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 15. August 2021

Schiebelhut, Yannik

Inhaltsverzeichnis

Abkürzungsverzeichnis

AJAX Asynchronous Javascript and XML

API Application Programming Interface

EWM Extended Warehouse Management

ewm-sim EWM Simulator

GKE Google Kubernetes Engine

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

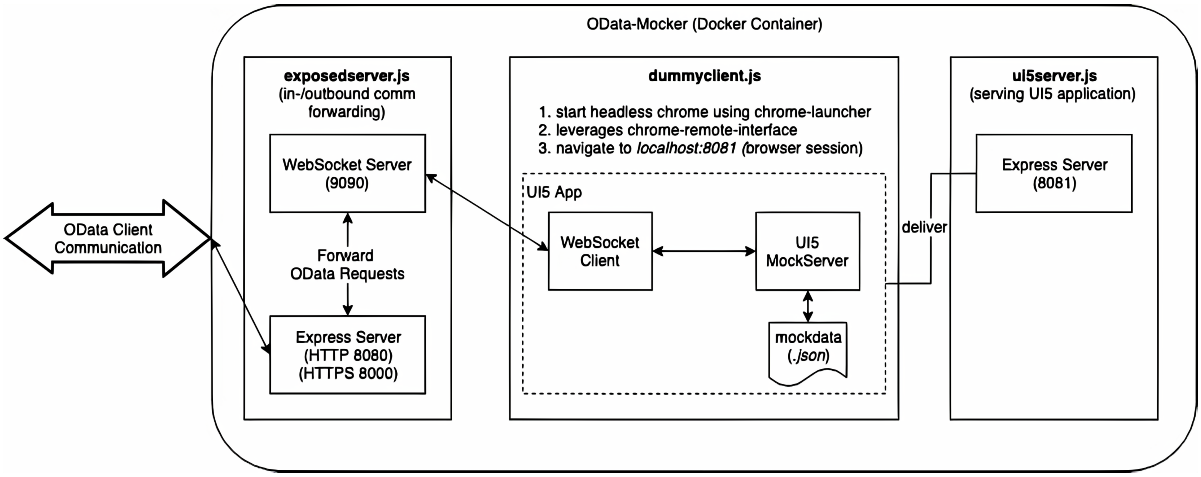

Abbildung 1.1: Aufbau des ewm-sim v1

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.

Verweis auf Studentenprojekt, Zusammenarbeit mit Roboter-Anbindung und Unity Simulation

2 Theoretische Grundlagen

Zur Bearbeitung dieses Projekts wurden einige Technologien eingesetzt, für die hier zunächst Vor- und Nachteile sowie theoretische Grundlagen aufgeführt werden sollen. Falls eine Entscheidung für eine entsprechende Technologie getroffen werden musste, soll diese anhand der erläuterten Kriterien 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. [1]

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. [2]

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. [3]

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 Kubernetes

Wie in 2.1 beschrieben, können mithilfe von Containern leicht Applikationen in einer neuen Umgebung eingerichtet werden. Kubernetes stellt eine quelloffene Option zur erleichterten und automatisierten Verwaltung von containerisierten Services dar. Dabei wird von Kubernetes ein deklarativer Ansatz verfolgt. Das bedeutet: Der Nutzer beschreibt den gewünschten Zustand – zum Beispiel über Konfigurationsdateien oder die Konsole – und Kubernetes ermittelt selbstständig die Schritte, die zur Erreichung und Aufrechterhaltung notwendig sind. Mittels Kubernetes ist es auch möglich, Dienste dynamisch zu skalieren (d. h. es werden abhängig von der aktuellen Anzahl der Nutzer eines Dienstes die Ressourcen einer Anwendung entweder erhöht oder wenn möglich gespart) sowie Lastausgleich und Redundanz zwischen verschiedenen Instanzen desselben Services herzustellen. [4, 5]

Instanzen eines Kubernetessystem werden auch als Cluster bezeichnet. Ein Cluster besteht aus beliebig vielen Nodes. Letztere können dabei beliebige Maschinen sein (in der Regel virtuelle Maschinen oder physische Server) auf denen die eigentlichen Anwendungen laufen. Die Interaktion mit einem Cluster findet über den sogenannten *Kubernetes Master* statt. Er stellt eine zentrale Verwaltungsstelle dar, mit den Nodes wird praktisch nie direkt interagiert. [6]

Google Kubernetes Engine (GKE) Kubernetes kann theoretisch auf jedem Heimcomputer installiert und betrieben werden. Primär findet es jedoch Anwendung im Cloud Computing. Hierbei können fertige Kubernetes Cluster gemietet werden. Dies geschieht normalerweise in Form von virtuellen Servern, deren Spezifikationen nach den eigenen Bedürfnissen gewählt werden können, ohne dass selbst Hardware angeschafft werden

muss. Somit ist es möglich, auch für kurze Projekte eine größere Infrastruktur zu deployen oder bei Bedarf und mithilfe der Skalierungsmöglichkeit von Kubernetes mit wenigen Klicks die Ressourcen einer Anwendung beziehungsweise eines Clusters anzupassen. Weiterhin ist es vorteilhaft, dass nur für die tatsächlich genutzte Rechenzeit und -last gezahlt werden muss. Google ist mit GKE einer der größeren Anbieter von Kubernetes Clustern. [**gke**] Die Entwicklungsabteilung, im Rahmen derer dieses Projekt entwickelt wird, nutzt GKE, um dort dynamisch Testumgebungen aufbauen zu können.

2.3 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.

Node Package Manager (npm) Eine weitere nützliche Funktionalität von Node.js ist der Node Package Manager. 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 Logger mit erweiterter Funktionalität zur Verfügung. npm hilft außerdem bei der strukturierten Verwaltung eines Node.js Projektes. Er stellt einen Assistenten bereit, um automatisiert eine sogenannte package.json zu generieren und auf das Projekt anzupassen. In ihr werden grundlegende Eigenschaften des Projektes verzeichnet, so zum Beispiel

- Name
- Beschreibung

- kennzeichnende Schlüsselwörter
- Lizenzen
- Autoren und Mitwirkende
- Projektwebseite und
- Quellcode-Repository.

Darüber hinaus werden hier allerdings auch die mit npm installierten Pakete inklusive deren Versionsnummer gespeichert, welche wiederum noch in allgemeine Abhängigkeiten und solche, die nur zu Entwicklungszwecken vonnöten sind, untergliedert werden. Möchte jemand anders das Projekt weiterentwickeln oder ein Deployment durchführen, so kann mithilfe von npm install im Handumdrehen die entsprechende Laufzeitumgebung dafür schaffen und Abhängigkeiten befriedigen. Des Weiteren können in der package.json auch Skripte festgelegt werden, die ein einfaches Ausführen, Testen oder Bauen des Projektes, sowie ähnliche Aufgaben ermöglichen. [7]

Die Wahl der zu verwendenden Programmiersprache fiel aus verschiedenen Gründen auf Node.js. Zunächst muss betrachtet werden, dass die simple Skriptsprache Node.js genau für den Einsatzzweck der serverseitigen Entwicklung konzeptioniert wurde. Weiterhin ist der SAPUI5 Mock Server selbst in JavaScript geschrieben. Mithilfe von einigen bereits existierenden npm-Modulen ist somit eine sehr einfache Integration mit den bereits vorhandenen Softwarekomponenten möglich. Auf Docker Hub existiert außerdem eine breite Auswahl an vorgefertigten Images für Node.js Umgebungen, sodass kein eigenständiges Containerimage von Grund auf hochgezogen werden muss.

2.4 Postman

Bei der Entwicklung eines Web-Servers ist es zu Testzwecken hilfreich, um nicht gar zu sagen unumgänglich, manuell Hypertext Transfer Protocol (HTTP)-Requests an diesen schicken zu können. Ganz grundlegende Anfragen können schon durch einen Web-Browser abgesetzt werden. Genau dies geschieht beim Aufruf jeder Website. Diese einfach im Browser abzusetzenden Anfragen stoßen jedoch schnell an ihre Grenzen, weshalb ein Programm mit erweitertem Funktionsumfang benötigt wird, bei dem detailliert Einfluss auf die Parameter von Requests genommen werden kann. Zunächst kommt ein einfaches

und auf vielen Systemen bereits vorinstalliertes Programm, welches einen solchen Funktionsumfang unterstützt, in den Sinn: *CURL*. Das konsolenbasierte Tool bietet allerdings keine Möglichkeit, Requests für spätere Verwendung zu speichern und erfordert mitunter die komplexe Kombination von Parametern, um das gewünschte Ergebnis zu erzielen. Anstelle dessen bietet sich Postman an. Das Programm stellt eine Lösung für alle oben beschriebenen Probleme von *CURL* dar. Es bietet eine übersichtliche Oberfläche, um die einzelnen Eigenschaften einer HTTP-Request bis ins kleinste Detail zu konfigurieren und ermöglicht es auch, diese zu speichern. Zur späteren Verwendung gespeicherte Anfragen können in Ordnern organisiert, mit einem Account synchronisiert oder auch als *Collection* im JSON-Format zum Austausch mit Mitarbeitern exportiert werden.

2.5 Git

Git ist ein quelloffenes Tool zur Versionsverwaltung, welches ursprünglich von Linus Torvalds zur Entwicklung des Linux-Kernels geschrieben wurde. Die klassische Bedienung erfolgt über die Kommandozeile. Mittlerweile gibt es aber auch einige grafische Bedienungsoberflächen, zudem wurde Git direkt in die gängigsten Entwicklungsumgebungen integriert. Dateien werden in sogenannten Repositories verwaltet, welchen 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 kann zum Beispiel einen Hauptbranch geben, auf dem der stabile Stand des Codes geführt wird und einen, auf dem ein neues Feature entwickelt wird. Branches können durch *mergen* ineinander überführt werden. Dies geschieht weitestgehend automatisch, es kann jedoch dabei auch zu Konflikten kommen. Beispielsweise kann dieser Fall eintreten, wenn dieselbe Zeile einer Datei in beiden Branches bearbeitet wurde. In dieser Situation muss dann manuell interveniert und der Konflikt beseitigt werden. Die Zustände des Quellcodes werden zum Zeitpunkt des jeweiligen Commits gespeichert und sind auch nachträglich noch einsehbar. So können neu hinzugekommene Fehler einfach mittels Rollbacks behoben werden.

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 werden können (push).

GitHub ist einer der größten Anbieter für das Hosting von Git-Repositories und bietet auf der Website noch zahlreiche weitere Möglichkeiten zum Management und der Dokumentation von Projekten. Hierzu gehören zum Beispiel ein in das Projekt integriertes Wiki, ein Tracker für Probleme (Issues), Statistiken zum Projekt und *Pull Requests* – ein Feature um Änderungen am Code anderer vorzuschlagen, auf den keine direkten Schreibrechte existieren.

2.6 Travis CI

In der professionellen Software-Entwicklung ist es üblich, automatisierte Tests (auch genannt Unittests oder Komponententests) für den Code zu schreiben, um diesen auf korrekte Funktionalität zu überprüfen. Das Tool Travis CI ist ein Dienst, der für die Durchführung genau dieser Art von Tests konzipiert wurde. In einer Konfigurationsdatei, die im Projektverzeichnis liegt, werden alle Randbedingungen (wie die genutzte Programmiersprache und das Betriebssystem, auf dem die Tests durchzuführen sind) sowie der genaue Ablauf der Tests festgelegt. Auch komplexere Abläufe, wie das Kompilieren von mehrschrittigen Builds oder parallelisierte Operationen, können mit Travis CI umgesetzt werden. Der Dienst wird dann automatisch aktiv, wenn beispielsweise neuer Code auf einem GitHub-Repository hochgeladen wird. Die Resultate der Tests können dann entweder auf der Website von Travis CI eingesehen werden oder auch als automatisch generierte Badge im Readme des GitHub-Repositorys oder auf der Projektwebsite eingebunden werden. Des Weiteren kann eine Benachrichtigung per E-Mail konfiguriert werden, sodass der für das Projekt verantwortliche direkte Rückmeldung darüber erhält, wenn etwas nach einem Update nicht mehr funktionsfähig sein sollte.

Travis CI ist eine gute und beliebte Möglichkeit, die Integrität des eigenen Codes zu garantieren. Allein deshalb schon sollte sie bei diesem Projekt zum Einsatz kommen. Ein weiterer Vorteil genau dieser Testsuite ist, dass sie bereits für EWM Cloud Robotics verwendet wird und somit die Tests des neu entstehenden ewm-sims direkt in das vorhandene Repository eingebunden werden können.

2.7 Jira

Die Abteilung in der EWM Cloud Robotics entwickelt wird, arbeitet nach der agilen Projektmanagement-Methodik *Scrum*. Dort findet Jira als Planungssoftware Verwendung. Sie erlaubt die Definition von Backlog-Items mit dazugehörigen Sub-Tasks, welche in Sprints verwaltet werden. In einem aktiven Sprint können mit wenigen Klicks Aufgaben einem Teammitglied zugewiesen oder neue Sub-Tasks erstellt werden. Die Hauptansicht zur Verwaltung ist in drei Spalten gegliedert, welche den aktuellen Status anzeigen: *To Do*, *In Progress* und *Done*. Zwischen Spalten hin und her geschoben werden Elemente eines Sprints bequem per Drag and Drop.

3 Vorbereitung

Die Abteilung, in der die Entwicklung von EWM-Cloud-Robotics vorangetrieben wird, legt einen starken Fokus auf innovative Entwicklung.

3.1 Analyse der bisherigen ewm-sim Implementierung

3.2 Einarbeitung in Basiskomponenten

Docker, Hello World, Dockerfile

Node.js, empfohlene Sprache, eventuelle Alternativen (gibt nicht viele, einfach optimal für diese Aufgabe → serverseitigen Code)

3.3 Suche nach einer geeigneten Basis

Kernaufgabe des ewm-sim ist die Simulation der OData-Schnittstelle eines vollständigen SAP-EWM-Systems. Grundsätzlich existieren viele Wege, über die dies erreicht werden kann. OData ist ein HTTP-basiertes Protokoll, welches eine offene Spezifikation darstellt. Also solche bieten sich grundsätzlich zwei mögliche Vorgehensweisen. Zum einen bestünde grundsätzlich die Möglichkeit, von Grund auf einen neuen Service zu implementieren, bei dem für sämtliche Requests, die an den Server gestellt werden können, das entsprechende Verhalten und mögliche Antworten abgedeckt werden müssen. Auf diese Weise wären der Entwicklung die größtmöglichen Freiheiten eingeräumt, brächte allerdings auf der anderen Seite auch eine Reihe von potenziellen Problemen und vermeidbaren Fehlern mit sich, da der OData Standard relativ umfangreich ist. Hinzu kommt, dass die Erfahrung in dieser ersten Praxisphase noch stark eingeschränkt war, weshalb dies wahrscheinlich zu einer Überforderung geführt hätte, das Projekt vermutlich nicht in der gesteckten

Zeit von zwei Monaten vollständig umsetzbar gewesen wäre und sich somit insgesamt ein alternativer Ansatz empfiehlt.

Dieser alternative Ansatz besteht aus vorgefertigten Modulen, die sich genau der Aufgabe widmen, eine einfache OData-Schnittstelle nachzubilden. Einige dieser Module finden sich beispielsweise in den npm-Repositories, wobei sie je nach Variante einen stark variierenden Funktionsumfang bieten.

Literaturverzeichnis

[1] *Was ist das Windows-Subsystem für Linux? | Microsoft Docs*. URL: https://docs.microsoft.com/de-de/windows/wsl/about (Einsichtnahme: 14. 07. 2021).

[2] *What is a Container? | App Containerization | Docker*. URL: https://www.docker.com/resources/what-container (Einsichtnahme: 14. 07. 2021).

[3] *Docker Hub - Container Image Library | Docker*. URL: https://www.docker.com/products/docker-hub (Einsichtnahme: 14. 07. 2021).

[4] Bloß, A. *Containerorchestrierung mit Kubernetes - Teil 4 — x-cellent technologies GmbH Blog*. 2019. URL: https://www.x-cellent.com/blog/containerorchestrierung-mit-kubernetes-teil-4/ (Einsichtnahme: 15. 07. 2021).

[5] *Was ist Kubernetes? | Kubernetes*. URL: https://kubernetes.io/de/docs/concepts/overview/what-is-kubernetes/%7B%5C#%7Dwas-bedeutet-kubernetes-k8s (Einsichtnahme: 15. 07. 2021).

[6] *Konzepte | Kubernetes*. URL: https://kubernetes.io/de/docs/concepts/%7B%5C_%7Dprint/ (Einsichtnahme: 15. 07. 2021).

[7] *package-lock.json | npm Docs*. URL: https://docs.npmjs.com/cli/v7/configuring-npm/package-lock-json (Einsichtnahme: 14. 07. 2021).