# Architekturdokumentation

## Dumb ways to innovate[1]

Eine spielerische Applikation zum Vorstellen der sovanta Innovation Factory for SAP BTP

Auftragnehmer: Sepia

Auftraggeber: sovanta AG


Softwareentwicklungsprojekt im Sommersemester 2023
an der Hochschule Mannheim


Version 1.0 vom 16.05.2023
Verantwortlich: Mike Menzel, Eren Saglam

[1] Arbeitstitel; Name wird noch geändert

Das Dokument wurde erstellt mithilfe des arc42-Templates.

Januar 2023

**Über arc42**
arc42, das Template zur Dokumentation von Software- und Systemarchitekturen.
Template Version 8.2 DE. (basiert auf AsciiDoc Version), Januar 2023
Created, maintained and © by Dr. Peter Hruschka, Dr. Gernot Starke and contributors.
Siehe https://arc42.org

# Änderungsverzeichnis

Das Architekturdokument wird iterativ weiterentwickelt und pro Version kommen neue Änderungen hinzu, die hier festgehalten werden.

| Version | Datum | Änderung | Autor | Prüfer |
|---|---|---|---|---|
| 1.0 | 16.05.2023 | Review des Dokuments und Korrektur von Fehlern | Mike Menzel, Fabian Hoppe | Stephan Halder, Eren Saglam |
| 0.3 | 15.05.2023 | Referenzen zur Anforderungsspezifikation und dem Projekthandbuch wurden eingefügt.<br>Beschreibungen wurden präzisiert.<br>Formulierungen wurden angepasst. | Fabian Hoppe, Mike Menzel | |
| 0.2 | 12.05.2023 | Diagramme der Baustein-, Verteilungs- und Laufzeitsicht wurden eingefügt.<br>Architekturentscheidungen wurden dokumentiert. | Mike Menzel, Eren Saglam | |
| 0.1 | 09.05.2023 | Grundstruktur des Dokuments wurde erstellt. | Eren Saglam | |

# Inhaltsverzeichnis

# 1. Einführung und Ziele

## 1.1 Aufgabenstellung

Es sollen eine prototypische Applikation und ein überzeugendes Konzept erstellt werden, die Besucher von Technik- und IT-Messen von den Möglichkeiten der sovanta Innovation Factory for SAP BTP überzeugen. Dabei sollen die Besucher der Messen durch das Produkt aufmerksam auf den Stand der sovanta AG werden. Außerdem soll durch das Anwenden des Produktes das Interesse an der Arbeit der sovanta AG geweckt werden.

## 1.2 Qualitätsziele

Die Qualitätsziele werden in diesem Dokument im Abschnitt 10. Qualitätsanforderungen beschrieben.

## 1.3 Stakeholder

Hier befindet sich eine Übersicht über die wichtigsten Stakeholder des Projekts.
Weitere Informationen zu den Stakeholdern sind im Projekthandbuch (Version 1) im Abschnitt 1 “Einleitung" und Abschnitt 2 “Der Kunde” enthalten.

| **Stakeholder** | **Kontakt** |
|---|---|
| Sepia (Auftragnehmer) | sepia.mannheim.2023@gmail.com |
| sovanta AG (Auftraggeber) | semesterprojekt2023@sovanta.com |
| Professoren (Betreuer) | p.knauber@hs-mannheim.de<br>w.schramm@hs-mannheim.de |

# 2. Randbedingungen

Im Folgenden werden die wichtigsten Randbedingungen genannt, die die Architektur betreffen.
Alle weiteren Randbedingungen für das Projekt sind der Anforderungsspezifikation (Version 2)[2] im Abschnitt “4.4.4 Randbedingungen” zu entnehmen.

| Randbedingung | Beschreibung |
|---|---|
| Hardware muss mobil sein | Die sovanta AG baut ihre Werbematerialien selbst auf. Daher muss alles, was am Stand genutzt wird, transportabel und mobil sein. |
| Nutzung der SAP BTP | Die sovanta AG wirbt mit ihrer Arbeit mit der SAP BTP und gibt deshalb auch vor, dass Services der SAP BTP verwendet werden. |
| Instabile Internetverbindung | Die Internetverbindung kann auf Messen instabil sein, weshalb das Produkt damit umgehen können muss. |

# 3. Kontextabgrenzung

Die folgende Abbildung zeigt, dass das System die SAP BTP als Nachbarsystem besitzt und dieses benutzt. Die Begründung dafür findet man unter 9.4 Nutzung von Cloud Foundry für Deployment.

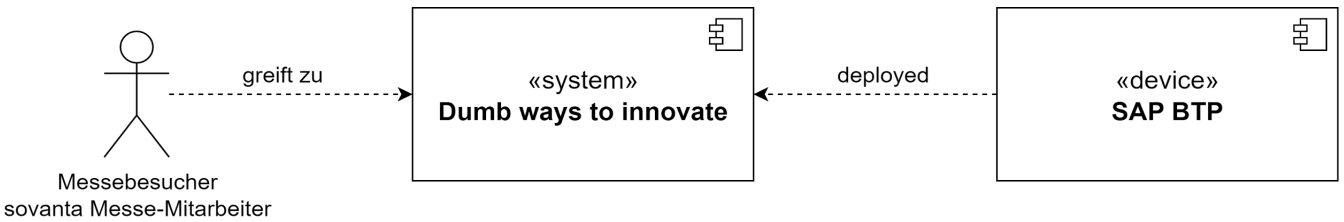



Abbildung 3.1 Kontextabgrenzung

Die Nutzer können auf das System durch die jeweiligen Frontends in einem Webbrowser zugreifen und die jeweiligen Funktionen nutzen.
Das System besitzt mit der SAP BTP ein Nachbarsystem. Die SAP BTP wird dazu genutzt, um das System zu deployen.

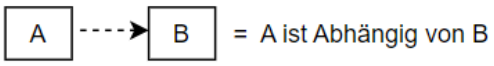
[2] Version 2 der Anforderungsspezifikation kann ab dem 24.05.2023 auf der Teamwebsite abgerufen werden.

# 4. Lösungsstrategie

Grundlegende Entscheidungen und Lösungsansätze, die Entwurf und Implementierung des Systems prägen, werden in 9. Architekturentscheidungen erläutert.

# 5. Bausteinsicht

Pfeil-Legende für die Komponentendiagramme in diesem Kapitel:




## 5.1 Ebene 1

Im Folgenden werden die Komponenten der ersten Ebene vorgestellt.
Die Komponente "Scoreboard Frontend" zeigt die Spieler mit ihren Scores an und ist damit so simpel, dass sie keine Verfeinerung benötigt.
Die Komponente "Database-Management-System" wird von PostgreSQL umgesetzt und daher nicht weiter verfeinert.

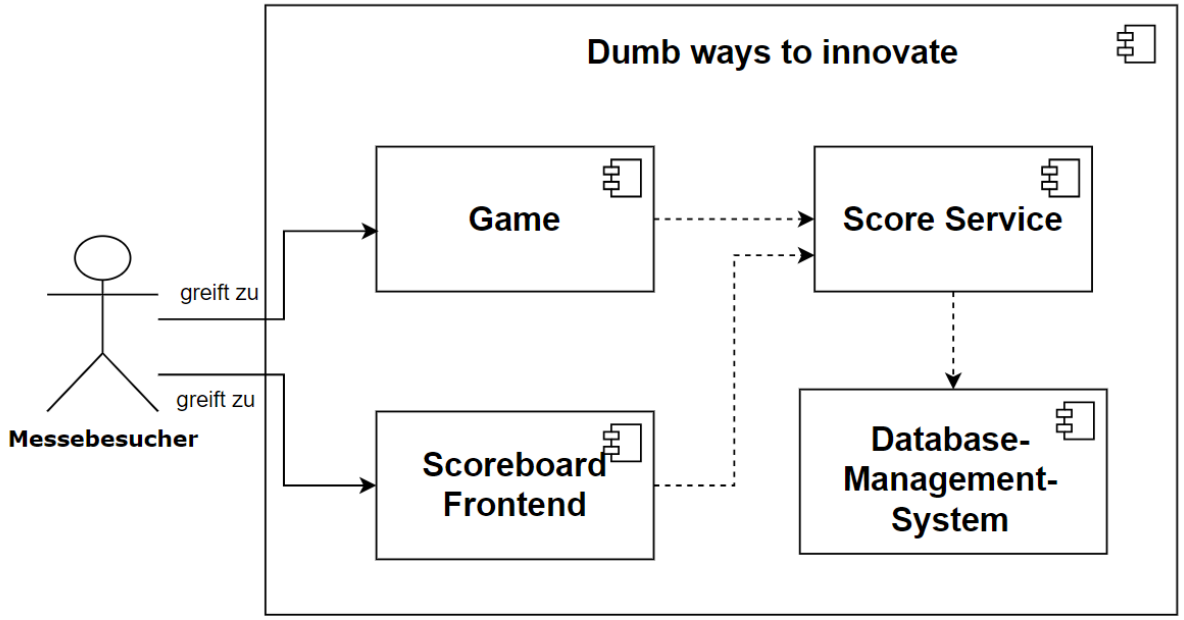



Abbildung 5.1.1 Bausteinsicht Ebene 1 Gesamtsystem

| Komponente | Beschreibung |
|---|---|
| Game | Eine Unity-Applikation, welche das Spiel darstellt. |
| Scoreboard Frontend | Eine React-Applikation, welche die Spieler mit ihren erreichten Scores aus dem Spiel anzeigt. |
| Score Service | Eine Spring-Boot-Applikation, welche den beiden Komponenten “Game” und “Scoreboard Frontend” eine REST-API bietet. Über die REST-API werden Daten über die Spieler mit ihren Scores ausgelesen oder neu angelegt. Zudem kommuniziert diese Komponente mit der Datenbank, um die Daten zu speichern und auszulesen. |
| Database-Management-System | Ein PostgreSQL Datenbankmanagementsystem, welches zur persistenten Speicherung der Spielerdaten dient. Die Daten werden für die Dauer der IT-Messe gespeichert. |

## 5.2 Ebene 2

Die Komponenten "Game" und "Score Service" aus der Ebene 1 werden im Folgenden verfeinert.

### 5.2.1 Ebene 2 Score Service

Die Komponente "Database Adapter" wird im Folgenden nicht weiter verfeinert, da diese von dem Spring Boot Framework umgesetzt wird. "Model" und "Score Controller" werden ebenfalls nicht weiter vereinfacht, da eine weitere Verfeinerung in ein Feindesign münden würde.

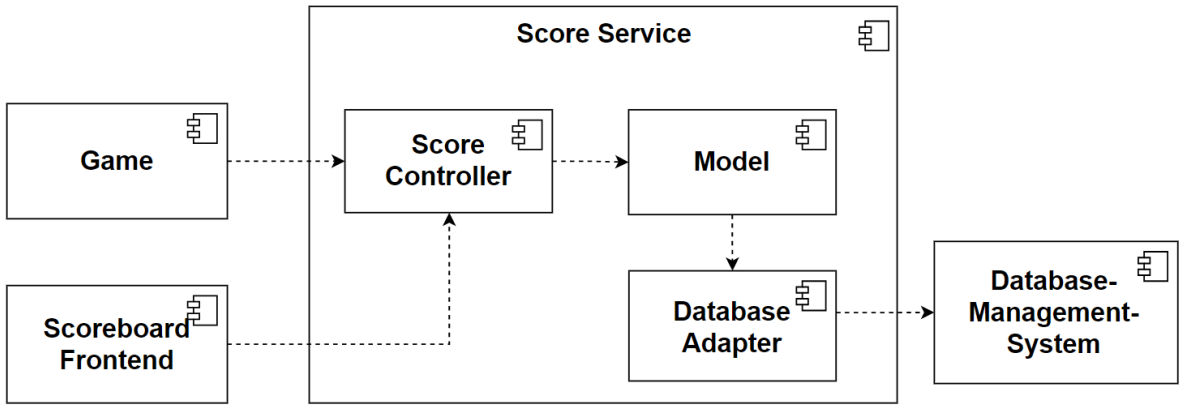



Abbildung 5.2.1.1 Bausteinsicht Ebene 2 Score Service

| Komponente | Beschreibung |
|---|---|
| Score Controller | Nimmt HTTP-Requests von den Komponenten “Game” und “Scoreboard Frontend” entgegen und verarbeitet diese entsprechend. Zudem finden hier Input Validierungen statt. |
| Model | Enthält die Geschäftslogik. |
| Database Adapter | Stellt eine Verbindung zu der Datenbank her. |

### 5.2.2 Ebene 2 Game

Die Komponente "Game Engine" wird im Folgenden nicht weiter verfeinert, da diese von Unity umgesetzt wird.
"Game Manager" wird ebenfalls nicht weiter vereinfacht, da eine weitere Verfeinerung in ein Feindesign münden würde.

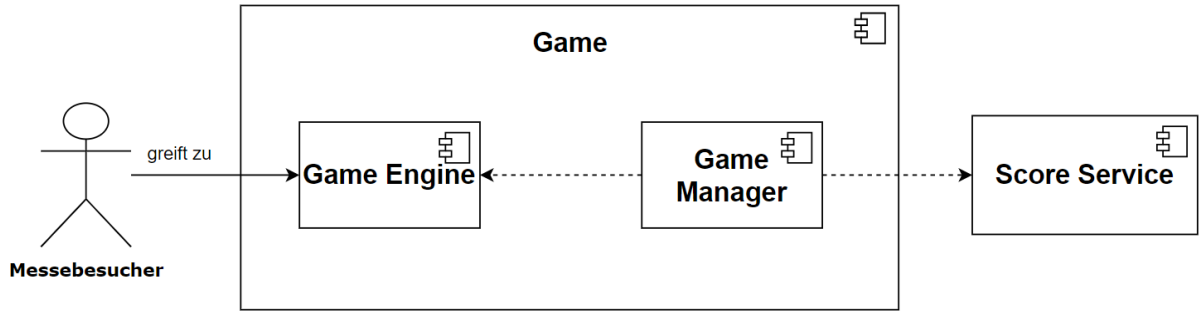



Abbildung 5.2.2.1 Bausteinsicht Ebene 2 Game

| Komponente | Beschreibung |
|---|---|
| Game Engine | Umfasst grundlegende Funktionalitäten für die Spieleentwicklung. Dazu gehören Input-Management, Game-Physics und Animation-Handling. Diese Funktionalitäten werden von Unity zur Verfügung gestellt. |
| Game Manager | Enthält die Spiellogik. Dazu gehören die Spiel-Initialisierung, die Steuerung des Spielablaufs und die Level-Auswahl. |

# 6. Laufzeitsicht

Im Folgenden werden die typischen und komplexesten Abläufe zwischen den in der Bausteinsicht definierten Komponenten dargestellt.

## 6.1 Ablauf “Spielerdaten speichern”

Die Abbildung 6.1 stellt den grundlegenden Ablauf dar, wie neue Spielerdaten (Spieler- und Unternehmensname sowie erreichter Score) nach dem Beenden des Spiels gespeichert werden.

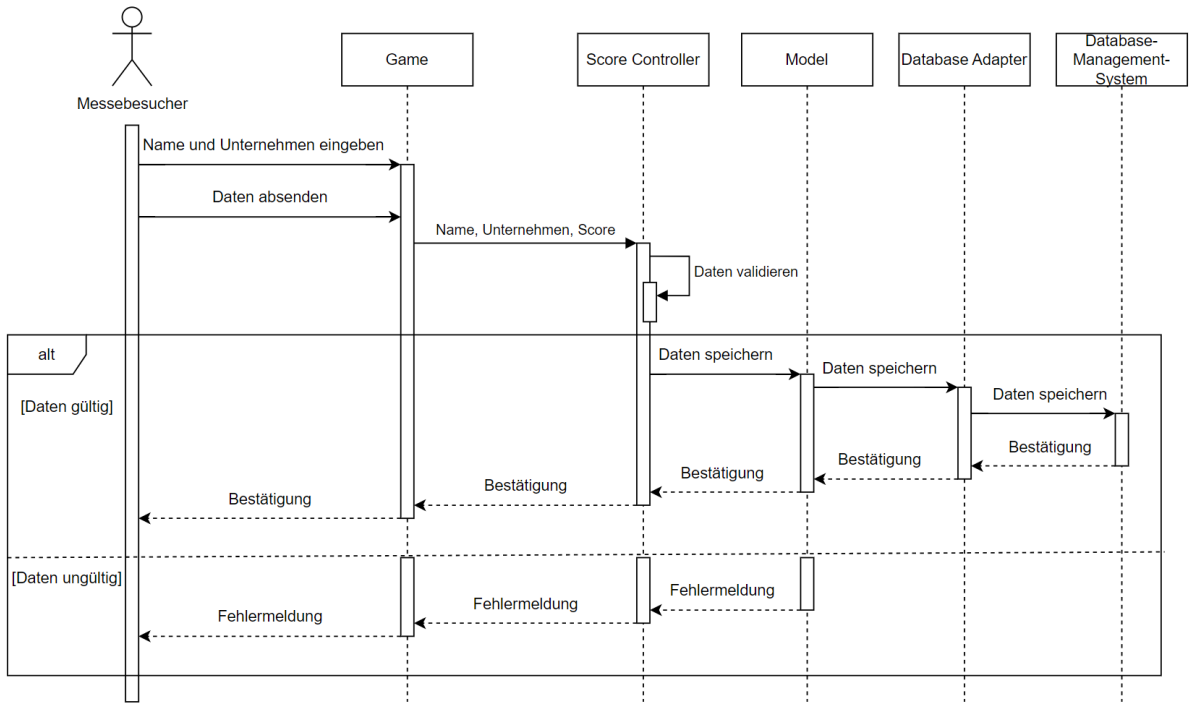
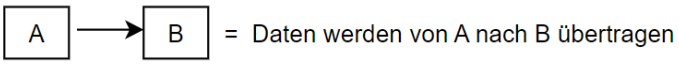



Abbildung 6.1.1 Sequenzdiagramm Spielerdaten Speicherung

# 7. Verteilungssicht

Pfeil-Legende für das Deployment-Diagramm in diesem Kapitel:



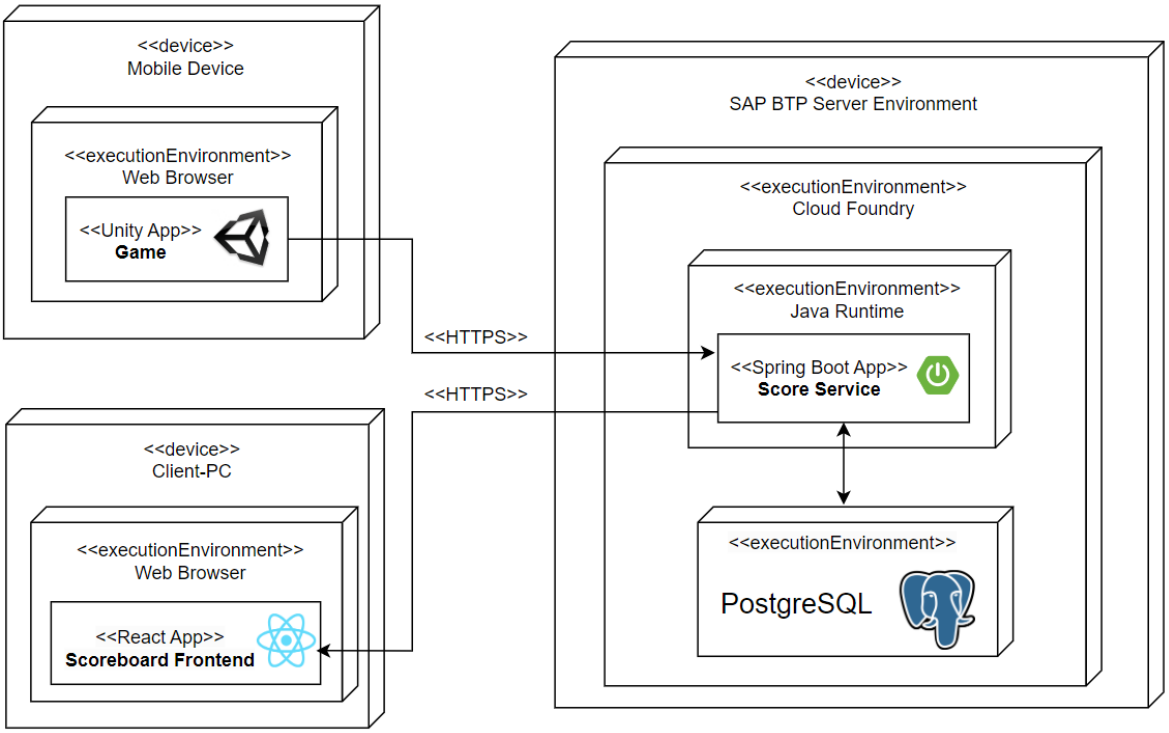




Abbildung 7.1.1 Verteilungssicht

**Begründung**

Da die Scores für eine hohe Aufmerksamkeitsgenerierung bei den IT-Messebesuchern auf einem großen Bildschirm angezeigt werden sollen, erhält das Scoreboard ein eigenes Frontend.

Da das Spiel auf den Geräten der Messebesucher ohne Installation ausgeführt werden soll, soll das Spiel für mobile Applikationen optimiert und in gängigen Browsern (Safari, Chrome) ausführbar sein.

Die Begründungen für die Verwendung von Unity, React, Spring Boot und der SAP BTP Cloud Foundry werden in den Architekturentscheidungen (Kapitel 9) erläutert.

# 8. Querschnittliche Konzepte

Angesichts der Größe und Komplexität unseres Projekts sowie der relativ einfachen übersichtlichen Architektur haben wir festgestellt, dass die Dokumentation von querschnittlichen Konzepten nicht erforderlich ist.

Die Entscheidung beruht auf der Tatsache, dass die spezifischen Herausforderungen, die normalerweise querschnittliche Konzepte erfordern, in unserem Projekt nicht vorhanden sind.

Wir sind uns jedoch bewusst, dass sich die Anforderungen im Laufe des Projekts ändern können. Falls sich neue Anforderungen ergeben oder die Komplexität zunimmt, sind wir bereit, die Architektur anzupassen und gegebenenfalls die Dokumentation um querschnittliche Konzepte zu erweitern, um die Software langfristig wartbar und skalierbar zu machen.

# 9. Architekturentscheidungen

## 9.1 Nutzung von Unity als Game-Engine

**Kontext:**
Es soll ein Spiel für IT-Messen entwickelt werden. Auf IT-Messen ist die Internetverbindung nicht selten instabil. Das Spiel soll auf Smartphones der Messebesucher und den Tablets von der sovanta AG ausführbar sein. Zudem besteht im Team eine geringe Erfahrung in der Spieleentwicklung.

**Begründung:**
Unity wird als Game-Engine verwendet, da aufgrund der starken Verbreitung und dem langjährigen Bestehen eine große Sammlung an Dokumentationen, Projekten und Tutorials existiert. Dies hilft bei der Einarbeitung des Teams.
Zudem hat die Tutorin des Teams Erfahrung mit Unity und kann daher bei Problemen als Ansprechpartnerin dienen.
Außerdem bietet Unity im Gegensatz zu Web-Game-Engines die Möglichkeit, das Spiel als Offline-Applikation zu erzeugen. Somit könnte das Produkt trotz schlechter Internetverbindung offline auf den Tablets der sovanta AG eingesetzt werden.

**Konsequenzen:**
Da die Einnahmen der sovanta AG in den letzten 12 Monaten bei über $200.000 liegen, muss für den Einsatz des Spiels eine Unity-Lizenz erworben werden, welche jährliche Kosten verursacht. Die sovanta AG hat bestätigt, dass im Falle eines Einsatzes des Produkts die Kosten in einem vertretbaren Rahmen liegen.

**Alternativen:**
Im Rahmen der Game-Engine-Auswahl wurden neben Unity die Game-Engines Pixie.js und Phaser betrachtet. Diese bieten den Vorteil, dass keine Lizenzgebühren ab einer bestimmten Einnahmegrenze anfallen. Jedoch gibt es für diese Game-Engines im Vergleich mit Unity weniger Dokumentation und Tutorials. Zudem sind Spiele, die mit diesen Game-Engines entwickelt worden sind, auf eine stabile Internetverbindung angewiesen.

## 9.2 Nutzung von React als Web-Frontend-Bibliothek

**Kontext:**
Das Team hat keine Erfahrung in der Nutzung von Web-Frontend-Bibliotheken.

**Begründung:**
Aufgrund der fehlenden Erfahrung des Teams mit Web-Frontend-Bibliotheken, war es wichtig, eine Technologie zum Erstellen von Web-Frontends auszuwählen, die leicht zu verstehen ist, sowie viel Dokumentation und Tutorials bietet. React bietet dies und wird daher für die Erstellung des Frontends verwendet.

**Konsequenzen:**
Durch die Einfachheit von React ist es möglich, schnell Web-Frontends zu erzeugen. Da React jedoch viel weniger Vorgaben macht, als es beispielsweise bei Angular der Fall ist, besteht in Kombination mit der Unerfahrenheit des Teams die Gefahr, Code zu schreiben, welcher schlecht strukturiert und kaum erweiterbar ist.

**Alternativen:**
Als Alternative wurde Angular betrachtet, da dieses Framework häufig von der sovanta AG eingesetzt wird. Nach einer Recherche wurde jedoch klar, dass Angular weitaus komplexer als React und daher für das Team ungeeignet ist.

## 9.3 Nutzung von Spring Boot für die Erstellung des Backends

**Kontext:**
Ein Backend wird benötigt, um die Spielerdaten (Name, Unternehmen und erreichter Score) in einer Datenbank zu speichern.
Das Team hat keine Erfahrung im Erstellen von Backend-Applikationen.
Im Rahmen des Studiums und privaten Projekten hat das Team fundiertes Wissen in der Programmiersprache Java erlangt.
Für das Deployment der Backend-Applikation wird Cloud Foundry der SAP BTP verwendet (siehe 9.4 Nutzung von Cloud Foundry für Deployment für die Begründung).

**Begründung:**
Spring Boot ermöglicht es, schnell Backend-Applikationen zu erzeugen, um eine Verbindung zu einer Datenbank herzustellen und eine REST-API aufzubauen. Aufgrund des langen Bestehens gibt es viel Dokumentation und Tutorials, die die Einarbeitung des Teams erleichtert.

**Konsequenzen:**
Die Konsequenzen können nicht eingeschätzt werden.

**Alternativen:**
Als Alternative zu Spring Boot bietet sich Quarkus an, mit dem sich nach der Empfehlung eines Tutors beschäftigt wurde. Hier gibt es jedoch größere Probleme beim Deployment auf der Cloud Foundry und es bietet für die Einfachheit der Backend-Applikation keine großen Vorteile gegenüber Spring Boot.

## 9.4 Nutzung von Cloud Foundry für Deployment

**Kontext:**
Die sovanta AG möchte, dass das Produkt SAP BTP Technologien und keine Konkurrenzprodukte der SAP BTP verwendet.

**Begründung:**
Da keine anderen Technologien der SAP BTP für einen sinnvollen Einsatz bei diesem Produkt in Frage kommen und das Deployment über die Cloud Foundry ohne größere Probleme funktioniert, wird Cloud Foundry für das Deployment des Scoreboard Frontends und des Unity Spiels verwendet. Dadurch wird die Vorgabe der sovanta AG, einen Service der SAP BTP zu verwenden, erfüllt.

**Konsequenzen:**
Die Konsequenzen können nicht eingeschätzt werden.

**Alternativen:**
Es wurden keine Alternativen betrachtet.

## 9.5 REST-API für die Kommunikation zwischen Frontend und Backend

**Kontext:**
Das Team hat keine Erfahrung im Erstellen von Kommunikationswegen zwischen Frontend und Backend.

**Begründung:**
REST-API sind weit verbreitet und es gibt dementsprechend viel Dokumentation und Tutorials. Zudem konnte im Rahmen des vertikalen Prototyping schnell eine REST-API über Spring Boot eingerichtet werden.

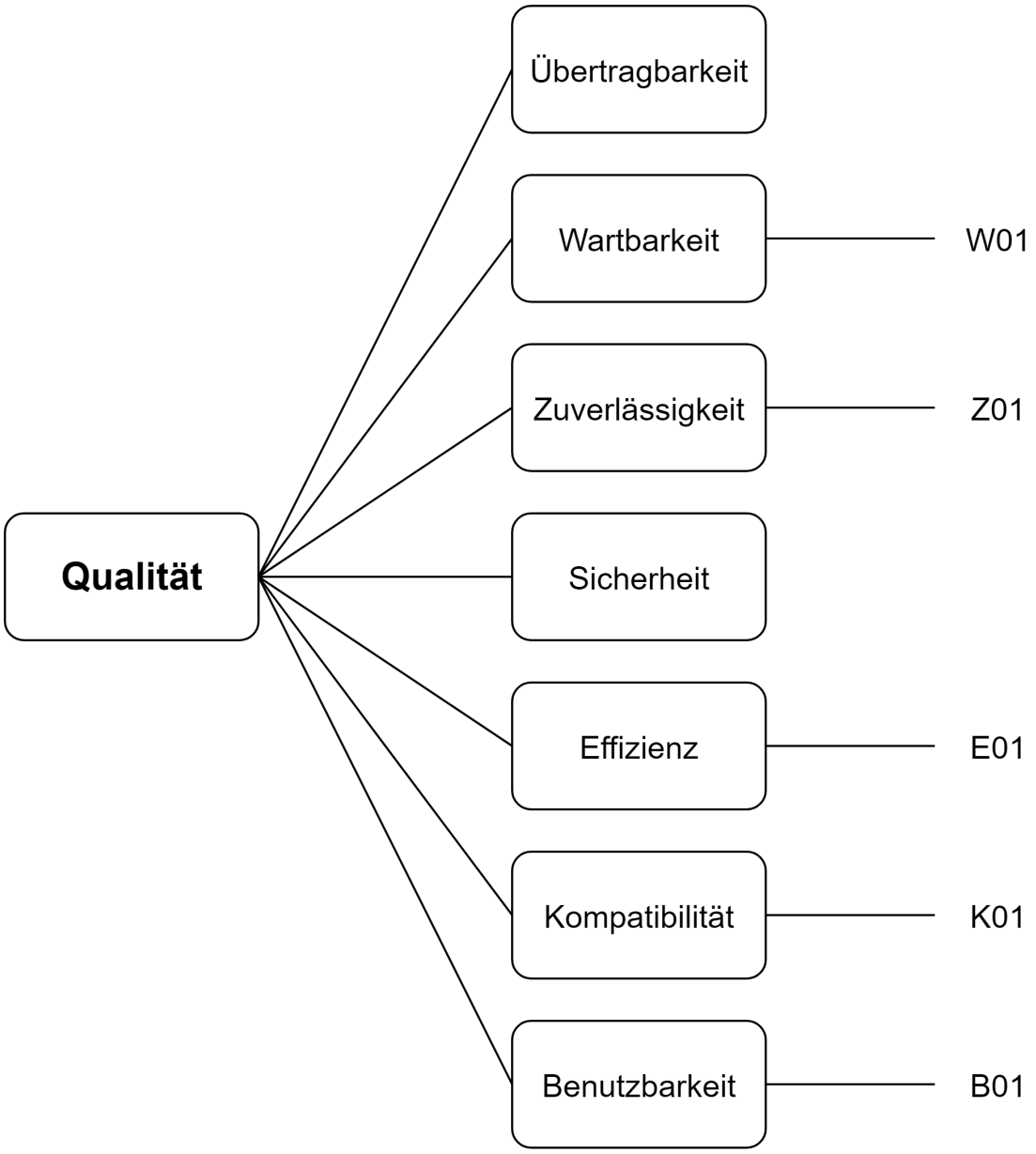
**Konsequenzen:**
Wird ein neuer Score gespeichert, kann es wenige Sekunden dauern, bis das Scoreboard Frontend diesen anzeigt. Das liegt daran, dass das Frontend die aktuellen Scores in regelmäßigen Abständen abfragt.

**Alternativen:**
Als Alternative bieten sich Websockets an, die eine permanente Verbindung zwischen Backend und Scoreboard erstellen. Dies hat den Vorteil, dass die Scores auf dem Scoreboard in Echtzeit aktualisiert werden können. Zurzeit ist es nicht besonders wichtig, neue Scores unmittelbar im Scoreboard anzeigen zu lassen und daher ist eine REST-API ausreichend.

# 10. Qualitätsanforderungen

Im Folgenden werden die Qualitätsanforderungen an das Produkt anhand eines Qualitätsbaums und Qualitätsszenarien beschrieben.
Genauer werden diese Qualitätsanforderungen in der Anforderungsspezifikation (Version 2) im Abschnitt “4.4.3 Nicht-funktionale Anforderungen” beschrieben.

## 10.1 Qualitätsbaum



## 10.2 Qualitätsszenarien

| ID | Beschreibung |
|---|---|
| W01 | Das Produkt soll bei Fehlverhalten klare Fehlermeldungen liefern, damit der Nutzer weiß, was er falsch gemacht hat.<br>Der Nutzer gibt am Ende des Spiels bei der Dateneingabe seinen Namen nicht an und versucht auf den “Sende”-Button zu drücken. Es wird eine Fehlermeldung ausgegeben, die den Nutzer darauf hinweist, seinen Namen anzugeben. |
| Z01 | Das Produkt soll den ganzen Tag von Messebesuchern am Stand der sovanta AG benutzt werden können. Deshalb wird sichergestellt, dass es den ganzen Tag möglich ist, das Spiel zu spielen und die Punktestände auf dem Scoreboard zu sehen. |
| E01 | Da auf Messen eine unbeständige Internetverbindung vorliegt, muss das Produkt effizient arbeiten, d.h. die zu übertragende Datenmenge muss so gering wie möglich gehalten werden.<br>Das Spiel wird geladen und danach ist es möglich zu spielen, ohne viele Daten aus dem Backend zu benötigen.<br>Eine Kommunikation mit dem Backend ist nur für die Übertragung der Punktestände nötig. Dafür muss die Internetverbindung nicht konstant stabil sein. |
| K01 | Das Produkt soll auf verschiedenen Geräten (z.B. Tablet und Smartphone) von verschiedenen Marken (z.B. Apple und Samsung) verwendet werden können. Das ist möglich durch die Verwendung des Produkts im Webbrowser (Safari und Google Chrome). |
| B01 | Da Messebesucher aus verschiedenen Bereichen das Produkt verwenden, muss das Produkt für diese Besucher einfach zu benutzen sein. Dafür müssen vorgegebene UX-Heuristiken beachtet werden. |

# 11. Risiken und technische Schulden

Nach einer eingehenden Analyse potenzieller architekturrelevanter Risiken sind wir zu dem derzeitigen Stand gekommen, dass uns keine solchen Risiken bekannt sind. Unser Team ist sich der allgegenwärtigen Natur von Risiken bewusst und steht bereit, auf neue Informationen und Erkenntnisse zu reagieren.

# 12. Glossar

| Begriff | Beschreibung |
|---|---|
| Cloud Foundry | Cloud Foundry ist eine Open-Source-Plattform-as-a-Service (PaaS), die Entwicklern ermöglicht, Anwendungen einfach in der Cloud zu erstellen.<br><br>https://www.cloudfoundry.org/ |
| SAP BTP | Die SAP Business Technology Platform (SAP BTP), eine für SAP-Anwendungen in der Cloud optimierte Innovationsplattform, vereint Funktionen für Anwendungsentwicklung, Datenmanagement und Analysen, Integration, Automatisierung und KI in einer zentralen Umgebung.<br><br>https://www.sap.com/germany/products/technology-platform.html |
| Sovanta Innovation Factory | Die sovanta Innovation Factory for SAP BTP ist ein Konzept für Entwicklung und Bereitstellung von Innovationen auf der SAP BTP.<br><br>https://sovanta.com/innovation-factory-for-sap-btp/ |
| SAP | SAP steht für “Systemanalyse und Programmentwicklung”. Es handelt sich um einen international führenden deutschen Softwarekonzern, der umfangreiche Unternehmenssoftware Lösungen entwickelt und vertreibt.<br><br>https://www.sap.com/germany/index.html |
| UX | UX steht für “User Experience” (Nutzererfahrung) und bezieht sich auf das Gesamterlebnis einer Person bei der Interaktion mit einem Produkt, einer Dienstleistung oder einem System. |
| Spring Boot | Spring Boot ist ein Open-Source-Framework für die schnelle Entwicklung von Java-Anwendungen. Es bietet eine einfache Konfiguration und die Unterstützung verschiedener Funktionen an.<br><br>https://spring.io/ |
| Quarkus | Quarkus ist ein Open-Source-Framework für die schnelle Entwicklung von Java-Anwendungen. Es ist für Cloud-native Architekturen optimiert, hat eine geringe Startzeit und weist einen geringen Speicherbedarf auf.<br>https://quarkus.io/ |

| | |
|---|---|
| Unity | Unity ist eine 2D- und 3D-Game-Engine.<br><br>https://unity.com/de |
| React | React ist eine Bibliothek zum Erstellen von grafischen Benutzeroberflächen für Web-Anwendungen.<br><br>https://react.dev/ |
| PostgreSQL | PostgreSQL ist ein Datenbankmanagementsystem.<br><br>https://www.postgresql.org/ |
| Pixie.js | Pixi.js ist eine JavaScript-Bibliothek für die Erstellung von 2D-Grafiken und Animationen im Web.<br><br>https://pixijs.com/ |
| Phaser | Phaser ist ein 2D-Spiele-Framework.<br><br>https://phaser.io/ |
| Angular | Angular ist ein Web-Framework zum Erstellen von Frontends für Webanwendungen.<br><br>https://angular.io/ |