Blackout SAD

Table of Contents

Disclaimer: Inhalt teils mit Antworten von ChatGPT (3.5, OpenAI) bearbeitet.

1. Einführung und Ziele

In diesem Kapitel wird auf die Aufgabenstellung, die funktionale Anforderung an das Produkt und dessen Qualitätsziele eingegangen. Des Weiteren wird ein genauer Blick auf die Stakeholder und deren Erwartungshaltung geworfen. Es bietet somit einen Überblick über die Rahmenbedingungen für das geplante System.

1.1. Aufgabenstellung

Die fachliche Aufgabenstellung besteht darin, ein neues System zu entwickeln oder ein bestehendes zu modifizieren, um eine bestimmte fachliche Aufgabe zu unterstützen oder die Qualität zu verbessern. Es wird darauf hingewiesen, dass detaillierte Anforderungsdokumente vorhanden sind, die auf die entsprechenden Versionen und Ablageorte verweisen. Die Motivation für das Projekt liegt in der zukünftigen Nutzung und dem Streben nach Effizienzsteigerung.

1.1.1. Was ist Blackout?

"Blackout" ist ein Produkt, genauer gesagt ein Quizspiel, das auf einem Raspberry Pi läuft. Es besteht aus einer Java-Software, die auf dem Raspberry Pi installiert ist. Das Spiel kann entweder von einer einzelnen Person oder von zwei Spielern an zwei Displays gespielt werden. Jedes Display verfügt über einen Lautsprecher, über den Sounds abgespielt werden können.

Das Ziel von "Blackout" ist es, Wissen zum Thema Energie zu vermitteln und gleichzeitig eine unterhaltsame Spielerfahrung zu bieten. Das Spiel folgt einem bestimmten Narrativ, das von "Kulana", einem Mädchen, und ihren Kolleginnen und Kollegen in einem Haus auf dem Land handelt. Kulana hat viel Wissen über Energie und Klima und lebt nachhaltig, indem sie versucht, möglichst viel Strom selbst zu produzieren und zusätzlichen Strom aus erneuerbaren Quellen zu beziehen.

Das Quiz im Spiel besteht aus fünf Fragen zum Thema Energie. Zu jeder Frage gibt es vier Antwortmöglichkeiten im Multiple-Choice-Format, von denen ein bis vier richtig sein können. Nach jeder Frage wird die richtige Antwort eingeblendet, die erreichten Punkte und die Lösung werden erklärt.

Am Ende jeder Quizrunde wird den Spieler:innen ein QR-Code angezeigt, über den sie auf eine Webseite gelangen können. Diese Webseite bietet eine detaillierte Auswertung der Quizrunde, einschließlich der Punktzahl, aller gestellten Fragen, der gegebenen Antworten und der richtigen Antworten.

"Blackout" wurde entwickelt, um den Besuch eines Energie-Museums zu gamifizieren und den Spielern eine zusätzliche Motivation zu bieten, die Exponate des Museums zu erkunden und zu bearbeiten. Es soll eine spannende Spielerfahrung bieten und gleichzeitig Wissen über Energie vermitteln.

Aufgabenstellung

1.1.2. Wesentliche Features

- Quizspiel: "Blackout" ist ein interaktives Quizspiel, das den Spielern Fragen zum Thema Energie stellt. Die Fragen sind im Multiple-Choice-Format gestaltet, und die Spieler müssen die richtigen Antworten auswählen.
- Team-Modus: Das Spiel bietet einen Team-Modus, in dem zwei Spieler zusammenarbeiten müssen, um die Fragen zu beantworten. Spieler 1 sieht nur die Frage, während Spieler 2 nur die Antwortmöglichkeiten sieht. Die Spieler müssen miteinander kommunizieren, um die richtige Antwort zu finden.
- 1:1-Modus: Im 1:1-Modus treten zwei Spieler gegeneinander an. Beide Spieler sehen sowohl die Frage als auch die Antwortmöglichkeiten auf ihren eigenen Displays. Am Ende vergleichen sie ihre Punktzahlen, um zu sehen, wer mehr Punkte erreicht hat.
- Einzel-Modus: Im Einzel-Modus kann das Quiz von einer einzelnen Person alleine gespielt werden. Hierbei geht es darum, die Fragen möglichst korrekt zu beantworten und eine hohe Punktzahl zu erzielen.
- Narrativ von "Kulana": Das Spiel verwendet das Narrativ von "Kulana", einem Mädchen, das mit ihren Kolleginnen und Kollegen in einem nachhaltigen Haus auf dem Land lebt. Kulana hat viel Wissen über Energie und Klima. Das Narrativ dient dazu, den Kontext des Spiels zu bereichern und den Spielern eine Verbindung zur Hauptfigur herzustellen.
- Rückmeldung und Auswertung: Nach jeder Quizrunde erhalten die Spieler eine Rückmeldung über ihre Punktzahl und die richtigen Antworten. Diese Rückmeldung wird durch einen QR-Code angezeigt, über den die Spieler auf eine Webseite gelangen, auf der sie eine detaillierte Auswertung ihrer Quizrunde erhalten.
- Gamification des Museumsbesuchs: "Blackout" wurde entwickelt, um den Besuch eines Energie-Museums zu gamifizieren. Das Spiel soll den Spielern eine zusätzliche Motivation bieten, die Exponate des Museums zu erkunden und zu bearbeiten. Durch das Erzielen hoher Punktzahlen im Quiz können die Spieler ihre Leistung beim Bearbeiten der Exponate verbessern.

1.2. Qualitätsziele

Table 1. Qualitätsziele

Qualitätsziel	Motivation und Beschreibung
Macht Spass und ist interaktiv (Benutzerfreundlichkeit)	Der Besuchende soll am Spiel Spass haben und vom Ausstellungsstück angezogen werden. Auch sollte es möglichst interaktiv sein und auf unterschiedliche Eingabearten setzen (nicht einfach nur ein Bildschirm, denn sonst könnte Blackout auch Zuhause gespielt werden)
Wissen prüfen (Funktionale Eignung)	Blackout soll dem Benutzer:innen aufzeigen, was er aus der Ausstellung mitgenommen hat und das Wissen gamifiziert abfragen.

Qualitätsziel	Motivation und Beschreibung
Interaktion und Spiel funktioniert (Zuverlässigkeit)	Der Benutzer soll ein funktionierendes Spiel vor sich haben, das mehrmals ohne Probleme durchgespielt werden kann. Des Weiteren sollen die Interaktionsmöglichkeiten jederzeit verfügbar sein und so das Erlebnis nicht eingeschränkt werden.
Hält als Ausstellungsstück bestand (Wartbarkeit)	Blackout wird an einer Ausstellung präsentiert, es soll nicht die ersten paar Mal funktionieren, sondern muss dem Dauerbetrieb einer Ausstellung standhalten. Wenn doch etwas Defekt ist, soll es möglichst einfach ersetzt werden können.

Qualitätsziele

1.3. Stakeholders

Stakeholders

2. Randbedingungen

Die Randbedingungen umfassen die technische, organisatorische und politische Randbedingung des Projektes. Diese sind unten nach ihren Randbedingungen und Erläuterungen aufgelistet.

2.1. Politische Randbedingungen

Table 2. politische Randbedingungen

ID	Einschränkung
E06	Das Spiel muss haptisch sein.
E04	Das Spiel muss interaktiv sein.
E05	Das Spiel muss mehrspielerfähig sein.
E08	Das Spiel muss transportierbar sein.
E03	Das Spiel muss unabhängig von einer Internetverbindung funktionieren.
E07	Das Spielergebnis muss vergleichbar, sprich kompetitiv sein.
E01	Die Sensoren und Aktoren müssen über einen Raspberry Pi verbunden betrieben werden.
E02	Die Software muss in Java geschrieben sein.

Politische Einschränkungen

2.2. Technische Randbedingungen

Table 3. Technische Randbedingungen

Randbedingung	**Erläuterung**
Hardware	Das Spiel Blackout soll auf einem Raspberry Pi 4 laufen. Dieser Einplatinencomputer bringt Begrenzungen bezüglich der Leistungsfähigkeit sowie der Ausgabemöglichkeiten.
Raspbian (Debian basierte Distribution)	Die Applikation soll auf Raspbian laufen.
Java 17	Die Applikation soll mit Java 17 entwickelt werden
Pi4J	Die Applikation soll mit der Pi4J Library entwickelt werden.
SQLite DB	SQLite soll als DB verwendet werden.
JUnit5	JUnit4 soll als Testframework verwendet werden
Team	Team besteht aus 7 Leuten.
Zeithorizont	Im Sommer 2023 soll die Applikation fertig sein.
Vorgehensmodell	Iterative and incremental development
Entwicklungswerkzeuge	Intellij IDEA

Randbedingung	Erläuterung
Versionsverwaltung	GIT
Architekturdokumentation	arc42
Kodierrichtlinien	Siehe Coding Conventions
Spiel	Die Spiele sollen nicht nur am Computer gespielt werden können, sondern haptische Elemente enthalten.
Spiel	Das Spiel soll transportabel sein.
Verbindung	Das Spiel soll auch ohne Internet funktionieren.
Qualitätsanforderung	Das Spiel soll weitere erforderliche nicht-funktionalen Anforderungen umsetzen: Robustheit, Wartbarkeit, Fehlertoleranz.
Dokumentation	Die Übergabe soll geplant und das Spiel entsprechend dokumentiert werden. kurze Spielanleitung. Anleitung zur Inbetriebnahme und Aufbewahrung
Tests	Funktionierende, dokumentierte und relevante Tests (sowohl automatisierte als auch manuelle Tests)
Source Code	Source Code (öffentliches git repository)

2.3. Randbedingung Kunden

Table 4. Kundenbedingungen und Lieferbedingungen

Erläuterung
Dokumentation zum Betrieb des Produktes (Start, Stop, Troubleshooting, Reset etc.)
Dokumentation zum Code und Devsetup
Dokumentation zum Deployment
Dokumentation zur Hardware
Raspberry PI Image mit lauffähigem Stand
Jar Datei mit StartScript
Dateien für den 3D-Druck
Produkt soll weiterentwickelt werden
Fragen sollen ohne dem Neubuilden angepasst werden

3. Kontextabgrenzung

In der Kontextabgrenzung werden die Grenzen zu potenziellen Nachbarsysteme oder Benutzer angeschaut. Dabei werden externe Schnittstellen aufgezeigt und die Verantwortlichkeit von Blackout bzw. dessen Scope genau definiert.

3.1. Fachlicher Kontext

Beim fachlichen Kontext werden die alle Kommunikationsbeziehungen mit dem System festgelegt. Sowie die Ein- und Ausgabedaten oder Schnittellen erklärt. Blackout interagiert nicht mit externen Schnittstellen, sondern nur mit menschlichen Benutzer.

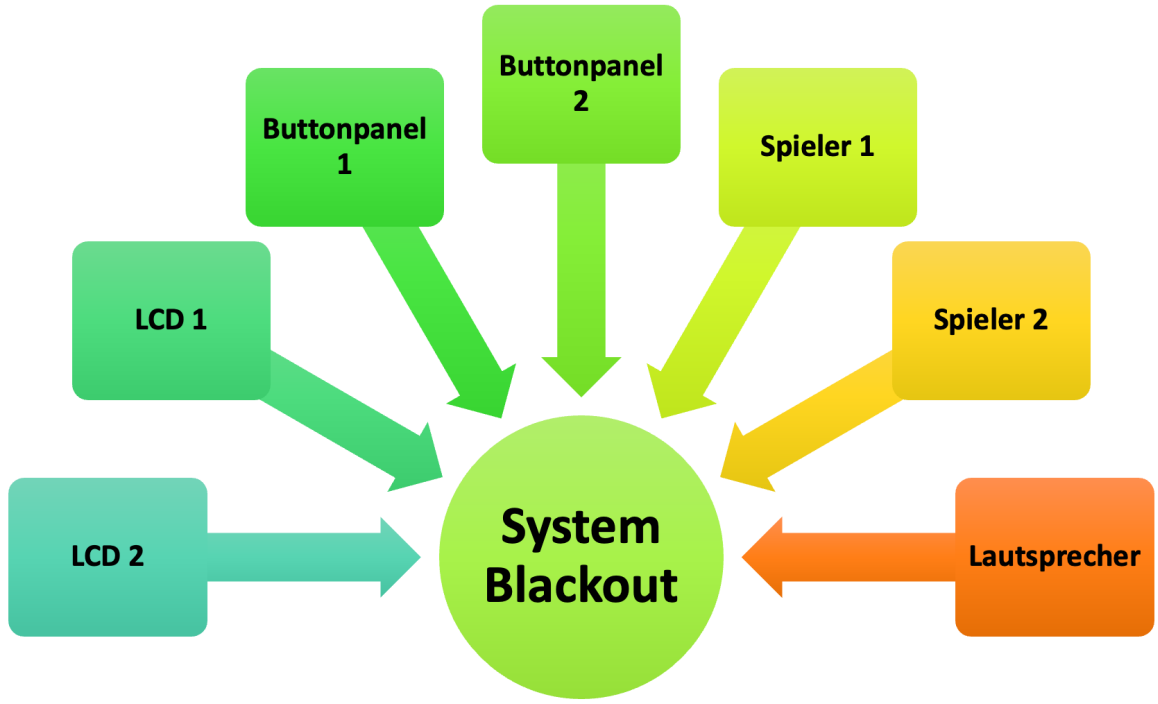


Figure 1. System-Kontextdiagramm

3.2. Spieler (Benutzer)

Blackout kann als alleine oder zu zweit gespielt werden. Diese Spieler interagieren als Einzige mit dem System Blackout.

3.3. Ein- und Ausgabearten von Blackout

Table 5. Ein- und Ausgabearten von Blackout

Kommunikationsart	Eingabe	Ausgabe
Environment Geräusche (Spiel)	Zustandswechsel	Ton und Visuelle Darstellung

Context Geräusche (Spiel)	Zustandswechsel	Ton und Visuelle Darstellung
Blackout-Frage	Buttoneingabe	Ton und Visuelle Darstellung
Buttonzustand	-	Visuelle Darstellung
Bild	-	Visuelle Darstellung

3.4. Technischer Kontext

Technische Schnittstellen zwischen Blackout und seiner Umwelt werden im technischen Kontext angeschaut. Zusätzlich werden die technischen Schnittstellen gemappt auf die fachlichen Schnittstellen und deren technischen Spezifikationen angeschaut.

Systemidee

Das Spiel Blackout soll auf einem Raspberry Pi laufen und auf einem, bzw. zwei Displays angezeigt werden (abhängig von der Anzahl Spieler*innen). Gesteuert wird es über 4 Hardware-Buttons. Damit sollen die Spieler*innen den Spielmodus auswählen sowie die Fragen beantworten. Die Audioausgabe soll für den Mehrspielermodus via Lautsprecher und Kopfhörer und für den Einzelmodus nur über Kopfhörer erfolgen. Am Schluss soll ein QR-Code angezeigt werden, der von den Spieler*innen gescannt werden kann. Damit erhalten sie die erreichte Punktzahl auch auf ihrem Handy.

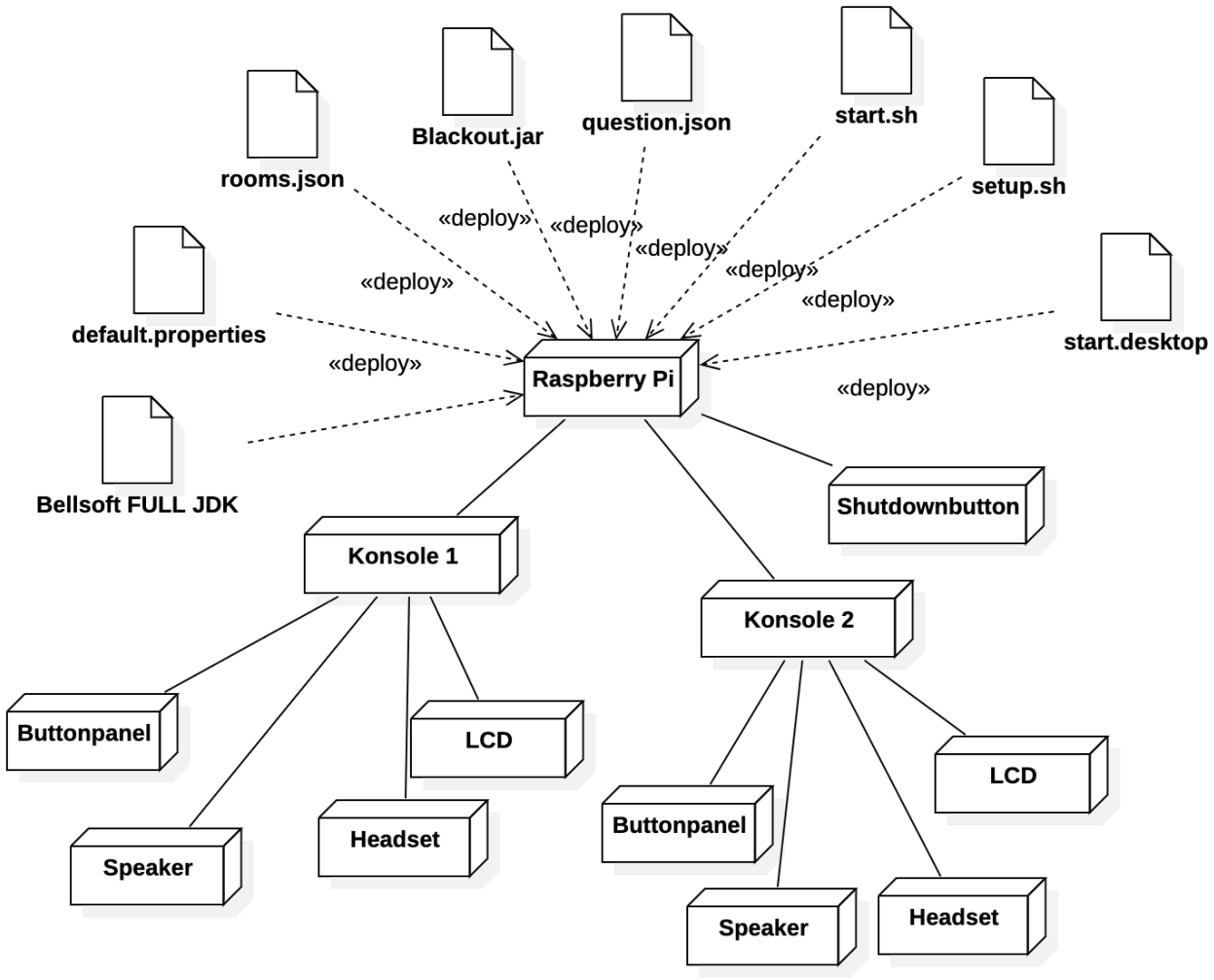


Figure 2. Deployment Diagramm

Table 6. Technische Schnittstellen

Schnittstelle	Beschreibung	Mapping
Raspberry PI	"Blackout" läuft auf einem Raspberry Pi, einer Einplatinen-Computereinheit. Der Raspberry Pi dient als Hardwareplattform, auf der die Java-Software des Spiels ausgeführt wird.	Ton und Visuelle Darstellung
Displays	Das Spiel wird auf ein oder zwei Displays angezeigt, je nachdem, ob es von einer einzelnen Person oder von zwei Spielern gespielt wird. Die Displays können über HDMI an den Raspberry Pi angeschlossen werden und zeigen die Fragen, Antwortmöglichkeiten und andere Spielinformationen an.	Visuelle Darstellung

Schnittstelle	Beschreibung	Mapping
Lautsprecher	Jedes Display verfügt über einen Lautsprecher, über den Sounds während des Spiels abgespielt werden können. Dies ermöglicht beispielsweise das Abspielen von Hintergrundmusik, Soundeffekten oder der Stimme von "Kulana" für zusätzliche Immersion.	Ton
Hardware-Buttons	"Blackout" wird über Hardware-Buttons gesteuert. Es gibt insgesamt 5 Hardware-Buttons auf jeder Seite, die von den Spielern verwendet werden, um die Antworten auszuwählen oder andere Aktionen im Spiel auszuführen.	Ton und Visuelle Darstellung
Webseiten-Auswertung	Am Ende jeder Quizrunde wird den Spielern ein QR-Code angezeigt. Über diesen Code gelangen sie auf eine Webseite, auf der eine detaillierte Auswertung der Quizrunde angezeigt wird. Die Kommunikation zwischen dem Spiel und der Webseite erfolgt über das Internet oder eine lokale Netzwerkverbindung.	Visuelle Darstellung

4. Lösungsstrategie

4.1. Erreichen der Qualitätsziele

Table 7. Erreichen von Qualitätszielen

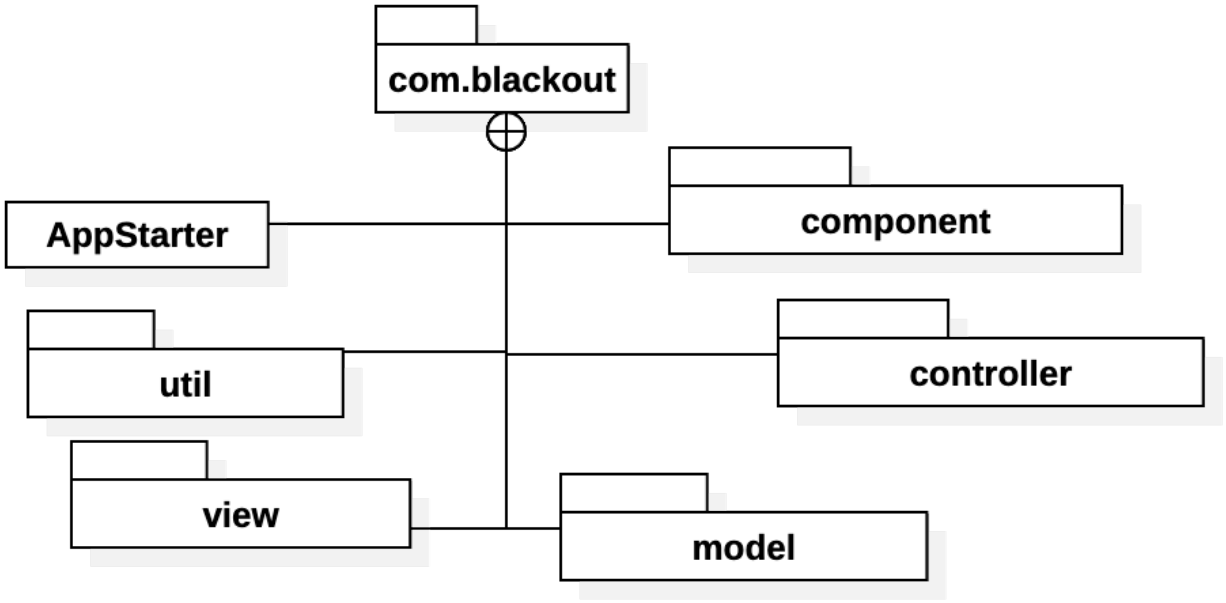
Qualitätsziel	Architekturentscheide für die Erfüllung
Funktionale Eignung	Blackout dient als Spiel für die Gamification des Museumsbesuchs
	Blackout ist Mehrspielerfähig
	Blackout ist sehr interaktiv gestaltet
Zuverlässigkeit	Blackout ist mit bewährten Komponenten gebaut
	Blackout verwendet besonders stabile Komponenten
	Blackout als Softwaresystem ist auf Ausfallsicherheit getrimmt
Wartbarkeit	Blackout verwendet stabile Komponenten die nicht oft ausgetauscht werden müssen
	Blackout verwendet einfach beziehbare Komponenten
	Blackout als Softwaresystem ist erweiterbar und simple konzipiert, dass Wartungen und Erweiterungen möglich sind
Benutzerfreundlichkeit	Blackout hat ein simples Eingabekonzept
	Blackout verwendet ein simples Spielprinzip
	Blackout ist ohne Probleme verständlich

Qualitätsziele

5. Bausteinsicht Level 1

Die Bausteinsicht listet die Bausteine des Systems auf und zeigt deren Abhängigkeiten auf.

Bausteinsicht Level 1 als Klassendiagramm



5.1. Whitebox Gesamtsystem

In diesem Abschnitt wird die grobe Zerlegung des Gesamtsystems genauer betrachtet und die einzelnen Ebenen sowie die dazugehörige Whiteboxen angeschaut.

Begründung

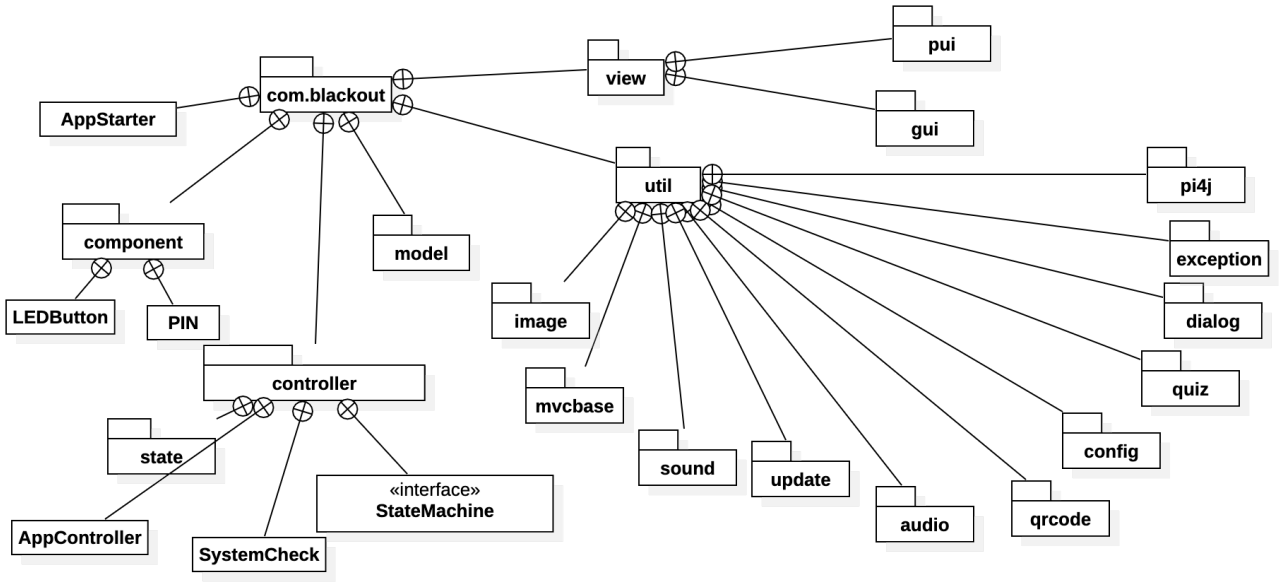
Das System Blackout ist nach der MVC Architektur aufgebaut, diese Architektur definiert somit die Struktur der Software.

Table 8. Bausteine

Subsystem	Kurzbeschreibung
AppStarter	Die Main-Klasse der Applikation.
controller	Enthält alle Controller für die Applikation
model	Enthält alle Modelle der Applikation
view	Enthält alle GUI und PUI der Applikation
component	Enthält alle Komponenten die eine Hardware-Komponenten repräsentieren.
util	Enthält alle Hilfsklassen der Applikation

6. Bausteinsicht Level 2

Bausteinsicht Level 2 als Klassendiagramm



Begründung

Hier wird die zerlegung des Systems detaillierte angeschaut. Sowie die wichtigsten Systeme dokumentiert. Zuvor erklärte Bausteine werden nicht mehr nochmals erwähnt.

Table 9. Bausteine

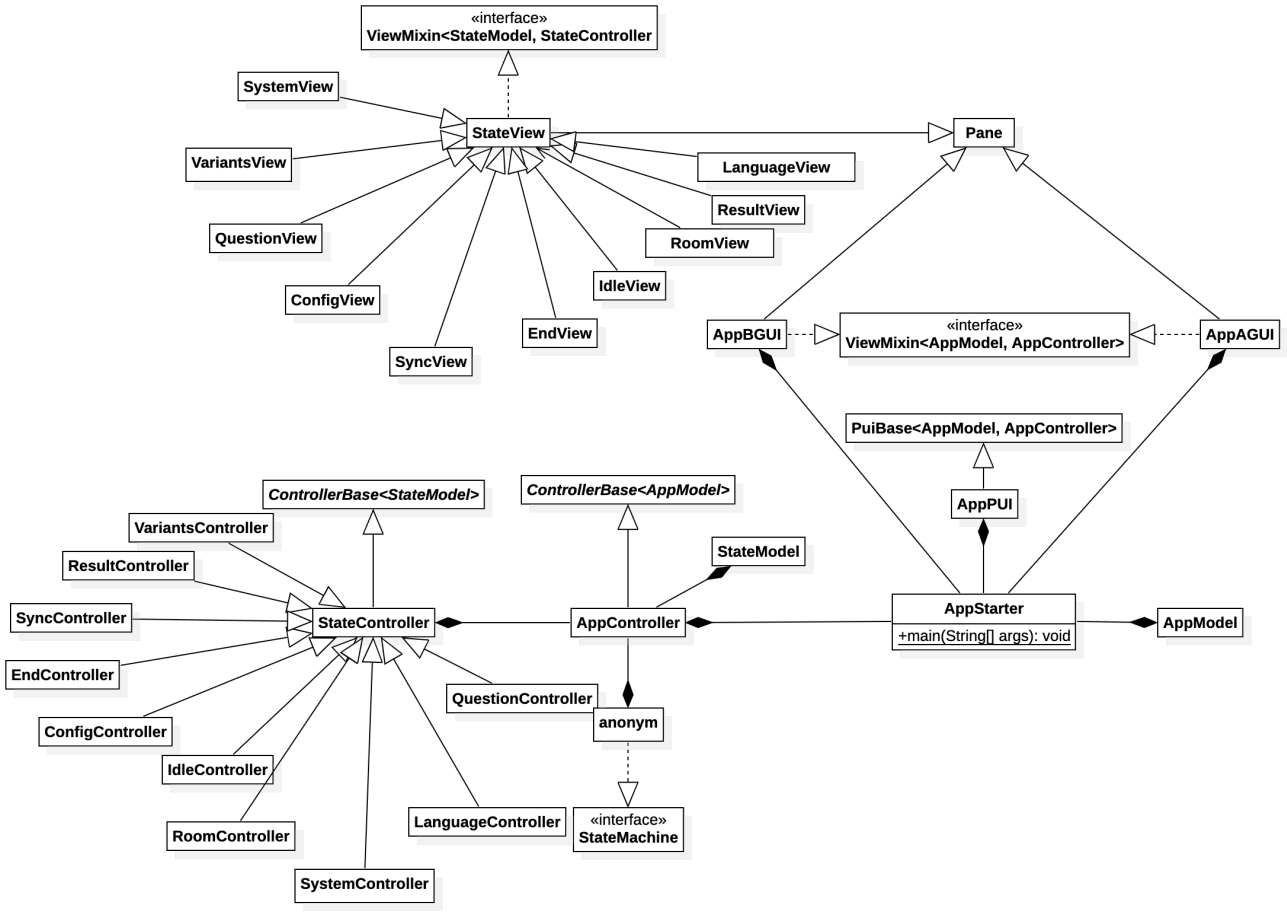
Subsystem	Kurzbeschreibung
LEDButton	Der Komponent welcher der physische LedButton repräsentiert
PIN	Hilfsklasse für die PIN Nummerierung des Raspberry Pi
state	Hier sind alle StateController versorgt
AppController	Der AppController ist der Hauptcontroller der Application
SystemCheck	Die SystemCheck Klasse ist für den SystemCheck der Buttons zuständig
StateMachine	Das Interface StateMachine dient den StateController dazu ihren State zu wechseln.
image	Hier ist die Hilfsklasse für das Laden von Bildern versorgt
mvcbase	Hier sind die Basisklassen und Interfaces für die MVC-Architektur abgelegt
sound	Hier sind die Hilfsklassen für das Abspielen von Sound versorgt
update	Hier ist die Hilfsklasse für das Aktualisieren der Quizfragen versorgt
audio	Hier sind ist die AudioEngine versorgt
qrcode	Hier ist die Hilfsklasse für das Erstellen von QR-Codes versorgt

Subsystem	Kurzbeschreibung
config	Hier ist die Hilfsklasse für das Laden der Config versorgt
quiz	Hier ist die Logik für das Quiz versorgt
dialog	Hier ist die Logik des Dialog-Hilfsklasse versorgt
exception	Eigene Exceptions der Applikation
pi4j	Hilfsklasse für die Pi4j Library
gui	Alle GUIs
pui	Das PUI der Applikation

6.1. MVC

Die Applikation implementiert die MVC Architektur, daher gibt es Klassen für den Controller, View, und Model. Da Blackout eine Statemachine implementiert (Siehe: Kapitel 8 Konzepte) gibt es für einzelnen States auch Klassen für Controller, View und Model.

MVC Architektur als Klassendiagramm



6.2. Audio

Für die Applikation implementierten wir eine eigene Audio-Engine (Siehe: Kapitel 8 Konzepte).

Audio Architektur als Klassendiagramm

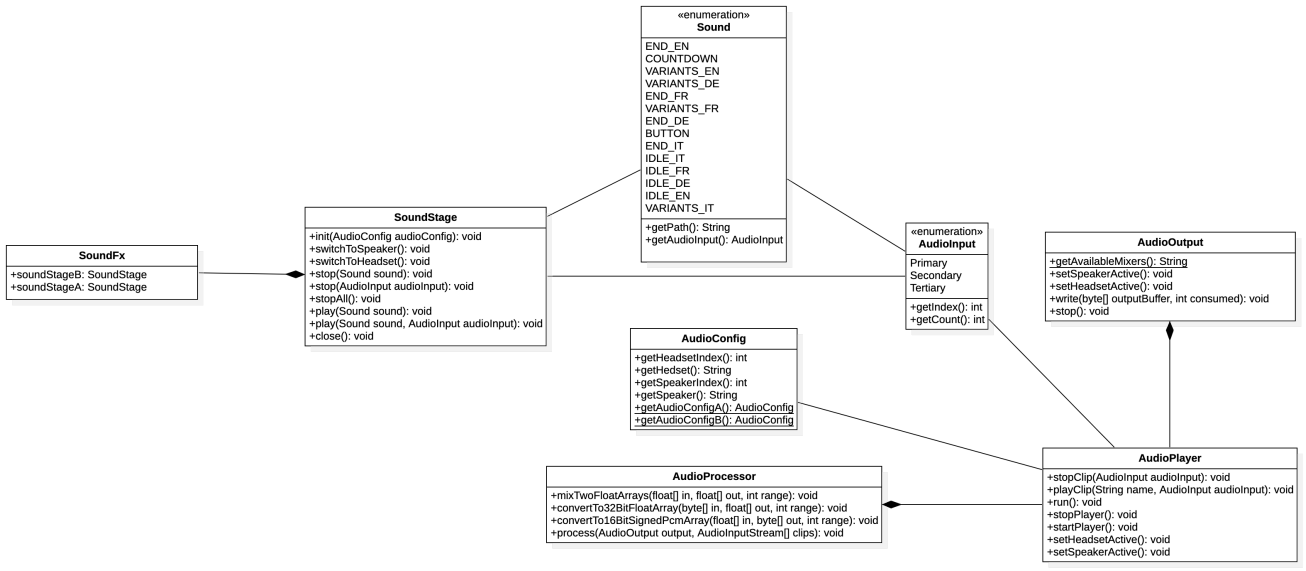


Table 10. Bausteine Audio

Klasse	**Kurzbeschreibung**
Sound	Repräsentiert alle Sound die gespielt werden können.
SoundFx	Managerklasse für die SoundStages
SoundStage	Klasse für das Verwalten des Audios auf einer Konsole
AudioConfig	Config für das Erstellen eines AudioPlayers
AudioProcessor	Klasse für das Mixen, Konvertieren und Schreiben von Audio
AudioPlayer	Klasse für das Spielen des Audios auf einer Konsole
AudioOutput	Klasse für einen Audioausgang

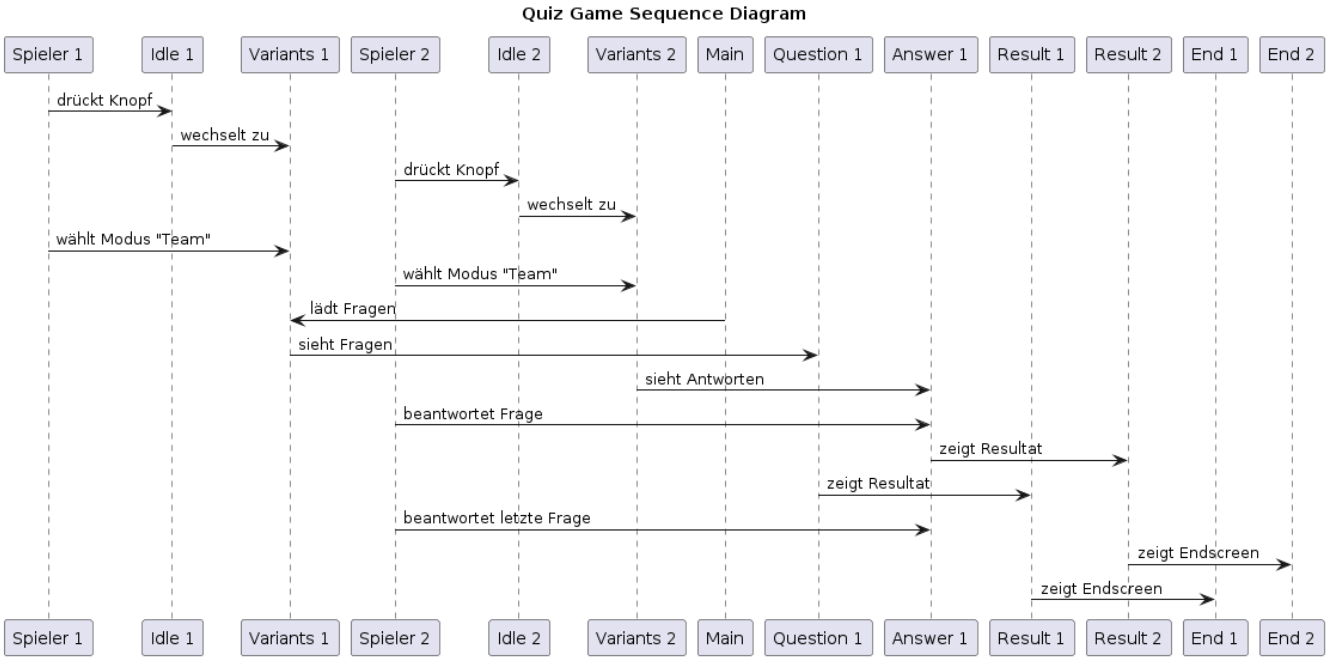
7. Laufzeitsicht

In der Laufzeitsicht werden die konkreten Abläufe und deren Beziehungen zwischen den Bausteinen betrachtet.

7.1. Einfacher Spielablauf

Der einfache Spielablauf zeigt der Ablauf eines Spieles.

Sequenzdiagramm einfacher Spielablauf



7.1.1. Beschreibung

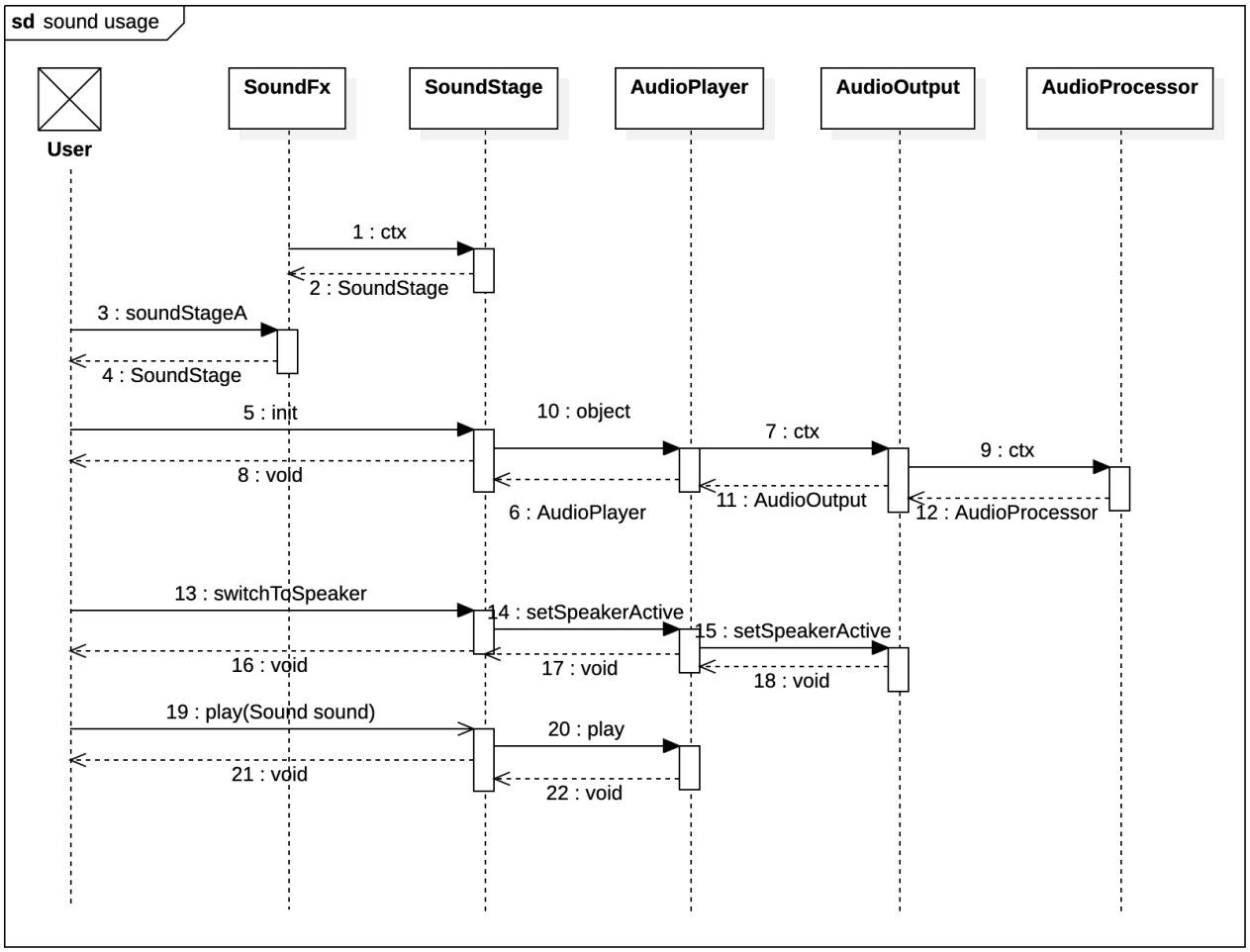
- "Spieler 1" drückt Knopf in "Idle 1"
- "Idle 1" wechselt zu "Variants 1"
- "Spieler 2" drückt Knopf in "Idle 2"
- "Idle 2" wechselt zu "Variants 2"
- "Spieler 1" wählt Modus "Team"
- "Spieler 2" wählt Modus "Team"
- "Main" lädt Fragen "Main"
- "Spieler 1" sieht Fragen "Question 1"
- "Spieler 2" sieht Antworten "Answer 1"
- "Spieler 2" beantwortet Frager "Answer 1"
- "Spieler 1" sieht Resultat "Result 1"
- "Spieler 2" sieht Resultat "Result 2"
- "Spieler 2" beantwortet letzte Frage "Answer 1"

- "Spieler 1" sieht Endscreen "End 1"
- "Spieler 2" sieht Endscreen "End 2"

7.2. Audio Verwendung

Hier wird die Laufzeitansicht beim Verwenden des Audios aufgezeigt.

Sequenzdiagramm Audio Verwendung



7.2.1. Beschreibung

- Zuerst werden die Klassen erstellt beim Start der Applikation
- Danach muss die SoundStage initialisiert werden
- Ab dann kann auf der SoundStage das Audio verwendet werden → siehe bsp. wechseln zu Lautsprecher

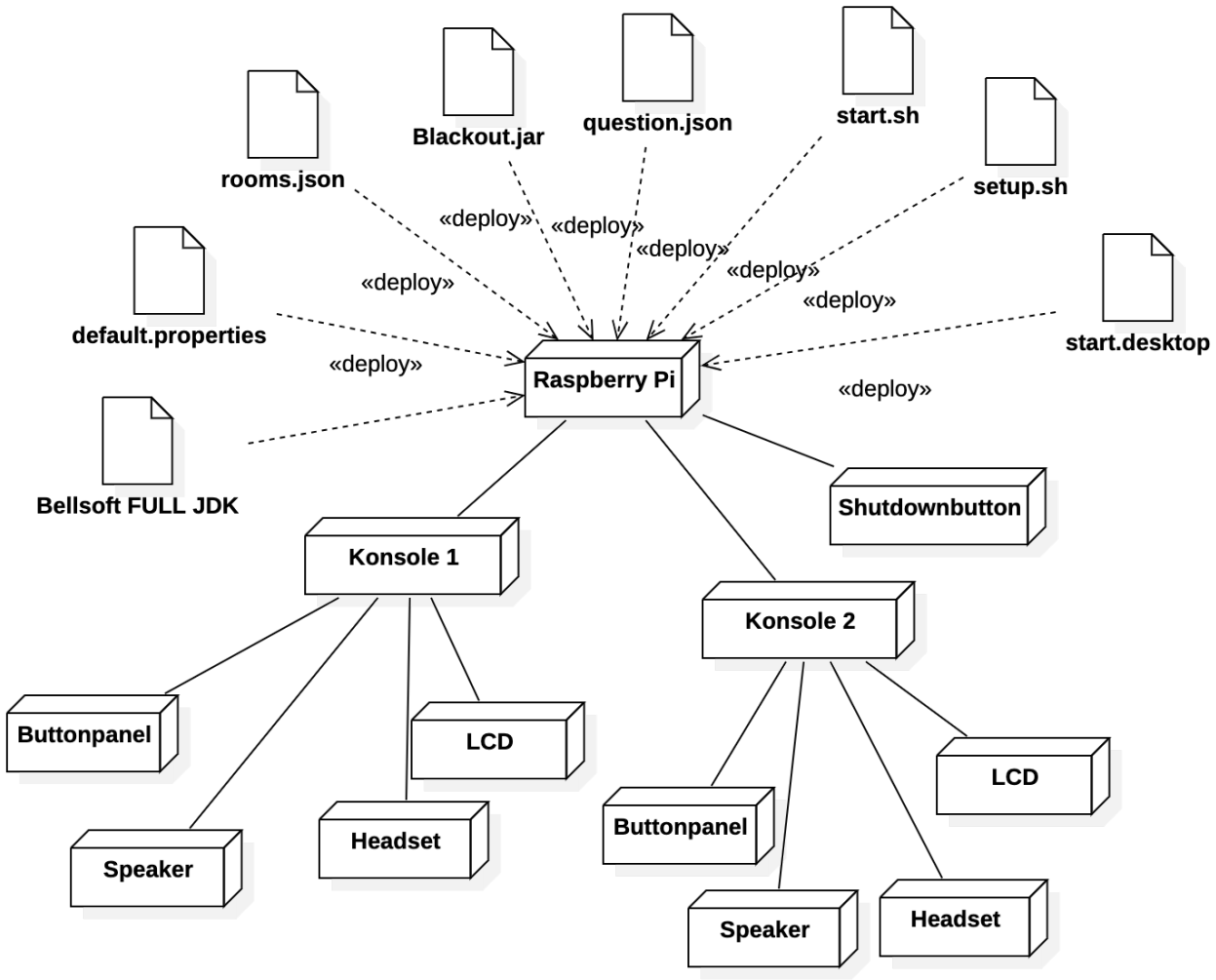
8. Verteilungssicht

In der Verteilungssicht wird die technische Infrastruktur, auf der das System läuft, genauer betrachtet und die Softwarebausteine auf der Infrastruktur.

8.1. Infrastruktur Ebene 1&2

Hier wird die Verteilung des Gesamtsystems beleuchtet.

Deployment Diagram



Begründung

> Das System Blackout läuft auf einen Raspberry Pi, dabei werden die .jar und die Config auf das System deployt. Das System holt, dann selbständig die Datenbank der Fragen. Das System interagiert über die GPIO Schnittstelle des Raspberry PIs mit den Buttons.

Des Weiteren werden folgende Dinge noch mitgeliefert (Siehe Randbedingungen Kunde):

- Dokumentation
- Raspberry PI Image

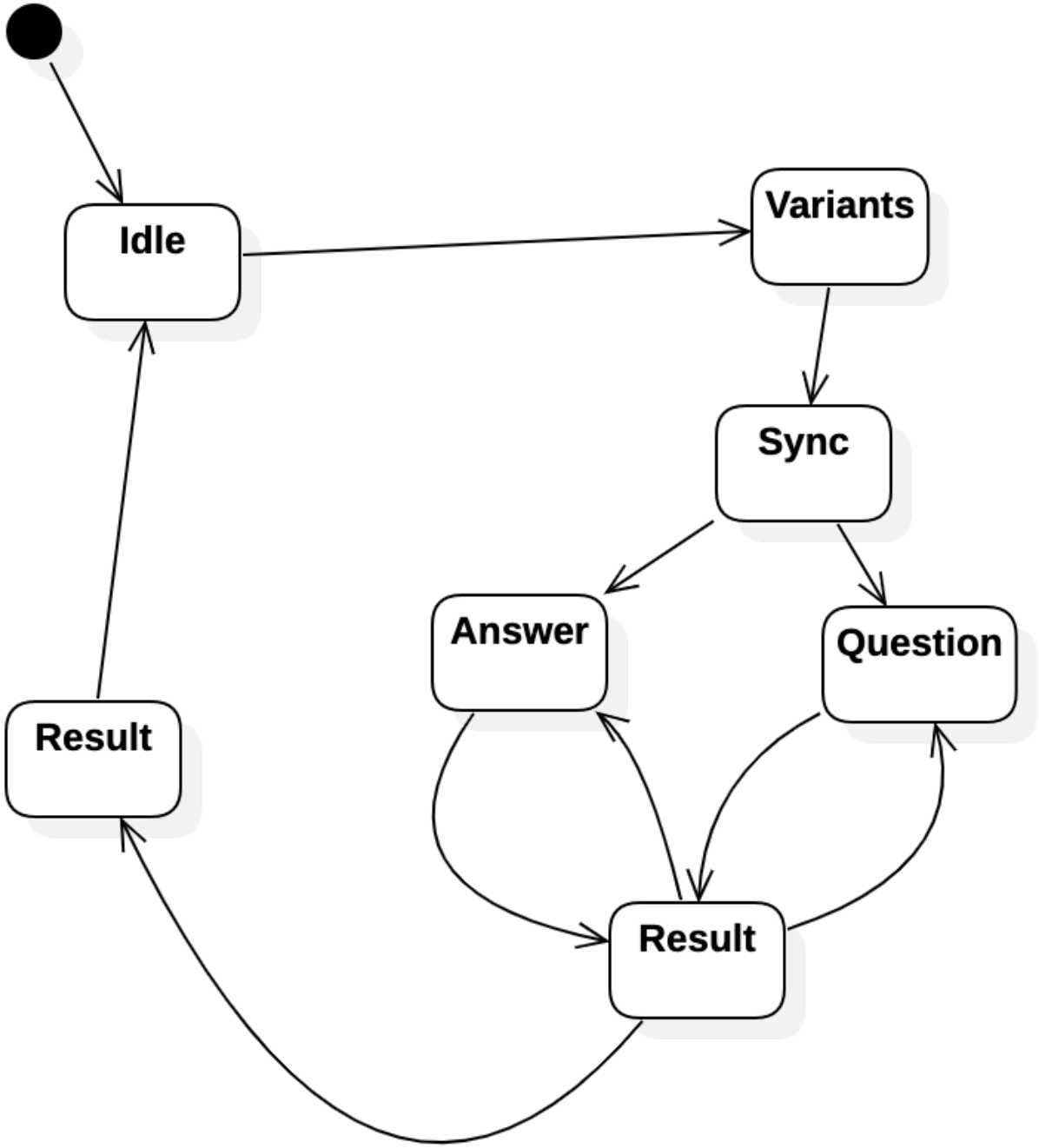
9. Querschnittliche Konzepte

Der Abschnitt Querschnittliche Konzepte beschreibt übergreifende Regelungen und Lösungsansätze.

9.1. States

Das Spiel beinhaltet mehrere Zustände. Von diesen Zuständen aus kann zu anderen oder wieder zurückgewechselt werden. Dadurch dass das Spiel mehrere Aspekte hat, sind diese am besten mit Zuständen repräsentiert. Hier sehen wir alle mögliche State der BlackoutApplikation.

Activitydiagramm



Gründe für die States:

Spielstand speichern: Durch die Verwendung von States kann "Blackout" den Spielstand speichern. Das bedeutet, dass der aktuelle Fortschritt des Spiels, wie beispielsweise die aktuelle Frage, die bereits gegebenen Antworten und die erzielten Punkte, gespeichert werden können. Dadurch können die Spieler das Spiel unterbrechen und später an derselben Stelle wieder fortsetzen, ohne

den Fortschritt zu verlieren.

Spiellogik steuern: States ermöglichen es dem Spiel, die Logik und den Ablauf des Quizspiels zu steuern. Das Spiel kann beispielsweise den aktuellen Zustand des Spiels verfolgen, ob sich die Spieler im Team-Modus oder 1:1-Modus befinden, ob eine Frage gestellt wird oder ob die Antwortzeit abgelaufen ist. Basierend auf diesen Zuständen kann das Spiel die entsprechenden Aktionen ausführen, wie das Anzeigen der nächsten Frage oder die Auswertung der Antworten.

Spielerinteraktion verfolgen: States ermöglichen es dem Spiel, die Interaktion der Spieler zu verfolgen und entsprechend zu reagieren. Das Spiel kann beispielsweise speichern, welche Antwort(en) ein Spieler ausgewählt hat, um dies bei der Bewertung und Punktevergabe zu berücksichtigen. Durch das Verfolgen der Spielerinteraktion kann das Spiel auch die Kommunikation zwischen den Spielern im Team-Modus überprüfen und entsprechende Rückmeldungen oder Hinweise geben.

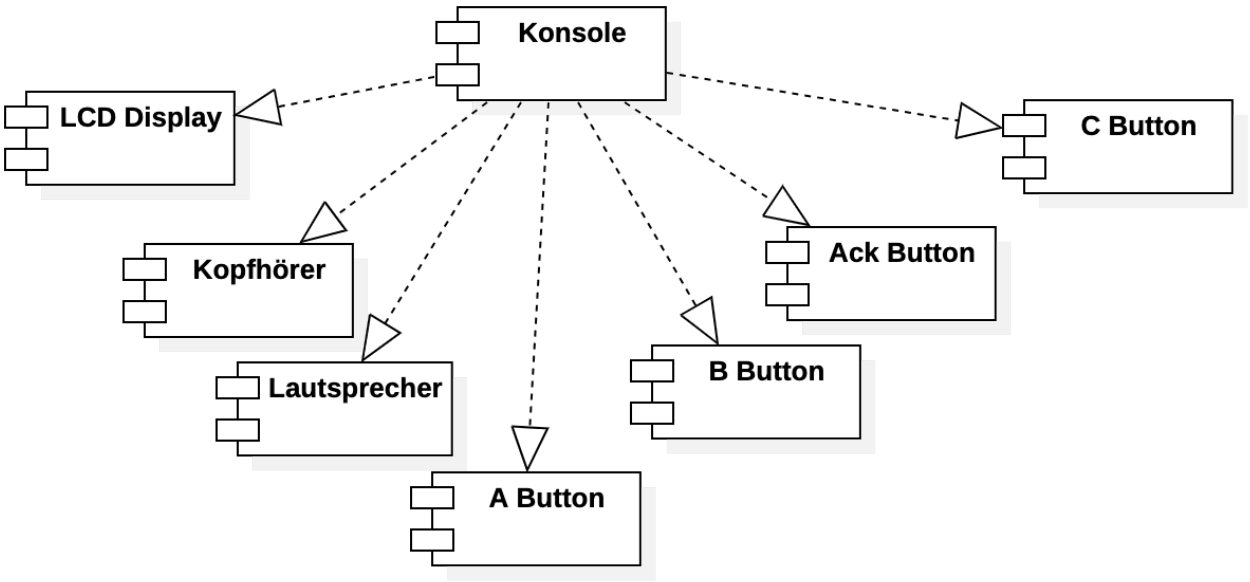
Auswertung und Ergebnisse: States sind wichtig, um die Ergebnisse und Auswertungen der Quizrunden zu speichern. Das Spiel kann den erreichten Punktestand, die richtigen Antworten, die Anzahl der beantworteten Fragen und andere statistische Informationen in den States speichern. Diese Informationen können dann verwendet werden, um den Spielern am Ende einer Runde eine detaillierte Auswertung zu präsentieren oder sie zur weiteren Analyse an eine Auswertungswebseite zu senden.

Durch die Verwendung von States kann "Blackout" also den Spielstand speichern, die Spiellogik steuern, die Spieler-Interaktion verfolgen und Ergebnisse speichern. Dadurch wird eine reibungslose Spielerfahrung ermöglicht und das Spiel kann personalisierte Rückmeldungen und Auswertungen bieten.

9.2. Konsole

Die Spieler haben eine eigene Konsole, mit dieser Konsole interagieren sie mit dem System Blackout. Somit sind die Eingaben und Ausgaben für die einzelnen Spieler sauber abgekapselt.

Konsole als Komponenten diagramm



9.3. MVC Architektur mit dem Projektor Pattern

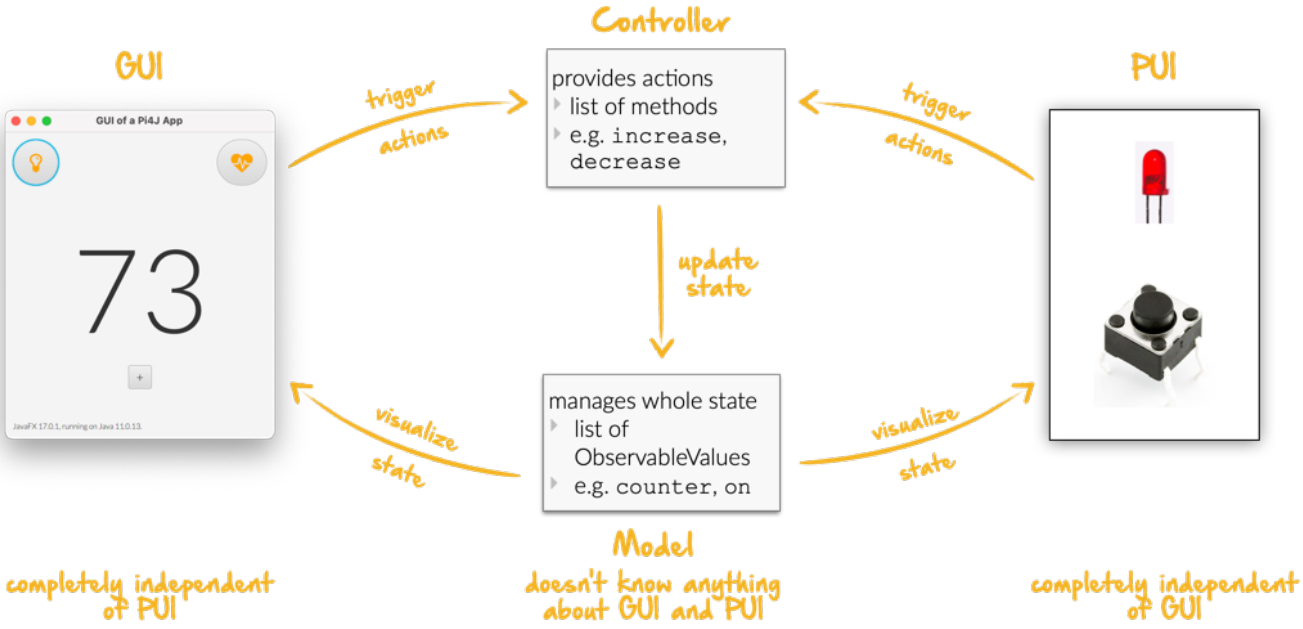
Damit wir eine saubere Architektur haben, haben wir uns für MVC mit dem Projektor entschieden.

Die MVC-Architektur (Model-View-Controller) ist ein Entwurfsmuster für die Softwareentwicklung. Sie besteht aus drei Komponenten: dem Modell, das die Daten und die Geschäftslogik enthält, der Ansicht, die für die Darstellung der Benutzeroberfläche zuständig ist, und dem Controller, der die Kommunikation zwischen Modell und Ansicht verwaltet. line

Das Modell repräsentiert die Daten und die Logik der Anwendung, während die Ansicht die Darstellung der Daten für den Benutzer übernimmt. Der Controller steuert die Interaktionen des Benutzers mit der Anwendung und aktualisiert das Modell und die Ansicht entsprechend.

Die MVC-Architektur ermöglicht eine klare Trennung von Daten, Präsentation und Steuerungslogik, was zu einer besseren Wartbarkeit, Skalierbarkeit und Wiederverwendbarkeit von Code führt. Sie wird häufig in Webanwendungen, Desktop-Anwendungen und anderen Softwareprojekten eingesetzt.

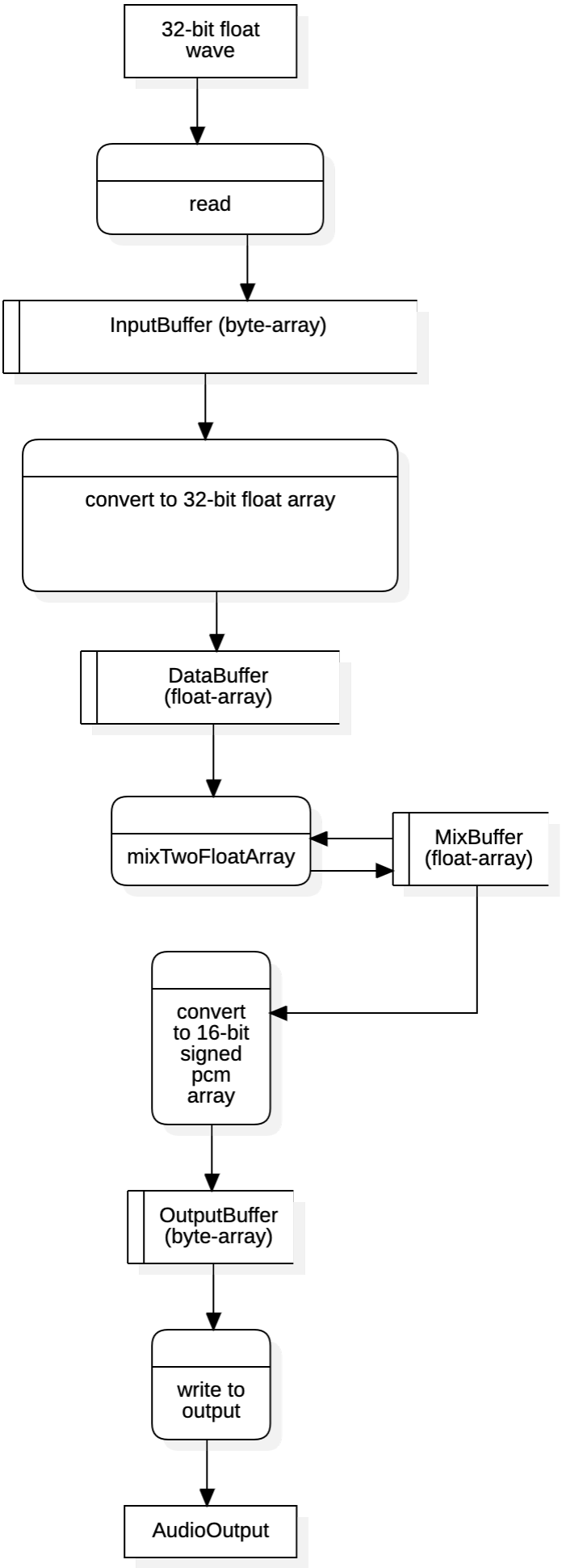
MVC geklaut von Dieter Holz Original



9.4. Audio

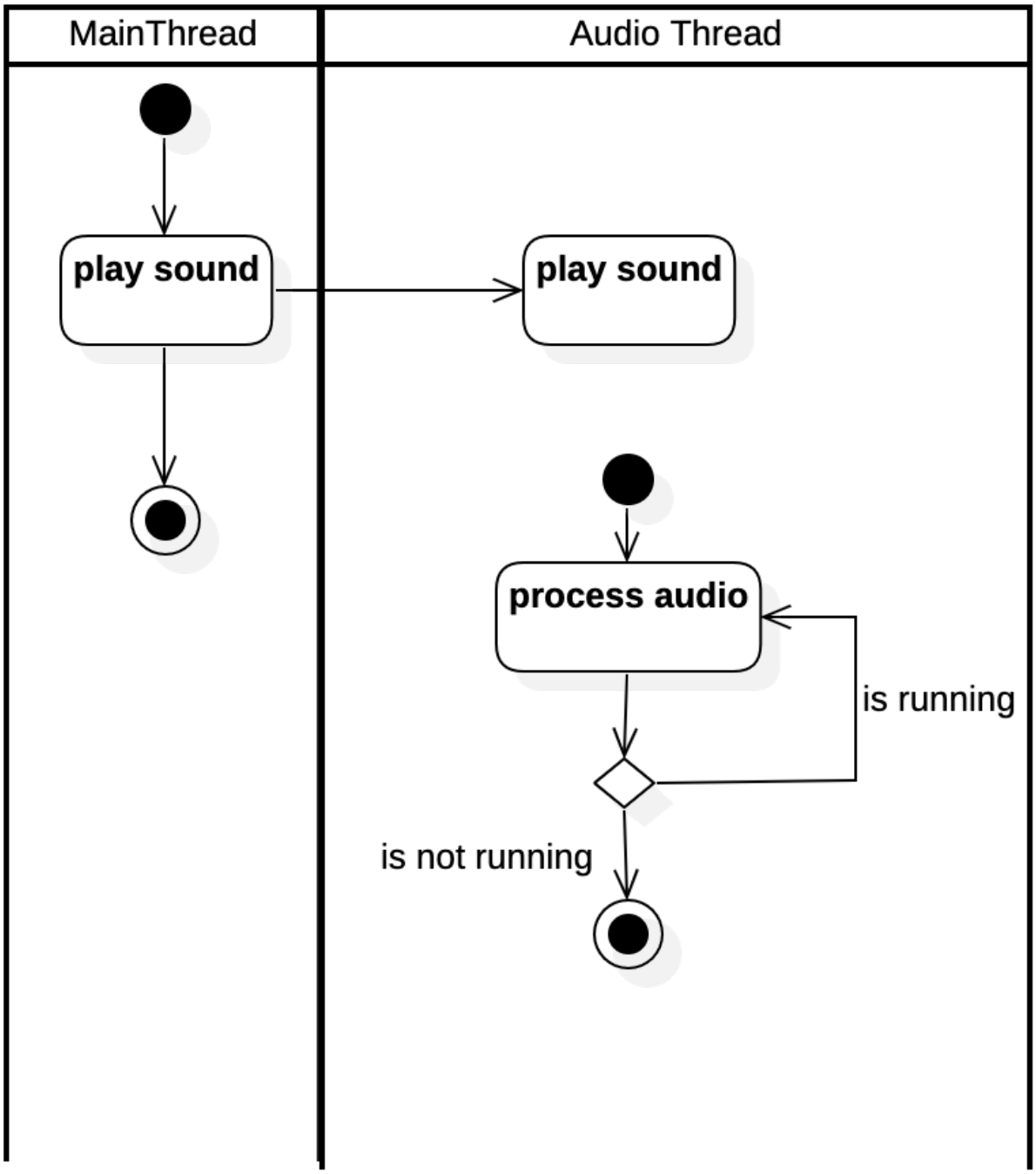
Wir haben die Anforderung für unser Projekt Audio auf total 4 Audiogeräten abzuspielen, jedoch nur auf 2 Audiogeräten gleichzeitig. Das Audio muss nicht synchron sein, sondern auf den 2 aktuellen Audiogeräten spielt unterschiedliches Audio. Den der Spieler hat jeweils eine eigene Konsole, auf der er das Spiel spielen kann, diese Konsole hat 2 Audioausgänge → Lautsprecher und Kopfhörer. Da keine Library diese Anforderung abdeckt, entwickelten wir eine eigene Audio Engine (Wir verwenden java.sound.api als low level Library um mit den Audiogeräten zu interagieren). Unsere Audio Engine kann Wave Dateien mit 44'100 hz und 32-bit float auf den mehreren Audioausgängen gleichzeitig abspielen. Jeweils können 3 Audiodateien gleichzeitig auf drei Kanälen abgespielt werden. Unten ist das Audio Processing als Flowchart ersichtlich

Audio Processing als Flowchart Diagramm


32-bit float wave
read
InputBuffer (byte-array)
convert to 32-bit float array
DataBuffer (float-array)
mixTwoFloatArray
MixBuffer (float-array)
convert to 16-bit signed pcm array
OutputBuffer (byte-array)
write to output
AudioOutput

Damit das Audio ohne Probleme spielt, läuft es auf einem separaten Thread (Wir verwenden java.util.concurrent für das Threading). Hier ist die Interkation zwischen dem Main-Thread und dem AudioThread aufgelistet.

Audio Threading



9.5. Raspberry Pi Konfiguration

Für die Konfiguration des Raspberry Pi haben wir eine config.sh Datei erstellt, welche die Dateien an den richtigen Ort kopiert, sowie die richtige JDK von Bellsoft herunterlädt. Des Weiteren muss, die Taskleiste ausgeblendet werden und im Dateiexplorer, das Öffnen von Scripten ohne Warnungen eingestellt werden. Auch sollte die Scripten mit chmod +x "Dateiname" ausführbar

gemacht werden.

10. Architekturentscheidungen

10.1. Software Entscheidungen

Aufgrund den Randbedingungen Kunde müssen folgende Entscheidungen getroffen werden.

10.1.1. Detaillierte Dokumentation für den Betrieb

Für den betrieb der Software muss eine detaillierte Anleitung geschrieben werden, sodass eine fachunkundige Person das Produkt bedienen kann. Der Inhalt soll folgendes Umfassen:

- Starten von Blackout
- Betrieb
- Stoppen
- Troubleshooting
- Fragen aktualisieren

Dafür müssen wir eine Testperson finden, welche diese Anleitung durchtestet → Planung noch offen.

10.1.2. Raspberry PI Image

Mit PI Gen kann ein Custom Image erstellt werden. Dafür muss eine Dev Anleitung geschrieben werden. Anleitung Todo → Detailliert Dokumentieren und testen

10.1.3. JAR Datei

Damit eine FAT JAR Datei gebaut werden kann, kann das Shade Plugin verwendet werden. Auch muss das Java Module System ausgeschaltet werden. Dafür muss eine JDK mit integrierten JavaFX verwendet werden, weil sonst die App nicht ohne diese Dependency ausgeführt werden kann. Auch kann die JavaFX Abhängigkeit nicht mit eingepackt werden, da nur Plattformspezifische Abhängigkeiten eingebaut werden können bspw. gebaut auf windows nur windows version eingepackt. Daher verwenden wir die JDK von Bellsoft.

10.1.4. Startscript

Für das Startscript kann Bash verwendet werden. Für das automatisierte Starten der Applikation kann im autostart eine .desktop Verknüpfung erstellt werden.

10.1.5. Fragen ohne Neubuilden ändern

Damit Fragen ohne Fachkenntnisse geändert werden können, wird ein NodeJS server entwickelt, der eine Applikation für das Anpassen der Fragen bietet. Des Weiteren ist er für das Hosten der QR-Website zuständig. Die Java App versucht sich mit dem NodeJS server zu verbinden, um dann die aktuelleste Version herunterzuladen. Sonst wird als Fallback auf die Lokale Version zurückgegriffen. Auch soll die Java App die Möglichkeit bieten, dass die URL zum NodeJS Server

angepasst werden kann, falls er sich ändert (Mit Kunde abgesprochen). TODO → Zugang absichern

10.1.6. Troubleshooting

Damit ein einfaches Troubleshooting durchgeführt werden kann, soll die Java APP ein Testmodus haben. Dieser Testmodus testet die Buttoneingabe, die LED der Buttons und den Ton. Soll jeweils beim Start angezeigt werden oder über ein Menu aufgerufen werden (Mit Kunde abgesprochen).

10.2. Hardware Entscheidungen

Die Hardware Entscheidungen, zeigen auf welche Entscheidungen im Bereich Hardware getroffen worden sind.

10.2.1. Nicht einrastende Buttons

Damit immer der Ursprungszustand des Spiels automatisch wiederhergestellt werden kann und kein Primeo Mitarbeit:innen das Spiel in den Ursprungszustand zurücksetzen muss, haben wir uns für nicht einrastende Buttons entschieden. So gehen diese immer wieder in den Standardzustand zurück.

Figure 3. Nicht einrastender Button

Das spezielle an diesen Buttons ist das sie einen grünen LED Ring haben. Wir zeigen damit auf welche Buttons ausgewählt sind und so eine einfache Zuordnung zu den Antworten existiert.

10.2.2. GPIO Raspberry PI

Damit die Buttons am Raspberry Pi angeschlossen und die integrieren LED angesteuert werden können, müssen sie an den GPIO Pins angeschlossen werden. Damit diese Übersichtlich geordnet sind und einfach angeschlossen werden können, haben wir ein Button Expansion Board gebaut. Darauf sind Schraubklemmen angelötet und mit den Pins des Raspberry Pi verbunden. Dies hat der Vorteil dass bei einem Defekt die Kabel einfach ausgetauscht werden können. Damit die Kabel der Buttons übersichtlich geordnet sind, haben wir ein Flachbandkabel eingesetzt. Nachfolgend wird mit dem Blockdiagramm und Schaltplan der Aufbau gezeigt.

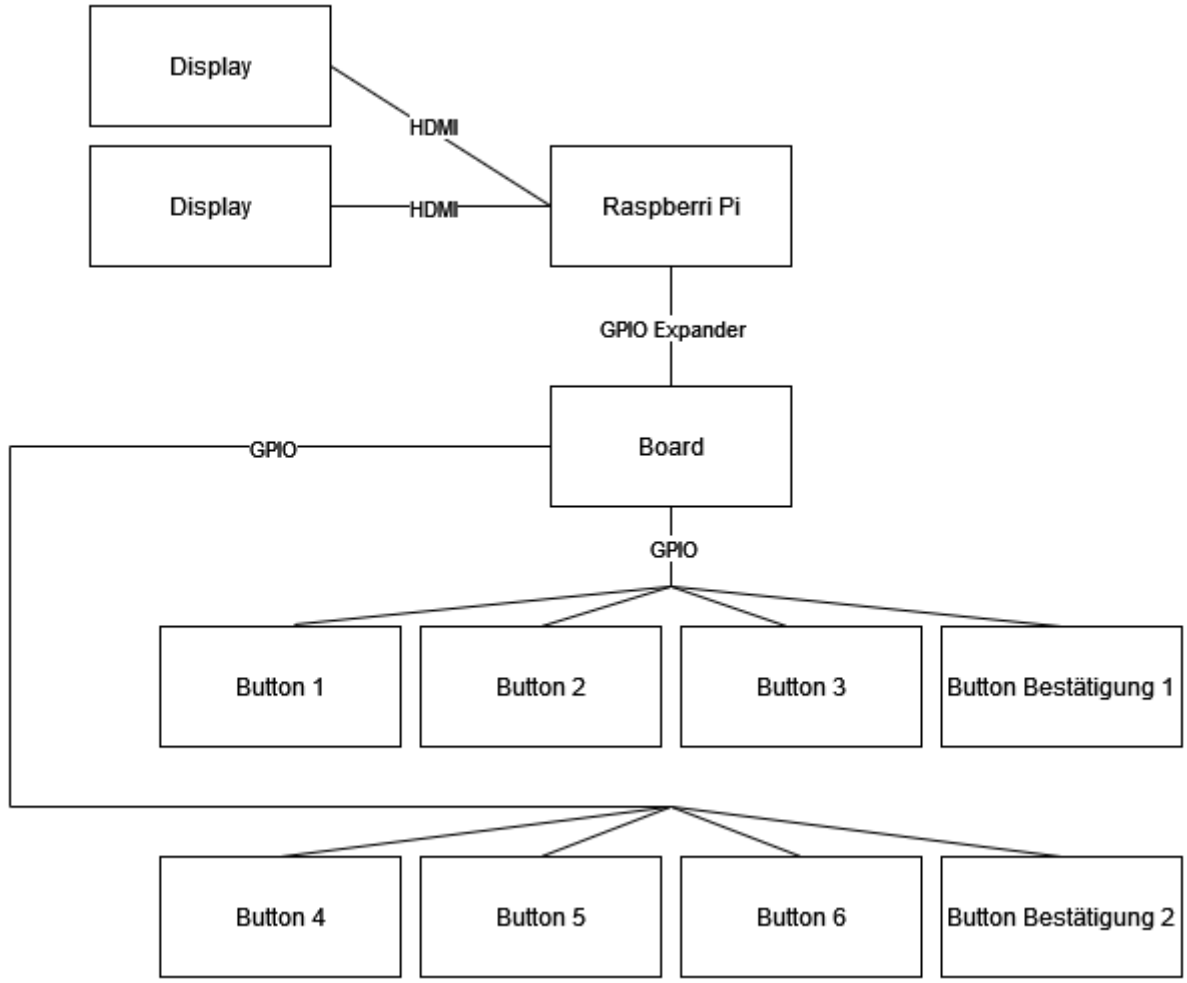
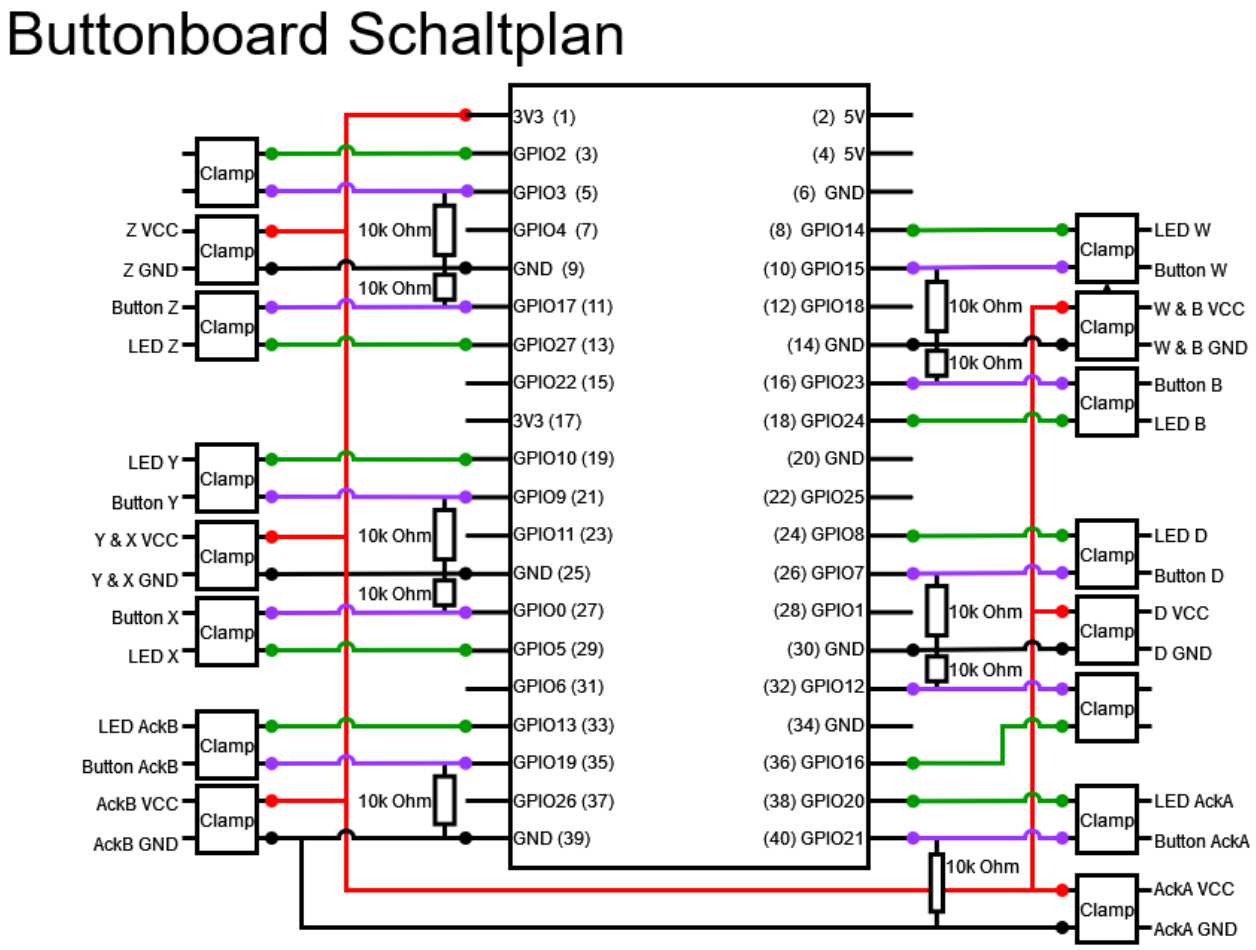


Figure 4. Blockdiagramm



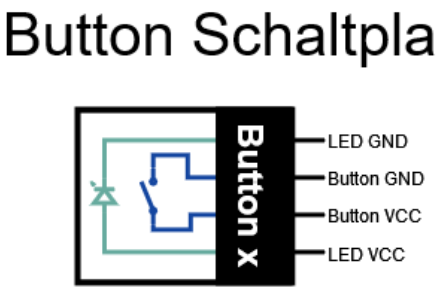


Figure 5. Expansion Board

10.2.3. 3D Druck nicht verfügbar

Da der Stand für den 3D Druck zu gross ist, soll auf einen anderen Werkstoff ausgewichen werden. Das Gehäuse besteht aus Holz und Plexiglas und wird mit Nägel, Schrauben und Holzleim zusammengehalten. Der Oberbau wird aus Schrauben und Heissleim zusammengesetzt. Mit dem Gehäuse aus Holz wird sichergestellt, dass das Produkt sicher vor Beschädigungen ist und Teile einfach ausgetauscht werden können.

11. Qualitätsanforderungen

Im Abschnitt Qualitätsanforderungen werden die Qualitätsziele weiter verfeinert. Die heruntergebrochene Anforderungen werden in ihren Szenarien weiter beleuchtet.

11.1. Qualitätsbaum

Der Qualitätsbaum stellt die Verfeinerung der Qualitätszielen dar.

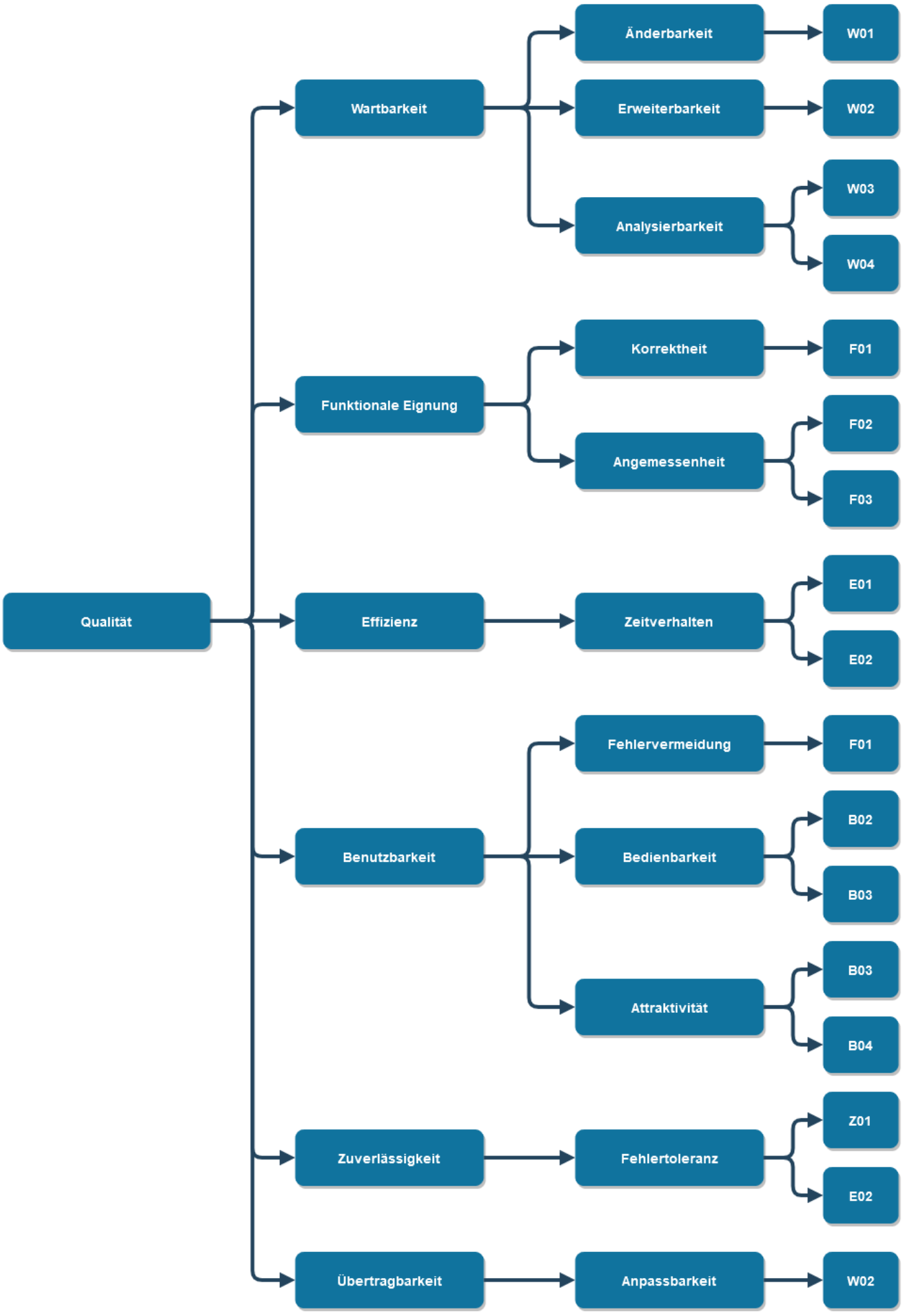


Figure 6. Qualitätsbaum

11.2. Qualitätsszenarien

Bei den Qualitätsszenarien wird genauer auf die einzelnen Szenarien eingegangen.

Table 11. Qualitätsszenarien

ID	Szenario
W01	*Ein Entwickler kann seine Ideen ohne grosse Änderung und ohne Übersetzung des vorhandenen Codes in bestehende Strategien integrieren.*
W02	*Komplexere Fragen können leicht implementiert und in die Lösung integriert werden. Wenn ein Java-Programmierer unseren Fragenkatalog abändern möchte, kann er die Frage ohne grossen Aufwand einfügen.*
W03	*Jemand mit Grundkenntnissen in UML und dem Verständnis wie ein Quiz Spiel funktioniert, möchte einen Einstieg in die Architektur von unserem Spiel. Lösungsstrategie und Entwurf erschliessen sich ihr oder ihm innerhalb von 30 Minuten.*
W04	*Ein Software-Architekt, der das Spiel anwenden möchte, sucht zu einem beliebigen Thema einen konkreten Beispielinhalt und findet ihn unverzüglich in der Dokumentation.*
F01	*Wenn einen Fehler auftritt, soll der Benutzer möglichst wenig davon mitbekommen und das System soll sich wieder fangen können oder mit Ein- und Ausschalten wieder in einen fehlerlosen Zustand zurückgesetzt werden.*
F02	*Blackout wechselt bei Untätigkeit im Hauptmenü in den Wartezustand.*
F03	*Blackout reagiert auf die Buttons und tätigt die entsprechende Auswahl.*
E01	*Während eines Spiels reagiert Blackout auf die Buttons innerhalb von zwei Sekunden mit der Auswahl.*
E02	*Falls keine Eingabe mehr kommt kehrt Blackout nach einer bestimmten Zeit wieder in den Idle Zustand. Diese Zeit ist einstellbar nach den Wünschen von Primeo.*
B02	*Das Design der Bedienungskomponenten / UI wird so gestaltet, dass es zusammen mit der Spielanleitung ohne weitere Erklärungen verstanden wird.*
B03	*Das System lässt sich einfach Konfigurieren und hochfahren. Die Fragen können einfach eingelesen werden.*
B04	*Nach dem Spielen gibt das System eine Rückmeldung bei der ersichtlich ist wo man das Wissen aufarbeiten kann.*
Z01	*Blackout soll einfache Eingabe und Ausgabemöglichkeiten bieten, sodass die Besucher:innen ohne Vorkenntnisse bedienen können.*

12. Risiken und technische Schulden

Im Kapitel Risiken und technische Schulden werden die Architekturrisiken und technische Schulden aufgelistet und deren Massnahmen um diese abzubauen, vermeiden oder minimieren.

Table 12. Risiken und technische Schulden Hardware

Risiken/Schulden	Beschreibung	Massnahme
Display ist kaputt	*Das Display funktioniert nach einem Test nicht mehr*	*Prüfen, wo das Problem liegt und evtl. ersetzen*
Gewisse Buttons leuchten	*Gewisse Buttons leuchten, obwohl sie nicht eingeschaltet sind*	*Laut Recherche normal und auch gemessen mit einem Messgerät einfach beim Start deaktivieren mit einem Script*
Elektr. Schaltplan nicht vorhanden	*Der elektronische Schaltplan ist nicht vorhanden.*	*In der Projektwoche gas geben und zusammen im Team integrieren*
Hardware ist noch nicht integriert	*Die Hardware ist nicht integriert.*	*In der Projektwoche gas geben und zusammen im Team integrieren*
3D-Druck klappt nicht	*Das 3D-Modell ist zu gross für den Anbieter*	*Neuer Anbieter suchen oder Modell aus Holz bauen*
Lange Lieferzeiten	*Die Lieferzeiten für das Model ist sehr lang*	*Anrufen, E-Mail oder persönlich abholen*

Table 13. Risiken und technische Schulden Software

Risiken/Schulden	Beschreibung	Massnahme
Softwarearchitektur könnte vereinfacht werden	*Bezüglich der Softwarearchitektur gibt es noch optimierungspotential*	*Zusammensitzen und anschauen und bei genügend Zeit umsetzen*
Codedokumentation	*Siehe Schuldtitel*	*Dokumentieren, Dokumentieren, Dokumentieren …*

13. Glossar

Im Glossar sind Projektspezifische Begriffe genau definiert, sodass alle Beteiligten die gleiche Sprache sprechen und einen zentralen Ort um Begriffsdefinitionen nachzuschlagen.

Table 14. Glossar

Begriff	**Definition**
1vs1	*Spieler:in spielen mindestens zu zweit und gegeneinander.*
Antwortauswahl	*Durch das Betätigen der Knöpfe 1-3.*
Antwortmöglichkeiten	*Antwortmöglichkeiten, welche im Quiz-Spiel vorkommen.*
Anzeigedisplay	*Das Anzeigedisplay dient als Standartausgabe des Spiels.*
Auswählen	*Mittels drücken desButtons die Antworten anwählen.*
Bestätigung	*Durch das Betätigen des Enterknopfs.*
Blackout	*Blackout ist der Name des Spiels, welches das Team Wechselspiel umsetzt.*
Countdown	*3 Sekunden Countdown vor dem Spielstart.*
Cube	*Das ganze physische Spiel inklusive Ober- und Unterbau.*
Einzelmodus	*Spieler:in spielt alleine. Sieht Fragen und Antwortmöglichkeiten.*
Exponate	*Ausstellungsstücke für den Primeo Energie Kosmos.*
Feedbackmodus	*Im Feedbackmodus werden Fragen aus dem besuchten Raum ausgewählt und abgefragt.*
Fragen	*Fragen, welche im Quiz-Spiel vorkommen.*
Gamifizierung	*Integration von gaming-typischen Elementen. Feedback, Highscore System.*
GPIO	*General Purpose Input Output - Pins beim Raspberry Pi.*
Haptische Elemente	*Auswahl- und Eingabebuttons.*
Hardware	*Alle Einzelteile, welche zum Produkt gehören.*
Kulana	*Junge Expertin für Energiefragen namens Kulana, welche die Spielenden durch das Quiz navigiert. Das Narrativ begleitet das gesamte Projekt.*
Lernwelt Energie	*Abteilung des Energie Kosmos' – über Klima- & Energieinteressierte: für Schulen, die Öffentlichkeit, Firmen und Partner.*
Oberbau	*Der obere Teil des Cubes. Kann separat abgehoben werden. Er beinhaltet die Displays und den Ausschaltbutton.*
Primeo	*Energie Kosmos Science und Erlebnis Center für Klima & Energie der Primeo Energie AG.*
QR-Code	*Leitet auf die Webseite mit dem Score weiter.*
Quiz-Spiel	*Frage-Antwort-Spiel.*

Begriff	Definition
Raum	*Örtlichkeit mit Exponaten bei Primeo Energie AG.*
Score/Highscore	*Spielresultat am Ende, welches auf unserer Webseite präsentiert wird.*
Spielmodus	*Auswahl zwischen Einzelmodus und 1vs1-Modus.*
Startbildschirm	*Der erste Screen der erscheint für die Besucher.*
Statistik	*Die Analyse der Antworten die richtig oder falsch beantwortet wurden. Wird über den gesammten Zeitraum gesammelt.*
Team Wechselspiel	*Gruppenname IP12.*
Timeout	*Wenn Spieler:innen die Frage nicht beantworten, wird das Ergebnis nicht angezeigt.*
Unterbau	*Der untere Teil des Cubes. Kann mittels einer Schublade unten geöffnet werden. Er beinhaltet den Raspberry Pi, alle Kabel und Buttons.*
Vorschlagsmodus	*Das Spiel enthält einen Vorschlagsmodus für die Schüler:innen. Im Vorschlagsmodus werden 3 Fragen zufällig ausgewählt.*
Webseite	*Zugang zur Auswertung der Fragen und Eingabe neuer Fragen.*