meisten Container der Implementierung können jedoch in den laufenden Betrieb genommen werden.

Abstract

The goal of this practical transfer report is the container virtualization of the web infrastructure of the VR classroom project at the University of Potsdam. For the comprehensibility of the report, container virtualization is compared with conventional hardware virtualization and a nomenclature is explained.
This is followed by a brief description of the VR classroom and the web infrastructure and by a requirements analysis. This describes the requirements and demands of the program to be created. The conceptual design deals with the structure and requests of the web infrastructure. The implementation is done in several sections, some of which could not be implemented. However, most of the containers of the implementation can be put into operation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

1 Einleitung

Anfang der 2000er-Jahre hat die Virtualisierung den Alltag vieler EntwicklerInnen bedeutend verändert. Nach Einführung der Virtualisierung konnten verschiedene Betriebssysteme auf einem Computer ausgeführt, Tests von Programmen in verschiedenen Umgebungen isoliert durchgeführt und verschiedene Software-Stacks parallel installiert werden.
Virtuelle Maschinen (VM) spielen weiterhin für bestimmte Anwendungszwecke eine große Rolle, wie z.B. für isolierte Tests oder als Arbeitsumgebung. In den letzten Jahren kam bei der Softwareentwicklung jedoch ein Trend auf, um von virtuellen Maschinen auf Container umzusteigen.
Die Containertechnologie ermöglicht es Softwarekomponenten, wie z.B. Webserver, Datenbanken und weitere Programme ohne den Overhead einer virtuellen Maschine auszuführen. Das führt zu einem schnelleren Aufsetzen, geringeren Ressourcenbelastung, höhere Skalierbarkeit und Lastverteilungsmöglichkeiten als bei der herkömmlichen Virtualisierung. [Kof21] (S.11)

Das VR-Klassenzimmer ist ein Projekt des Instituts für Informatik der Universität Potsdam, dessen Webinfrastruktur aus mehreren einzelnen unabhängigen Services besteht.
Wenn das Projekt an anderen Hochschulen zum ersten Mal genutzt werden soll, müssen derzeit alle Services einzeln aufgesetzt werden. Das ist fehleranfällig und bindet Ressourcen. Eine Lösung dafür ist die Nutzung der Containerisierung für alle Webkomponenten des Projekts.
Die Containervirtualisierung wird im Rahmen des Praxistransferberichtes der Hochschule für Wirtschaft und Recht (HWR) und des Instituts für Informatik der Universität Potsdam konzipiert und implementiert.

2 Vergleich mit der Hardware-Virtualisierung

Die herkömmliche Virtualisierung ist eine Methode in der Informationstechnik, um Ressourcen und ihre Auslastung zu optimieren. Es existieren verschiedene Arten der Virtualisierung, wobei der Fokus in dieser Arbeit auf der Hardware-Virtualisierung liegt. Die meisten Virtualisierungstechniken ermöglichen die Erstellung von virtuellen Maschinen (VM). [Chr20] (S.253) Die jeweiligen Hardwareressourcen wie z.B. Prozessorleistung, Arbeitsspeicher usw. werden von einem Hypervisor bzw. einem Virtual Machine Manager (VMM) zugewiesen. Dies ist auf der linken Seite der Abbildung 1, S.3 zu sehen. In jeder VM wird ein Betriebssystem initialisiert, auch Gastbetriebssystem genannt, in dem proprietäre Anwendungen laufen können. [Chr20] (S.261-262)

Die Containervirtualisierung versucht dies in einem anderen Ansatz umzusetzen.
Als Best Practice wird meist nur eine Anwendung je Container verwendet. Ein Container stellt eine standardisierte Softwareeinheit dar, welche im Vergleich zu VMs leichtgewichtiger ist und alle Ressourcen, die eine Anwendung benötigt, beinhaltet.[1] Dies ist rechts in der Abbildung 1, S.3 schematisch dargestellt.
Des Weiteren wird die Infrastruktur des Hostsystems direkt von den Containern genutzt (das Dateisystem und der Kernel). Dies unterstützt die Leichtgewichtigkeit, welche zusätzlich dadurch begünstigt wird, dass Betriebssysteme, Binärdateien und Codebibliotheken bei gleicher Version geteilt und nicht wie bei einer VM einzeln erstellt werden müssen. Der Overhead wird dadurch so gering wie möglich gehalten, wodurch eine schnellere Aufsetzung, geringere Ressourcenbelastung, mehr Skalierbarkeit und Lastverteilungsmöglichkeiten gelingen. Container können somit als isolierter Prozess eines Hostsystems betrachtet werden.

Da Virtuelle Maschinen nicht den Kernel des Hostsystems nutzt, bieten sie mehr Sicherheit gegenüber Container, welche mit dem Kernel des Hosts interagieren. In der Praxis schließen sich daher Container und VMs nicht aus, sondern die Containervirtualisierungssoftware wird in einer VM ausgeführt. Beide Technologien können auf einem Server ausgeführt werden und benötigen ein Hostbetriebssystem. [Kof21] (S.11 und S.48-49)

[1] https://www.docker.com/resources/what-container Abrufdatum: 03.01.2022

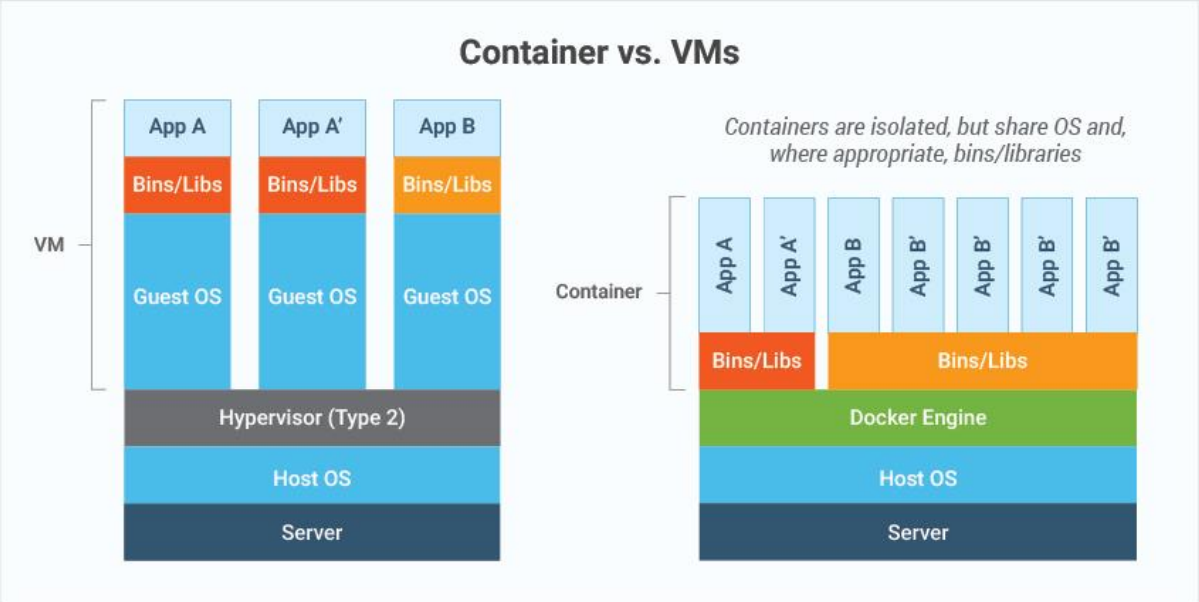

Abbildung 1: Vergleich von Hardware-Virtualisierung und Container[2]

2.1 Docker Grundlagen und Nomenklatur

Die beliebteste Software zur Erstellung von Containern ist nach Umfrage über Plattformen „Docker“ und gilt zudem als Industriestandard.[3] Docker ist eine standardisierte Open Source Container-Software, die mit Windows und Linux kompatibel ist. [Kof21] (S.12) Sie bietet eine Vielzahl an Funktionen und Features, wobei einige in der folgenden Nomenklatur erläutert werden, die für die Implementierung relevant sind. Die Aufgaben werden von der Docker-Engine übernommen (siehe rechts in Abbildung 1, S.3). Unter den Funktionen befindet sich z.B. die Speicherverwaltung (Volumes), die Vernetzung (Networking), der Container Daemon (containerd) und vieles mehr.[4]
Die genaue Erklärung der Befehle und der Syntax erfolgt bei der (6.) Implementierung.

[2] https://www.eginnovations.com/blog/containers-vs-vms/ Abrufdatum: 03.01.2022
[3] https://insights.stackoverflow.com/survey/2020#technology-platforms-all-respondents5 Abrufdatum: 03.01.2022
[4] https://www.docker.com/products/container-runtime Abrufdatum: 03.01.2022

2.1.1 Image, Docker Hub

Das Docker Image ist der Ausgangspunkt für jeden Container. Es ist ein Read-only-Dateisystem, welches vom Container nie verändert wird. Dadurch lassen sich beliebig viele Container aus einem Image erstellen.
Images können aus dem Internet heruntergeladen werden. Dafür gibt es eine Registry, auch Docker Hub[5] genannt. Diese werden überwiegend von der Docker-Community bereitgestellt. Es existieren aber auch offizielle Images, welche von der Docker Firma gewartet werden. [Kof21] (S.43-44)

2.1.2 Dockerfile

Eine Dockerfile gibt Instruktionen vor, aus denen ein eigenes Image gebaut wird. Dazu muss eine Datei in einem Verzeichnis erstellt werden. Es werden mit Schlüsselwörtern die Eigenschaften eines Images festgelegt. Eine genaue Beschreibung der Syntax kann der Dokumentation[6] entnommen werden. [Kof21] (S.81)

2.1.3 docker compose

Ein Container kann mittels eines Befehls (docker run) gestartet werden. Entsprechend kann ein sehr großer Aufwand vorliegen, wenn Hunderte von Container gestartet werden müssen. Aus diesem Grund existiert das Tool docker compose[7]. Mittels einer „docker-compose.yml“ Datei werden in der YAML Syntax Instruktionen vorgegeben, wie mehrere Container zusammenhängend durch einen Befehl erstellt werden. [Kof21] (S.99) Die Container stehen dabei in einem Kontext zueinander z.B. Frontend, Backend und Datenbank einer Anwendung.

2.1.4 Portweiterleitung und Netzwerke

Da Container als isolierter Prozess anzusehen sind, können Anwendungen, wie z.B. eine Webapplikation anfangs nicht mit der Außenwelt interagieren. Dies wird jedoch durch eine Portweiterleitung (port forwarding) ermöglicht. Es muss beim Erstellen bzw. Starten eines Containers die Flag Option „—publish“ bzw. „-p“ angefügt werden, wie z.B. „-p 8080:80“. Die Docker-Engine erstellt damit eine Firewall Regel (bzw. Ausnahme), womit der Port 80 im Container mit dem Port 8080 des Hostsystems zugewiesen wird. Somit kann über „localhost:8080“ z.B. eine Webapplikation erreicht werden.[8]
Container können außerdem in Netzwerken zugeordnet werden. Das ist wichtig für die netzwerkinterne Namensauflösung, welche mit der Flag „--name" angegeben wird. [Kof21] (S.72) Somit müssen IP-Adressen, welche sich stetig ändern können, nicht gesondert betrachtet werden. Das Docker-compose Tool erstellt bei Ausführung ein eigenes Netzwerk.

[5] https://hub.docker.com/ Abrufdatum: 03.01.2022
[6] https://docs.docker.com/engine/reference/builder/ Abrufdatum: 03.01.2022
[7] https://docs.docker.com/compose/ Abrufdatum: 07.01.2022
[8] https://docs.docker.com/config/containers/container-networking/ Abrufdatum: 04.01.2022

3 VR-Klassenzimmer

Virtual Reality (Virtuelle Realität oder VR) ist eine dreidimensionale computergenerierte Umgebung. Dabei wird das Ziel verfolgt, eine Situation nachzubilden, mit welcher der Nutzer (User) interagieren kann. Mittels gezielten Feedbacks und Reflexion lässt sich dadurch Wissen aneignen. [Lac20] (S.76) Diese Umgebung kann über ein Head-Mounted-Display (sogenannte VR-Brille) oder über die Software Unity[9] dargestellt werden. Das VR-Klassenzimmer ist ein Projekt von Axel Wiepke, Raphael Zender, sowie Eric und Dirk Richter [Wie19] und weiteren Mitarbeitern der Universität Potsdam[10]. Das Projekt dient dem Verhaltenstraining von Lehramtsstudentinnen und Lehramtsstudenten (LuL) und soll ihnen die Möglichkeit bieten sich auf den späteren realen Unterricht vorzubereiten, wodurch auch Ressourcen gespart werden. Dabei üben sie, praxisnah und zielgerichtet auf Unterrichtsstörungen einzugehen, um dadurch ihre Klassenmanagementkompetenzen unter Beweis zu stellen und gegebenenfalls zu verbessern. Es wird ein Klassenzimmer nachgebildet, indem sich virtuelle Schülerinnen und Schüler (vSuS) befinden (siehe Abbildung 2, S.5).

Abbildung 2: Screenshot aus dem VR-Klassenzimmer, Die Schüler Harley und Jonas melden sich

Die vSuS sind optisch in der zweiten Sekundarstufe und ethnisch divers. Im Klassenzimmer sind verschiedene Anordnungen der Tische und verschiedene Umgebungen möglich. Ein Coach/eine Coachin kann über eine Weboberfläche (siehe Anhang Abbildung 3, S.6) viele Aspekte des VR-Klassenzimmers kontrollieren, wie z.B. Skripte ausführen oder manuell die vSuS auswählen und Verhalten auslösen. [Wie19] Die Aktionen der vSuS basieren auf realen Unterrichtssituationen in einer Klasse und werden nach dem Buch „Techniken der Klassenführung“ [Kou06] kategorisiert. Zu der Webinfrastruktur gehören die Weboberfläche in Form einer React-Web App und ein Learning Managementsystem. Durch die Containervirtualisierung sollen diese Anwendungen schneller und leichter gestartet werden. [Wie19]

[9]https://unity3d.com/de Abrufdatum: 03.01.2022
[10]https://gitup.uni-potsdam.de/mm_vr/vr-klassenzimmer/wikis/Kernteam Abrufdatum: 04.01.2021

3.1 React Web-App

React ist eine JavaScript Bibliothek, um User-Interface-Komponenten zu erstellen.[11] Mittels React ist eine Weboberfläche erstellt worden (siehe Abbildung 3, S.6). Zudem besteht die Weboberfläche aus weiterer Software bzw. Abhängigkeiten, welche in einer package.json definiert werden. Über den Paketmanager npm[12] für Node.js werden diese installiert. Die Dateien zu der Weboberfläche sind in dem Verzeichnis „website-control~“ des VR-Klassenzimmers zu finden.

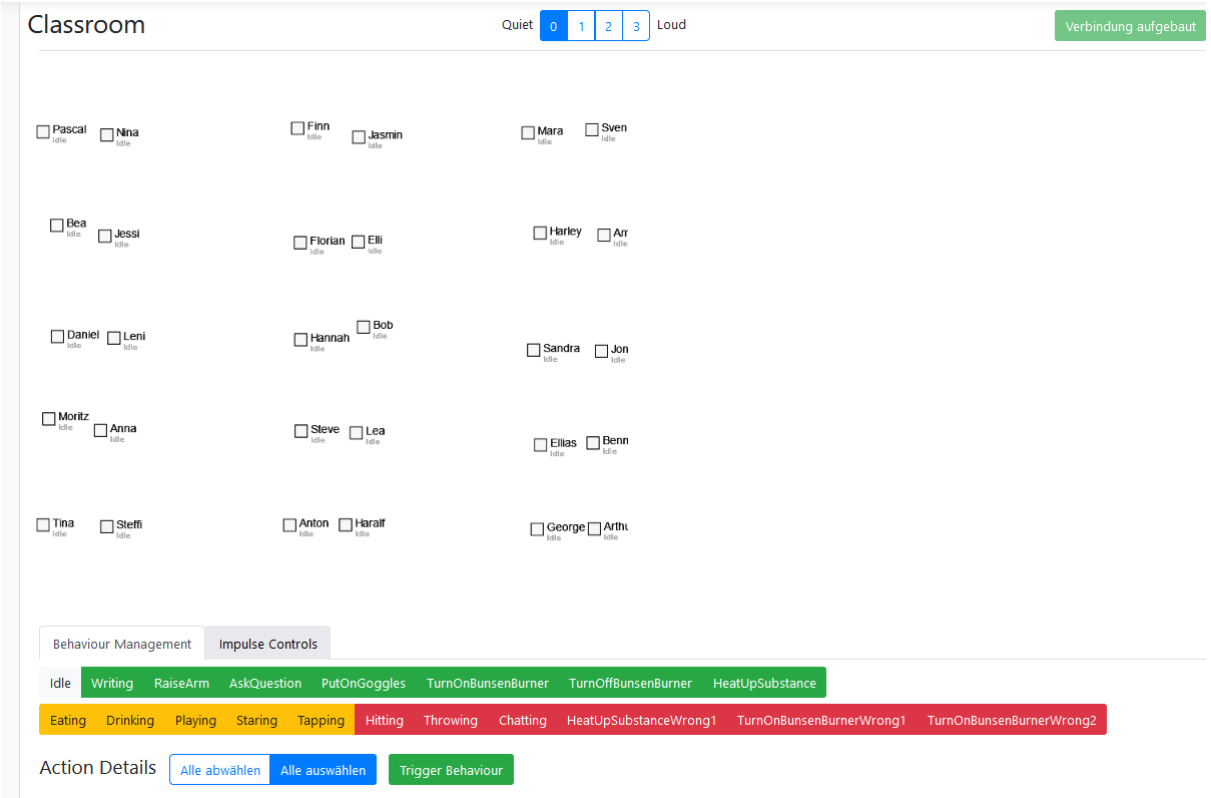

Abbildung 3: Weboberfläche des VR-Klassenzimmers mit auswählbaren Schülern und kategorisierten Verhalten, weitere Funktionen sind möglich

[11] https://reactjs.org/ Abrufdatum: 04.01.2022
[12] https://www.npmjs.com/ Abrufdatum: 04.01.2022

3.2 Learning Management System

Ein Learning Management System (LMS) versucht E-Learning (Electronic Learning) zu vereinfachen.[13] Das derzeit genutzte LMS heißt Learning Locker.[14] Learning Locker ist ein Open Source Learning Record Store (LRS) das Daten im xAPI Format aus der Anwendung schreibt, speichert und ausgibt.[15] Ein LRS ist ein System, welches Daten und Anfragen erhalten und verarbeiten kann.[16] Die xAPI (Experience Application Programming Interface, auch Tin Can API) ist eine Spezifikation für Lerntechnologie, die das Abrufen und Speichern von Daten für Analysen erleichtert. Die standardisierte Syntax hilft Lernerfahrungen und deren Ergebnisse über mehrere Plattformen zu verstehen und zu vergleichen.[17]
Learning Locker ist in drei Anwendungen aufgeteilt. Die Learning Locker Anwendung (Frontend), der xAPI-Service und eine Mongo Datenbank.[18]

4 Anforderungsanalyse

Die korrekte Funktionalität muss nach Implementierung weiterhin erhalten bleiben. Das bedeutet, dass die React Web-App eine Verbindung mit Unity aufbauen kann und das Learning Management System Daten aus Unity lesen und speichern kann. Es sollten außerdem Versionen für die Images festgelegt werden, um die Funktionalität bzw. Stabilität zu gewährleisten. Standardisierte Ports (Best Practise Ports) sollten, wie z.B. bei MongoDB die 27017 festgelegt sein bzw. weiterhin bestehen bleiben. Des Weiteren müssen Daten, die in MongoDB geschrieben werden, persistiert und eine Verbindung mit dem xAPI-Service erstellt werden. Zudem müssen die Daten auch in der Learning Locker Anwendung angezeigt werden.
Alle Container müssen mit einem docker-compose Befehl gestartet werden und die Dockerfile und docker.compose.yml sollten übersichtlich und verständlich sein. Letztendlich sollte die Implementierung die Performanz, die Portierbarkeit, die Skalierbarkeit und die Nutzbarkeit steigern.

[13] https://ieeexplore.ieee.org/abstract/document/5492522 Abrufdatum: 04.01.2022
[14] https://lrs.soft.cs.uni-potsdam.de/login Abrufdatum: 04.01.2022
[15] https://docs.learninglocker.net/welcome/ Abrufdatum: 04.01.2022
[16] https://xapi.com/learning-record-store/ Abrufdatum 06.01.2022
[17] https://xapi.com/overview/ Abrufdatum: 04.01.2022
[18] https://docs.learninglocker.net/overview-architecture/ Abrufdatum: 05.01.2022

5 Konzeption

Wie unter (3.) VR-Klassenzimmer erwähnt, besteht die Webinfrastruktur aus vier Anwendungen, welche mit dem VR-Klassenzimmer direkt und indirekt interagieren. Das folgende nicht standardisierte Diagramm stellt die Anfragen zwischen den Anwendungen und deren Ports dar. Zu beachten ist, dass im Rahmen der Recherche keine Festlegung bzw. Best Practise zu Containervirtualisierungsdiagrammen existiert, weswegen das folgende Diagramm nur der Übersichtlichkeit und Anschaulichkeit dient.

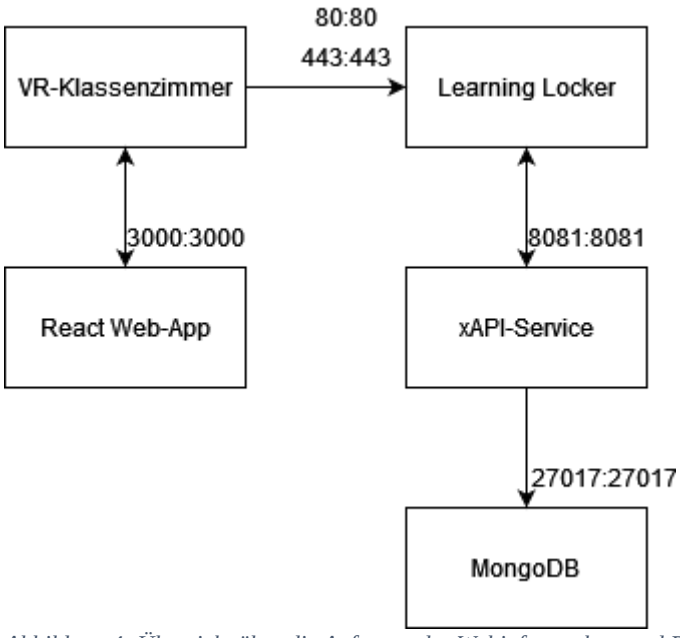

Abbildung 4: Übersicht über die Anfragen der Webinfrastruktur und Port Mapping der Container

Die Rechtecke in Abbildung 4 bilden die Container und die gerichteten Pfeile die Richtung der Anfragen ab. Die nebenstehenden Ports der jeweiligen Anwendung beziehen sich auf die Verbindung vom Hostsystem und Container (Host-Port:Container-Port). Das LMS und die React-Web-App sind eigenständige Services, daher besteht kein direkter Austausch zwischen den beiden. Die Anfragen von der Weboberfläche wie z.B. die Aktion eines vSuS werden an das VR-Klassenzimmer geschickt. Das VR-Klassenzimmer stellt zudem eine Anfrage auf eine Verbindung mit der Weboberfläche. Die Verbindung des VR-Klassenzimmers und den Learning Locker besteht aus zwei Ports (http und https), die für die Webdarstellung nötig sind und wird mittels einer Datei[19] im VR-Klassenzimmer hergestellt. Der xAPI Service, liest und schreibt Daten in die Learning Locker Anwendung und trägt die Daten in die MongoDB ein, welche dort persistiert werden.

[19]vr-klassenzimmer/Assets/ownScripts/xAPI/LrsController.cs
https://gitup.uni-potsdam.de/mm_vr/vr-klassenzimmer/-/blob/master/Assets/ownScripts/xAPI/LrsController.cs
Abrufdatum: 06.01.2022

6 Implementierung

Die Implementierung erfolgt in mehreren Abschnitten. Die Services der Webinfrastruktur des VR-Klassenzimmers laufen unabhängig voneinander, weshalb die Reihenfolge der Implementierung beliebig ist. Für die Images werden bestimmte Versionen genutzt, die periodisch aktualisiert und getestet werden sollten. Zuletzt werden die Implementierungen zusammen in einer docker-compose.yml Datei zusammengefasst, um das Starten der Anwendung zu vereinfachen.

6.1 React Web-App (Implementierung)

Die Weboberfläche ist ein von den Entwicklern des VR-Klassenzimmers erstellte Anwendung, welche in einem Container exportiert werden muss. Selbsterklärend gibt es davon kein Image auf Docker Hub. Aus diesem Grund ist eine Dockerfile in dem Verzeichnis „website-control~“ geschrieben worden.

```
FROM node:14.17.0-alpine3.13

COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm","start"]
```

Mit FROM wird die Version des zu erstellenden Containers gewählt. In diesem Fall ist die Wahl auf Alpine (Linux-Distribution) mit der vorinstallierten Node.js Version 14.17.0 gefallen, da Alpine besonders leichtgewichtig ist und Node.js für die Implementierung benötigt wird. Danach wird die „package.json“ in den Container kopiert und der Befehl „npm install“ ausgeführt, wodurch Abhängigkeiten für die Weboberfläche installiert werden. Anschließend muss der Inhalt des aktuellen Verzeichnisses „website-control~“ in den Container kopiert werden. Dabei muss der Ordner „node_modules“ ausgeschlossen werden, da sonst Fehler hervorgerufen werden. Es wird eine „.dockerignore“ Datei benötigt, welche „/node_modules“ beinhaltet. Dadurch wird der Ordner beim Kopiervorgang nicht betrachtet. Der Ausruck „EXPOSE 3000“ dient als Platzhalter und ist ein Best Practise um darzustellen, dass der Port 3000 im Container verwendet wird. Dieser Port ist schon vorher bei der Weboberfläche angewendet und ist übernommen worden. Zuletzt wird „npm start“ ausgeführt, wobei der Unterschied zwischen RUN und CMD ist, dass RUN bei der Erstellung des Images ausgeführt wird und CMD beim Start des Containers.
Um den Container zu starten, muss mittels der Konsole zuerst ein Image gebaut werden. Mit Befehl 1 (siehe Anhang Tabelle 1, S.15) wird dies umgesetzt und mit Befehl 2 wird unter Portweiterleitung der Port 3000 auf dem Host mit dem Container verbunden. Über „localhost:3000“ kann die Weboberfläche genutzt werden.

6.2 Learning Management System

Das Learning Management System besteht aus drei Komponenten, welche nach Best Practise in einzelnen Containern ausgeführt werden sollten.
Zuerst muss mit Befehl 3 ein Netzwerk erstellt werden, das für die netzwerkinterne Namensauflösung relevant ist (siehe 2.2.4).
Über den Befehl 4 (siehe Anhang Tabelle 1, S.15) kann ein MongoDB Container mit dem Port 27017 gestartet werden. Gespeicherte Daten in dem Container sind nach dem Löschen des Containers nicht mehr verfügbar. Mittels des Volume-Flags des Befehls 4 werden Daten an einen Datenträger des Hostsystems kopiert und werden somit redundant gespeichert. Mit Befehl 5 kann manuell mit der Datenbank interagiert werden. Der Befehl 6 lädt aus dem Docker Hub den xAPI-Service herunter und startet ihn über Port 8081. Zudem wird eine Umgebungsvariable überschrieben, welche durch die Namensauflösung die IP-Adresse von der MongoDB zuweist und somit eine Verbindung zwischen den beiden Services aufgebaut wird.
Falls es bei größeren Aufgaben zu einer erhöhten Anzahl von Umgebungsvariablen kommt, wird empfohlen die Umgebungsvariablen in eine „.env“ Datei auszulagern.[20] Zudem spiel die Startreihenfolge der Services eine Rolle, da der MongoDB Container länger braucht, um fertig erstellt zu werden. Dies können je nach Hardware ein paar Sekunden dauern, weswegen empfohlen wird zuerst den MongoDB Container und nach einigen Sekunden den xAPI-Service Container zu starten.
Die Implementierung des Learning Locker hat nach vielen verschiedenen Ansätzen und Versuchen nicht funktioniert. Da eine „package.json“ im Repository[21] vorliegt, ist eine ähnliche Implementierung wie bei der React Web-App versucht worden, welche jedoch Fehler hervorgerufen hat. Die Dokumentation[22] für die Installation und einer Custom Installation[23] sind ausprobiert worden, haben aber auch Fehler aufgeworfen. Im Rahmen der Recherche sind einige Custom Installationen bzw. Docker Implementierungen von externen Entwicklern ausprobiert worden. Einige Umsetzungen haben funktioniert, stellten jedoch eine andere Version dar und wiederrum andere waren fehlerhaft. Da diese Durchführungen nicht offiziell sind, wird auch kein Support und Sicherheit garantiert, weshalb sie nicht genutzt werden.[24]

[20] https://github.com/LearningLocker/xapi-service Abrufdatum: 05.01.2022
[21] https://github.com/LearningLocker/learninglocker Abrufdatum: 05.01.2022
[22] http://docs.learninglocker.net/guides-installing/ Abrufdatum: 05.01.2022
[23] https://docs.learninglocker.net/guides-custom-installation/ Abrufdatum 06.01.2022
[24] http://docs.learninglocker.net/guides-custom-installation/ Abrufdatum: 05.01.2022

6.3 docker compose

Die docker-compose.yml Datei versucht das Starten der Anwendungen mittels eines Befehls zu vereinfachen und kombiniert sinnbildlich alle vorherigen Schritte.

```
version: '3'

services:
  react:
    container_name: reactcontainer
    build: .
    ports:
      - 3000:3000
  mongo:
    container_name: mongocontainer
    image: mongo:4.4.11
    ports:
      - 27017:27017
    volumes:
    - ./mongo-data:/data/db:rw
  xapi:
    container_name: xapicontainer
    image: learninglocker/xapi-service:4.1.2
    depends_on:
      - mongo
    ports:
      - 8081:8081
    environment:
      - MONGO_URL=mongodb://mongocontainer
volumes:
  mongo-data: {}
```

Eine docker-compose Datei wird in der YAML Syntax (Einrückung wichtig) verfasst. Zuallererst erfolgt die Deklarierung einer Version für die docker-compose.yml Datei. Anschließend werden die verschiedenen Services gelistet. Für die React Web-App wird ein Name für den Container und die Ports ausgewählt. Der Ausdruck „build: .“ weist auf die Dockerfile im aktuellen Verzeichnis hin. Die vorher genannte „.dockerignore“ Datei wird immer noch benötigt. Beim Mongo Service wird statt des Dockerfiles ein Image aus Docker Hub genutzt und ein Datenträger (Volume) erstellt. Da MongoDB länger zum Starten braucht als der xAPI-Service, wird die „depends_on“ Funktion genutzt, damit der xAPI-Service erst startet, wenn der Mongo Container schon ausgeführt wird. Das kann bei erster Ausführung (Erstellung der Container) oder je nach Hardware auch nicht ausreichen, wodurch empfohlen wird, zuerst mit Befehl 7 die MongoDB separat und mit Befehl 8 die anderen Services zu starten. Die Container werden dann in einem eigenen Netzwerk erstellt (siehe Abbildung 5, S.11).

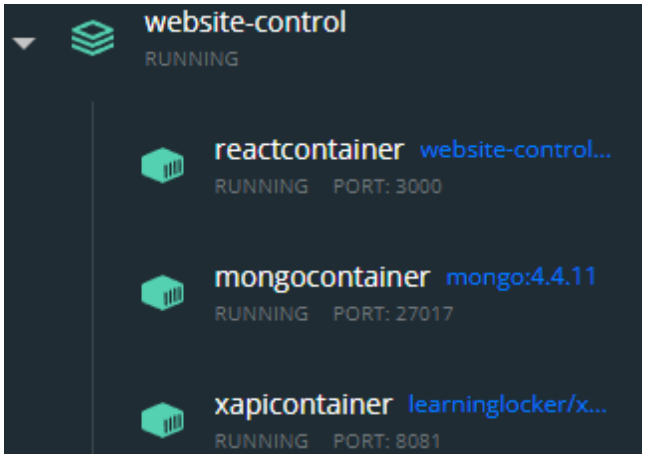

Abbildung 5: Docker Services nach ausführen der Befehle (docker-compose)

7 Fazit und Ausblick

Die Containervirtualisierung spielt in der heutigen Welt der Entwickler eine wichtige Rolle und löst in bestimmten Bereichen die Hardware-Virtualisierung ab.
Dieser Praxistransferbericht hat das Thema nur grob angeschnitten, wobei eine Vielzahl von Funktionen und Features möglich sind. Die Implementierung konnte nicht vollständig umgesetzt werden, wobei die meisten Anwendungen jedoch einsatzfähig sind. Die React Web-App kann daher als Container in den laufenden Betrieb genommen werden, wodurch die Performanz und Nutzbarkeit gesteigert wird.
Entsprechend sind auch nicht alle Anforderungen der Implementierung erfüllt. Zukünftig wäre es möglich die React Web-App von dem Learning Management System netzwerktechnisch zu trennen, wodurch eine höhere Sicherheit zu gewährleistet wird. Des Weiteren gibt es einige Möglichkeiten, um die Verbindung zwischen xAPI und MongoDB stabiler und einfacher auszuführen. Im xAPI-Service könnte die Anzahl der Versuche bzw. die Zeit angepasst werden, die gebraucht wird, um eine Verbindung mit der MongoDB zu suchen. Eine andere Möglichkeit wäre es, das Starten des xAPI-Service Container einige Sekunden warten zu lassen. Dies stellt aber eine „unsaubere“ und inkonsistente Lösung dar. Des Weiteren sollte die Learning Locker Anwendung von einem Entwickler-Team noch in Docker umgesetzt werden. Die Versionen der Images können für einige Zeit genutzt werden, wobei sie periodisch aktualisiert und getestet werden sollten, damit bestimmte Sicherheitslücken ausgeschlossen werden können. Ein passendes und aktuelles Beispiel ist die Log4J bzw. Log4Shell Sicherheitslücke.

8 Literaturverzeichnis

[Chr20] Christian Baun: *Betriebssysteme kompakt*. Berlin, Heidelberg: Springer Vieweg, 2020.

[Kof21] Michael Kofler und Bernd Öggl: *Docker - Das Praxisbuch für Entwickler und DevOps-Teams*. Bonn: Rheinwerk Computing, 2021.

[Kou06] Jacob S. Kounin: *Techniken der Klassenführung. Standardwerke aus Psychologie und Pädagogik. Reprints.*. Münster: Waxmann, 2006.

[Lac20] Maximilian Lackner und Horst Orsolits: *Virtual Reality und Augmented Reality in der Digitalen Produktion*. Wiesbaden: Springer Gabler, 2020.

[Wie19] Axel Wiepke, Eric und Dirk Richter und Raphael Zender: *Einsatz von Virtual Reality zum Aufbau von Klassenmanagement-Kompetenzen im Lehramtsstudium*. N. Pinkwart und J. Konert (Hrsg.). DELFI 2019. Bonn: Gesellschaft für Informatik e.V.: 2019, S. 133-144. Abgerufen 12. Jan. 2021. [online] https://dl.gi.de/handle/20.500.12116/24390

9 Eidesstattliche Erklärung

Ich erkläre ehrenwörtlich:

1. dass ich meinen Praxistransferbericht selbständig verfasst habe,

2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,

3. dass ich meinen Praxistransferbericht bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Werder (Havel), den 06.01.2022
Ort, Datum

Yüxi Zhang

10 Anhang

Nr.	Befehl	Erklärung
1	docker build . -t reactimage	Bauen eines Images aus einem Dockerfile aus dem aktuellen Verzeichnis, Name: „reactimage“
2	docker run -d -p 3000:3000 --name reactcontainer reactimage	Starten eines Containers mit dem Namen „react“ aus dem Image reactimage, ohne Flag -d (detached) würde Prozess in Konsole angezeigt werden (nicht unbedingt notwendig)
3	docker network create lrs	Erstellung des Netzwerk lrs für Netzwerkinterne Namensauflösung, keine Angabe von IP-Adressen nötig
4	docker run -d -p 27017:27017 --network lrs -v C:\Pfad\Ordner\mongo-data:/data/db:rw --name mongocontainer mongo:4.4.11	Starten eines MongoDB Containers mit der Portweiterleitung 27017, Flag -v (volume, Host-Pfad:Container-Pfad:Rechte) um Daten zu persistieren
5	docker exec -it mongocontainer bash	Interagieren mit der MongoDB über eine Kommandozeile, zum Testen „mongo“ eingeben
6	docker run -d -p 8081:8081 --network lrs -e MONGO_URL=mongodb://mongocontainer --name xapicontainer learninglocker/xapi-service:4.1.2	Starten des xAPI-Service mit Portweiterleitung 8081, xAPI braucht eine Verbindung mit MongoDB, Verbindung mittels Überschreibens einer Umgebungsvariable
7	docker-compose up -d mongo	Starten des Mongo Services aus der docker-compose.yml Datei
8	docker-compose up -d	Starten aller konfigurierten Services der docker-compose.yml Datei

Tabelle 1: Übersicht über Befehle der Docker Implementierung