Verteilte Systeme Praktikum

### Middleware Gruppen Kommunikation Dokumentation

### Wintersemester 2022/2023

### Gruppe: Beta 5

#### Dominik Martin, Can Heintze, Dominik Mueller

### Gruppe: Gamma 8

#### Valeria Silver, Gerrik Heitmann

# 

# Einführung und Ziele

## Aufgabenstellung

**Inhalt**

Dieses Dokument beschreibt die Kommunikation zwischen den Gruppen Beta 5 und Gamma 8 für das Zusammenspiel beim Tron Spiel.

Folgende Anforderungen sollten erfüllt werden:

### Use Cases

**Nummer:** UC-1

**Titel:** Gemeinsames Spielen

**Beschreibung:**

Beide Gruppen sollen ein gemeinsames Spiel spielen können. Dabei sollen alle Gruppenmitglieder jeweils mit der Applikation aus der Gruppe ein gemeinsames Spiel Spielen können. So, dass Beta 5 drei Spiel views startet und Gamma 8 zwei Spiel views startet.

**Nummer:** UC-2

**Titel:** Kommunikation der Gruppen Applikationen

**Beschreibung:**

Die Gruppen Applikationen sollen in der Lage sein, zu kommunizieren, zum Beispiel über eine zusammen definierte Schnittstelle wie einen REST Server oder ähnliches.

## Motivation

Die wesentliche Motivation für uns, diese Middleware zu implementieren ist es, die PVL zu erhalten. Weitere Motivationspunkte wären aber auch, neues zu lernen und unser bisheriges Wissen zu vertiefen.

## Qualitätsziele

|  |
| --- |
| **Qualitätsziele:** |
| Gut definierte Schnittstellen |
| Kompatibilität zu einer anderen Gruppe (Mindestens zwei Teams müssen miteinander spielen können) |
| Fehlertoleranz (Wenn ein Spieler abstürzt, egal welcher Spieler, dann geht das Spiel trotzdem weiter) -> Stabilität |
| Die Transparenz Kriterien sollten so gut wie möglich erfüllt werden. |

## Stakeholder

**Inhalt**

Unsere Stakeholder sind die Entwickler (Studenten), der Kunde (Professor) und die Spieler (Studenten).

Die Stakeholder mit deren Kontakt werden in der unteren Tabelle aufgelistet:

| Rolle | Kontakt | Erwartungshaltung |
| --- | --- | --- |
| *Entwickler* | [Dominik.martin@haw-hamburg.de](mailto:Dominik.martin@haw-hamburg.de) | *Ein gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Entwickler* | [*Can.heintze@haw-hamburg.de*](mailto:Can.heintze@haw-hamburg.de) | *Ein gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Entwickler* | [*Dominik.mueller@haw-hamburg.de*](mailto:Dominik.mueller@haw-hamburg.de) | *Ein gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Entwickler* | [Valeria.silver@haw-hamburg.de](mailto:Valeria.silver@haw-hamburg.de) | *Eing gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Entwickler* | [Gerrik.heitmann@haw-hamburg.de](mailto:Gerrik.heitmann@haw-hamburg.de) | *Eing gutes Spiel zu programmieren & die PVL zu erhalten* |
| *Kunde* | [*Martin.becke@haw-hamburg.de*](mailto:Martin.becke@haw-hamburg.de) | *Ein Lauffähiges Spiel, bei dem 6 Spieler gleichzeitig ein komplettes Spiel ohne Fehler durchspielen können.* |
| *Spieler / Nutzer* | *n/a* | *Ein funktionierendes Spiel spielen und dabei Spaß haben* |

# Lösungsstrategie

### Funktionale Anforderungen

|  |  |  |  |
| --- | --- | --- | --- |
| **Methoden Signatur** | **Beschreibung** | **Klasse** | **UseCase** |
| PUT /players | Erstellt einen Spieler | REST Server | Use Case übergreifend |
| GET /players | Gibt alle derzeitig vorhanden Spieler zurpück | REST Server | Use Case übergreifend |
| GET /players/1 | Gibt einen bestimmten Spieler mit der jeweiligen ID zurück | REST Server | Use Case übergreifend |
| DELETE /players | Löscht alle Spieler | REST Server | Use Case übergreifend |
| DELETE /players/1 | Löscht einen bestimmten Spieler mit der jeweiligen ID | REST Server | Use Case übergreifend |
| POST /players/1/direction | Setzt die Richtung (Nutzereingabe) für einen bestimmten Spieler mit der jeweiligen ID | REST Server | Use Case übergreifend |
| GET /players/1/direction | Gibt die Richtung (Nutzereingabe) für einen bestimmten Spieler mit der jeweiligen ID zurück | REST Server | Use Case übergreifend |
| PUT /countdown | Setzt einen Countdown für den Spielstart | REST Server | Use Case übergreifend |
| GET /countdown | Gibt den aktuellen Stand des Countdowns für den Spielstart zurück | REST Server | Use Case übergreifend |

# Bausteinsicht

**Komponenten Diagramm auf oberster Ebene:**

Diagram

Description automatically generated

**Komponenten Diagramm auf zweiter Ebene:**

Diagram

Description automatically generated

Diagram

Description automatically generated

# Laufzeitschicht

**Sequenzdiagramme**

Diagram

Description automatically generated

Diagram

Description automatically generated

# Architekturentscheidungen

**REST Server:**

Um eine Kommunikation zur anderen Gruppe zu ermöglichen, haben wir uns für die Implementierung eines REST-Servers entschieden, welcher als „Single Soruce of truth“ fungiert.

Die Nutzer Eingaben, welche im Tron Spiel entgegengenommen werden, werden vom Tron – Controller an den REST-Server gesendet, von beiden Gruppen. Diese Nutzer Eingaben, werden dann bei jedem Spiel Tick vom Tron – Model wieder vom REST-Server abgefragt. So, dass die Nutzereingaben, einfach nur über den REST-Server geroutet werden.

Wenn die Nutzereingaben vom Model abgefragt wurden, sind, berechnet das Model ganz normal die Spielelogik und das Spiel geschehen und sendet somit Gruppen intern (über RPC) die Updates an die Tron-View.