Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektarbeit

Andreas Müller, Claus Torben Haug, Jan-Dennis Bartels, Marjan Bachtiari

MBC-Ping-Pong

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Andreas Müller, Claus Torben Haug, Jan-Dennis Bartels, Marjan Bachtiari

MBC-Ping-Pong

Projektarbeit eingereicht im Rahmen der Wahlpflichtfach

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Martin Becke

Eingereicht am: 30. November 2016

Andreas Müller, Claus Torben Haug, Jan-Dennis Bartels, Marjan Bachtiari

Thema der Arbeit
MBC-Ping-Pong

Stichworte
Ping-Pong, NodeJS, JavaScript, WebRTC

Kurzzusammenfassung
In diesem Dokument wird das Projekt im MBC-Ping-Pong, das im Rahmen des Wahlpflichtfaches Modernebrowserkommunikation an der HAW-Hamburg erstellt wird, behandelt. Hierbei handelt es sich um eine Pong Clone welcher auf einem zentralen Bildschirm spielbar ist und mnittels WebRTC gesteuert wird. Es wird auf die Architektur, JavaScript, Frontend und Backend eingegangen.
Andreas Müller, Claus Torben Haug, Jan-Dennis Bartels, Marjan Bachtiari

Title of the paper
MBC-Ping-Pong

Keywords
Ping-Pong, NodeJS, JavaScript, WebRTC

Abstract
This document is about the Project MBC-Ping-Pong, which is made for the elective course Modernebrowserkommunikation at HAW-Hamburg. Its about a Pong clone, played on a central screen nd controled via WebRTC. The architecture, JavaScript, frontend and backend are discussed.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Architektur

1.1 Arbeitsablauf zur Bearbeitung eines Issue

Um eine erfolgreiche Zusammenarbeit zu gewährleisten, sind allgemein gültige Regeln nötig. Insbesondere wird festgelegt, wie die einzelnen Arbeitsschritte ablaufen sollten, um ein Issue zu bearbeiten. Zudem werden weiterhin die Zuständigkeiten geregelt.

1.1.1 Verwaltung der zu bearbeitenden Issues

Die zu bearbeitenden Issues werden auf GitHub unter Issues (https://github.com/Transport-Protocol/MBC-Ping-Pong/issues) gepflegt. Um den Verlauf eines Issues darzustellen wird das Kanbanboard von GitHub (https://github.com/Transport-Protocol/MBC-Ping-Pong/projects) genutzt.

1.1.2 Erstellen der zu bearbeitenden Issues

Prinzipiell kann und darf jedes Projektmitglied zu jeder Zeit Issues erstellen. Gerade bei Bugs ist dies ein gewünschtes vorgehen. In der Regel sollten dies jedoch aus Gruppensitzungen hervorgehen und durch den Architekten ausformuliert werden.
Ein Issue besteht aus drei Absätzen:

- **Beschreibung**
  In der Beschreibung wird allgemein auf den Kontext des Issues eingegangen.
- **Anforderung**
  In Anforderung wird die Zielvision dargestellt.
- **Abnahmekriterien**
  In Abnahmekriterien werden alle Punkte aufgeführt, die notwendig sind, um das Issue als erfolgreich bearbeitet anzusehen.

1.1.3 Das Kanbanboard

Das Kanbanboard ist in fünf Abschnitte eingeteilt:

- **Selected for Development**
  Diese Spalte enthält alle Issues, die der Architekt zur Bearbeitung in nächster Zeit ausgewählt hat. Hier enthaltene Issues sind entweder durch den Architekten einem bestimmten Teammitglied zugeordnet. Diese sollten dann auch vorrangig bearbeitet werden. Oder (dies sollte der Normalfall sein) sie sind niemandem zugeordnet, dann kann sich jedes Teammitglied entscheiden, ob er das Issue bearbeitet. Gründe für das direkte zuweisen können unteranderem sein, dass es eine entsprechende vorhergehende Absprache gab, dass der Architekt das Issue speziell einem Bereich zugehörig sieht bzw. eine spezielle Paarung erreichen möchte, oder aber auch, weil ein Issue schon zu lange in SSelected for Development"verweilt. Hat sich ein Teammitglied für ein Issue entschieden, trägt er sich als Bearbeiter ein und zieht es in auf Ïn Development".

- **In Development**
  In dieser Spalte verweilen alle Issues, an denen gerade entwickelt wird. Wenn die Entwicklung an einem Issue abgeschlossen ist, zieht der Bearbeiter das Issue weiter auf "Needs Review".

- **Needs Review**
  Hier verweilen alle Issues, deren Entwicklung abgeschlossen ist, aber noch nicht geprüft wurde, ob die Abnahmebedingungen erfüllt sind. Normalerweise sollte die Abnahme durch den Architekten erfolgen. Issues, die der Architekt bearbeitet hat, muss das Review von einem anderen Teammitglied gemacht werden. Ein Issue bei dem das Review durchgeführt wird, wird in die Spalte Ïn Review"verschoben.

- **In Review**
  Hier sind alle Issues enthalten, die sich gerade im Review befinden. Sind alle Abnahmekriterien erfüllt, und sind durch die Bearbeitung des Issue keine neuen Probleme/Fehler hinzugekommen, wird es in die Spalte "Done"verschoben und das Issue geschlossen. Ist dies nicht der Fall, wird ein Entsprechender Kommentar mit einer möglichst detaillierten Beschreibung des Problems an das Issue angehängt, und es wieder auf Ïn Development"geschoben.

- **Done**
  Diese Spalte enthält alle abgeschlossenen Issues.

1.1.4 Git

Hier sind die Verhaltensweisen für die Nutzung von Git aufgeführt. Alles hier nicht aufgeführte kann von jedem Teammitglied nach eigenen ermessen gehandhabt werden.

- **Branches**
  Für jedes Issue wird ein Branch erstellt, außer es handelt sich um reine Dokumentation (im Ordner Docu). Ein Branchname folgt folgendem Muster: "#<IssueNummer> <KurzerName>". Dadurch lässt sich
- **Commits**
  Commits folgen folgendem Namensschema: "#<IssueNummer> <Beschreibung>".
- **Push und Pull**
  Es sollte möglichst häufig gepusht werden, um einen eventuellen Datenverlust zu vermeiden. Beim Pull sollte mit -rebase"gearbeitet werden, um die Historie möglichst sauber zu halten.
- **Merge und Pullrequest**
  Bevor ein Issue auf "Needs Review"geschoben wird, ist der Master in den Branch zu mergen und ein Pullrequest (https://github.com/Transport-Protocol/MBC-Ping-Pong/pulls) zu erstellen. Derjenige der das Issue reviewt hat, merget den Branch dann mithilfe des Pullrequests in den Master und löscht ihn.

1.2 Meilensteine

In diesem Abschnitt werden die Meilensteine festgelegt. Hierbei wird beschrieben, was wan erreicht sein sollte.

1.2.1 Projekt Aufsetzen

- **Beschreibung**
  Die Grundlegenden für die Entwicklung notwendigen anfangs Infrastrukturen sind aufgesetzt.
- **Kriterien**
  - **NodeJS-Server aufsetzen**
    Der NodeJS Server ist aufgesetzt und stellt eine statische Website zur Verfügung

  - **Docker**
    Eine einheitliche Umgebung wird durch Docker und Docker-Compose ermöglicht.

- **Beendet:** 25.11.2016

1.2.2 Prototyp (Technik)

- **Beschreibung**
  Um die identifizierten technischen Risiken schnellst möglich in den Griff zu bekommen, werden diese möglichst früh bearbeitet. In dem Prototyp (Technik) soll gezeigt werden, das die kritische Technik funktioniert. Dies wird anhand von kleine losgelösten Beispielen, die aber nahe der Zielarchitektur sind gezeigt.

- **Kriterien**
  - **Darstellung**
    Es wird gezeigt, das im Webbrowser eine Flüssige Darstellung möglich ist.
  - **Kollisionserkennung**
    Es wird gezeigt, dass eine Kollisionserkennung erreichbar ist.
  - **Kommunikation mittels WebRTC**
    Architektur bedingt ist die Nutzung von WebRTC unumgänglich. Es ist zu zeigen, dass eine Verbindung von mehreren Handys zum Darstellungsmedium möglich ist.
  - **Steuerung**
    Die Steuerung soll über den Touchscreen geschehen. Es ist zu zeigen, dass es möglich ist, die Position des Fingers auf dem Touchscreen im Browser auszulesen.
  - **Größen der Handys/Tabletts**
    Unterschiedliche Handys und Tabletts haben verschiedene Größen und Formen. Somit ist ein Konzept zu erarbeiten, welches diesem Problem bei der Steuerung gerecht wird.

- **Beendet:** 16.12.2016

1.2.3 Release 1.0 (Zwei Spieler)

- **Beschreibung**
  In diesem ersten Release ist eine Basisversion des Spieles implementiert. Es können zwei Spieler gegeneinander Spielen, indem sie ihre Schläger mit den Handys steuern. Gleichzeitig ist diese Version die minimal Version und enthält alle "MustFeatures.

- **Kriterien**
  - **Schläger**
    Für jeden Spieler existiert ein Schläger, der mit dem Handy Steuer
  - **Ball**
    Es gibt ein Ball, der sich über das Spielfeld bewegt. Kollidiert er mit einem Schläger oder der Wand, an der sich kein Schläger befindet, prallt er davon ab. Es gilt hierbei, dass der Einfallwinkel dem Ausfallwinkel entspricht. Zudem beschleunigt der Ball, wenn er mit einem Schläger kollidiert. Wenn der Ball mit der Wand hinter einem Schläger kollidiert, wir er in die Ausgangsposition versetzt und erhält die Ausgangsgeschwindigkeit.
  - **Punkte**
    Immer wenn der Ball mit der Wand hinter einem Schläger kollidiert, erhält der andere Spieler einen Punkt.
  - **Spielende**
    Das Spiel endet automatisch nach X Spielen (wobei gilt: $X \in \mathbb{N} \wedge X \mod 2 = 1$).
    Das genaue X ist noch zu definieren.
- **Beendet:** 13.01.2017

1.2.4 Release 1.X (Diverse Features)

- **Beschreibung**
  Basierend auf der Version 1.0 wird das Spiel weiterentwickelt. Jedoch sind alle Features die hier bearbeitet werden können "CanFeatures. Daher kann es sein, dass das Release 1.X äquivalent zu dem Release 1.0 ist. Zudem sind alle hier genannten möglichen Features noch nicht näher spezifiziert und auf ihre Machbarkeit geprüft. Es gilt jedoch, dass je Umgesetztes Feature die Versionsnummer im Minorbereich um Eins steigt.
- **mögliche Kriterien**
  - **N Spieler**
    Mehr als 2 Spieler
  - **Zusätzliche Hindernisse**
    Auf dem Spielfeld sind zusätzliche Hindernisse.
  - **Highscore-Liste**
    Es wird eine Highscore-Liste geführt und angezeigt.

  - **TBD**
    To be discussed.

- **Beendet:** 24.02.2017

1.3 Highlevel View

In diesem Abschnitt wird die Grob-/Gesamtarchitektur betrachtet. Hierbei wird nicht nur auf wesentliche Schnittstellen und Komponenten eingegangen. Zur engeren Auswahl standen zwei mögliche Ansätze. Es werden beide betrachtet, und erläutert warum der Ansatz 2 umgesetzt werden wird.

1.3.1 Ansatz 1

Der Ansatz 1 verfolgt den Klassischen Client-Server-Ansatz. Hierbei dient der Server als zentrale Instanz, die alle signifikanten Logikoperationen übernimmt. Die Clients dienen lediglich zur Ein-/Ausgabe.

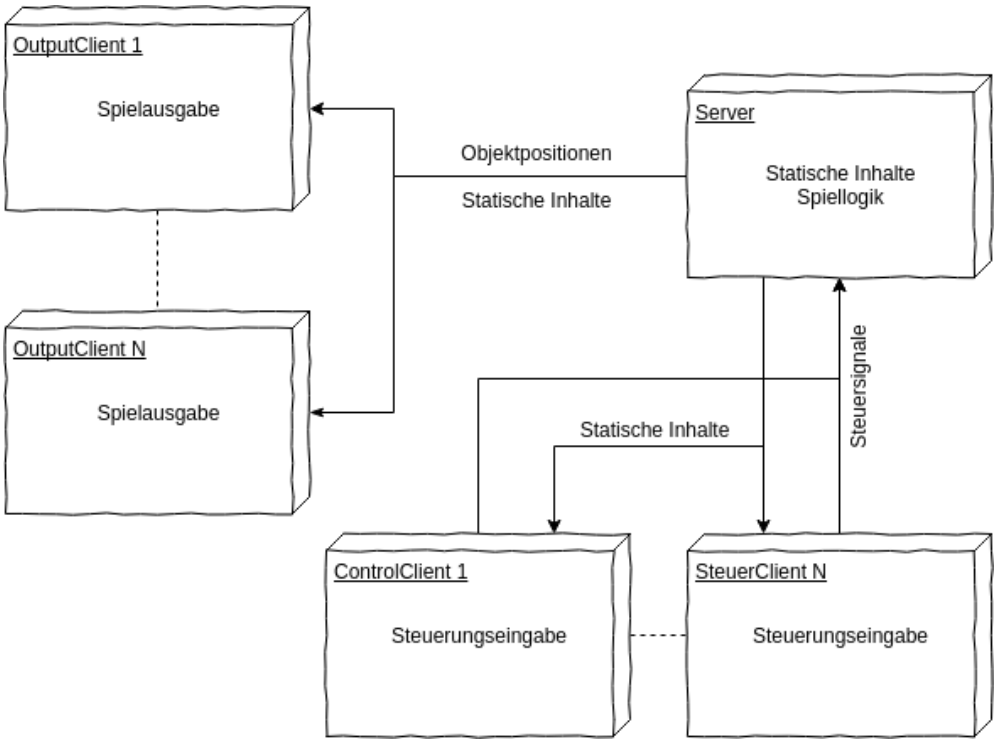

Abbildung 1.1: Highlevel Ansatz 1

Server

Der Server stellt die statischen Inhalte (HTML, CSS, JS) bereit. Zudem enthält er die gesamte Spiellogik. Der Server empfängt Steuerinformationen vom ControlClient, verarbeitet diese und sendet einen aktuellen Gamestate an den OutputClient.

ControlClient

Der ControlClient erfasst die Steuereingaben des Nutzers und sendet sie an den Server.

OutputClient

Der OutputClient empfängt den Gamestate vom Server und aktualisiert die Anzeige entsprechend.

Analyse

Einerseits ist dieser Ansatz architektonisch sehr einfach zu umzusetzen, da es eine zentrale Instanz gibt und Seperation of Concerns Architektur bedingt unterstützt wird. Zudem ist es möglich ein Spiel auf mehreren OutputClients darzustellen. Da sich die Clients mit dem Server verbinden, ist der Einsatz von WebSockets möglich. Mit socket.io gibt es eine sehr gute Abstraktionsschicht für WebSockets. Andererseits kann durch den Einsatz von WebSockets ein nicht zu vernachlässigendes Delay entstehen, da diese auf TCP basieren. Um diesem zu begegnen ist der Einsatz des WebRTC Protokollstacks notwendig. Zudem muss der Server die Zuordnung der ControlClients und OutputClients zu einem Spiel organisieren. Außerdem sind zwei Netzwerkübertragungen von Nöten, damit die Eingabe des Spielers auf dem OutputClient sichtbar wir. Dadurch wird das Netzwerk doppelt belastet, und es ist zweimal das Übertragungsdelay vorhanden.

1.3.2 Ansatz 2

Server

Der Server stellt die statischen Inhalte (HTML, CSS, JS) bereit. Zudem fungiert er als zentrale Instanz für den WebRTC-Protokollstack

ControlClient

Der ControlClient erfasst die Steuereingaben des Nutzers und sendet sie an den OutputClient.

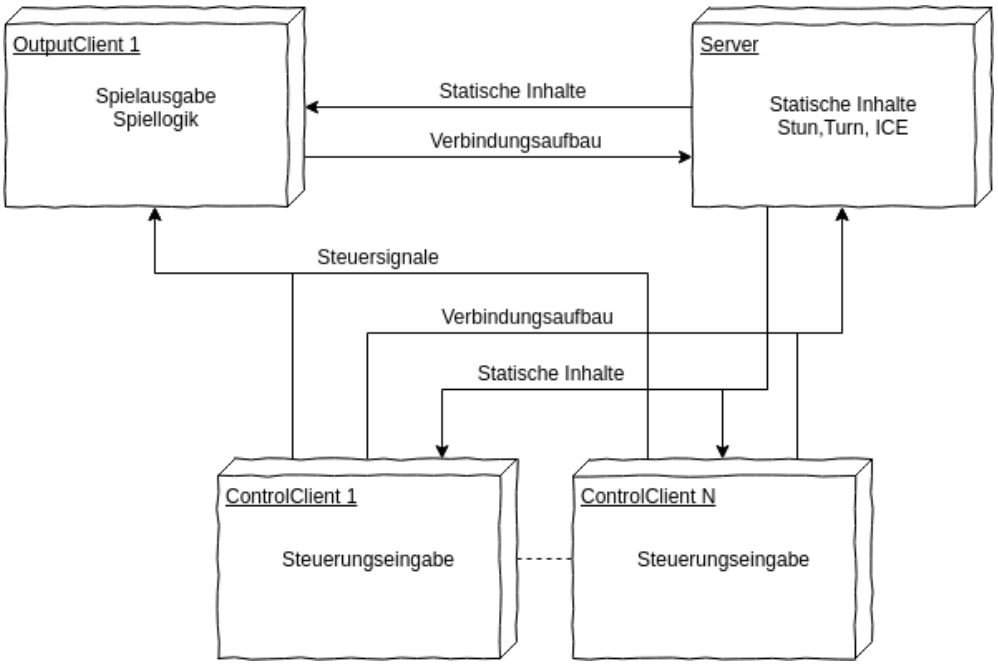

Abbildung 1.2: Highlevel Ansatz 2

OutputClient

Der OutputClient ist ein Fat-Client. Er enthält die gesamte Spiellogik und empfängt über den WebRTC-Protokollstack direkt die Steuereingaben des Spielers. Zudem zeigt er das Spiel an.

Analyse

Der Ansatz 2 ist architektonisch recht Komplex, da die Aufteilung auf Client und Server (der Fat-Client-Ansatz) unterstützt Seperation of Concern nicht direkt. Es muss während der Entwicklung verstärkt darauf geachtet werden, das die Trennung von Spiellogik und Ausgabe getrennt wird. Zu dem ist man auf den WebRTC-Protokollstack angewiesen, da die Clients direkt miteinander Kommunizieren müssen. Außerdem ist man auf einen einzelnen OutputClient beschränkt. Andererseits erfolgt die Steuerung direkt an dem OutputClient, dadurch ist das kleinst mögliche Delay zwischen Eingabe,Verarbeitung und Ausgabe gewährleistet. Außerdem ermöglicht der Einsatz von WebRTC den Einsatz von UDP als Transportprotokoll, wodurch das Delay weiter verkleinert werden kann, da UDP Verbindungslos ist.

1.3.3 Fazit

Auch wenn der Ansatz 2 zunächst komplexer erscheint und einen geringeren Funktionsumfang bittet, da nur ein AusgabeClient pro spiel unterstützt wird und WebRTC eingesetzt werden muss, überwiegen doch die Vorteile dieses Ansatzes. Durch die fehlende zweite Netzwerkübertragung und der mögliche Einsatz von UDP, werden Delays minimiert. Gleichzeitig wird mehr Rechenleistung auf die Clients ausgelagert und eine aufwändige Verwaltung, welcher Spieler zu welchem Spiel gehört ist auch nicht notwendig.

1.4 Risiken

In diesem Kapitel wird auf die Risiken eingegangen, die zu Schwierigkeiten bei der Projektdurchführung führen können. Die Auswirkungen und Eintrittswahrscheinlichkeit werden in 3 Kategorien eingeteilt: "1:Gering, 2:Mittel, 3: Hoch". Das potenzielle Risiko ist das Produkt aus Auswirkungen und Eintrittswahrscheinlichkeit.

1.4.1 Technische Risiken

Zunächst wird auf die technischen Risiken eingegangen

Darstellung ist nicht möglich

- **Beschreibung:**
  Gerade bei Ping-Pong wird die Bewegung des Balls irgendwann sehr schnell. Dies muss trotzdem im Browser darstellbar sein, ohne das es ruckelt.
- **Eintrittsgründe;**
  - Das gewählte Grafikframework ist nicht leistungsfähig genug.
  - Das gewählte Grafikframework wurde nicht richtig genutzt.
  - Das Zielsystem ist nicht Leistungsfähig genug.
- **Folgen:**
  - Das Spielen ist nicht möglich => Projektfehlschlag.
- **Eintrittswahrscheinlichkeit:** 2
- **Auswirkungen:** 3
- **Risiko:** 6

- **Maßnahmen:**
  - Bereits im Prototyp eine Beispielimplementierung durchführen.
  - Möglichst früh einen Test auf der Zielplattform absolvieren.

Kollisionserkennung funktioniert nicht

- **Beschreibung:**
  Durch die schnelle Ballbewegung bei Ping-Pong ist eine gute Kollisionserkennung notwendig.
- **Eintrittsgründe;**
  - Die gewählte Physicsengine ist nicht leistungsfähig genug.
  - Die gewählte Physicsengine wurde nicht richtig genutzt.
  - Das Zielsystem ist nicht Leistungsfähig genug.
- **Folgen:**
  - Das Spielen ist nicht möglich => Projektfehlschlag.
- **Eintrittswahrscheinlichkeit:** 2
- **Auswirkungen:** 3
- **Risiko:** 6
- **Maßnahmen:**
  - Bereits im Prototyp eine Beispielimplementierung durchführen.
  - Möglichst früh einen Test auf der Zielplattform absolvieren.

Kommunikation mittels WebRTC funktioniert nicht

- **Beschreibung:**
  Um die Übertragung in akzeptabler Geschwindigkeit zu gewährleisten, ist der Einsatz von WebRTC unausweichlich. WebRTC ist jedoch absolutes Neuland für das gesamte Team.
- **Eintrittsgründe;**
  - WebRTC wird nicht korrekt genutzt.

  - Im Zielnetzwerk wird die Verwendung durch Firewalls behindert.

- **Folgen:**
  - Die Architektur muss von Fat-Client Ansatz auf einen Serverzentrierten Ansatz umgestellt werden => Projektverzögerung.
- **Eintrittswahrscheinlichkeit:** 2
- **Auswirkungen:** 2
- **Risiko:** 4
- **Maßnahmen:**
  - Bereits im Prototyp eine Beispielimplementierung durchführen.
  - Möglichst früh einen Test auf der Zielplattform absolvieren.

Steuerung ist nicht möglich

- **Beschreibung:**
  Das auslesen der Position des Fingers auf dem Bildschirm über den Browser funktioniert nicht, bzw. nicht schnell genug.
- **Eintrittsgründe;**
  - Zu altes Handy verwendet (Browser unterstützt es nicht).
  - Schnittstelle nicht korrekt verwendet
- **Folgen:**
  - Das Spiel ist nicht Steuerbar => Projektfehlschlag.
- **Eintrittswahrscheinlichkeit:** 1
- **Auswirkungen:** 3
- **Risiko:** 3
- **Maßnahmen:**
  - Bereits im Prototyp eine Beispielimplementierung durchführen.

1.4.2 Konzeptuelle Risiken

Steuerung ist nicht Fair

- **Beschreibung:**
  Gerade mit dem Prinzip des Browser basierten Ansatzes werden sehr viele unterschiedliche Endgerätetypen angesprochen. Jedes Endgerät hat jedoch eine andere Bildschirmgröße und Pixeldichte.
- **Eintrittsgründe;**
  - Es möchten Personen mit unterschiedlichen Endgeräten gegeneinander antreten
- **Folgen:**
  - Das Spiel ist unfair => geringere Akzeptanz.
- **Eintrittswahrscheinlichkeit:** 3
- **Auswirkungen:** 1
- **Risiko:** 3
- **Maßnahmen:**
  - Das Risiko wird durch die weitergehende Analyse, und dem Entwurf eines möglichst Fairen Steuerungskonzeptes verringert.
  - Ggf. zum Teil ignorieren

1.4.3 Organisatorische Risiken

Temporärer Personalausfall

- **Beschreibung:**
  Es kann jederzeit zu einem temporären Personalausfall kommen.
- **Eintrittsgründe;**
  - Krankheit.
  - Klausuren.
  - Unmotiviertheit.
- **Folgen:**

  - Arbeitskraftmangel => Projektverzögerung.
  - Fachwissensmangel => Projektverzögerung.
- **Eintrittswahrscheinlichkeit:** 3
- **Auswirkungen:** 1
- **Risiko:** 3
- **Maßnahmen:**
  - Puffer einplanen.
  - Möglichst wenig Must-Anforderungen definieren.

Kompletter Personalausfall

- **Beschreibung:**
  Es kann jederzeit zu einem temporären Personalausfall kommen.
- **Eintrittsgründe;**
  - Krankheit.
  - Unmotiviertheit.
- **Folgen:**
  - Arbeitskraftverlust => Projektverzögerung.
  - Fachwissensverlust => Projektverzögerung.
- **Eintrittswahrscheinlichkeit:** 2
- **Auswirkungen:** 2
- **Risiko:** 4
- **Maßnahmen:**
  - Puffer einplanen.
  - Möglichst wenig Must-Anforderungen definieren.

2 JavaScript

3 Backend

4 Frontend

See also One und Two (2010).

Literaturverzeichnis

[One und Two 2010] One, Author ; Two, Author: A Sample Publication. (2010)

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 30. November 2016 Andreas Müller, Claus Torben Haug, Jan-Dennis Bartels, Marjan Bachtiari