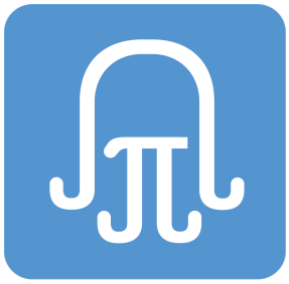
Architekturdokument

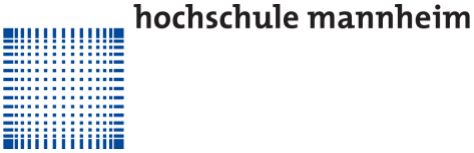


Auftragnehmer: OctoPi

Verantwortlich: Philip Dell, Timo Wenz

Im Auftrag von

sovanta AG (Heidelberg)



Softwareentwicklungsprojekt an der Hochschule Mannheim

Sommersemester 2023

Version 1.0

10.05.2023

Inhaltsverzeichnis

1 Versionsverzeichnis

Version	Änderung	Datum	Autor(en)
1.0	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	26.04.2023	Julian Wernz

	Einleitung geschrieben		
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 Dokument beschreibt die wesentlichen Anforderungen und treibenden Kräfte im Rahmen des SEPs. Diese müssen zur Umsetzung der Softwarearchitektur und Entwicklung der Applikation berücksichtigt werden. Im Laufe des Sommersemester 2023 wird für diese eine grundlegende Version erstellt, welche Erweiterungsmöglichkeiten bietet. Das Team OctoPi arbeitet in 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 für Technikmessen für sovanta zu entwickeln, welche den Besuchern dieser Messen die “sovanta Innovation Factory for SAP BTP” spielerisch vorstellen und die einzelnen Features der BTP dadurch näher bringen soll. 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.

2.3 Qualitätsziele

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. Das bedeutet, dass unsere Applikation schnell und effizient wieder nach einer Wartung oder Reparatur einsatzbereit ist.

2.4 Stakeholder

Siehe Projekthandbuch (v1.0) | Kapitel 5 Stakeholder (OctoPi 2023 : 8)

3 Randbedingungen

Siehe Anforderungsspezifikation (v1.0) | Kapitel 7.2.3 Randbedingungen (OctoPi 2023 : 31)

4 Kontextabgrenzung

Die Kontextabgrenzung bezeichnet die Applikation zwischen fachlichem und technischem Bereich. Diese ermöglicht eine effektive Planung und Entwicklung unter Berücksichtigung der Anforderungen und Erwartungen der Stakeholder.

4.1 Fachlicher Kontext

Die Nutzer brauchen kein Vorwissen, um die Applikation bedienen zu können. Sie soll die Anwender darauf vorbereiten, mit den Mitarbeitern von sovanta zu reden, indem sie die Nutzer mit der sovanta Innovation Factory vertraut machen.

4.2 Technischer Kontext

Zur Umsetzung werden Tablets benutzt. Diese bieten eine einfache, aber interaktive Möglichkeit, da die Bedienung den Nutzern bereits bekannt ist. Die Touchfunktion bietet die Möglichkeit zum einen den Spieler zu bewegen und zum anderen die Spiele zu bedienen. Da die Applikation als Spiel gestaltet ist, wird die Bedienung durch Tippen und Swipen realisiert. Das Mapping der Bewegung zu den Touch Inputs und die Reaktion auf diese soll flüssig erfolgen und intuitiv gestaltet sein. Dadurch können wir sicherstellen, dass die Vielzahl an Personen auf der Messe gut mit der Bedienung klarkommen und somit zufrieden sind

5 Lösungsstrategie

Wie schon im Abschnitt 2.2 erwähnt, handelt es sich um eine auf unterschiedlich großen Messen verwendete Applikation. Deshalb bietet es sich an, sie auf mobilen Endgeräten laufen zu lassen, um so gewährleisten zu können, dass sie bei so vielen Messen wie möglich zum Einsatz kommen kann (siehe auch Abschnitte 10.1 und 3).

Die Implementierung der Frontend Anwendung wird mit Hilfe von Three.js realisiert, da hiermit das Qualitätsziel “Wartbarkeit” gewährleistet und die Anforderung “Erweiterbarkeit” erfüllt werden kann (siehe auch Abschnitte 2.3 und 10.2.6). Die eben genannte Anforderung führte auch zur Entscheidung, das Framework TypeScript zu nehmen und nicht JavaScript (siehe auch Abschnitte 10.4 bzw 10.4.6). Zudem ermöglicht das gewählte Framework die Laufzeitfehler zu verringern und die Anforderung “Ausfallsicherheit” besser garantieren zu können (siehe auch Kapitel 10.4 bzw. 10.4.5 und Kapitel 7.2.2.2 Anforderungsspezifikation (v1.0) OctoPi 2023: 23).

6 Bausteinsicht

Die Frameworks React.js und Three.js benutzen wir als Frontend Libraries. Im Backend wird grundsätzlich nur ein statischer Webserver verwendet, der in diesem Fall auf der SAP Business Technology Platform deployed ist.

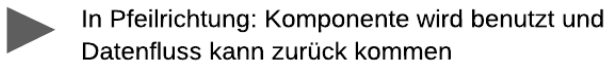


Abbildung 01 : Diagrammschlüssel

6.1 Whitebox Gesamtsystem

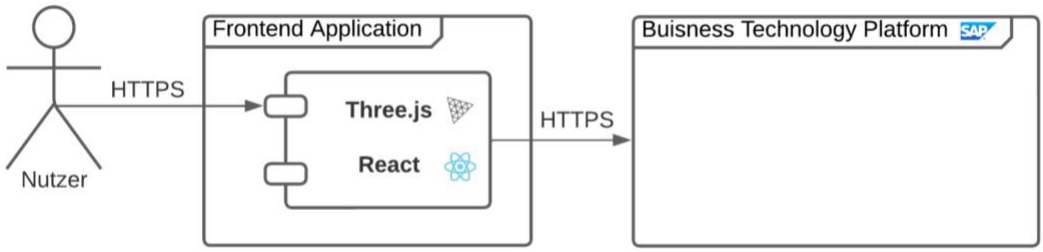


Abbildung 02 : Level 1 Bausteinsicht

Begründung

Der Nutzer interagiert mit der Frontend Applikation, welche via HTTPS aus dem Backend (Business Technology Plattform) geladen wird.

Name	Beschreibung / Verantwortung
Frontend Applikation	Ermöglicht die Interaktion zwischen dem Nutzer und der Applikation.
Business Technology Platform	Ist das Backend unserer Applikation. Hierüber können Cloud Komponenten der SAP laufen. Die BTP liefert alle im Frontend benötigten Daten.
HTTPS	Ist die Schnittstelle, mit der der Nutzer auf das Frontend zugreift und die Daten aus dem Backend bekommt.

6.2 Ebene 2

In Ebene zwei werden die beiden Blöcke aus der Whitebox-Gesamtsystem-Sicht verfeinert. Dafür beschäftigen sich die beiden nachfolgenden Diagramme mit der zweiten Ebene des Backends und des Frontends.

6.2.1 Business Technology Platform

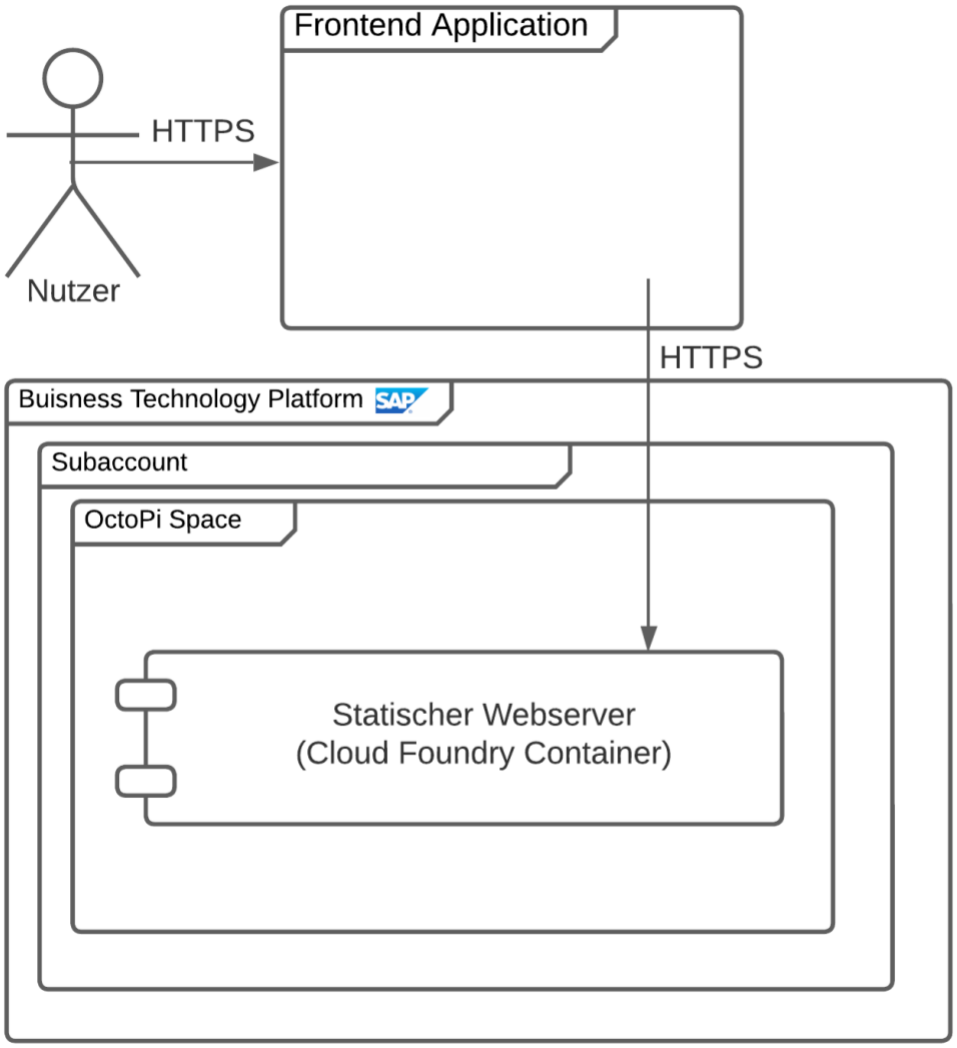


Abbildung 03 : Bausteinsicht Level 2 Backend Whitebox

Begründung

Die Applikation wird über die SAP BTP ‘deployed’. Dafür haben wir einen Subaccount von sovanta zur Verfügung gestellt bekommen, welcher den Team-Space beinhaltet. In diesem

kann man statische Cloud Foundry Container anlegen, auf die die Nutzer per HTTPS zugreifen können.

Name	Beschreibung / Verantwortung
Subaccount	Von der sovanta bereitgestellte Ressourcen, welche für die verschiedenen Teams aufgeteilt und auf welche dann die Container laufen werden. Alle erzeugten Kosten kommen hier zusammen und werden von sovanta eingesehen.
OctoPi Space	Sind dem Subaccount untergeordnet, in diesem können mehrere Container deployed werden. Der Space ist unserem Team zugeordnet.
Statischer Webserver (Cloud Foundry Container)	Ist ein Cloud-Container, in dem ein statischer Webserver läuft. Hierauf wird zugegriffen, wenn Daten aus dem Backend in das Frontend geholt werden.

6.2.2 Frontend Applikation

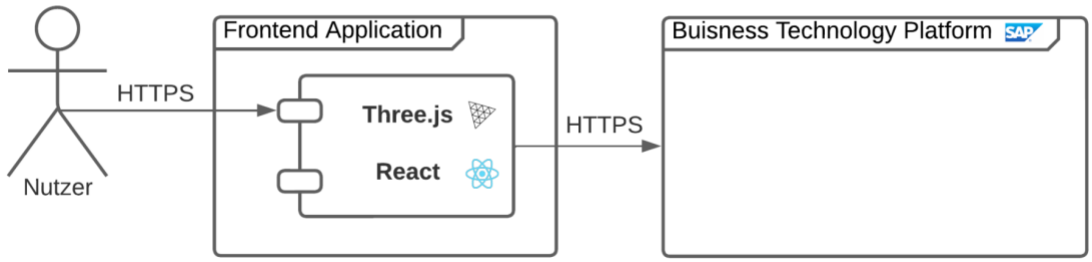


Abbildung 04 : Bausteinsicht Level 2 Frontend Whitebox

Begründung

Das Frontend besteht aus React Komponenten, welche über HTTPS vom Backend kommen. Diese Komponenten nutzen und kreieren Elemente, die mithilfe des HTML-Renderers des

Browsers dem Nutzer angezeigt werden. Zusätzlich kommt die Library Three.js in Verbindung mit React-Three/Fiber zum Einsatz, um React Komponenten auf den Three.js Canvas zu rendern, die dann durch den Browser dem Nutzer angezeigt werden.

Name	Beschreibung/Verantwortung
Three.js und React	Frontend Libraries, mit denen die Applikation 3D Grafiken rendert und Allgemein UI Elemente darstellt. Sie werden aus dem Backend über HTTPS geladen und mit ihnen kann der Nutzer interagieren.

7 Laufzeitsicht

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

7.1 Webseite aufrufen

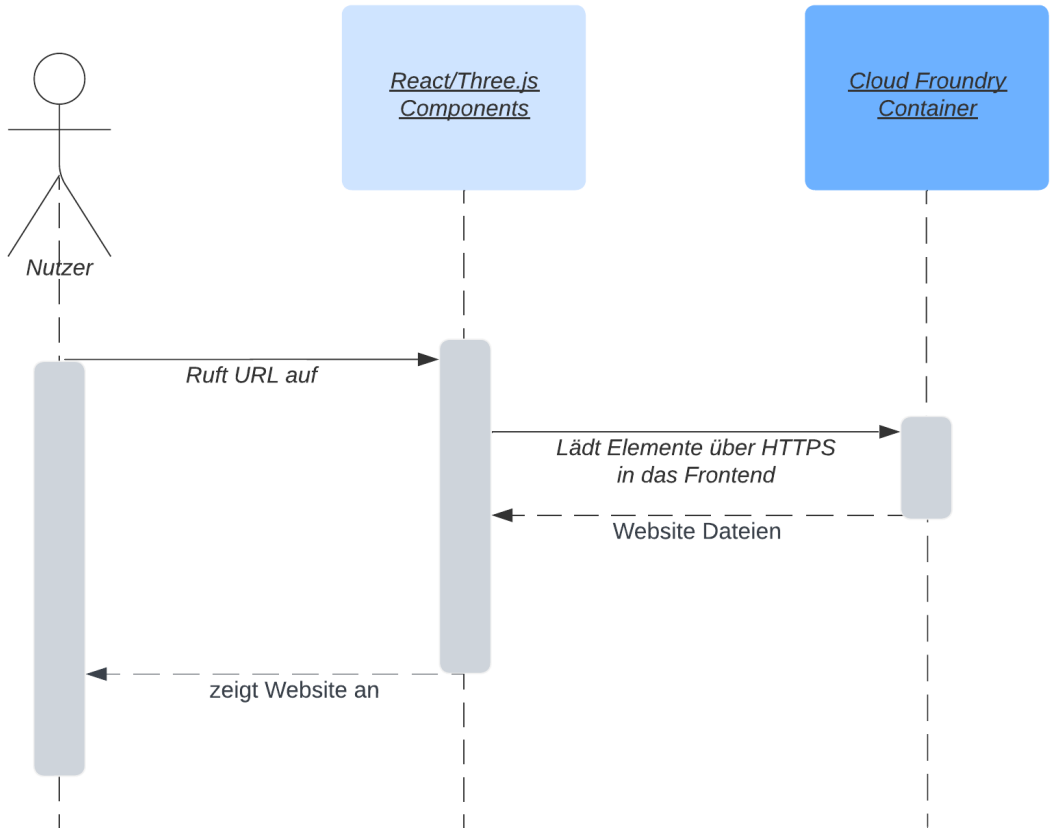


Abbildung 05 : Laufzeitsicht Elemente aus Cloud Foundry laden

Das dargestellte Sequenzdiagramm zeigt einen Nutzer, der die angegebene URL aufruft. Darauf lädt die React-und Three.js-Komponenten/Webseite Daten aus dem Backend, welche dann dafür sorgen, dass die Webseite dem Nutzer angezeigt wird.

8 Verteilungssicht

Unsere Verteilungssicht besteht aus zwei Teilen. Die Frontend Applikation läuft beim Client im Browser und das Backend liegt in der BTP Cloud Foundry. Da unsere Applikation in Bezug auf die Verteilung wenig komplex ist, haben wir uns dagegen entschieden, ein Diagramm zur Verteilungssicht darzustellen.

9 Querschnittliche Konzepte

9.1 Fachliche Konzepte

Auf Messen gibt es viele verschiedene Stände, dabei ist es nicht so leicht, aus der Masse herauszustechen und die Messebesucher auf den eigenen Stand aufmerksam zu machen. Das genannte Problem soll diese Applikation ändern, indem die Messebesucher direkt interessiert zu dem Stand gelockt werden. Nach der Benutzung der Applikation besteht die Möglichkeit, in ein Gespräch mit den sovanta Mitarbeitern zu kommen.

9.2 User Experience

Der Nutzer kommt an den Stand und möchte mit der Applikation interagieren. Hierzu benutzt er eines der zur Verfügung gestellten Geräte. Diese sollten dann unabhängig davon, wie lange er die Applikation benutzt, eine gute Erfahrung erzeugen. Im Genauen bedeutet das, dass die Oberfläche sehr einfach gestaltet und somit schnell verständlich ist. Dies wird einerseits auf der Hardware Ebene durch die Touchfunktion gewährleistet. Die Applikation soll andererseits auch einfach zu verstehen sein.

9.3 Architektur und Entwurfsmuster

Für die Entwicklung sollten alle Teile möglichst gut modularisiert werden, um sie wiederverwendbar zu machen. Insbesondere gilt das für die React Komponenten, welche vielfach vorkommen können.

9.4 Entwicklungskonzepte

Siehe Projekthandbuch(v 1.0) | Kapitel 6.1.2 Vorgehensmodell (OctoPi : 10)

9.5 Betriebskonzepte

Die Applikation soll einen kompletten Messetag stabil laufen und falls sie abstürzt, sollte man sie schnell wieder starten können, ohne große Verzögerungen. Als Versionierungssystem wird Git (GitHub) genutzt und über Cloud Foundry per CI / CD (Continuous Integration & Continuous Delivery) Pipeline deployed.

10 Architekturentscheidungen

10.1 Three.js als 3D Framework

10.1.1 Fragestellung

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

10.1.2 Rahmenbedingung

Die Anwendung muss auf Tablets beziehungsweise Touch Geräten laufen und bedienbar sein. Es sollte auf den jeweiligen Endgeräten flüssig und reibungslos laufen.

10.1.3 Annahmen

Die Anwendung benötigt keine besonderen Grafikleistungen und ist auf mobilen Geräten (Tablets, Notebooks) lauffähig, ohne dass es zu Problemen wie beispielsweise der Überhitzung der CPU führt.

10.1.4 Entscheidungskriterien

Das verwendete Framework bietet einfaches Importieren von Objekten, die mit anderen Programmen generiert werden (z.B. “.obj” Dateien).

10.1.5 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 Web Export und einfaches Umgehen mit extern generierten Objektdateien.

P5.js setzt mehr manuelles Kreieren 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.

10.1.6 Entscheidungen

Wir haben uns für Three.js entschieden, da wir dort die meiste Freiheit mit dem wenigsten Overhead haben und die Beispiele, die wir im Netz finden konnten, uns den Eindruck vermitteln, dass wir damit eine grafisch schönere Anwendung bauen können.

10.2 Implementierung der Frontend Anwendung mithilfe von Three.js

10.2.1 Fragestellung

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

10.2.2 Rahmenbedingung

SAP BTP soll verwendet werden und es ist nur ein Semester Zeit.

10.2.3 Annahmen

Das Framework unterstützt entweder JavaScript oder TypeScript Entwicklung.

10.2.4 Entscheidungskriterien

Die Dokumentation vom gewählten Tool 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.

10.2.5 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.

10.2.6 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 bietet die Aufteilung in Komponenten von React eine Möglichkeit, den Code aufgeräumter zu halten. Außerdem erhoffen wir uns dadurch die Applikation einfacher erweitern zu können.

10.3 Reine Frontend Application

10.3.1 Fragestellung

Brauchen wir ein Backend für unsere Applikation?

10.3.2 Rahmenbedingung

Die SAP BTP bietet die Möglichkeit, mehrere Instanzen laufen zu lassen, was ein beliebiges Backend gut möglich macht.
Die Anwendung wird auf Tablets verwendet, die potenziell nicht die beste Internet Bandbreite haben.
Zur Speicherung und Übertragung von Daten zwischen mehreren Clients wird ein Backend benötigt.

10.3.3 Annahmen

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

10.3.4 Entscheidungskriterien

Die Wichtigkeit von persistenten Daten und Datenübertragung zwischen den verschiedenen Clients ist bei uns von Bedeutung.

10.3.5 Betrachtete Alternativen

Das Backend mit SAP CDS in Verbindung mit Datenbank ist nur mithilfe von SAP HANA sinnvoll zu implementieren, was extra kostet und eine manuelle Freischaltung durch sovanta bedeutet.

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.

10.3.6 Entscheidungen

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

10.4 JavaScript oder TypeScript

10.4.1 Fragestellung

Wollen wir für das Projekt JavaScript oder TypeScript verwenden?

10.4.2 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.

10.4.3 Annahmen

Die gewählten Frameworks sind mit beiden Sprachen gleichermaßen kompatibel.

10.4.4 Entscheidungskriterien

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

10.4.5 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, die diese frühzeitig entdeckt werden können. Allgemein führt TypeScript zu weniger Laufzeitfehlern.

10.4.6 Entscheidungen

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

11 Qualitätsanforderungen

In diesem Kapitel werden die Qualitätsanforderungen mit jeweils einem speziellen Szenario beschrieben. Die wichtigsten dieser Anforderungen wurden bereits im Kapitel 2.3 (Qualitätsziele) hervorgehoben.

Die unterhalb verwendeten Qualitätsanforderungen findet man detailliert in der Anforderungsspezifikation (v1.0) unter dem Kapitel 7.2.2 Nichtfunktionale Anforderungen

(OctoPi 2023 : 22-31).

11.1 Qualitätsbaum

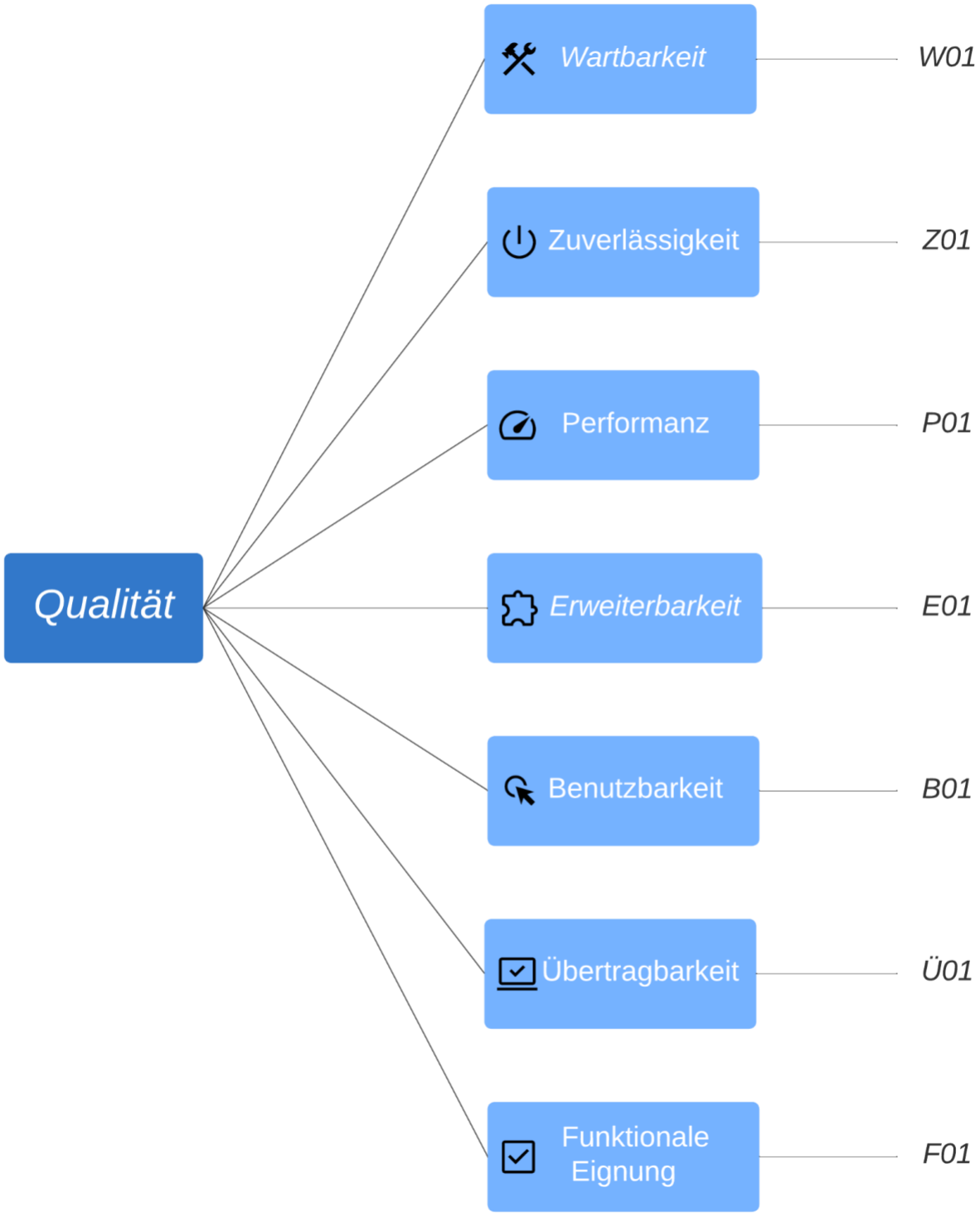


Abbildung 06 : Qualitätsanforderungen Qualitätsbaum

11.2 Qualitätsszenarien

ID	Beschreibung
W01	Die Applikation muss präzise Angaben über den aufgetretenen Fehler ausgeben.
Z01	Ein Messebesucher kommt auf eine Messe und will mit der Applikation interagieren. Dies kann er, da die Anwendung zuverlässig ist und fehlerfrei über einen Messetag läuft.
P01	Ein Messebesucher bekommt eine gute Experience mit der Applikation, das Bild sollte flüssig sein und Lade-Screens nur kurz angezeigt werden (< fünf Sekunden).
E01	Die Applikation ist modular aufgebaut und gut dokumentiert, sodass ein Mitarbeiter die Implementierung von weiteren Minispielen, ohne große Einarbeitung, bewerkstelligen kann.
B01	Ein Entwickler kommt zu dem Stand und interagiert mit der Applikation. Er weiß genau, was er machen muss, um eine Aktion zu erreichen, da alle Funktionen so sind, wie er sie bereits kennt.
Ü01	Die Firmenmitarbeiter bauen ihren Messestand auf und wollen die Applikation auf den Tablets laufen lassen, das funktioniert, da diese schnell übertragen werden kann.
F01	Während des Spielens, bleibt ein Spieler nicht an einer Stelle stecken, bei der ein weiterer Fortschritt unmöglich ist.

12 Risiken und technische Schulden

In diesem Kapitel werden die möglichen Risiken und technischen Schulden, welche im Laufe des Projekts auftreten können, aufgelistet. Zu jedem Risiko/Schuld gibt es jeweils präventive sowie reaktive Maßnahmen.

Nr	Benennung	Maßnahmen
1	Browser Probleme	**Präventiv:** Applikation auf vielen möglichen Browsern testen. **Reaktiv:** Cache leeren, Update, Neustart, Cookies löschen…
2	Internetausfall	**Präventiv:** Lokale Variante der Applikation auf dem Gerät speichern. **Reaktiv:** Gegebenenfalls über Mobilfunk verbinden.
3	Hardware unterstützt nicht WebGL.	**Präventiv:** Applikation vor der Messe auf der Hardware testen. **Reaktiv:** Anderes Gerät verwenden.
4	Die verwendete Library React-Three.js führt zu komplizierten Workarounds, da React nicht für Spieleentwicklung gemacht ist.	**Präventiv:** Research zu verschiedenen Möglichkeiten. **Reaktiv:** Features weglassen oder einfachere Alternativen in der Implementierung suchen.

13 Abbildungsverzeichnis

Alle Abbildungen sind Eigenkreationen.

14 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.
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
HTML5	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/