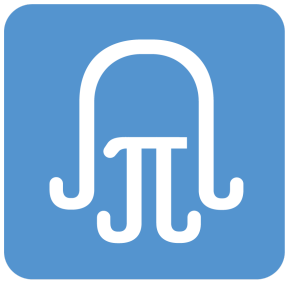
Architekturdokument



Auftragnehmer: OctoPi

Verantwortlich: Philip Dell, Timo Wenz

Im Auftrag von

sovanta AG (Heidelberg)

Softwareentwicklungsprojekt an der Hochschule Mannheim

Sommersemester 2023

Version 2.0

06.06.2023

Inhaltsverzeichnis

1 Versionsverzeichnis

Version	Änderung	Datum	Autor(en)
2.0	2. Abgabe Architekturdokument	06.06.2023	Philip Dell, Timo Wenz
1.7	Qualitätssicherung	06.06.2023	Philip Dell, Julian Wernz
1.6	Diagramme überarbeitet	05.06.2023	Philip Dell
1.5	Legende für Bausteinsicht angepasst	05.06.2023	Philip Dell, Timo Wenz
1.4	Bausteinsicht überarbeiten	26.05.2023	Philip Dell, Julian Wernz
1.3	Kapitel aufgrund des Feedbacks streichen	23.05.2023	Philip Dell
1.2	Architekturdokument an Feedback (Inhaltsverzeichnis) weiter anpassen	19.05.2023	Timo Wenz
1.1	Architekturdokument an Feedback anpassen	16.05.2023	Philip Dell
1.0	1. Abgabe Architekturdokument	09.05.2023	Philip Dell, Timo Wenz
0.8	Qualitätssicherung	09.05.2023	Thomas Martin, Steven Schmitt

0.7	Überarbeitung der Diagramme und allgemeine Überarbeitung und Qualitätssicherung	08.05.2023	Timo Wenz, Philip Dell, Jasmin Tschernoch, Julian Wernz
0.6	Architekturentscheidungen erweitern	06.05.2023	Timo Wenz
0.5	Qualitätsanforderungen, Risiken, Architekturentscheidungen, Lösungsstrategie	05.05.2023	Philip Dell, Thomas Martin, Jasmin Tschernoch, Timo Wenz, Julian Wernz
0.4	Diagramme verbessert und querschnittliche Konzepte angepasst	04.05.2023	Philip Dell, Timo Wenz, Julian Wernz
0.3	Struktur und Aufbau arc42	03.05.2023	Timo Wenz, Julian Wernz
0.2	Versionsverzeichnis angefügt	27.04.2023	Philip Dell
0.1	Inhaltsverzeichnis erstellt und Einleitung geschrieben	26.04.2023	Julian Wernz
0.0	Dokument aus arc42 Template erstellt	26.04.2023	Julian Wernz

Für die Versionierung dieses Dokuments gelten folgende Regeln, welche teamintern beschlossen wurden:

1. Die Versionsnummer besteht aus zwei Zahlen, getrennt durch einen Punkt.

2. Die erste Zahl der Versionsnummer wird ganzzahlig erhöht, sobald das Dokument für eine Abgabe bereit ist. Die Zahl nach dem Punkt wird zusätzlich auf 0 gesetzt.
3. Die Erhöhung der zweiten Zahl der Versionsnummer erfolgt, sobald der Inhalt um mindestens ein Kapitel erweitert oder der Inhalt bestehender Abschnitte geändert wurde.
4. Die initiale Version eines Dokuments ist 0.0.

2 Einführung und Ziele

2.1 Einführung

Das Architekturdokument legt die grundlegende Struktur, Komponenten und Funktionalitäten eines Systems dar und dient als Referenz für die Entwicklung und Wartung. Diese müssen zur Umsetzung der Softwarearchitektur und Entwicklung der Applikation berücksichtigt werden. Im Laufe des Sommersemester 2023 wird dieses Architekturdokument erstellt. Das Team OctoPi arbeitet an der Hochschule Mannheim und setzt sich aus sechs Mitgliedern zusammen. Die Beteiligten sind Philip Dell, Thomas Martin, Steven Schmitt, Jasmin Tschernoch, Timo Wenz und Julian Wernz.

Das arc42 Template dient als Vorlage für das vorliegende Dokument (https://arc42.org). Dieses wird verwendet, da es im Rahmen der Software-Engineering 2 Lehrveranstaltung der Hochschule behandelt wurde und alle Teammitglieder damit vertraut sind.

2.2 Aufgabenstellung

Ziel des Projektes ist es, eine prototypische Applikation im Bereich der Technikmessen für die Softwarefirma sovanta AG zu entwickeln (im Folgenden als sovanta adressiert). Diese soll den Besuchern dieser Messen die “sovanta *Innovation Factory* for *SAP BTP*” spielerisch vorstellen und ihnen die einzelnen Features der BTP näher bringen. Das Produkt soll die Benutzer informieren und einen guten ersten Eindruck der sovanta hinterlassen.

In den folgenden Abschnitten werden die zu erfüllenden Aufgaben beschrieben. Die Liste aller funktionalen und nicht funktionalen Anforderungen ist in der Anforderungsspezifikation zu finden. Siehe Anforderungsspezifikation | Kapitel 7 Systemleistungen (OctoPi 2023).

2.3 Qualitätsziele

Das Kapitel Qualitätsanforderungen entfällt, da alle wichtigen Anforderungen im Anforderungsdokument gefunden werden können. Zusätzlich entfallen die Qualitätsszenarien, da sie bei diesem Projekt keine Relevanz haben.

Priorität	Qualitätsziel	Beschreibung
Hoch	Zuverlässigkeit	Die Software soll zuverlässig sein und fehlerfrei über einen Messetag laufen.
Hoch	Benutzerfreundlichkeit	Die Bedienung des Systems soll für alle Nutzer in unter einer Minute erlernbar und intuitiv sein.
Mittel	Performant	Das System bedient sich flüssig, zeigt Änderungen sofort an und es treten keine spürbaren Verzögerungen auf.
Mittel	Wartbarkeit	Das System sollte einfach gewartet und repariert werden können. Dies bedeutet, dass unsere Applikation schnell und effizient wieder nach einer Wartung oder Reparatur einsatzbereit ist.

2.4 Stakeholder

Siehe Projekthandbuch v2.0 | Kapitel 3 Stakeholder (OctoPi 2023 : 7)

3 Randbedingungen

Siehe Anforderungsspezifikation | Kapitel 7.2.3 Randbedingungen (OctoPi 2023 : 31)

4 Kontextabgrenzung

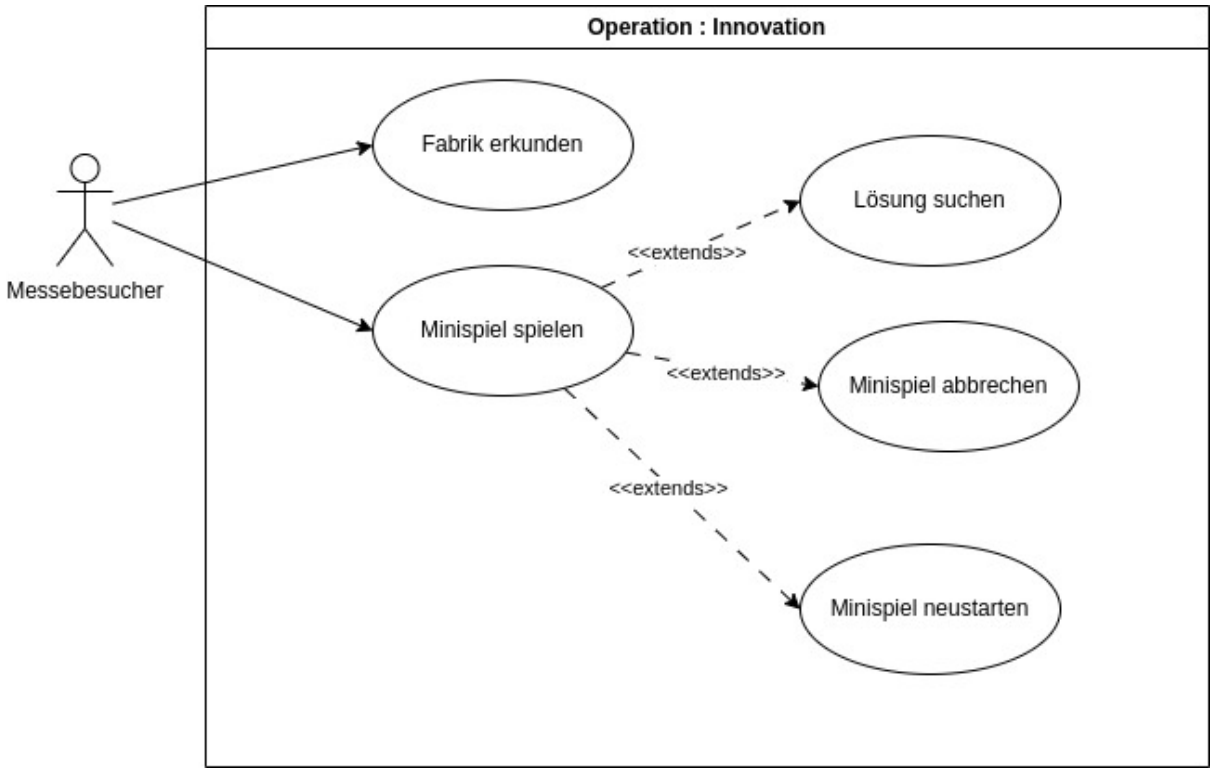


Abbildung 01: Use Case Diagramm

5 Bausteinsicht

Im *Frontend* benutzen wir als Team zwei Frameworks, *React.js* und *Three.js*. Im *Backend* wird grundsätzlich nur ein statischer Webserver verwendet, der in diesem Fall auf der SAP BTP deployt ist.

Die Komponente in der Pfeilrichtung wird benutzt und ein Datenfluss kann zurückkommen.

Ein Container beinhaltet beliebig viele Komponenten und bietet eine Möglichkeit, sie zu gruppieren und auf höheren Ebenen abstrahiert zu behandeln.

Eine Komponente ist eine Einheit von Funktionen, die noch weiter untergliedert werden kann.

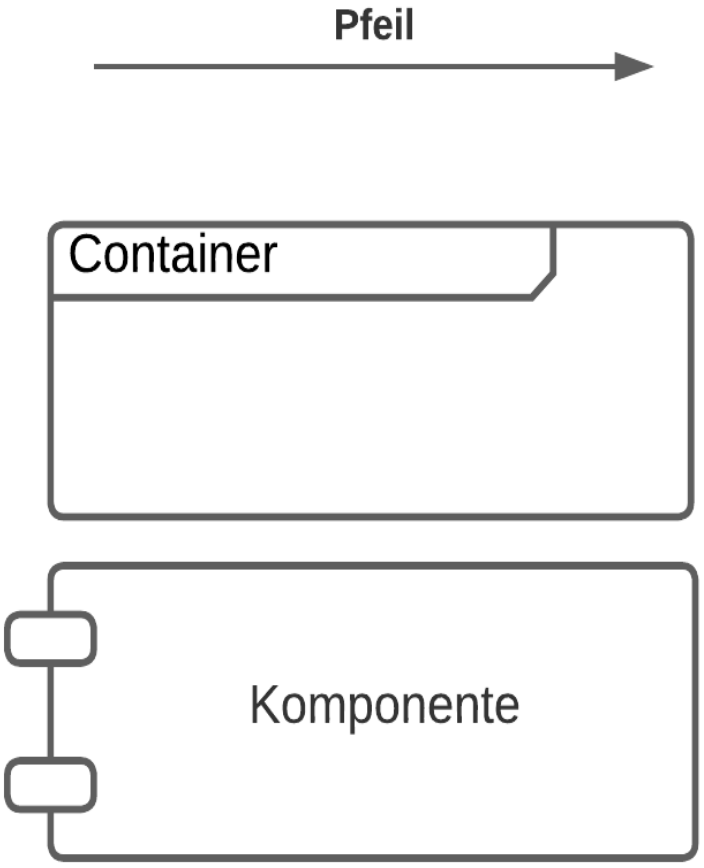


Abbildung 02: Legende

5.1 Gesamtsystem

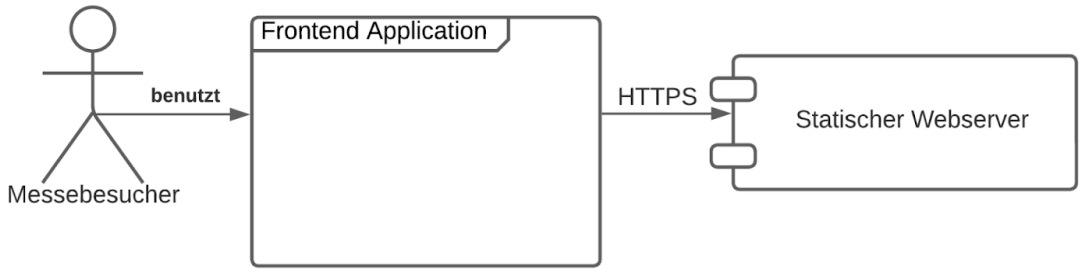


Abbildung 03: Level 1 Bausteinsicht

Name	Beschreibung / Verantwortung
Frontend Applikation	Hierdurch kann der Messebesucher mit dem Spiel interagieren. Die Frontend Applikation beinhaltet die gesamte Spiellogik, die dann im Browser auf den zur Verfügung gestellten Tablets des Messestandes läuft.
Statischer Webserver	Der statische Webserver liefert vorgefertigte Dateien wie *HTML*, *CSS* und Bilder an Clients, ohne sie dynamisch zu generieren. In unserem Fall läuft dieser auf der BTP in einem *Cloud Foundry* Container.
HTTPS (Hypertext Transfer Protocol Secure)	Ist das Netzwerk-Protokoll, über welches der Browser die Dateien aus dem Backend in das Frontend lädt.
Messebesucher	Da das Produkt für Messen gedacht ist, wird es auch von Messebesuchern am Stand verwendet. Mit Touch-Eingaben

	interagieren sie direkt mit dem Frontend im Browser. Dies findet auf Tablets des Messestandes statt.

5.2 Bausteinsicht Verfeinerung

Hier werden die Blöcke aus der Gesamtsystem-Sicht verfeinert. Dafür beschäftigen sich die beiden nachfolgenden Diagramme mit der zweiten und dritten Ebene des Frontends (siehe Abb. 04, Abb. 05). Das Backend wird nicht weiter verfeinert, da es sich um einen einfachen Webserver handelt. Mehr Informationen zum Backend können in der Verteilungssicht (Kapitel 7) genauer gefunden werden.

5.2.1 Frontend Applikation Ebene 2

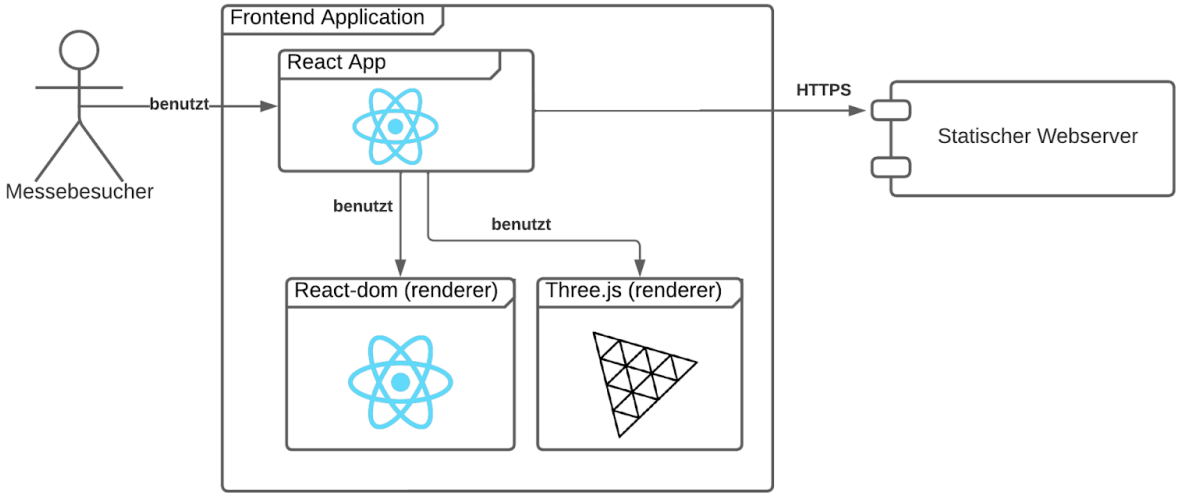


Abbildung 04: Bausteinsicht Ebene 2 Frontend

Name	Beschreibung/Verantwortung
Three.js (Renderer)	Der Renderer in Three.js ist verantwortlich für die Darstellung von 3D-Grafiken mithilfe von WebGL innerhalb des React-Kontexts. Er verarbeitet Geometrien, Materialien, Beleuchtung und

Name	Beschreibung/Verantwortung
	Kameras (View Port Kamera), um die 3D-Szene in einem Canvas-Element dem Messebesucher anzeigen zu können.
React-dom (Renderer)	Wird verwendet, um die React Komponenten der React App zu rendern, indem sie in DOM-Elemente umgewandelt und dem Messebesucher angezeigt werden.
React App	Die für das Projekt implementierten Logiken und Komponenten bilden in Kombination die React App. Verfeinert in Kapitel 5.2.3

5.2.2 Frontend Applikation Ebene 3

Die Komponenten in diesem Abschnitt sind Englisch benannt, da die Komponenten Skripts oder Ordnern im Projekt zugeordnet sind, welche auf Englisch programmiert werden.

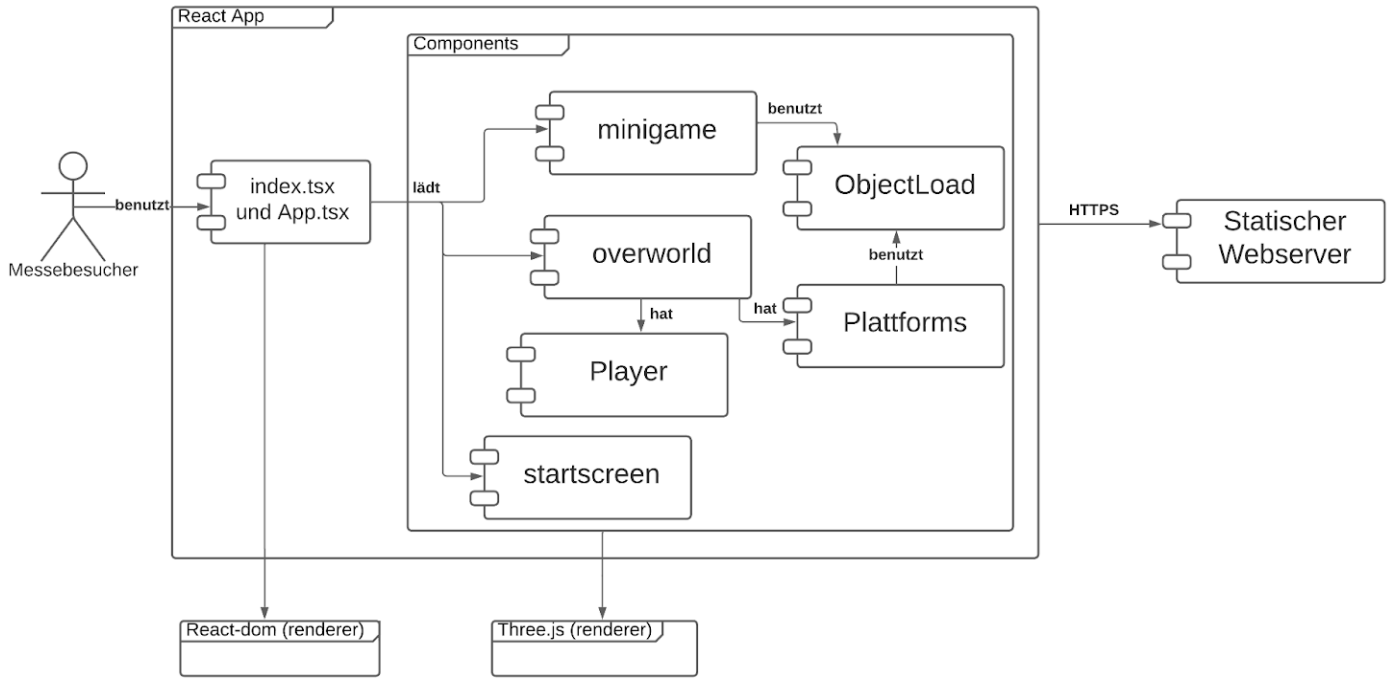


Abbildung 05: Bausteinsicht Ebene 3 Frontend

Name	Beschreibung/Verantwortung
index.tsx und App.tsx	Sie sind dafür verantwortlich, dem Messebesucher die richtigen Komponenten sowie die Szenen anzuzeigen. Zusätzlich wird mithilfe von ihnen zwischen den Szenen gewechselt. Alle geladenen Szenen bekommen zudem die Möglichkeit, über eine Hook das Laden einer anderen Szene auszulösen.
Components	Ist der Ordner des Projekts, in dem alle Spiellogik-Komponenten enthalten sind. Er wird in verschiedene Szenen (Overworld, Minigames, …) unterteilt und liefert allgemein nutzbare Komponenten wie z. B. ObjectLoad.
ObjectLoad	Die ObjectLoad Komponente ist für das vereinfachte Laden von Objekten (.obj und .mtl) zuständig. Dafür werden die Pfade relativ zum ,,/public'' Ordner des Projektes übergeben. In diesem Ordner können zudem alle 3D Objekte inklusive Materialien gefunden werden, sowie neue 3D Objekte hinzugefügt werden.
Platforms	Alle Plattformkomponenten der Overworld können in diesem Ordner gefunden werden. Jede Plattform Komponente hat einen Untergrund und mehrere Objekte, welche auf diesem platziert werden. Diese Objekte werden mithilfe von ObjectLoad geladen. Jede Platform muss intern die Referenz auf ihr Mesh an eine Hook übergeben, um so eine Kollision mit dem Player zu ermöglichen.
Player	Der Player kann von Messebesuchern über die Plattformen der Overworld gesteuert werden. Er kann sich nur auf den Plattformen und den Treppen befinden. Das wird ermöglicht, indem ihm die Meshes aller Plattformen und Treppen übergeben werden, welche dann auf Kollision überprüft werden.

Name	Beschreibung/Verantwortung
overworld	Ist die Szene, in der die Plattformen sowie der Player und die zugehörige Bewegungs- und Kollidieren Logik zu finden sind. Sie wird direkt nach dem Startscreen geladen und aus ihr kann man die Minispiele starten.
minigame	In diesem Ordner werden die einzelnen Minigames implementiert, welche auf jeder Plattform der Overworld zur Verfügung stehen und betreten werden können.
startscreen	Diese Komponente zeigt die Startanimation und den Startbutton an, die beim Starten oder Neu-Laden der Applikation angezeigt werden. Zudem dient sie als Screensaver, wenn die Applikation eine gewisse Zeit nicht benutzt wurde.

6 Laufzeitsicht

Dieses Kapitel stellt die Laufzeitsicht in Form von Sequenzdiagrammen dar (Insgesamt drei Sequenzdiagramme). Hierbei werden die einzelnen Ebenen näher betrachtet, bzw. deren genaue Abläufe dargestellt. Das Laufzeitszenario spricht alle wichtigen Komponenten an und ist somit für alle weiteren Funktionen der Applikation repräsentativ.

6.1 Webseite aufrufen

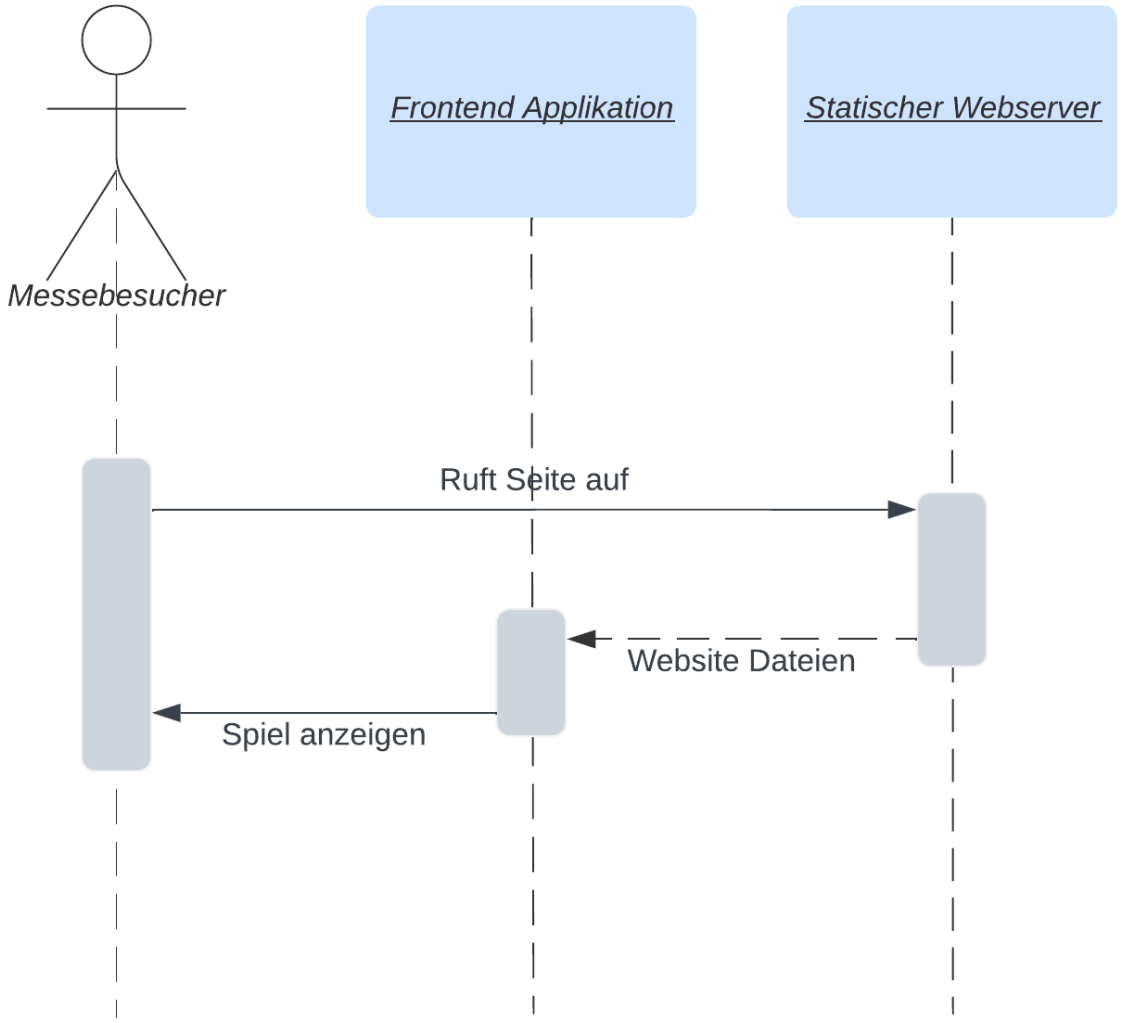


Abbildung 06: Laufzeitsicht Elemente aus Cloud Foundry laden

Das dargestellte Sequenzdiagramm zeigt einen Messebesucher, der die angegebene URL aufruft. Damit lädt er alle Website Dateien in den Browser und somit die Frontend Applikation, in welcher die restliche Logik dafür sorgt, das Spiel anzuzeigen.

6.2 Szene wechseln

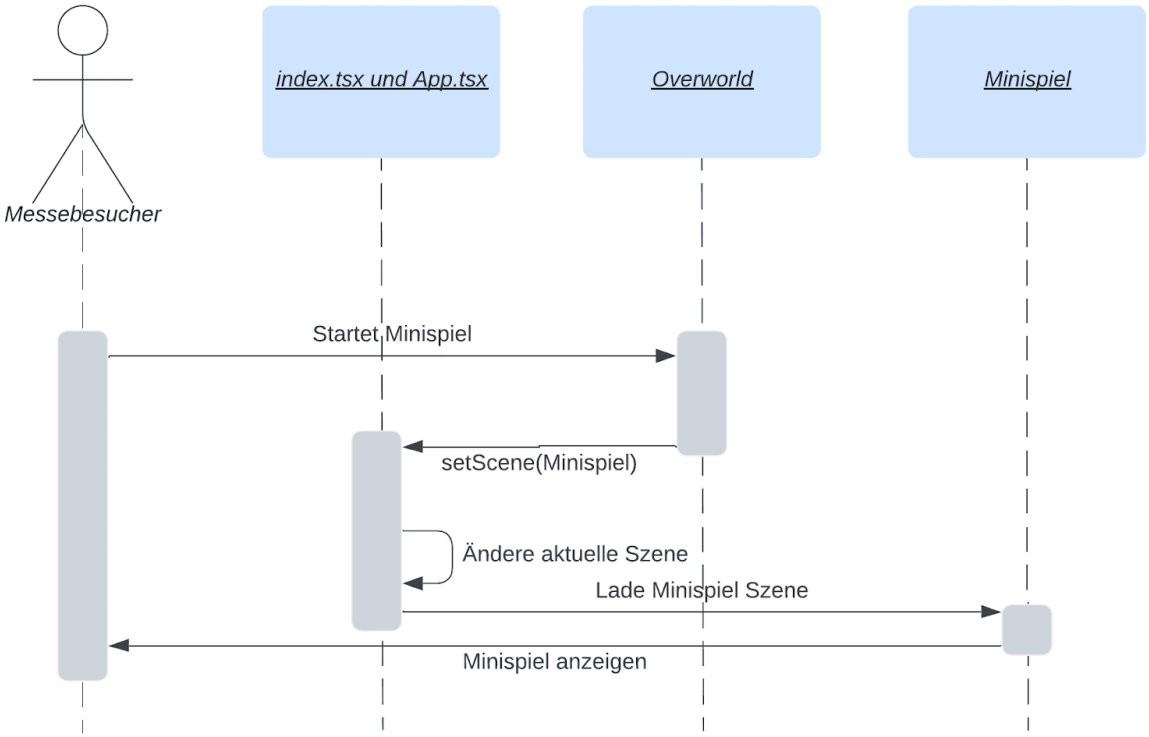


Abbildung 07: Laufzeitsicht, Szene wechseln (Minispiel starten)

Das dargestellte Sequenzdiagramm zeigt einen Messebesucher, welcher ein Minispiel über die Overworld startet. Dies führt zu einem Szenenwechsel, es wird das entsprechende Minispiel geladen und dem Messebesucher dieses im Browser angezeigt.

6.3 Spiel abschließen

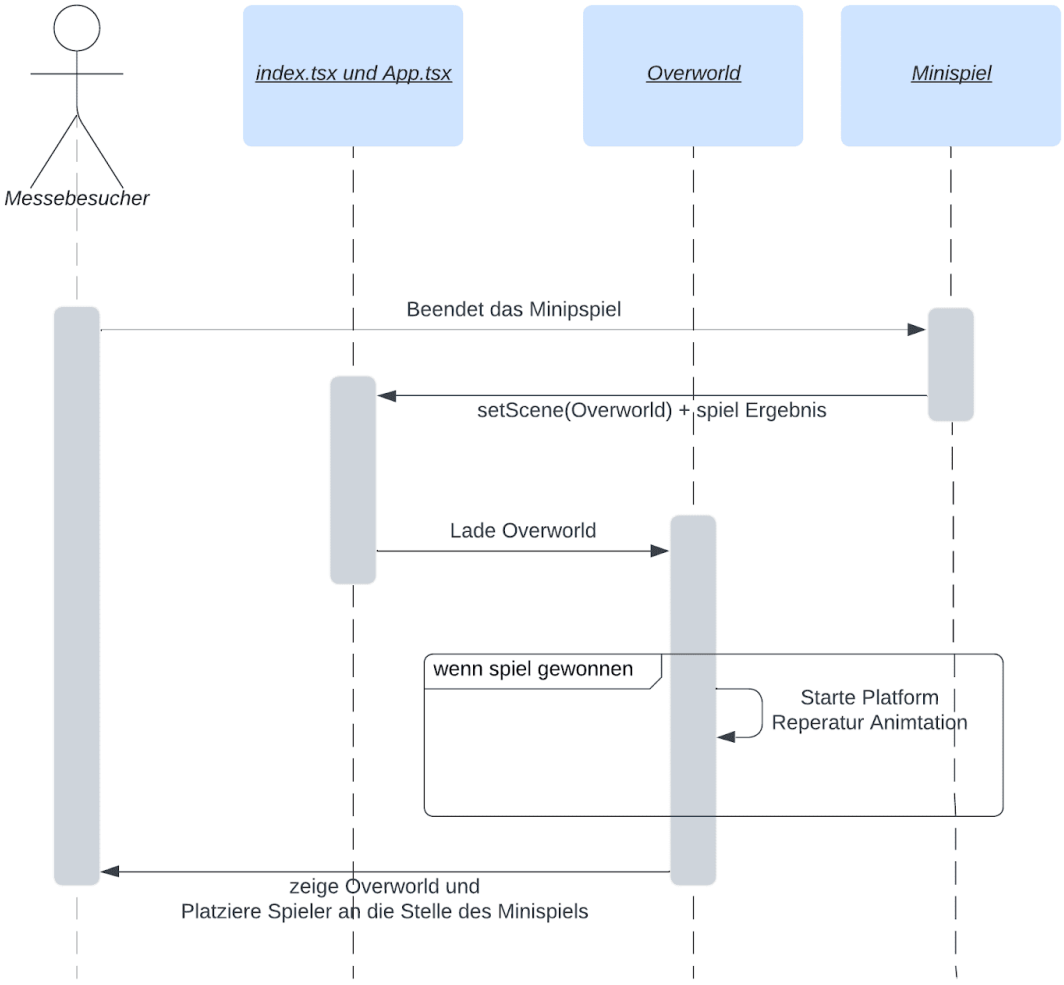


Abbildung 08: Laufzeitsicht Minispiel abschließen

In diesem Sequenzdiagramm beendet ein Messebesucher das Minispiel. Es wird hierbei die Szene gewechselt und die Overworld wird geladen. Falls der Messebesucher das Minispiel gewonnen bzw. absolviert hat, wird eine Reparatur Animation der entsprechenden Plattform abgespielt. Danach wird ihm die Overword mit dem Spieler angezeigt.

7 Verteilungssicht

Unsere Verteilungssicht besteht aus zwei Teilen. Die Frontend Applikation, welche beim Client im Browser läuft, sowie das Backend liegt in der BTP Cloud Foundry.

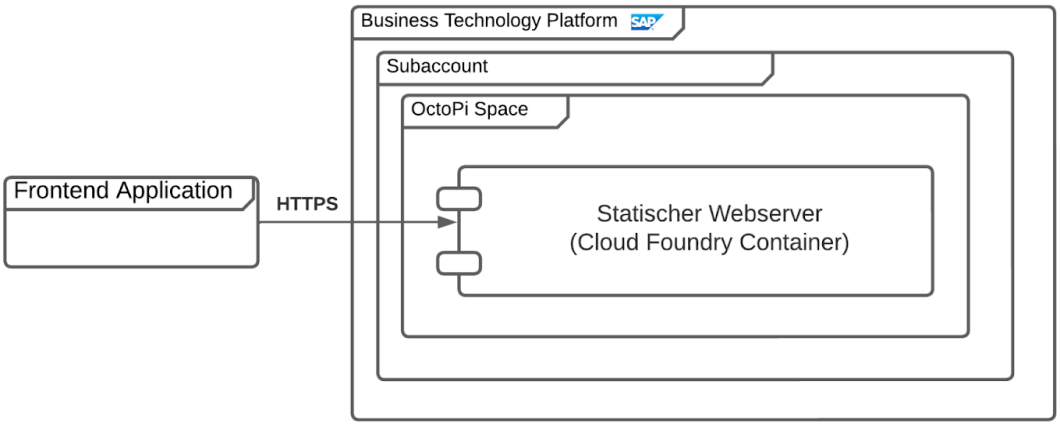


Abbildung 09: Verteilungssicht

Name	Beschreibung / Verantwortung
Frontend Application	Das Frontend (beschrieben in Kapitel 5.1) läuft auf dem Tablet im Webbrowser des Messestandes.
Business Technology Platform	Die SAP Cloud Platform ist eine Plattform-as-a-Service (PaaS), die Unternehmen ermöglicht, Anwendungen zu entwickeln, zu integrieren und bereitzustellen, indem sie Cloud-Services nutzt.
Subaccount	Von der sovanta bereitgestellte Server-Ressourcen, welche für die verschiedenen Teams des *SEPs* in Spaces aufgeteilt wurden. Alle durch die Cloud Foundry Container erzeugten Laufzeitkosten kommen hier zusammen.

Name	Beschreibung / Verantwortung
OctoPi Space	Sind dem Subaccount untergeordnet, in diesem können mehrere Cloud Foundry Container deployt werden. Der Space ist unserem Team zugeordnet.
Statischer Webserver (Cloud Foundry Container)	Der statische Webserver liefert vorgefertigte Dateien wie HTML, CSS und Bilder an Clients, ohne sie dynamisch zu generieren. In unserem Fall läuft dieser auf der BTP in einem Cloud Foundry Container. Im genauen handelt es sich dabei um einen Static Buildpack Container der SAP, welche im Hintergrund durch einen Nginx Webserver dargestellt wird.

8 Architekturentscheidungen

8.1 Three.js als 3D Framework

8.1.1 Fragestellung

Welches Framework verwenden wir, um 3D Objekte im Browser darzustellen?

8.1.2 Rahmenbedingung

Die Anwendung muss auf Tablets beziehungsweise Touch Geräten laufen und bedienbar sein. Es muss auf den jeweiligen Endgeräten für die Nutzer als flüssig und reibungslos wahrgenommen werden, was im Nutzertest überprüft werden soll.

8.1.3 Annahmen

Die Anwendung benötigt keine zusätzliche Grafik-Hardware. Sie läuft also auf mobilen Endgeräten (Tablets, Notebooks), ohne dass es zu Problemen wie beispielsweise Überhitzung der CPU führt.

8.1.4 Entscheidungskriterien

Das verwendete Framework bietet einfaches Importieren von 3D Objekten, die mithilfe von anderen Programmen (z. B. *Womp*, *Blender*) generiert werden, um die von den Designern erstellten Objekten einfach verwenden zu können (z. B. “.obj” Dateien).

8.1.5 Betrachtete Alternativen

Unity Game Engine hat einen größeren Einarbeitungsaufwand und benötigt eine kostenpflichtige Lizenz für die Verwendung im Unternehmenskontext. Gleichzeitig bietet es eine große Vielfalt an Möglichkeiten, einen Webexport und einfaches Umgehen mit extern generierten Objektdateien.

P5.js setzt mehr auf manuelles Erstellen von Komponenten voraus und bedeutet daher mehr Aufwand. Gleichzeitig bietet es die Möglichkeit, sehr frei die Gestaltung und den Look zu beeinflussen.

Babylon.js sieht visuell weniger ansprechend aus, bietet aber eine einfache Bearbeitung durch einen Web-Editor.

8.1.6 Entscheidungen

Wir haben uns für Three.js entschieden, da wir dort am wenigsten eingeschränkt sind (Keine Engine) und gleichzeitig eine direkte Integration in den Browser haben. Zusätzlich haben die Beispiele, die wir im Netz finden konnten, den Eindruck vermittelt, dass es möglich ist, damit eine grafisch schönere Anwendung, im Vergleich zu anderen 3D Libraries, bauen zu können.

8.2 Implementierung der Frontend Anwendung mithilfe von Three.js

8.2.1 Fragestellung

Wie wollen wir die 3D Webanwendung mithilfe von Three.js umsetzen?

8.2.2 Rahmenbedingung

SAP Business Technology Platform sollte verwendet werden und der Account auf der BTP Cloud Foundry wird für uns ein Semester bereitgestellt.

8.2.3 Annahmen

Das Framework unterstützt *JavaScript* oder *TypeScript* Entwicklung.

8.2.4 Entscheidungskriterien

Die Dokumentation von der gewählten Library und des Frameworks ist gut und ausführlich. Das Verstehen und Nutzen ist mit wenigen Tagen Einarbeitungszeit verbunden und es sind viele Beispiele vorhanden, die zur Implementierung beitragen können.

8.2.5 Betrachtete Alternativen

Three.js als Standard JavaScript Anwendung bietet eine gute Dokumentation, ist allerdings aufwändig in der Programmierung, führt zu mehr geschriebenem Code und ist weniger gut strukturiert.

Angular in Kombination mit Three.js ist mithilfe von Angular Three einfach integrierbar und kombiniert die Vorteile von Angular. Allerdings ist Angular noch keinem Teammitglied bekannt und der Funktionsumfang ist größer als bei React.

React ist weit verbreitet und ermöglicht einen übersichtlichen Aufbau von Webseiten. Es findet sich viel Dokumentation und es gibt eine gute Integration mit Three.js (React-Three/Fiber), was die Arbeit mit Three.js sehr vereinfacht.

8.2.6 Entscheidungen

Die Kombination von Three.js und React wird verwendet, da sie die Vorteile von Three nutzt und gleichzeitig eine einfachere Verwendung der Komponenten ermöglicht. Zusätzlich sorgt die Aufteilung in Komponenten von React für eine bessere Lesbarkeit des Programmcodes. Außerdem erhoffen wir uns dadurch die Applikation einfacher erweitern zu können.

8.3 Reine Frontend Application

8.3.1 Fragestellung

Brauchen wir ein Backend für unsere Applikation?

8.3.2 Rahmenbedingung

Die SAP BTP bietet die Möglichkeit, mehrere Instanzen von unterschiedlichen Cloud Foundry Containers laufen zu lassen, was ein Backend gut möglich macht.
Die Anwendung wird auf Tablets laufen. Da sich die Messestände nicht direkt neben dem WLAN-Router befinden, muss damit gerechnet werden, dass es zu langsameren Internet oder sogar zu einem Ausfall kommt. Zur Speicherung und Übertragung von Daten zwischen dem Frontend bzw. dem Messebesucher wird ein Backend benötigt.

8.3.3 Annahmen

Backend Deployment bedeutet mehr Aufwand und Einarbeitung in SAP BTP Cloud Foundry, um die Instanzen sinnvoll und sicher zu verknüpfen.

8.3.4 Entscheidungskriterien

Die Wichtigkeit von persistenten Daten und Datenübertragung zwischen den verschiedenen Clients ist in unserem Fall nicht von Bedeutung.

8.3.5 Betrachtete Alternativen

Das Backend mit SAP *CDS* in Verbindung mit einer Datenbank ist nur mithilfe von *SAP HANA* sinnvoll zu implementieren. Diese Möglichkeit ist nur durch Mehrkosten und zusätzlichen Aufwand durch eine manuelle Freischaltung der sovanta möglich.

Mithilfe unseres Backends wie *Express.js* und *PostgreSQL* kann es gut funktionieren und würde den Zweck erfüllen, Daten zu speichern und damit Datenfluss zwischen den Clients zu schaffen. Allerdings benötigt die Umsetzung eine Ausarbeitung von Backend Logik und eine Einarbeitung in die SAP BTP Datenbank Authentifikation.

8.3.6 Entscheidungen

Die Applikation wird ohne Backend umgesetzt, da es bei der Umsetzung unserer Idee nicht von Bedeutung ist, Informationen zwischen Clients zu teilen.

8.4 JavaScript oder TypeScript

8.4.1 Fragestellung

Wollen wir für das Projekt JavaScript oder TypeScript verwenden?

8.4.2 Rahmenbedingung

Wir bauen unser Produkt auf dem *Node.js* Runtime Environment auf. Zum Erstellen des Projektes benutzen wir ein Build Tool namens Vite. Mit diesem Tool lässt sich das Projekt mit einfachen Schritten in TypeScript umwandeln.

8.4.3 Annahmen

Die gewählten Frameworks sind mit beiden Sprachen gleichermaßen kompatibel.

8.4.4 Entscheidungskriterien

Wichtig sind dabei die Einfachheit der Programmierung, die Wartbarkeit und Qualität des zu entwickelnden Produktes und die Möglichkeit, jederzeit agile Änderungen am Projekt vornehmen zu können.

8.4.5 Betrachtete Alternativen

JavaScript ist eine dynamisch typisierte Skriptsprache, die eine schnelle Implementierung ohne Angaben von genauen Typen ermöglicht und daher weniger *Overhead* erzeugt. Gleichzeitig führt JavaScript zu einer erschwerten Fehlersuche, da man viele Freiheiten in Formatierungstypen, Interpretation, etc. hat.

TypeScript ist eine stark typisierte Skriptsprache, welche die Leserlichkeit verbessert. Es werden Fehler direkt angezeigt und es gibt somit weniger Bugs, da diese frühzeitig entdeckt werden können. Allgemein führt TypeScript zu weniger Laufzeitfehlern.

8.4.6 Entscheidungen

Es wird TypeScript verwendet, da es für einen besseren Durchblick sorgt, indem beispielsweise alle Variablen typisiert sein müssen und es allgemein zur besseren Einhaltung der Clean Code Regeln führt.

8.5 Wie sollen 3D Objekte geladen werden?

8.5.1 Fragestellung

Wie sollen 3D Objekte geladen werden, um die gegebenen Anforderungen am besten umzusetzen?

8.5.2 Rahmenbedingung

Die 3D Objekte werden in Womp erstellt und als .obj Dateien exportiert. Die Möglichkeit, sie zu konvertieren und zu komprimieren, ist gegeben.

Die Objekte müssen vom Backend ins Frontend geladen und im Canvas gerendert werden.

8.5.3 Annahmen

Three.js unterstützt alle betrachteten Alternativen mit vergleichbarem Aufwand.

8.5.4 Entscheidungskriterien

Am wichtigsten ist die Ladegeschwindigkeit bei möglichst wenig Einschränkung der Qualität.

Zudem muss der benötigte Aufwand betrachtet werden, um von .obj Dateien zur finalen Lösung zu kommen.

8.5.5 Betrachtete Alternativen

Die Verwendung von .obj-Dateien über den OBJ-Loader von Three.js ist sehr einfach, da kein zusätzlicher Schritt erforderlich ist, damit es funktioniert. Allerdings ist dies der langsamste Weg, um die 3D-Objekte zu laden. Um die Ladezeit von .obj Files zu verbessern, können die für Texturen, Farben etc. enthaltenen PNG Dateien komprimiert werden.

Das Konvertieren der .obj-Dateien in .gltf-Dateien und die Verwendung des GLTFLoader von Three.js fügt einen zusätzlichen Konvertierungsschritt hinzu, ermöglicht jedoch ein schnelleres Laden der Dateien, während das Dateiformat immer noch für uns lesbar ist.

Das Konvertieren der .obj-Dateien in .glb-Dateien und die Verwendung des GLTFLoader von Three.js fügt einen zusätzlichen Konvertierungsschritt hinzu, ermöglicht jedoch ein viel schnelleres Laden der Dateien.

Das Konvertieren der .obj-Dateien in JavaScript über den Three.js Editor ermöglicht uns schnelle Ladezeiten, erfordert jedoch viel manuellen Konvertierungsaufwand für jedes Objekt.

Die Dracon-Kompression von glb-Dateien kann die Netzwerklast weiter verringern, erhöht jedoch die Dekompressionszeit im Frontend.

8.5.6 Entscheidungen

Die Pipeline besteht aus zwei Schritten. Sie beginnt mit der Komprimierung der PNGs der Texturdateien der .obj-Datei. Anschließend konvertieren wir die resultierenden Dateien in eine .glb Datei.

9 Abbildungsverzeichnis

Alle Abbildungen sind Eigenkreationen.

10 Glossar

Begriff	Definition
Angular	Frontend Framework, basierend auf TypeScript.
Babylon.js	Babylon.js ist eine 3D Engine.
Backend	Ein Server, der für die Maschineninteraktion gedacht ist. Es findet keine direkte Interaktion vom Benutzer zum Backend statt.
Blender	Ist ähnlich wie Womp, zum Erstellen von 3D Objekten
CDS	SAP Core Data Services https://cap.cloud.sap/docs/cds/
Clockify	Kostenlose App zum Zeiterfassen. https://clockify.me/
Cloud Foundry	Cloud Foundry ist eine Cloud-basierte CD Platform.
CSS	Cascading Stylesheet
Discord	Eine Plattform, die zur Kommunikation genutzt wird. https://discord.com/
DOD	Definition of Done

Epic Sum Up	Epic Sum Up ist ein Add-On für *Jira*, mit der man eingetragene Zeiten einfach summieren kann.
Express.js	Backend Web Server Framework basierend auf Node.js. https://expressjs.com/
FA	Funktionale Anforderungen
Frontend	Teil der Software, welcher mit dem Nutzer interagiert und auf dem Rechner des Nutzers läuft.
Github	Softwareversionierungsservice, welcher Git benutzt und weitere Zusatzfunktionen bietet. https://github.com/
Google Docs	Online-Service, welcher Mehrbenutzerbetrieb für Word-Dokumente, Sheets und Slides anbietet. https://docs.google.com
HTML	Strukturbasierte Programmiersprache basierend auf XML, welche für Webseiten benutzt wird.
HTTPS	Hypertext Transfer Protocol Secure
Human Centered Design	Reale Personen sind in der Mitte der Entwicklung.
Innovation Factory	Die Innovation Factory besteht aus sechs Bereichen und wird mit den vorhandenen Features der BTP realisiert. Die Umgebungen lauten Design, Engineering, Produktion, Parts, Shipment und Monitoring.

	(https://sovanta.com/innovation-factory-for-sap-btp/)
JavaScript	Eine Programmiersprache, die für den Browser entwickelt wurde und mittlerweile auch mithilfe von Node.js im Backend laufen kann.
Jira	Online Projektmanagement Tool, welches primär für Softwareentwicklung eingesetzt wird. https://www.atlassian.com/software/jira
NFA	Nichtfunktionale Anforderungen
Node.js	Eine plattformübergreifende JavaScript Laufzeitumgebung. https://nodejs.org
Overhead	Zusätzliche Arbeit, Probleme, Rechenzeit etc. sind zusammengefasst.
p5.js	Eine JavaScript Library für Datenvisualisierung.
Postgresql	Postgresql ist eine relationale Datenbank.
R	Randbedingung
React-Three-Fiber	Ein React-Renderer für Three.js.
React.js	Frontend Framework basierend auf Node.js JavaScript. https://react.dev/
SAP BTP	SAP Business Technology Platform: ,,Eine Technologieplattform, die Daten und Analysen, künstliche Intelligenz, Anwendungsentwicklung,

	Automatisierung und Integration in einer einheitlichen Umgebung vereint.“ (https://www.sap.com/germany/products/technology-platform/what-is-sap-business-technology-platform.html)
SAP HANA	SAP HANA ist eine Cloud-basierte Datenbank, entwickelt von SAP.
SEP	Softwareentwicklungsprojekt
Three.js	JavaScript Frontend Library für 3D Rendering. https://threejs.org/
TypeScript	Eine Erweiterung der Programmiersprache JavaScript. https://www.typescriptlang.org/
UX	User Experience - Die Erfahrung, die ein Nutzer mit einem Produkt macht.
VS Code	Visual Studio Code ist ein kostenloser Quelltext-Editor. https://code.visualstudio.com/
Web-API	Application Programming Interface, welches Interaktionen über HTTP-Requests bietet.
WhatsApp	Internet Nachrichten App für Mobilgeräte. https://www.whatsapp.com/
Womp	Programm, um 3D Objekte zu erzeugen.