Template LaTeX Wiki von BAzubis für BAzubis

Projektarbeit 1 (T3_1000)

im Rahmen der Prüfung zum

Bachelor of Science (B.Sc.)

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Karlsruhe

von

Yannik Schiebelhut

Abgabedatum:	04. Oktober 2021
Bearbeitungszeitraum:	01.10.2020 - 03.10.2021
Matrikelnummer, Kurs:	3354235, TINF20B1
Ausbildungsfirma:	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf, Deutschland
Betreuer der Ausbildungsfirma:	Helge Dickel
Gutachter der Dualen Hochschule:	DH-Vorname DH-Nachname

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Projektarbeit 1 (T3_1000) mit dem Thema:

Template LaTeX Wiki von BAzubis für BAzubis

gemäß § 5 der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. September 2017 selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, den 12. Juli 2021

Schiebelhut, Yannik

Inhaltsverzeichnis

Abkürzungsverzeichnis

AJAX Asynchronous Javascript and XML

API Application Programming Interface

EWM Extended Warehouse Management

ewm-sim EWM Simulator

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

JSON JavaScript Object Notation

npm Node Package Manager

OData Open Data Protocol

SDK Software Development Kit

SSH Secure Shell

WSL Windows-Subsystem für Linux

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1 Einleitung

1.1 Was ist der ewm-sim?

In der vierten industriellen Revolution verändert sich auch der Arbeitsalltag in Lagerhallen. Mobile Roboter finden verstärkt Einsatz, um die Arbeiter zu unterstützen. Das Projekt Extended Warehouse Management (EWM) Cloud Robotics der SAP hat das Ziel, die Integration von Robotern verschiedenster Hersteller in ein Netzwerk auf Basis von Google Cloud Robotics zu ermöglichen und dieses an ein SAP EWM-System anzubinden. Zu Demonstrations- und Entwicklungszwecken wird eine Simlationsumgebung erstellt, in der ein virtuelles Warenlager präsentiert wird, in dem Roboter beispielhafte Aufträge bearbeiten. Um nun zu vermeiden, dass ein vollständiges EWM-System für solch eine Simulation deployed werden muss, wurde der „ewm-sim“ eingeführt. Er stellt einen kleinen Web-Server dar, welcher die Schnittstelle, über die die Roboter ihre Aufträge vom EWM-System erhalten, detailgetreu nachbildet.

1.2 Kritikpunkte der ursprünglichen Implementierung

Wie bereits erwähnt, soll der ewm-sim die Schnittstelle eines EWM-Systems nachbilden. Hierbei handelt es sich um einen Open Data Protocol (OData)-Service. Für die Implementierung wurde hier auf den bestehenden Mock Server von SAPUI5 gesetzt, der normalerweise in der Frontendentwicklung dazu dient, entsprechende Schnittstellen einer Datenbank nachzubilden. Dieser ist jedoch nicht für die Backend-Entwicklung vorgesehen. Leider bringt er somit das Problem mit sich, dass er sich nur innerhalb der Laufzeitumgebung einer SAPUI5-App verwenden lässt, welche wiederum zwangsläufig in einem Browser laufen muss. In der Abbildung 1.1 ist der Aufbau des bisherigen ewm-sim veranschaulicht. Er stellt eine Art gekapseltes System dar. Die Anfragen, die an den nach außen hin geöffneten Webserver geschickt werden, landen über den WebSocket Server bei einer headless Instanz von Google Chrome, in welchem wiederum eine SAPUI5-App

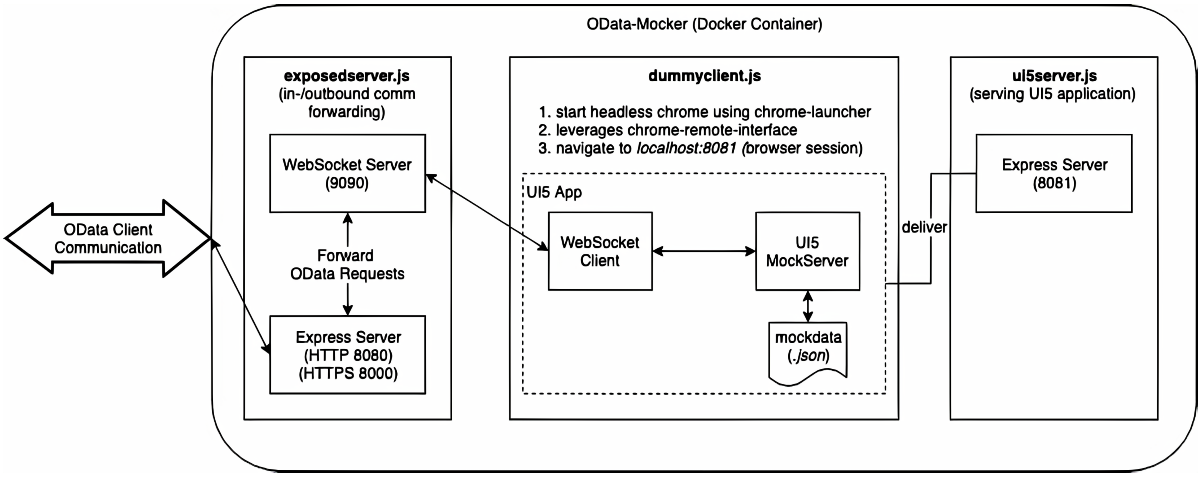

Abbildung 1.1: Aufbau des ewm-sim v1

ausgeführt wird. Diese ist mit dem SAPUI5 Mockserver verknüpft, welcher die Daten, die er bereitstellen soll, aus einer JavaScript Object Notation (JSON)-Datei einliest.

Wie gut zu erkennen ist, bringt diese Implementierung einen großen Overhead und somit mögliche Fehlerquellen mit sich. Ziel des in dieser Praxisarbeit behandelten Projekts soll es sein, das Konzept des ewm-sim zu optimieren und diesen daraufhin im Anschluss neu zu implementieren.

2 Theoretische Grundlagen

Zur Bearbeitung dieses Projekts wurden einige Technologien eingesetzt, für die hier zunächst Vor- und Nachteile sowie theoretische Grundlagen aufgeführt werden sollen. Anhand dessen soll die Entscheidung für die entsprechenden Technologien nachvollziehbar dargelegt werden.

2.1 Docker

Docker ist eine freie Implementierung des Konzepts der Containervirtualisierung. Ein sogenannter Docker Container ist plattformübergreifend lauffähig. Voraussetzung ist lediglich, dass der Docker Daemon auf dem Host-System installiert ist und im Hintergrund ausgeführt wird. Damit die Container, wie beschrieben, auf alles großen Betriebssystemen laufen können, setzen sie üblicherweise auf Linux auf. Somit wird unter Windows das Windows-Subsystem für Linux (WSL) genutzt, um die Container auszuführen. Dies ist eine schlanke Integration des Linux-Kernels in das Windows-Betriebssystem, welche durch Optimierung und Reduktion auf essenzielle Bestandteile den deutlich größeren Overhead von herkömmlicher Virtualisierung erspart.

Kennzeichnend für das Konzept von Docker sind die sogenannten Container. Diese stellen eine vollständige Zusammenstellung aller Komponenten dar, die eine bestimmte Anwendung zur Ausführung benötigt. Dadurch ist es sehr einfach, eine Applikation schnell und einheitlich auf einem neuen System zu deployen und dabei direkt alle Abhängigkeiten mitzuinstallieren und richtig zu konfigurieren.

Docker bietet außerdem eine „Docker Hub“ genannte Plattform an. Sie bietet eine Möglichkeit, eigene Container Images (also Systemabbilder/Bauanleitungen für eine bestimmte Container-Umgebung) hochzuladen, sowie die von anderen Nutzern hochgeladenen Images zu nutzen. Diese können zum Beispiel mit einem einzigen Kommandozeilenbefehl heruntergeladen und deployed werden. Sogar als Basis für neue, eigene Images können Images von Docker Hub verwendet werden.

Die angestrebte Anwendung als Docker Container zu konzipieren ist neben den oben aufgezählten Vorteilen auch vor allem deshalb naheliegend, weil die erste Version des ewm-sim bereits in dieser Form bereitgestellt wurde. Somit kann bei entsprechender Umsetzung eine identische oder zumindest sehr ähnliche Schnittstelle zur Anbindung an die anderen Softwarekomponenten des Projektes genutzt werden und Umweltfaktoren, die einen reibungslosen Betrieb verhindern würden, sind von vorneherein ausgeschlossen.

2.2 Node.js

Seit der ursprünglichen Einführung im Jahre 1995 hat die interpretierte Skriptsprache „JavaScript“ rasch an Beliebtheit und Bedeutung gewonnen. Heutzutage ist sie aus der Web-Entwicklung nicht mehr wegzudenken. Node.js stellt eine Möglichkeit dar, mithilfe derer JavaScript nicht mehr nur clientseitig im Browser, sondern auch serverseitig ausgeführt werden kann. Ein großer Vorteil davon liegt darin, dass Web-Entwickler, die bereits viel mit JavaScript arbeiten und dementsprechend damit vertraut sind, nur noch eine Sprache benötigen, um sowohl Frontend als auch Backend zu entwickeln. Zudem ermöglicht Node.js die parallelisierte Ausführung von Code. Dies bedeutet, dass ein Web-Server nicht wie traditionell üblich eine Schlange von Anfragen bilden und diese nacheinander beantworten muss, sondern er die Anfragen stattdessen gleichzeitig beantworten kann.

npm-Module Eine weitere nützliche Funktionalität von Node.js ist der Node Package Manager (npm). Mit ihm können sehr einfach von der Community erstellte Bibliotheken installiert und in ein Programm eingebunden werden. Auf diese Weise stehen beispielsweise fertige Frameworks für Web-Server, Unit-Tests oder erweiterte Logger zur Verfügung.

Die verwendeten Pakete werden in der package.json aufgezeichnet. Diese Datei dient zudem als eine Konfigurationsdatei für das Projekt. Dort können unter anderem Daten zum Autor, Lizenzen und Repository hinterlegt werden, sowie Skripte für npm definiert werden (zum Beispiel zum Starten, Testen oder Bauen des Projekts).

2.3 Postman

Bei der Entwicklung eines Web-Servers ist es hilfreich, manuell Hypertext Transfer Protocol (HTTP)-Requests an diesen schicken zu können. Ganz grundlegende Anfragen können theoretisch schon durch einen Web-Browser abgesetzt werden, diese stoßen jedoch schnell an ihre Grenzen. Deshalb bietet sich das Programm Postman an. Es bietet eine übersichtliche Oberfläche, um die einzelnen Eigenschaften einer HTTP-Request zu konfigurieren und ermöglicht es auch, diese zu speichern. Mit Postman ist es möglich, jede Art von Anfrage an einen Server auszulösen und diese für wiederholten Gebrauch in sogenannten „Collections“ zu sortieren, welche sich auch als JSON-Datei exportieren lassen.

2.4 Kubernetes / Google Cloud Platform

2.5 Git

Git ist ein quelloffenes Tool zur Versionsverwaltung, welches ursprünglich von Linus Torvalds zur Entwicklung des Linux-Kernels geschrieben wurde. Die Bedienung erfolgt in der Regel über die Kommandozeile. Dateien werden in sogenannten Repositories verwaltet, welchem sie durch einen „Commit“ hinzugefügt oder aktualisiert werden. In einem Repository kann es zudem beliebig viele „Branches“ geben, zwischen denen flexibel gewechselt werden kann. Die Dateien eines Branches sind unabhängig, d.h. es gibt zum Beispiel einen Hauptbranch, auf dem der stabile Stand des Codes geführt wird und einen, auf dem ein neues Feature entwickelt wird. Branches können durch „mergen“, weitestgehend automatisch, ineinander überführt werden.

2.5.1 GitHub

Eine große Stärke von Git ist die Möglichkeit zur einfachen Kollaboration. Hierfür muss ein Repository auf einem Server liegen, von dem aktuelle Änderungen auf den lokalen Rechner heruntergeladen (pull) oder veröffentlicht (push) werden können. GitHub ist einer der größten Anbieter für das Hosten von Git-Repositories und bietet auf der Website

noch zahlreiche weitere Möglichkeiten zum Management und der Dokumentation von Projekten, wie zum Beispiel ein in das Projekt integriertes Wiki.

2.6 Travis-CI

In der professionellen Software-Entwicklung ist es üblich, automatisierte Tests für den Code zu schreiben. Das Tool Travis-CI ist ein Dienst, der automatisch aktiv wird, wenn beispielsweise neuer Code auf einem GitHub-Repository hochgeladen wird. In einer Konfigurationsdatei, die im Projektverzeichnis liegt, können alle Randbedingungen (wie Programmiersprache und Betriebssystem) sowie der genaue Ablauf des Tests festgelegt werden. Auch komplexere Abläufe, wie mehrschrittige Builds oder parallelisierte Operationen, können mit Travis-CI umgesetzt werden.

2.7 Jira

Jira ist ein Tool, das in der agilen Projektmanagement-Methode „Scrum“ Anwendung findet. Dort können Backlog-Items mit dazugehörigen Sub-Tasks definiert und in Sprints verwaltet werden. In einem aktiven Sprint können mit wenigen Klicks Aufgaben einem Teammitglied zugewiesen oder neue Sub-Tasks erstellt werden. Elemente eines Sprints können bequem per Drag and Drop zwischen Spalten hin und her geschoben werden, welche den aktuellen Status anzeigen.

2.8 Visual Studio Code

3 Vorbereitung

3.1 Analyse der bisherigen ewm-sim Implementierung