



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 L^AT_EX 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 11. Dezember 2020

Schiebelhut, Yannik

Inhaltsverzeichnis

Formelverzeichnis	III
Abkürzungsverzeichnis	IV
Abbildungsverzeichnis	V
Tabellenverzeichnis	VI
Quellcodeverzeichnis	VII
1 Einleitung	1
1.1 Was ist der ewm-sim?	1
1.2 Kritikpunkte der ursprünglichen Implementierung	1
2 Theoretische Grundlagen	3
2.1 Docker	3
2.2 Node.js	4
2.3 Postman	5
2.4 Kubernetes / Google Cloud Platform	5
2.5 Git	5
2.6 Travis-CI	5
2.7 Jira	5

Formelverzeichnis

A	mm ²	Fläche
D	mm	Werkstückdurchmesser
d_{\min}	mm	kleinster Schaftdurchmesser
L_1	mm	Länge des Werkstückes Nr. 1
	Grad	Freiwinkel
	Grad	Keilwinkel

Abkürzungsverzeichnis

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
EWM	Extended Warehouse Management
HTTPS	Hypertext Transfer Protocol Secure
npm	Node Package Manager
SDK	Software Development Kit
SSH	Secure Shell
WSL	Windows-Subsystem für Linux

Abbildungsverzeichnis

1.1	Aufbau des ewm-sim v1	2
-----	---------------------------------	---

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, ein Roboternetzwerk auf Basis von Google Cloud Robotics an ein SAP EWM-System anzubinden. Zu Demonstrationszwecken wird eine Simulationsumgebung erstellt, in der ein virtuelles Warenlager präsentiert werden, in dem Roboter beispielhafte Aufträge bearbeiten. Um nun zu vermeiden, dass ein vollständiges EWM-System für solch eine Simulation deployed werden muss, wurde „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 OData-Service. Für die Implementierung wurde hier auf den bestehenden MockServer von SAPUI5 gesetzt. Dieser ist jedoch zur Frontend- und nicht zur Backend-Entwicklung vorgesehen. Leider bringt er das Problem mit sich, dass er sich nur innerhalb einer SAPUI5-App verwenden lässt, welche wiederum in einem Browser läuft. In der Abbildung 1.1 ist der Aufbau des bisherigen ewm-sim veranschaulicht. Er stellt eine Art gekapseltes System dar. Die Anfragen, die gegen den nach außen hin geöffneten WebServer geschickt werden, landen über den WebSocket Server bei einer headless Instanz von Google Chrome, in welchem wiederum die SAPUI5-App ausgeführt wird. Diese ist mit dem SAPUI5 Mockserver verknüpft, welcher die Daten, die er bereitstellen soll, aus json Dateien einliest.

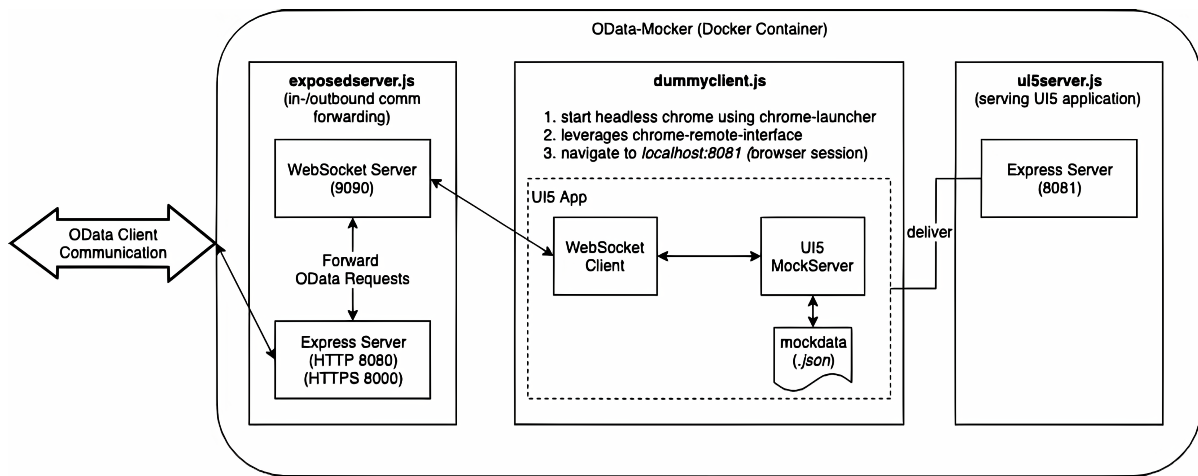


Abbildung 1.1: Aufbau des ewm-sim v1

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

2 Theoretische Grundlagen

Zur Bearbeitung dieses Projekts wurden einige Technologien eingesetzt, für die hier zunächst die theoretischen Grundlagen erläutert werden sollen.

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. Damit die Container, wie beschrieben, auf allen 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 Windows und erspart den Overhead von herkömmlicher Virtualisierung.

Kennzeichnend für das Konzept von Docker sind die sogenannten Container. Diese stellen eine 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 mit zu installieren und richtig zu konfigurieren.

Docker bietet außerdem eine Plattform an, welche Docker Hub genannt wird. Sie bietet eine Möglichkeit, eigene Docker Images hochzuladen, sowie die von anderen Nutzern hochgeladenen Images zu nutzen. Diese können zum Beispiel mit einem einzigen Kommandozeilenbefehl herunter geladen werden und sogar als Basis für neue, eigene Images dienen.

2.2 Node.js

In den letzten Jahren hat JavaScript in der Web-Entwicklung zunehmend an Bedeutung gewonnen. Node.js stellt eine Möglichkeit dar, wie JavaScript nicht mehr nur clientseitig im Browser ausgeführt werden kann, sondern auch serverseitig. 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 Front- als auch Backend zu entwickeln. Zudem kann JavaScript asynchron ausgeführt werden. Dies bedeutet, dass ein Web-Server nicht wie traditionell üblich eine Schlange von Anfragen bilden muss und diese nacheinander beantworten, sondern er kann die Anfragen gleichzeitig beantworten.

Fachliche
Richtig-
keit
prüfen
und kor-
rigieren

2.2.1 npm-Module

Eine weitere nützliche Funktion von Node.js ist der Node Package Manager (npm). Mit ihm können sehr einfach, von der Community erstellte, Bibliotheken installiert und in das 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

2.4 Kubernetes / Google Cloud Platform

2.5 Git

2.5.1 GitHub

2.6 Travis-CI

2.7 Jira