Architekturdokumentation

Learn to innovate

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

hochschule mannheim

Version 2.0 vom 13.06.2023
Verantwortlich: Mike Menzel, Fabian Hoppe

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

Alle Wörter, die im Glossar näher erläutert werden, sind bei ihrem ersten Erscheinen in *kursiver Schrift* markiert.

Änderungsverzeichnis

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

Version	Datum	Änderung	Autor	Prüfer
2.0	13.06.23	- Randbedingungen und Qualitätsziele und -anforderungen anpassen - Überarbeitung 2. Ebene Game - Verfeinerung des Scoreboards	Mike Menzel, Fabian Hoppe	Stephan Halder
1.2	12.06.23	- Deployment-Pfeile für Deployment Diagramm - Alphabetische Sortierung Glossar - Reihenfolge der Architekturentscheidungen - Referenzen auf Architekturentscheidungen bei der Bausteinsicht - Erklärung und Begründung für externes Formular für Spielerdaten-Eingabe - Aktualisierung Sequenzdiagramm für “Spielerdaten speichern” - füge Ablauf “Spielen” hinzu - füge Ablauf “Weiterleiten zu Formular” hinzu	Mike Menzel	

1.1	09.06.23	- Kursive Schreibung der Wörter, die im Glossar stehen - Verwenden von “Learn to innovate” anstelle von “Produkt” und “Dumb ways to innovate” - Pfeilrichtung bei Abbildung 3.1 geändert - Abbildungsnummern in Texten verwenden	Mike Menzel	
1.0	16.05.23	Review des Dokuments und Korrektur von Fehlern	Mike Menzel, Fabian Hoppe	Stephan Halder, Eren Saglam
0.3	15.05.23	Referenzen zur Anforderungsspezifikation und dem Projekthandbuch wurden eingefügt. Beschreibungen wurden präzisiert. Formulierungen wurden angepasst.	Fabian Hoppe, Mike Menzel	
0.2	12.05.23	Diagramme der Baustein-, Verteilungs- und Laufzeitsicht wurden eingefügt. Architekturentscheidungen wurden dokumentiert.	Mike Menzel, Eren Saglam	
0.1	09.05.23	Grundstruktur des Dokuments wurde erstellt.	Eren Saglam	

Inhaltsverzeichnis

1. Einführung und Ziele

1.1 Aufgabenstellung

Das Team Sepia entwickelt im Rahmen des Software-Entwicklungsprojektes im Sommersemester 2023 an der Hochschule Mannheim ein Produkt für die *sovanta AG*. Ausgeliefert werden sollen eine prototypische Applikation und ein überzeugendes Konzept, welche Besucher von Technik- und IT-Messen von den Möglichkeiten der *sovanta Innovation Factory for SAP BTP* überzeugen sollen. Das Produkt muss im Rahmen einer Abschlusspräsentation am 26.06.23 am Standort Heidelberg der sovanta AG präsentiert und ausgeliefert werden.
Durch das Produkt sollen die Besucher der Messen aufmerksam auf den Messestand der sovanta AG werden. Außerdem soll durch das Nutzen des Produktes das Interesse an der Arbeit der sovanta AG geweckt werden.

1.2 Qualitätsziele

Hier werden die architekturrelevanten Qualitätsziele für “Learn to innovate” beschrieben. Die Anforderungsnummern sind aus der Anforderungsspezifikation (Version 3) übernommen. Konkrete Qualitätsszenarien und die Zuordnung in Qualitätskriterien befinden sich in Kapitel 10. Qualitätsanforderungen.
Weitere Qualitätsanforderungen für “Learn to innovate”, die aber nicht architekturrelevant sind, sind in der Anforderungsspezifikation (Version 3) zu finden.

Anforderungs-nummer	**Beschreibung**
NFA-2	Das Spiel soll auch bei kurzen Netzwerkausfällen weiter nutzbar sein.
NFA-3	Durch den modularen Aufbau soll es einfach möglich sein, weitere Level hinzuzufügen.
NFA-4	Das System soll auf verschiedenen Geräten (z.B. Tablet und Smartphone) von verschiedenen Marken (z.B. Apple und Samsung) verwendet werden können.

1.3 Stakeholder

Hier befindet sich eine Übersicht über die wichtigsten Stakeholder des Projekts.
Weitere Informationen zu den Stakeholdern sind im Projekthandbuch (Version 3) enthalten.

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

2. Randbedingungen

Im Folgenden werden die Randbedingungen genannt, die die Architektur betreffen.
Weitere Details zu den Randbedingungen für das Projekt sind der Anforderungsspezifikation (Version 3) zu entnehmen.

Randbedingung	Beschreibung
Instabile Internetverbindung	Die Internetverbindung kann auf Messen instabil sein, weshalb “Learn to innovate” damit umgehen können muss.
Hardware muss mobil sein	Die sovanta AG baut ihre Werbematerialien selbst auf. Daher muss alles, was am Stand genutzt wird, transportabel und mobil sein.

3. Kontextabgrenzung

Abbildung 3.1 zeigt, dass das System die *SAP BTP* als Nachbarsystem besitzt und dieses benutzt. Die Begründung dafür findet man unter 9.3 Nutzung von *Cloud Foundry* für Deployment.

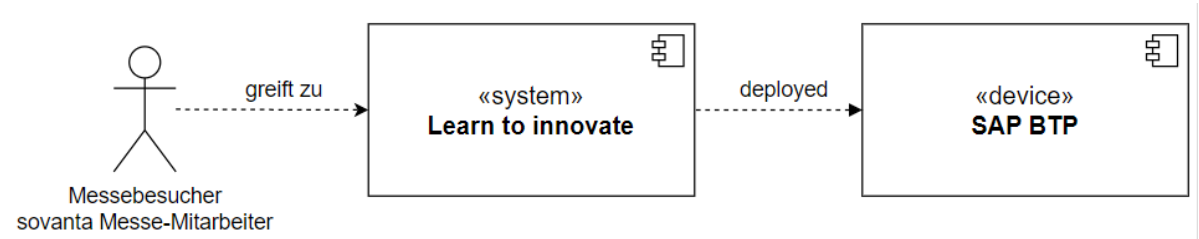


Abbildung 3.1 Kontextabgrenzung

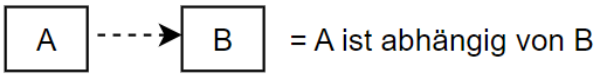
Die Nutzer können auf das System mittels Webbrowser zugreifen.
Das System besitzt mit der SAP BTP ein Nachbarsystem. Die SAP BTP wird dazu genutzt, um das System zu deployen.

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

Abbildung 5.1.1 zeigt die Komponenten der ersten Ebene.

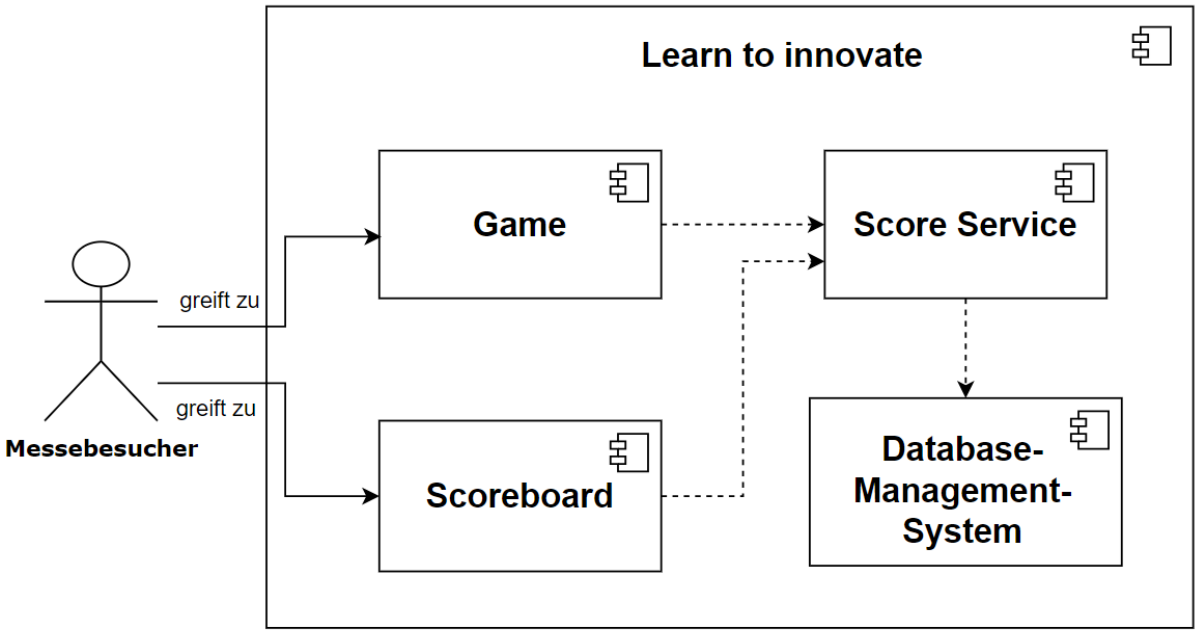


Abbildung 5.1.1 Bausteinsicht Ebene 1 Gesamtsystem

Komponente	Beschreibung
Game	Eine *Unity*-Applikation, welche das Spiel darstellt. Die Verwendung von Unity wird in 9.1 Nutzung von Unity als Game-Engine begründet.
Scoreboard	Eine *React*-Applikation, welche die Spieler mit ihren erreichten Scores aus dem Spiel anzeigt. Da die Scores für eine hohe Aufmerksamkeitsgenerierung bei den IT-Messebesuchern auf einem großen Bildschirm angezeigt werden sollen, erhält das Scoreboard ein eigenes Frontend. Zudem findet hier über ein Formular die Eingabe des Spieler- und Unternehmensnamens statt, um sich für das Scoreboard einzutragen (siehe 9.6 Externes Formular für die Spielerdaten-Eingabe für eine Begründung). Die Verwendung von React wird in 9.2 Nutzung von React als Web-Frontend-Bibliothek begründet.

Score Service	Eine Spring-Boot-Applikation, welche den beiden Komponenten “Game” und “Scoreboard” eine REST-API bietet (siehe 9.4 REST-API für die Kommunikation zwischen Frontend und Backend für eine Begründung). 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. Die Verwendung von *Spring Boot* wird in 9.5 Nutzung von Spring Boot für die Erstellung des Backends begründet.
Database-Management-System	Ein *PostgreSQL* Datenbankmanagementsystem, welches zur persistenten Speicherung der Spielerdaten dient. Die Daten werden für die Dauer der IT-Messe gespeichert. Diese Komponente wird von PostgreSQL umgesetzt und daher nicht weiter verfeinert.

5.2 Ebene 2

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

Ebene 2 Score Service

Die Abbildung 5.2.1 zeigt die Komponenten der zweiten Ebene des Score Services.

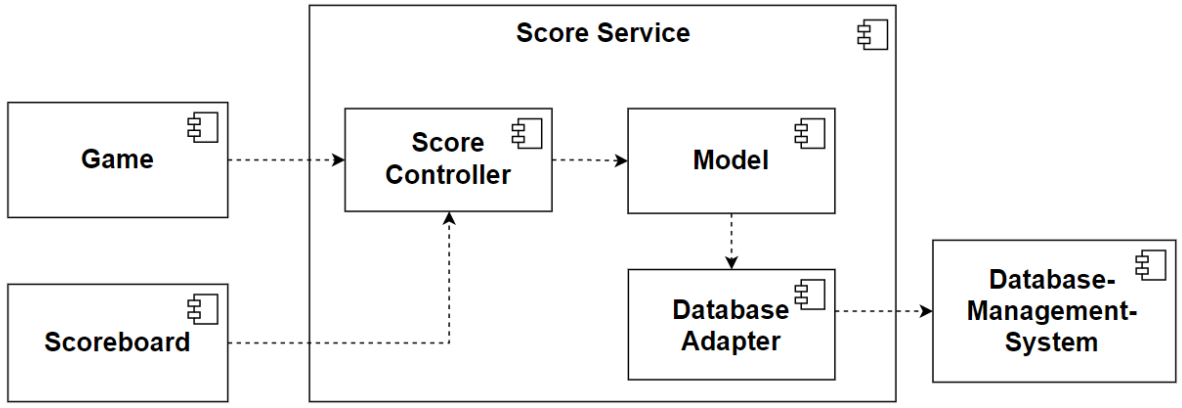


Abbildung 5.2.1 Bausteinsicht Ebene 2 Score Service

Komponente	Beschreibung
Score Controller	Nimmt HTTP-Requests von den Komponenten “Game” und “Scoreboard” entgegen und verarbeitet diese entsprechend. Zudem finden hier Input Validierungen statt. Diese Komponente wird nicht weiter verfeinert, da eine weitere Verfeinerung in ein Feindesign münden würde.
Model	Enthält die Geschäftslogik. Diese Komponente wird nicht weiter verfeinert, da eine weitere Verfeinerung in ein Feindesign münden würde.
Database Adapter	Stellt eine Verbindung zu der Datenbank her. Diese Komponente wird im Folgenden nicht weiter verfeinert, da diese von dem Spring Boot Framework umgesetzt wird.

Ebene 2 Game

Abbildung 5.2.2 zeigt die Komponenten der zweiten Ebene der Komponente Game.

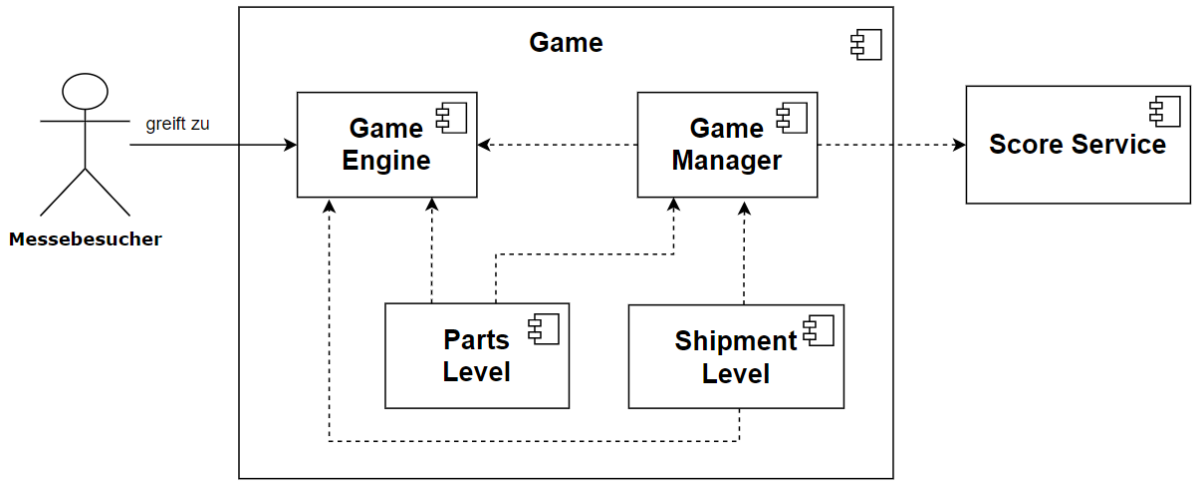


Abbildung 5.2.2 Bausteinsicht Ebene 2 Game

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. Diese wird nicht weiter verfeinert, da diese von Unity umgesetzt wird.
Game Manager	Enthält die Spiel-Initialisierung, die Steuerung des Spielablaufs, das Setzen der Level-Schwierigkeit und die Level-Auswahl. Neue Levels müssen dem Game Manager

	bekannt gemacht werden, sodass diese in das Spiel eingebunden werden (siehe NFA-3 unter 1.2 Qualitätsziele). Diese Komponente wird nicht weiter verfeinert, da eine weitere Verfeinerung in ein Feindesign münden würde.
Parts Level	Enthält die Spiellogik für das Level zum Bereich *Parts der sovanta Innovation Factory for SAP BTP*. Diese Komponente wird nicht weiter verfeinert, da eine weitere Verfeinerung in ein Feindesign münden würde.
Shipment Level	Enthält die Spiellogik für das Level zum Bereich *Shipment der sovanta Innovation Factory for SAP BTP*. Diese Komponente wird nicht weiter verfeinert, da eine weitere Verfeinerung in ein Feindesign münden würde.

Ebene 2 Scoreboard

Abbildung 5.2.3 zeigt die Komponenten der zweiten Ebene der Komponente Scoreboard.

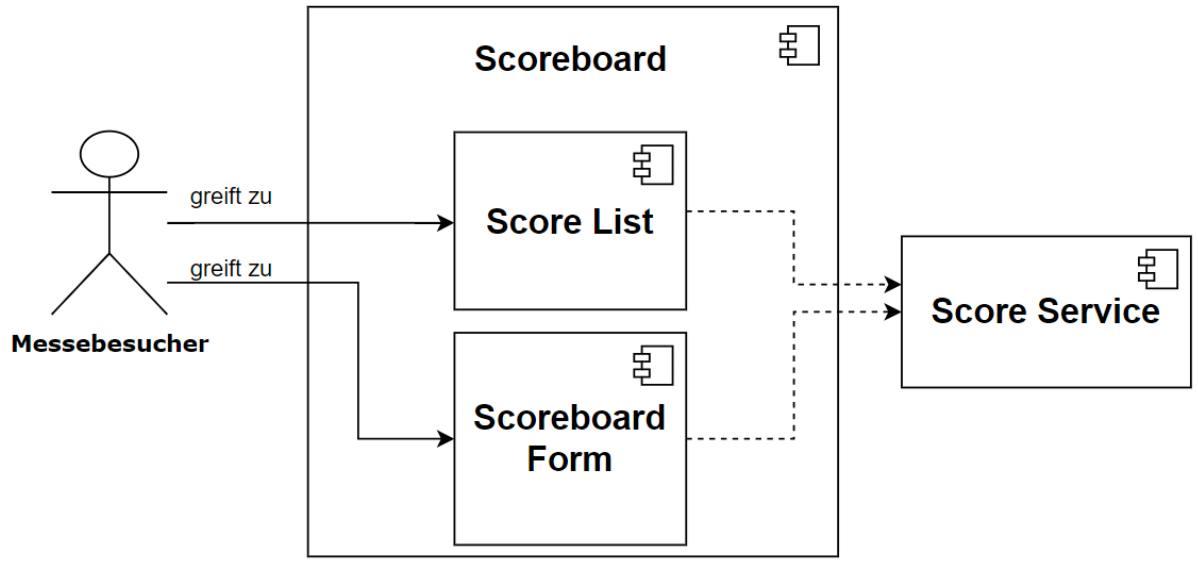


Abbildung 5.2.3 Bausteinsicht Ebene 2 Scoreboard

Komponente	Beschreibung
Score List	Zeigt die Scores der Einzelspieler und Unternehmen an. Diese Komponente ist so simpel, dass sie nicht weiter verfeinert wird.
Scoreboard Form	Bietet ein Formular zur Eingabe des Spieler- und Unternehmensnamens, sodass sich Spieler bei der Score-Liste eintragen können. Diese Komponente ist so simpel, dass sie nicht weiter verfeinert wird.

6. Laufzeitsicht

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

6.1 Ablauf “Weiterleiten zu Formular”

Die Abbildung 6.1.1 zeigt die Weiterleitung zum Formular zur Eingabe des Spieler- und Unternehmensnamens (siehe 9.6 Externes Formular für die Spielerdaten-Eingabe für eine Begründung). Daraufhin folgt typischerweise 6.2 Ablauf “Spielerdaten speichern”.
Der Umweg über eine ID dient dazu, den Score nicht direkt (beispielsweise per GET-Parameter in der URL) an das Scoreboard zu übergeben, um das Absenden beliebiger Scores zu erschweren.

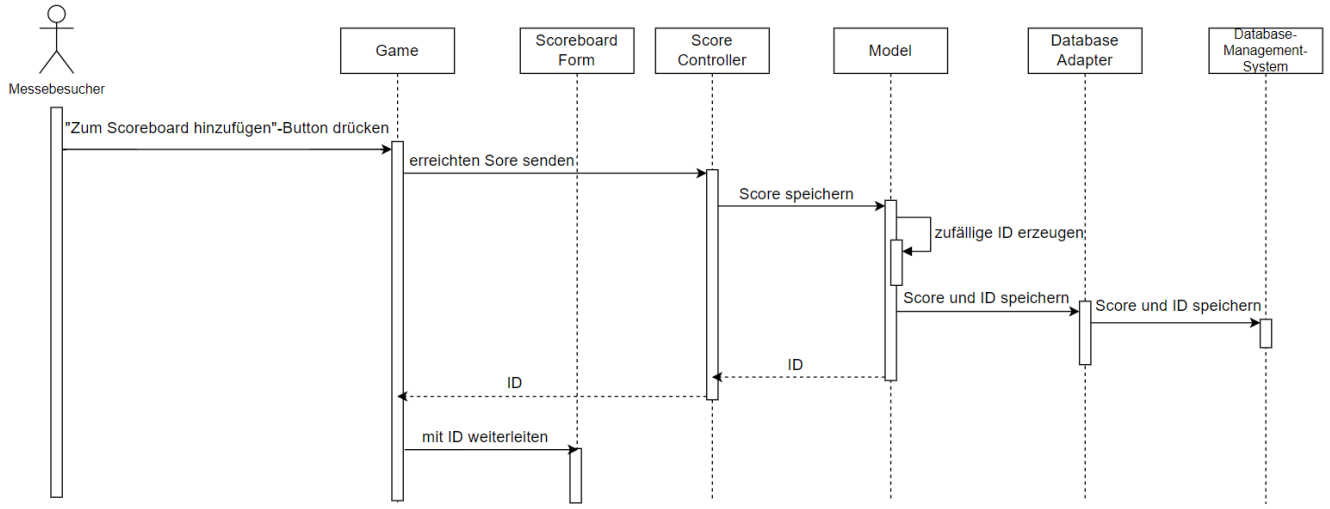


Abbildung 6.1.1 Sequenzdiagramm Weiterleitung zum Formular

6.2 Ablauf “Spielerdaten speichern”

Die Abbildung 6.2.1 stellt den grundlegenden Ablauf dar, wie neue Spielerdaten (Spieler- und Unternehmensname sowie erreichter Score) nach dem Beenden des Spiels gespeichert werden. Der Ablauf knüpft an 6.1 Ablauf “Weiterleiten auf Formular” an.

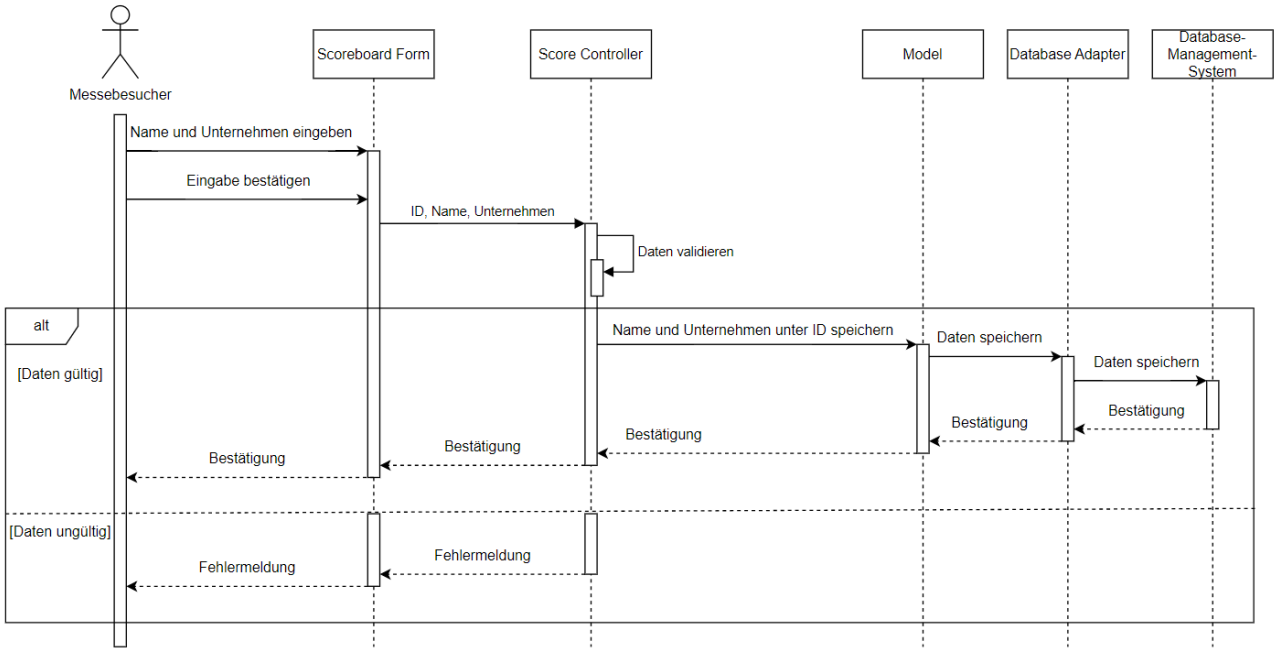


Abbildung 6.2.1 Sequenzdiagramm Spielerdaten Speicherung

6.3 Ablauf “Spielen”

Abbildung 6.3.1 zeigt den grundlegenden Ablauf für das Spielen des Spiels.

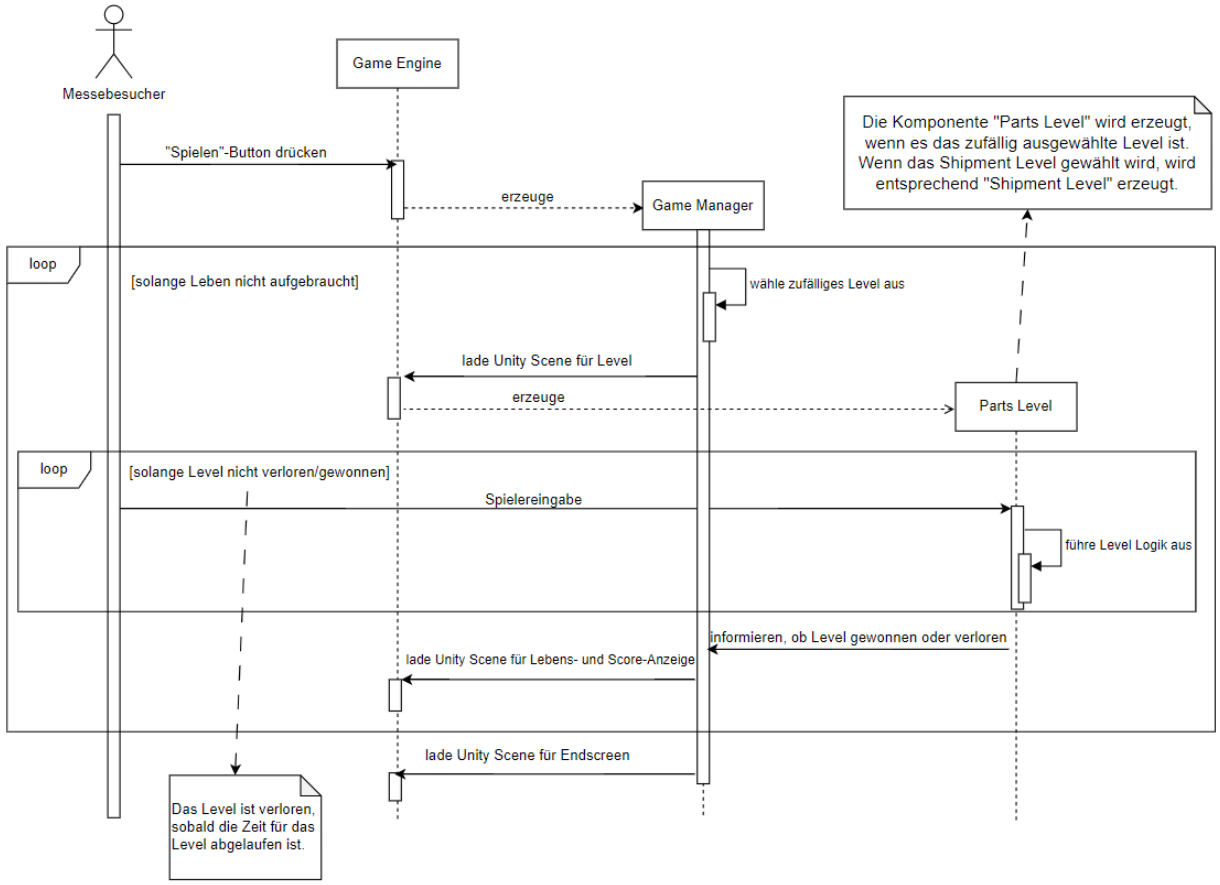
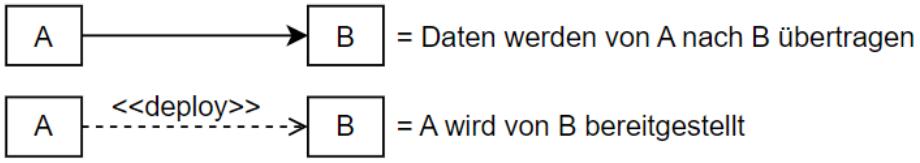


Abbildung 6.3.1 Sequenzdiagramm Spielen

7. Verteilungssicht

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



Die Abbildung 7.1 zeigt die Verteilungssicht von “Learn von innovate”.
Die Begründungen für die Verwendung von Unity, React, Spring Boot und der SAP BTP Cloud Foundry werden in den Architekturentscheidungen (Kapitel 9) erläutert.

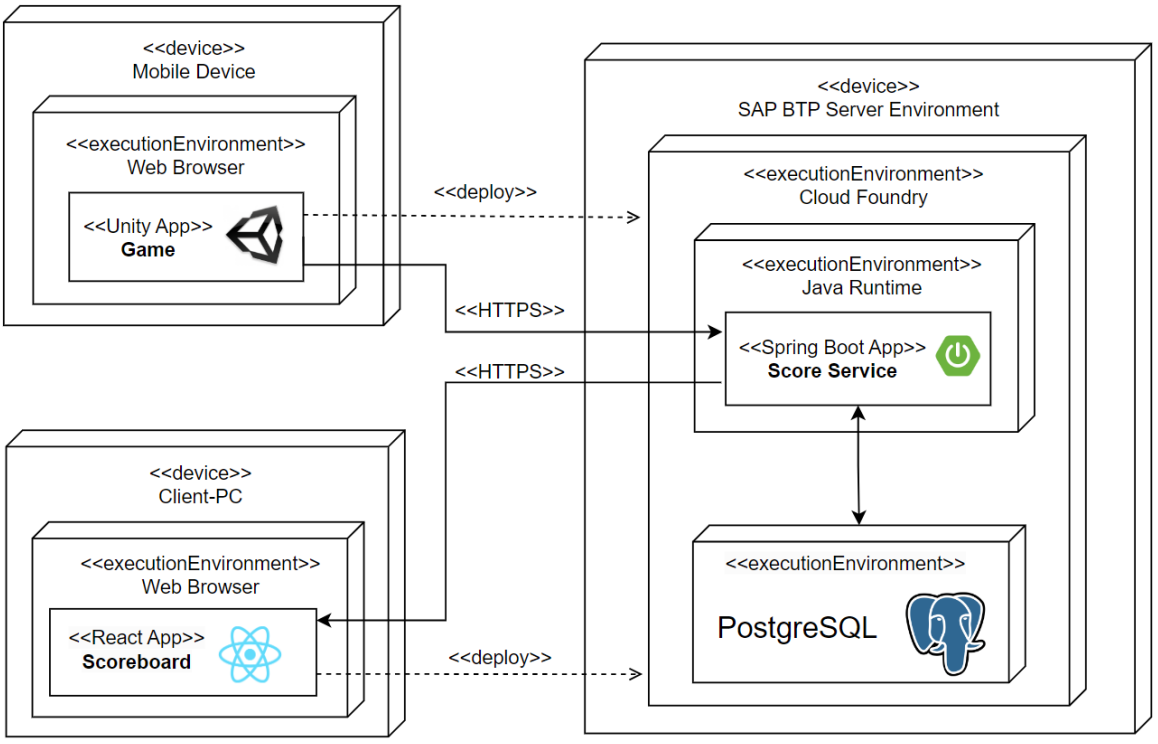


Abbildung 7.1 Verteilungssicht

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 (siehe 2. Randbedingungen). Das Spiel soll auf Smartphones der Messebesucher und den Tablets der sovanta AG ausführbar sein (siehe K01 unter 10.2 Qualitätsszenarien). 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.
Zusätzlich werden bei Unity Spielen für Browser zu Beginn alle benötigten Dateien für das Spielen geladen. So kann das Spiel auch dann weitergespielt werden, wenn die Internetverbindung ausfällt (siehe REFERENZ AUF QUALIT SZENARIO)
Außerdem bietet Unity im Gegensatz zu Web-Game-Engines die Möglichkeit, das Spiel als Offline-Applikation zu erzeugen. Somit könnte das “Learn to innovate” 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 von “Learn to innovate” die Kosten in einem vertretbaren Rahmen liegen.

Alternativen:
Im Rahmen der Game-Engine-Auswahl wurden neben Unity die Game-Engines *Pixi.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 Cloud Foundry für Deployment

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

Begründung:
Da keine anderen Technologien der SAP BTP für einen sinnvollen Einsatz bei “Learn to innovate” in Frage kommen und das Deployment über die Cloud Foundry ohne größere Probleme funktioniert, wird Cloud Foundry für das Deployment des Scoreboards 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.

9.4 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 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.

9.5 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 9.3 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:
Das Team hat sich nach der Empfehlung eines Tutors mit *Quarkus* beschäftigt. 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.

9.6 Externes Formular für die Spielerdaten-Eingabe

Kontext:
Das Spiel wird mit Unity entwickelt und soll in Browsern ausgeführt werden können, sodass Messebesucher das Spiel auf ihrem eigenen mobilen Gerät spielen können.
Um aus einem Unity-Spiel eine Browser-Applikation zu generieren, muss man den WebGL Build von Unity verwenden.

Begründung:
Der WebGL Build von Unity wird offiziell nicht für mobile Geräte unterstützt (siehe Unity Dokumentation), wodurch die Verwendung der Systemtastatur mobiler Geräte für Eingaben nicht möglich ist. Dadurch kann die Eingabe des Spieler- und Unternehmensnamens nicht direkt im Spiel durchgeführt werden. Da das Spiel im Browser ausführbar sein soll (siehe K01 unter 10.2 Qualitätsszenarien), wird ein externes Formular für die Eingabe des Spieler- und Unternehmensnamens verwendet.

Konsequenzen:
Um sich für das Scoreboard einzutragen, wird der Spieler vom Spiel auf ein externes Formular weitergeleitet. Dies kann für den Spieler ungewohnt erscheinen, da er erwarten könnte, seine Eingabe direkt im Spiel zu tätigen.

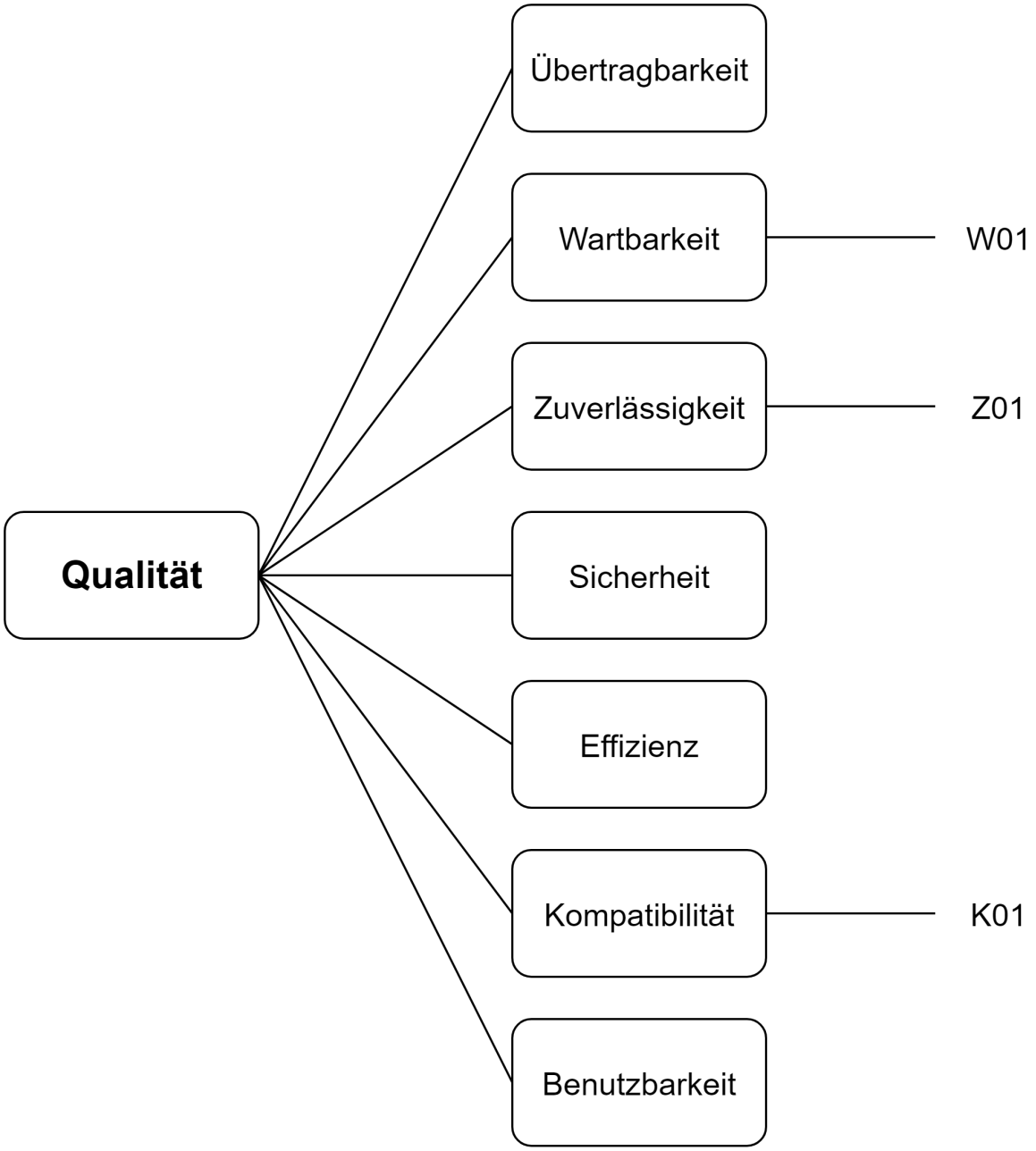
Alternativen:
Alternativ wäre es möglich, eine eigene Systemtastatur für mobile Geräte innerhalb des Spiels zu entwickeln. Dies ist jedoch aus der Sicht der Benutzerfreundlichkeit ungeeignet, da sich die Systemtastaturen abhängig vom Betriebssystem stark unterscheiden. Zudem wäre der Entwicklungsaufwand dafür hoch.

10. Qualitätsanforderungen

Im Folgenden werden die architektur-relevanten Qualitätsanforderungen an “Learn to innovate” anhand eines Qualitätsbaums und zugehörigen Qualitätsszenarien beschrieben. Genauer werden diese Qualitätsanforderungen in der Anforderungsspezifikation (Version 3) beschrieben.

10.1 Qualitätsbaum

In Abbildung 10.1.1 ist ein Qualitätsbaum dargestellt, welcher die Qualitätsanforderungen den jeweiligen Qualitätskriterien zuordnet.



10.2 Qualitätsszenarien

In der folgenden Tabelle werden Qualitätsszenarien zu zugehörigen Qualitätsanforderungen beschrieben. Die Szenarien beschreiben konkrete Abläufe, welche den Einfluss der jeweiligen Qualitätsanforderung auf die Architektur der Software verdeutlichen.

ID	Beschreibung	Zugehörige Qualitäts-anforderung
W01	Durch den modularen Aufbau soll es einfach möglich sein, weitere Level hinzuzufügen. Ein sovanta Mitarbeiter entwickelt ein neues Level für das Spiel. Um dieses Level im Spiel hinzuzufügen, muss der Mitarbeiter das Level dem Game-Manager bekannt machen. Der Game-Manager kümmert sich dann darum, dass das Level ausgeführt wird.	NFA-3
Z01	Das Spiel soll auch bei kurzen Netzwerkausfällen weiter nutzbar sein. Beim Aufrufen der Internetseite des Spiels wird das ganze Spiel im Browser des Clients (Gerät der sovanta AG oder Gerät des Nutzers) geladen. Während der Nutzer spielt, bricht die Internetverbindung immer mal wieder für kurze Zeit ab. Dadurch dass das Spiel auf der Client-Seite komplett geladen wurde, kann der Nutzer einfach weiterspielen und bemerkt gar nicht, dass die Internetverbindung immer für kurze Zeit abbricht.	NFA-2
K01	Das System soll auf verschiedenen Geräten (z.B. Tablet und Smartphone) von verschiedenen Marken (z.B. Apple und Samsung) verwendet werden können. Ein Nutzer kommt an den Messestand der sovanta AG und möchte das Spiel spielen. Dafür benutzt er ein iPad Air 5 der sovanta AG am Stand. Die Mitarbeiter am Messestand informieren den Nutzer, dass er das Spiel auch im Webbrowser auf seinem eigenen mobilen Endgerät benutzen kann. Der Nutzer besitzt ein Samsung Galaxy S20 und scannt damit einen QR-Code, um auf die Internetseite mit dem Spiel zu kommen. Auf seinem Gerät funktioniert das Spiel genauso, wie es auch auf dem iPad der sovanta AG funktioniert hat.	NFA-4

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
Angular	Angular ist ein Web-Framework zum Erstellen von Frontends für Webanwendungen. https://angular.io/
Cloud Foundry	Cloud Foundry ist eine Open-Source-Plattform-as-a-Service (PaaS), die Entwicklern ermöglicht, Anwendungen einfach in der Cloud zu erstellen. https://www.cloudfoundry.org/
Parts der sovanta Innovation Factory for SAP BTP	Ein Bereich der sovanta Innovation Factory for SAP BTP, bei dem es um die Wiederverwendung von bereits entwickelten Komponenten geht. https://sovanta.com/innovation-factory-for-sap-btp/
Phaser	Phaser ist ein 2D-Spiele-Framework. https://phaser.io/
Pixi.js	Pixi.js ist eine JavaScript-Bibliothek für die Erstellung von 2D-Grafiken und Animationen im Web. https://pixijs.com/
PostgreSQL	PostgreSQL ist ein Datenbankmanagementsystem. https://www.postgresql.org/
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. https://quarkus.io/
React	React ist eine Bibliothek zum Erstellen von grafischen Benutzeroberflächen für Web-Anwendungen. https://react.dev/
SAP	SAP steht für “Systemanalyse und Programmentwicklung”. Es handelt sich um einen international führenden deutschen

Begriff	Beschreibung
	Softwarekonzern, der umfangreiche Unternehmenssoftware Lösungen entwickelt und vertreibt. https://www.sap.com/germany/index.html
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. https://www.sap.com/germany/products/technology-platform.html
Shipment der sovanta Innovation Factory for SAP BTP	Ein Bereich der sovanta Innovation Factory for SAP BTP, bei dem es um die Bereitstellung des Produkts für den Kunden geht. https://sovanta.com/innovation-factory-for-sap-btp/
sovanta AG	Die sovanta AG ist ein deutsches mittelständisches Softwareunternehmen und unser Kunde. https://sovanta.com/
sovanta Innovation Factory for SAP BTP	Die sovanta Innovation Factory for SAP BTP ist ein Konzept für Entwicklung und Bereitstellung von Innovationen auf der *SAP BTP*. https://sovanta.com/innovation-factory-for-sap-btp/
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. https://spring.io/
Unity	Unity ist eine 2D- und 3D-Game-Engine. https://unity.com/de