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: 29. 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

Listings

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

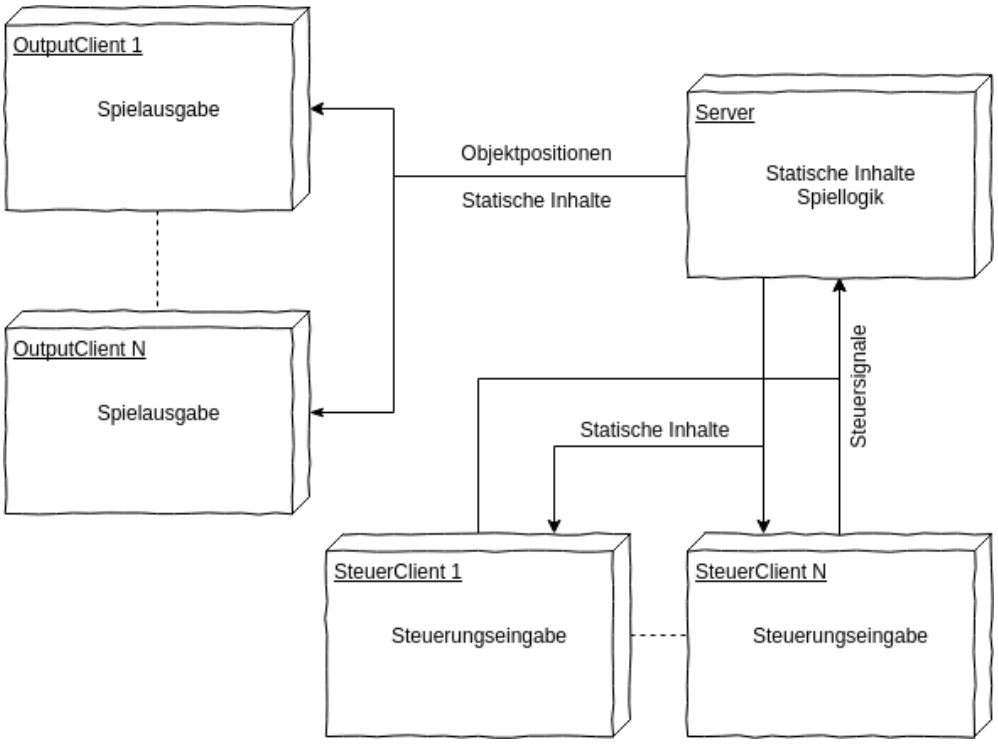

Abbildung 1.1: Highlevel Ansatz 1

1.3.2 Ansatz 2

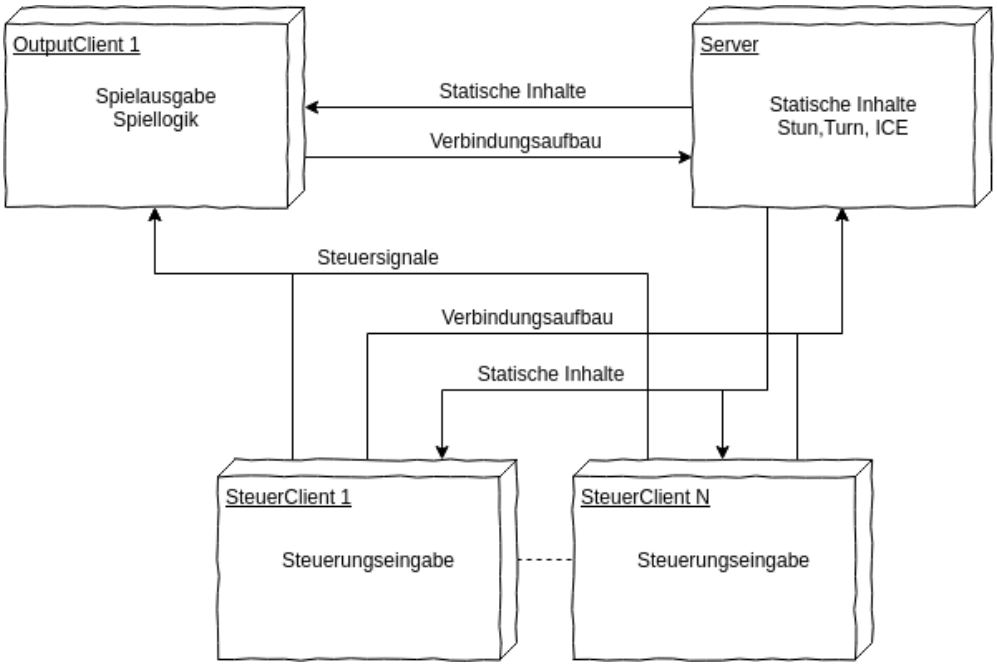

Abbildung 1.2: Highlevel Ansatz 2

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