Dokumentation TriviaPvP

# Einführung

Im Rahmen von Modul M326 wurde am GIBZ das hier beschriebene Projekt von Benjamin Klein, Jan Galliker und Timo Capitelli umgesetzt. Diese Applikation stellt eine realtime Quiz Applikation zur Verfügung und heisst TriviaPvP.

# Projektplanung

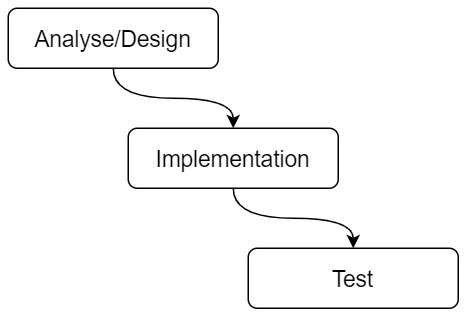
## Projektmanagement

Während dieses Projekts im Modul 326 wurde nach dem Wasserfallmodell gearbeitet.  
Das heisst, dass in einem ersten Schritt die Rahmenbedingungen dafür definiert wurden. Hierbei wurden die Anforderungen analysiert, das Ziel definiert, die eingesetzten Technologien erfasst und die Zeitplanung erstellt. Somit wurden in diesem Projekt die Schritte Analyse und Design in einem einzelnen Schritt zusammengefasst. Das macht auch Sinn, da das Projekt zu kurz ist, um die beiden Schritte einzeln auszuführen.

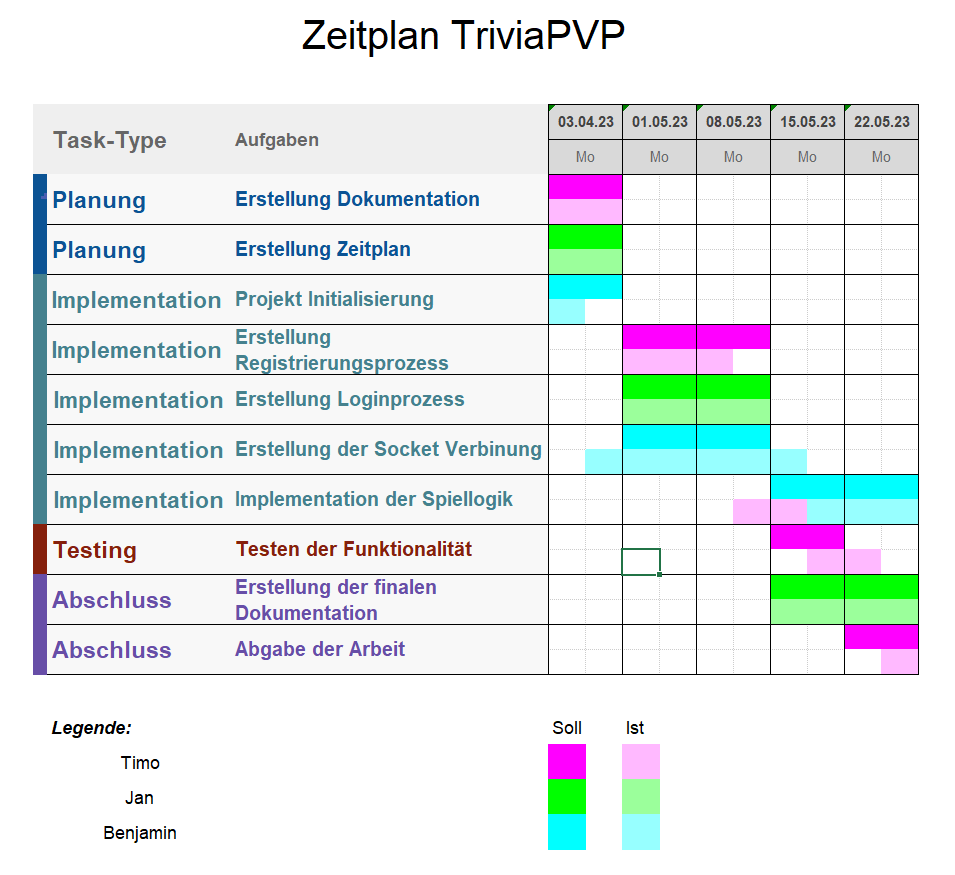
Im nächsten Schritt ging es um die Implementation. Sowohl das Frontend als auch das Backend wurden in diesem Schritt komplett implementiert.

In einem letzten Schritt wurde die gesamte Implementation getestet, um eine reibungslose Funktionalität zu garantieren.

Der Schritt der Deployen/Warten wurde nicht gemacht, da dies weder zum Projektauftrag gehört als auch aus zeitlichen Gründen nicht möglich gewesen wäre, die Applikation zu Deployen.



## Zeitplan



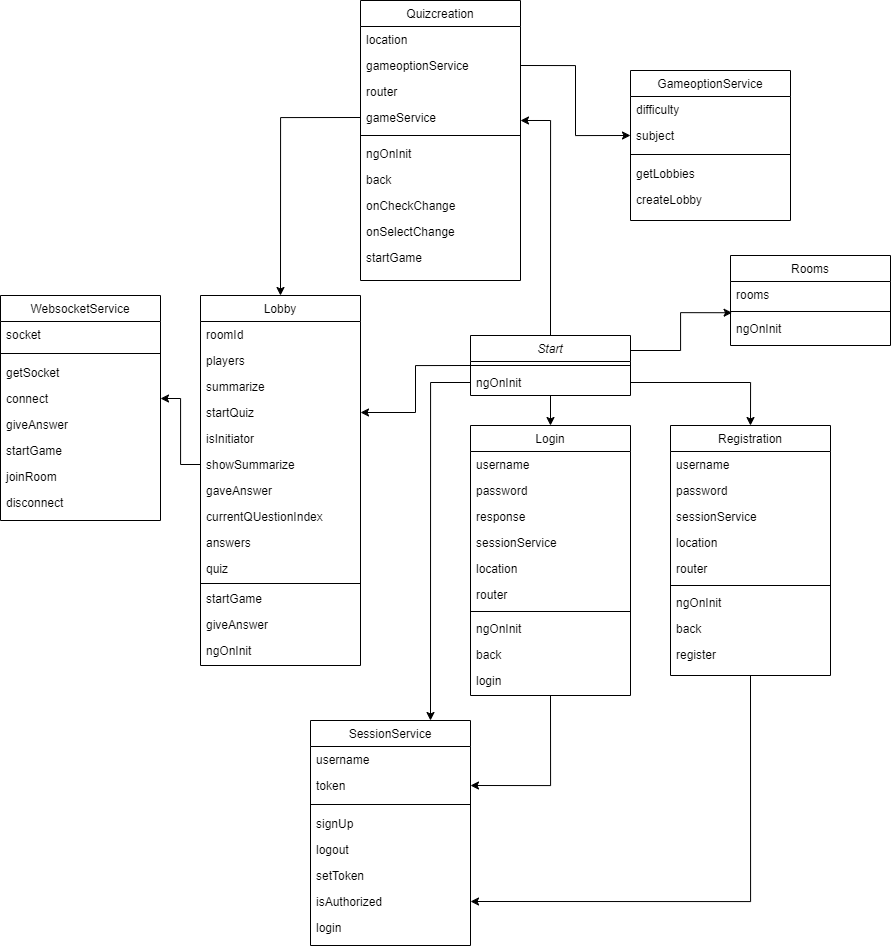
# Architektur

## Einführung

Diese Applikation stellt eine realtime Quiz Applikation zur Verfügung. Um dies zu erreichen, wurde eine REST-API Schnittstelle zu <https://opentdb.com> eingesetzt, damit die fertige Applikation eine grosse Auswahl an Fragen beziehen kann. Die eigentliche Applikation wurde in zwei Teile geteilt. Dem Frontend und dem Backend.

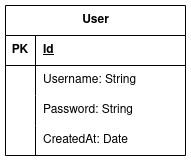
## Frontend

Wie bereits in der Projektplanung erwähnt, wurde das Frontend des TriviaPvP mit Angular geschrieben. Angular ist ein weit verbreitetes Framework für die Erstellung von Webapplikationen, welches die Grundlage bietet, eine Applikation modular aufzubauen. Daher sind alle geschriebenen Teile sogenannte Komponenten, welche individuell wiederverwendet werden können. Eine grobe Übersicht auf die Klassen dieses Angular Projektes gibt folgende Darstellung:



## Backend

Das Backend dieses Projekts wurde mithilfe von Node und Express.js erstellt. Diese sind für JavaScript erstellt worden. Um die Daten der Applikation zu persistieren, wird eine kostenlose MongoDB Atlas Datenbank im Hintergrund verwendet. Um die Authentifizierung des Users sicherzustellen, werden JSON Web Tokens verwendet. Des Weiteren kommuniziert das Backend mit Hilfe von JSON Objekten. Die gesamte Logik hierfür wurde in einem einzigen Javascript File namens index.js implementiert. Das einzige, das nicht in der Hauptdatei ist, ist die Definition des Modells, wie es in der Datenbank implementiert ist. Das Datenmodell ist wie folgt implementiert:



### Endpoints

| POST  /api/login | Eingabe: Benutzername und Passwort  Ausgabe: Status 400 oder JWT |
| --- | --- |
| POST  /api/signup | Eingabe: Benutzername und Passwort  Ausgabe: Status 400 oder JWT |
| POST  /api/create | Eingabe: Schwierigkeitsgrad und Thema  Ausgabe: Die Id vom Raum |
| GET  /api/rooms | Eingabe: Nichts  Ausgabe: Existierende Räume |
| GET  /api/join/:roomId | Eingabe: Die Id des Raums  Ausgabe: Id des Benutzers |

# Testkonzept

Weil wir für dieses gesamte Projekt nur sehr wenig Zeit zur Verfügung haben, mussten wir uns beim Testing aufs Wichtigste konzentrieren. Damit wir aber mit möglichst wenig Tests dennoch die meisten Funktionen testen wollten, haben wir uns dazu entschieden, Abnahmetests zu schreiben. So kann am Ende garantiert werden, dass zumindest die Applikation wie erwartet läuft. Dafür reichte für andere Tests, wie zum Beispiel Unittests, die Zeit nicht.

### Testcase 01

**Beschreibung:** Klickt man auf den Link ‘Registration’, kommt man auf die Seite der Registrierung. Erstellt man dort einen Benutzer mit Passwort und Benutzernamen, welcher noch nicht existiert, sieht man in der Konsole Browser Konsole keinen Fehler. Zudem wird man auf die Startseite weitergeleitet.

**Resultat:** Funktioniert!

### Testcase 02

**Beschreibung:** Klickt man auf den Link ‘Registration’, kommt man auf die Seite der Registrierung. Erstellt man dort einen Benutzer mit Passwort und Benutzernamen, welcher bereits existiert, sieht man in der Konsole Browser-Konsole einen Fehler, der einem sagt, dass der Benutzername bereits verwendet wird. Zudem wird man auf die Startseite weitergeleitet.

**Resultat:** Funktioniert!

### Testcase 03

**Beschreibung:** Klickt man auf den Link ‘Login’, kommt man auf die Seite des Login. Meldet man sich nun mit einem Benutzer an, den es nicht gibt, sieht man in der Browser-Konsole einen Fehler, dass kein Benutzer mit diesen Credentials gefunden wurde. Zudem wird man auf die Startseite weitergeleitet.

**Resultat:** Funktioniert!

### Testcase 04

**Beschreibung:** Klickt man auf den Link ‘Login’, kommt man auf die Seite des Login. Meldet man sich nun mit einem Benutzer an, den es gibt, sieht man in der Browser-Konsole keinen Fehler. Zudem wird man auf die Startseite weitergeleitet.

**Resultat:** Funktioniert!

### Testcase 05

**Beschreibung:** Klickt man auf den Link ‘Create Quiz’, kommt man auf die Seite der Quizerstellung. Wählt man eine Difficulty und eine Schwierigkeit aus und klickt auf den ‘Create Lobby’ Button, ohne angemeldet zu sein, gibt es in der Browser-Konsole einen Fehler 401.

**Resultat:** Funktioniert!

### Testcase 06

**Beschreibung:** Klickt man auf den Link ‘Create Quiz’, kommt man auf die Seite der Quizerstellung. Wählt man eine Schwierigkeit und ein Thema aus und klickt auf den ‘Create Lobby’ Button, während man angemeldet ist, wird man in die Lobby weitergeleitet.

**Resultat:** Funktioniert!

### Testcase 07

**Beschreibung:** Joint man einer Lobby, welche man erstellt hat, sieht man einen Text, den Link für die Lobby und einen Start Game Button.

**Resultat:** Funktioniert!

### Testcase 08

**Beschreibung:** Hat man erfolgreich ein Quiz erstellt und ist in einer Lobby, so wird eine Nachricht geschrieben, sobald ein anderer Benutzer derselben Lobby joint.

**Resultat:** Funktioniert!