

Konzeptbeschreibung

Team: SpeziRangers/NR.3

Mitglied 1: (Martin Beyer, 11909749)

Mitglied 2: (Nicolas Marte, 11909113)

Mitglied 3: (Islam Mechtijev, 11910366)

Mitglied 4: (Martin Neuner, 11917314)

Mitglied 5: (Clemens Prosser, 11907449)

Proseminargruppe: 6

Datum: 18.06.2021

1. Systemüberblick

Das System ist ein Webbasiertes Spiel, welches in zwei oder mehr Teams gespielt wird. Es wird mithilfe einer Webapplikation und einem IOT Würfel dargestellt. Ein Host definiert ein Themengebiet, aus welchem Fragen gestellt werden. Schließlich muss ein Teammitglied dem eigenen Team ein zufälliges Wort aus dem Themengebiet (mündlich, pantomimisch oder zeichnerisch) erklären. Der Würfel definiert die möglichen Punkte, die erlaubte Zeit und die Aktivität. Eine Runde endet, wenn die Zeit abgelaufen ist oder das ratende und erklärende Team der Ansicht ist, dass der Begriff korrekt erraten wurde und somit den Würfel dreht.

Nachdem der Würfel zur Bestätigung gedreht wurde, wird entschieden, ob das Wort fair erraten oder gegebenenfalls ein Regelverstoß durchgeführt wurde. Nach Abschluss dieser Abstimmung wird ein neues Team zum erklärenden Team und eine weitere Runde startet. Hat ein Team schließlich das Punktemaximum erreicht, gewinnt es.

Das Spiel soll ein unterhaltendes Quiz sein, kann aber auch sehr gut als Bildungsmittel eingesetzt werden. Zielgruppe sind Personen, welche ihre Erklär-, Zeichen- und Darstellungsfähigkeiten mit Kollegen/Freunden messen wollen.

2. Use Cases

2.1 Akteure

User

Ein User ist Mitglied eines Teams und besitzt ein Gerät mit Internetzugang und Browser. Er kann sich anmelden, Spiele betreten/verlassen, würfeln und erklären. Er besitzt ein Userprofil mit zahlreichen Statistiken. Ein User kann (gegebenenfalls mehrere) lokale Mitspieler erstellen. Im Browser ist die Weboberfläche zu sehen – wichtige Informationen, wie relevante Spielgeschehnisse, können hier gefunden werden.

Lokaler Mitspieler

Ein lokaler Mitspieler entspricht einem nicht registrierten Nutzer. Dieser teilt sich ein Endgerät mit einem User und wird von einem User erstellt. Er wird lediglich bei der zufälligen Auswahl des erklärenden Nutzers berücksichtigt – andere Aktionen werden vom dazugehörigen **User** ausgelöst.

Teams

In einem Team muss mindestens ein User beigetreten sein (wobei zum Start eines Spieles mindestens zwei User pro Team benötigt werden). Zu den Teammitgliedern zählen Nutzer und lokale Mitspieler (diese benötigen allerdings einen dazugehörigen User). Die Anzahl der Teammitglieder resultiert aus der Anzahl der beigetretenen Nutzer inklusive lokaler Mitspieler.

Erklärendes / Ratendes Team

Das erklärende / ratende Team ist jenes Team, wovon ein Teammitglied den aktuellen Begriff erklärt und alle anderen Teammitglieder versuchen, den Begriff zu erraten. Zu jedem Zeitpunkt eines Spieles existiert maximal ein erklärendes / ratendes Team.

Bestätigende Teams

Als bestätigendes Team bezeichnet man alle Teams exklusive dem erklärenden/ratendem Team. Sie bestätigen ein vorzeitiges Ende der Runde (sofern Begriff erraten / Regelverstoß

eingetreten) und entscheiden anschließend demokratisch, ob ein Begriff erfolgreich und fair erraten wurde oder etwaige Regelbrüche eingetreten sind.

Spielerverwalter

Ein Spielerverwalter stellt einen Nutzer mit erhöhten Berechtigungen dar (er kann somit in der Theorie genauso an einem Spiel regulär teilnehmen; es muss kein neuer User erstellt werden). Dieser sieht alle aktuell laufenden Spiele und deren Zwischenstände. Ebenso darf dieser neue Themengebiete erfassen und erweitern.

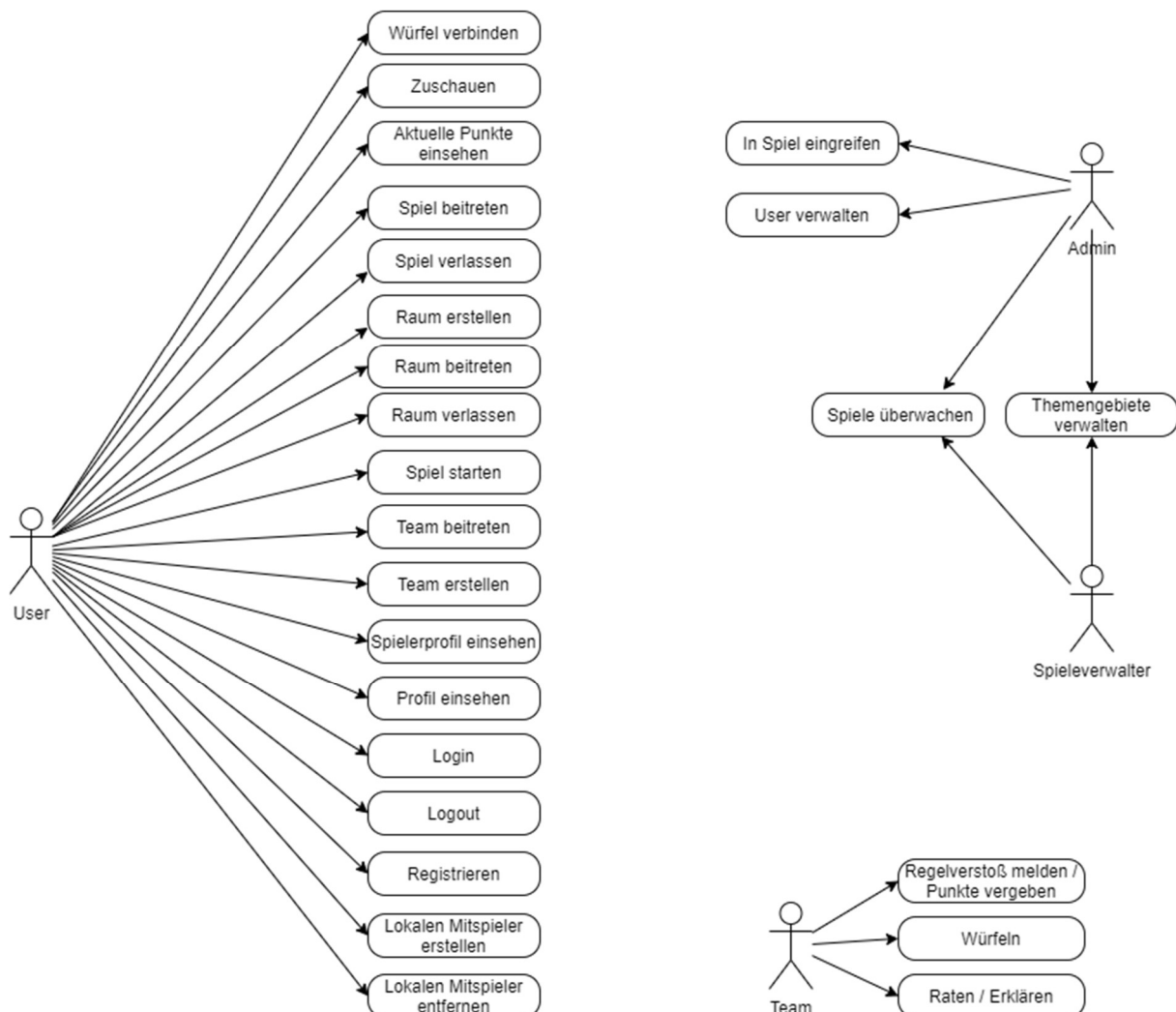
Admin

Ein Admin entspricht einem Spielerverwalter mit erhöhten Berechtigungen. Er kann zusätzlich alle Spieler verwalten (Anlegen, Bearbeiten, Löschen) und Berechtigungen vergeben (Spielerverwalter/Admin). Spiele und Zwischenstände können von einem Admin modifiziert werden.

2.2 Use-Cases

2.3.1 Use Case Diagramm

Nachfolgend befindet sich das Use Case Diagramm. Dieses beinhaltet die Funktionalitäten der jeweiligen Akteure. Zur Vereinfachung wurde in diesem Diagramm auf Vererbung verzichtet (z.B.: Administratoren \subseteq Spielerverwalter \subseteq Nutzer).



2.3.1 Akteur: User

Würfel verbinden

- *Vorbedingung:* Das System läuft und der Nutzer ist angemeldet und hat einen Raum erstellt. Es existiert ein TimeFlip Würfel und ein RaspberryPi, auf welchem das TimeCube Setup erfolgreich durchgeführt wurde
- *Ablauf:* Die Anwendung wird per Konsole gestartet, die Konfiguration wird korrekt durchgeführt
- *Erfolg:* Im jeweiligen Raum existiert eine Auswahl der Würfel. Es existiert ein Eintrag des Würfels mit dem eingegebenen Namen. Durch Auswahl wird der Würfel dem Raum zugewiesen.
- *Kein Erfolg:* Der RaspberryPi verbindet sich nicht mit dem Backend. (Dateneingabe falsch? Firewall offen?)
- *Involvierte Klassen:* Room, Cube, CubeCalibration

Zuschauen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in der Spielübersicht.
- *Ablauf:* Man tritt einem Raum bei, bei dem bereits ein Spiel läuft. Der User hat die Möglichkeit, dem Spiel zuzusehen (Punkte sehen, Begriff sehen, Verbleibende Zeit, ...)
- *Erfolg:* Der User hat Einsicht auf aktuellen Punktestand, Begriff und Verbleibende Zeit, sowie die Teams und deren Mitglieder.
- *Kein Erfolg:* Fehlermeldung wird angezeigt
- *Involvierte Klassen:* Room, Team, User

Registrieren

- *Vorbedingung:* Der User hat noch keinen Account. Das System läuft und man befindet sich auf der Startseite (Spielauswahl).
- *Ablauf:* Es wird auf „Registrieren“ gedrückt. Der zukünftige User gibt Benutzernamen, E-Mail und Passwort ein. Nach Bestätigung des Passworts wird auf „Benutzerkonto erstellen“ geklickt.
- *Erfolg:* Der User erhält eine Bestätigungsmeldung und wird zum Login weitergeleitet.
- *Kein Erfolg:* Der User erhält eine Fehlermeldung (Validierung schlug fehl)
- *Involvierte Klassen:* User

Login

- *Vorbedingung:* Der User hat einen Account. Das System läuft und man befindet sich auf der Startseite (Spielauswahl).
- *Ablauf:* Es wird auf „Einloggen“ gedrückt. Der User gibt seinen Benutzernamen und sein Passwort ein, daraufhin klickt er auf „Einloggen“
- *Erfolg:* Der User erhält eine Bestätigungsmeldung und wird zur Startseite weitergeleitet
 - Spieleverwalter: Schaltfläche für die Spieleverwalter-Optionen wird angezeigt
 - Admin: Schaltfläche für die Admin-Optionen wird angezeigt.
- *Kein Erfolg:* Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User

Logout

- *Vorbedingung:* Der User hat einen Account und ist angemeldet. Das System läuft und man befindet sich auf der Startseite (Spielauswahl).
- *Ablauf:* Der User wählt die Aktion „Ausloggen“
- *Erfolg:* Der User ist abgemeldet und zurück auf der Startseite.
- *Kein Erfolg:* Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User

Raum erstellen

- *Vorbedingung:* Der Spieler ist eingeloggt. Das System läuft und man befindet sich auf der Startseite (Spielauswahl). Ein Würfel ist verfügbar.
- *Ablauf:* Der User erstellt einen Raum und verknüpft einen freien Würfel mit diesem.
- *Erfolg:* Der neue Raum ist erstellt und der User ist dessen Host. Dem User wird der Raum mit allen beigetretenen Spielern (und erstellten lokalen Mitspielern) und Konfigurationsoptionen angezeigt.
- *Kein Erfolg:* Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User, Room, Cube

Raum beitreten

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich auf der Startseite (Spielauswahl). Ein virtueller Spielraum ist verfügbar.
- *Ablauf:* Der User wählt die Aktion „Spiel beitreten“
- *Erfolg:* Der User wird auf die Oberfläche des Spielraumes weitergeleitet
- *Kein Erfolg:* Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User, Room

Raum verlassen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem Spielraum.
- *Ablauf:* Der User wählt die Aktion "Spiel verlassen"
- *Erfolg:* Der User wird auf die Startoberfläche weitergeleitet
 - Host verlässt: Ein neuer Host wird ausgewählt
 - Verlässt der letzte User den Raum (ggf. existieren noch lokale Mitspieler desselben Nutzers), so wird der Raum geschlossen.
- *Kein Erfolg:* Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User, Room

Team beitreten

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem Spielraum. Das Spiel läuft noch nicht.
- *Ablauf:* Der User tritt einem existierenden Team bei, von welchem er noch kein Mitglied ist.
- *Erfolg:* Die aktuelle Oberfläche aktualisiert sich und der Spieler ist nun einem Team zugeordnet
- *Kein Erfolg:* Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User, Team, Room

Team erstellen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem Spielraum. Das Spiel läuft noch nicht.
- *Ablauf:* Der User drückt auf „Neues Team erstellen“ und wählt einen unbenutzten Teamnamen.
- *Erfolg:* Die aktuelle Oberfläche aktualisiert sich und der Spieler ist nun dem erstellten Team zugeordnet.
- *Kein Erfolg:* Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User, Team, Room

Themengebiet auswählen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem Spielraum. Der User ist der Spielhost und leitet somit den aktuellen Spielraum.
- *Ablauf:* Der Host drückt auf „Themengebiet auswählen“ und wählt ein Themengebiet aus.
- *Erfolg:* Allen Spielern im Raum wird das neue Themengebiet angezeigt. In der Spielauswahl wird das Thema des Raums aktualisiert.
- *Kein Erfolg:* Dem Host wird eine Fehlermeldung angezeigt.
- *Involvierte Klassen:* Topic, User, Room

Spiel starten

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem Spielraum. Der User ist der Spielhost und leitet somit den aktuellen Spielraum. Das Spiel ist nicht gestartet, es sind allerdings alle Bedingungen erfüllt (genügend Spieler + ausgewählter Würfel).
- *Ablauf:* Der Host klickt auf "Spiel starten".
- *Erfolg:* Alle eingeloggten Spieler werden über den Start benachrichtigt und können anschließend durch Bestätigung beitreten. Das Spiel beginnt, wenn genügt Spieler durch Bestätigung beigetreten sind.
- *Kein Erfolg:* Je nach Fehler erhalten die betroffenen Spieler oder zumindest der Host eine Fehlermeldung.
- *Involvierte Klassen:* Team, User, Room, Topic

Profil einsehen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft.
- *Ablauf:* Der User klickt auf die Schaltfläche "Profil"
- *Erfolg:* Der User sieht die Profilseite.
- *Kein Erfolg:* Eine Fehlermeldung wird angezeigt.
- *Involvierte Klassen:* User

Spielerprofil einsehen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft. Der Spieler findet eine Schaltfläche, welche auf ein anderes Spielerprofil verlinkt (beispielsweise in einem Raum).
- *Ablauf:* Der User wählt diese Schaltfläche aus und wird zum Spielerprofil des jeweiligen Spielers weitergeleitet.
- *Erfolg:* Der User sieht die Spielerprofilseite.
- *Kein Erfolg:* Eine Fehlermeldung wird angezeigt.
- *Involvierte Klassen:* User

Aktuelle Punkte einsehen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem aktiven Spiel eines Raums.
- *Ablauf:* Jedes Mitglied hat zu jeder Zeit des Spiels Einsicht auf den aktuellen Punktestand.
- *Erfolg:* Jedes Teammitglied sieht den aktuellen Punktestand
- *Kein Erfolg:* -

- *Involvierte Klassen:* Team, Room

Lokalen Mitspieler erstellen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem Raum und einem dazugehörigen Team.
- *Ablauf:* Man drückt auf „Lokalen Mitspieler erstellen“ und wählt einen unbenutzten Namen. Schließlich bestätigt man die Eingabe mit „Erstellen“.
- *Erfolg:* Ein lokaler Mitspieler wird im momentan ausgewählten Team hinzugefügt.
- *Kein Erfolg:* Fehlermeldung wird angezeigt
- *Involvierte Klassen:* Room, Team, User

Zuschauen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem gestarteten Raum.
- *Ablauf:* Man drückt auf die Schaltfläche „Zuschauen“.
- *Erfolg:* Der User tritt dem Spiel bei und wird (für sich) als aktiver Zuschauer markiert. Andere Spieler bemerken den neuen Zuschauer nicht. Der User wird nicht in der zufälligen Auswahl berücksichtigt. Das Spielgeschehen kann vom Nutzer in Echtzeit verfolgt werden.
- *Kein Erfolg:* Eine Fehlermeldung wird angezeigt
- *Involvierte Klassen:* Room, User, Game

Spiel beitreten

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem gestarteten Raum.
- *Ablauf:* Man drückt auf die Schaltfläche „Spiel beitreten“.
- *Erfolg:* Der User tritt dem Spiel bei, alle lokalen Mitspieler werden ebenso übernommen. Sie werden im Spielgeschehen berücksichtigt (Auswahl des erklärenden Spielers, ...)
- *Kein Erfolg:* Eine Fehlermeldung wird angezeigt
- *Involvierte Klassen:* Room, User, Game

Spiel verlassen

- *Vorbedingung:* Der User ist eingeloggt. Das System läuft und man befindet sich in einem Spiel.
- *Ablauf:* Man drückt auf die Schaltfläche „Spiel verlassen“.
- *Erfolg:* Der User verlässt das Spiel, alle lokalen Mitspieler des Users werden ebenso entfernt. Der User befindet sich inklusive lokaler Mitspieler im Raum.
- *Kein Erfolg:* Eine Fehlermeldung wird angezeigt

- *Involvierte Klassen:* Room, User, Game

2.3.2 Akteur: Teams

Regelverstoß melden / Punkte vergeben

- *Vorbedingung:* Der Spieler ist eingeloggt. Das System läuft und man befindet sich in einer aktiven Spielrunde.
- *Ablauf:* Durch Drehen des Würfels wird die Spielzeit beendet. Optional kann nun ein Spieler einen Regelverstoß mündlich aufzeigen. Alle Spieler (exklusive vom ratenden Team) können demokratisch abstimmen, ob der Begriff fair erraten oder ein Regelverstoß durchgeführt wurde. Sofern alle berechtigten Spieler abgestimmt haben (oder 30 Sekunden verstrichen sind), werden nach Auswertung Punkte hinzugefügt / entfernt.
- *Erfolg:* Die Punkte werden verändert und die Anzeigen aktualisiert.
- *Kein Erfolg:* Fehlermeldung wird angezeigt. Sofern 50% für Regelverstoß und 50% für einen fairen Ablauf sind, wird für das ratende Team entschieden (= es werden keine Punkte abgezogen).
- *Involvierte Klassen:* Team, User

Würfeln

- *Vorbedingung:* Der Spieler ist eingeloggt. Das System läuft und man befindet sich in einer aktiven Spielrunde sowie in der Rolle des erklärenden Teams.
- *Ablauf:* Der Würfel wird geworfen, anschließend wird die nach oben zeigende Fläche erkannt (Punkte, Zeit, Art der Aktivität).
- *Erfolg:* Die Spieler werden informiert
- *Kein Erfolg:* Fehlermeldung (z.B.: "Batterie ist leer")
- *Involvierte Klassen:* Team, Room, Cube

Raten/Erklären

- *Vorbedingung:* Das System läuft und man befindet sich in einer aktiven Spielrunde sowie in der Rolle des ratenden Teams. Die Zeit zum Raten hat bereits begonnen.
- *Ablauf:* Ein Teammitglied versucht den Begriff mittels zugeordneter Aktivität seinen Teamkollegen innerhalb der gegebenen Zeit zu erklären.
- *Erfolg:* Das Team errät den Begriff und bekommt Punkte (muss von den Gegnern verifiziert werden)
- *Kein Erfolg:*
 - Das Team errät den Begriff nicht und bekommt keine Punkte (muss von den Gegnern verifiziert werden)

- Es wird ein Regelverstoß durchgeführt und man wird bestraft.
- *Involvierte Klassen:* Team, Room

2.3.2 Akteur: Admin

User verwalten

- *Vorbedingung:* Das System läuft und User mit Berechtigung „Admin“ ist vorhanden.
- *Ablauf:* Auf seinem Startbildschirm sieht er die drei Aktionen (Erstellen, Bearbeiten und Löschen) mit denen er die Spielaccounts verwalten kann.
- *Erfolg:* Änderungen werden übernommen, er sieht ein positives Feedback.
- *Kein Erfolg:* Fehlermeldung wird angezeigt. (z. B.: Will User bearbeiten, der nicht mehr existiert)
- *Involvierte Klassen:* Admin, User

In Spiel eingreifen

- *Vorbedingung:* Das System läuft und User mit Berechtigung „Admin“ ist vorhanden.
- *Ablauf:* Auf seinem Startbildschirm sieht er eine Liste der aktiven Spiele. Er kann auf ein Spiel klicken und sich die Details dieses Spieles anschauen. In der Detailansicht kann er die Werte des Spiels verändern.
- *Erfolg:* Änderungen werden übernommen, er sieht ein positives Feedback und Spieler sehen Änderung.
- *Kein Erfolg:* Fehlermeldung wird angezeigt. (z. B.: Spiel existiert nicht mehr.)
- *Involvierte Klassen:* Admin, Room, User

Spiel überwachen

- Wird von Spielverwalter geerbt

Themengebiete verwalten

- Wird von Spielverwalter geerbt

2.3.3 Akteur: Spielverwalter

Spiel überwachen

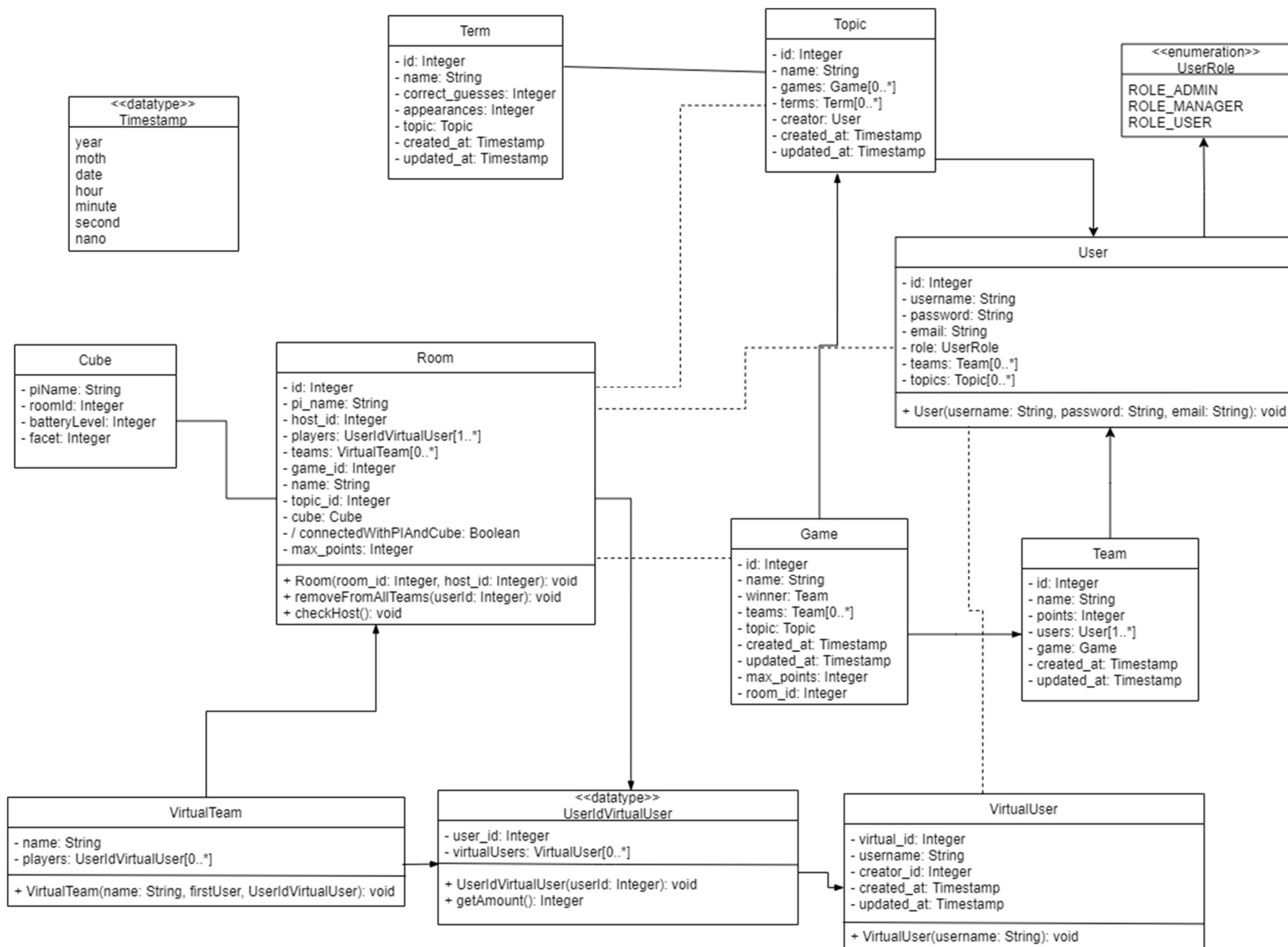
- *Vorbedingung:* Das System läuft und User mit Berechtigung „Spielverwalter“ oder „Admin“ ist vorhanden.
- *Ablauf:* Auf seinem Startbildschirm sieht er eine Liste der aktiven Spiele. Er kann auf ein Spiel klicken und sich die Details dieses Spieles einsehen.

- *Erfolg:* Er sieht die Details dieses Spiels
- *Kein Erfolg:* Fehlermeldung wird angezeigt. (z. B.: Spiel existiert nicht mehr.)
- *Involvierte Klassen:* Spieleverwalter, Admin, Room

Themengebiete verwalten

- *Vorbedingung:* Das System läuft und User mit Berechtigung „Spieleverwalter“ oder „Admin“ ist vorhanden.
- *Ablauf:* Der Nutzer sieht die Liste aller Themengebiete und kann diese editieren. Dazu gehörige Begriffe können eingesehen und modifiziert, entfernt oder hinzugefügt werden
- *Erfolg:* Themenpool wird verändert
- *Kein Erfolg:* Fehlermeldung wird angezeigt. (z.B.: „Themengebiet / Begriff existiert nicht mehr.“)
- *Involvierte Klassen:* Spieleverwalter, Admin, Topic, Term

3. Klassendiagramm



Im angeführten Klassendiagramm werden zur Vereinfachung **keine Setter und Getter** angeführt. Private Variablen weisen in der Regel **öffentlich zugängliche Getter / Setter** auf. Um eine möglichst **lose Kopplung** zwischen den Schichten zu erzeugen, werden **Data Transport Objects (DTOs)** verwendet. Diese sind ebenso nicht angeführt, allerdings dienen die Daten der angeführten Klassen als Basis.

Es existieren zahlreiche Use-Beziehungen (strichliert). Grund dafür sind fortgeschrittene **In-Memory Implementierungen**, welche für abweichende Zugriff- und Speicherstrukturen und Algorithmen sorgen.

3.1 Backend

3.1.1 Entities (Models)

Cube

Jeder nutzbare Cube ist mit einem **RaspberryPI** verbunden. Dieses Gerät verbindet sich anschließend mit dem Backend und registriert einen Cube. Informationen dazu (Aktuelle Seite, Batterie, ...) finden

sich in dieser Klasse. Hier wird kein DTO benötigt. Hierbei handelt es sich um ein Modell, welches ausschließlich **in-memory** verwendet wird.

Game

Dieses Modell dient zur Repräsentation von Spielen (Sowohl während das Spiel aktiv ist, als auch wenn es beendet ist). Diese werden persistent in der Datenbank hinterlegt. Aus einem Raum können sequenziell mehrere Spiele resultieren. Sofern dieses Modell in Webanfragen eingehend oder ausgehend verwendet wird, wird die Klasse **GameDto** verwendet.

Room

Dieses Modell dient zur Repräsentation von Räumen. Hierbei handelt es sich um ein Modell, welches ausschließlich **in-memory** verwendet wird. Aus einem Raum werden, sofern erforderliche Bedingungen erfüllt sind, Spiele erstellt. Hier wird kein DTO benötigt.

Team

Dieses Modell dient zur Repräsentation von Teams. Diese werden persistent in der Datenbank hinterlegt. Hier befinden sich alle persistenten Daten hinsichtlich Teams (Punkte, Name, Spieler, ...). Sofern dieses Modell in Webanfragen eingehend oder ausgehend verwendet wird, wird die Klasse **TeamDto** verwendet.

Term

Dieses Modell dient zur Repräsentation von Begriffen. Hier finden sich Informationen wie Begriffname, Anzahl der Vorkommnisse, Mehrere Begriffe werden zu einem Themengebiet gruppiert. Sofern dieses Modell in Webanfragen eingehend oder ausgehend verwendet wird, wird die Klasse **TermDto** verwendet.

Topic

Dieses Modell dient zur Repräsentation von Themengebieten. Hier finden sich Informationen wie Name und den Ersteller. Ein Themengebiet umfasst normalerweise mehrere Begriffe. Sofern dieses Modell in Webanfragen eingehend oder ausgehend verwendet wird, wird die Klasse **TopicDto** verwendet.

User

Dieses Modell dient zur Repräsentation von Nutzern. Hier findet man alle nutzerbezogenen Daten (Benutzername, Passwort, E-Mail, ...). Eine essentielle Eigenschaft ist die Nutzerrolle – diese wird für zahlreiche Absicherungen hinsichtlich unbefugtem Zugriff genutzt. Sofern dieses Modell in Webanfragen eingehend oder ausgehend verwendet wird, wird die Klasse **UserDto** verwendet.

UserIdVirtualUser

Diese Klasse dient zur In-Memory Zuweisung von lokalen Mitspielern zu Nutzern. Dieses Modell wird ausschließlich zur internen Verwaltung genutzt und ist nicht durch Webanfragen direkt zugänglich.

UserRole

Diese Klasse stellt einen Enum dar, konkret werden hier die unterschiedlichen Nutzerrollen definiert. Jeder Nutzer besitzt eine dieser Rollen. Je nach dem, welche Rolle zugewiesen wurde, erhält ein Nutzer unterschiedliche Berechtigungen.

VirtualTeam

Dieses Modell dient zur In-Memory Zuweisung von Nutzern (inkl. Lokaler Mitspieler) zu Teams. Dieses Modell wird im Zeitraum eines Raumes benutzt. Sofern ein Spiel persistent erstellt wird, wird das virtuelle Team in ein vollständig persistentes Team konvertiert. Sofern dieses Modell in Webanfragen eingehend oder ausgehend verwendet wird, wird die Klasse **VirtualTeamDto** verwendet.

VirtualUser

Dieses Modell dient zur Repräsentation von lokalen Mitspielern. Hier finden sich Informationen wie Name, dazugehöriger Nutzer, Ein Nutzer kann mehrere lokale Mitspieler erstellen. Ein lokaler Mitspieler weist immer exakt einen Nutzer auf. Dieses Modell wird ausschließlich **In-Memory** verwaltet. Sofern dieses Modell in Webanfragen eingehend oder ausgehend verwendet wird, wird die Klasse **VirtualUserDto** verwendet.

3.1.2 Weitere wichtige Klassen

ErrorResponse

Diese Klasse wird benutzt, sofern über die REST-API eine Fehlermeldung resultiert. Genauere Informationen dazu können in der beiliegenden `rest_documentation.md` nachgeschlagen werden.

SuccessResponse

Diese Klasse wird benutzt, sofern über die REST-API ein positives Ergebnis resultiert (Gegenstück zu ErrorResponse). Genauere Informationen dazu können in der beiliegenden `rest_documentation.md` nachgeschlagen werden.

WebSocketResponse

Diese Klasse wird benutzt, sofern via WebSockets eine Nachricht übermittelt wird. Genauere Informationen dazu können in der beiliegenden `websocket_documentation.md` nachgeschlagen werden.

3.2 RaspberryPI

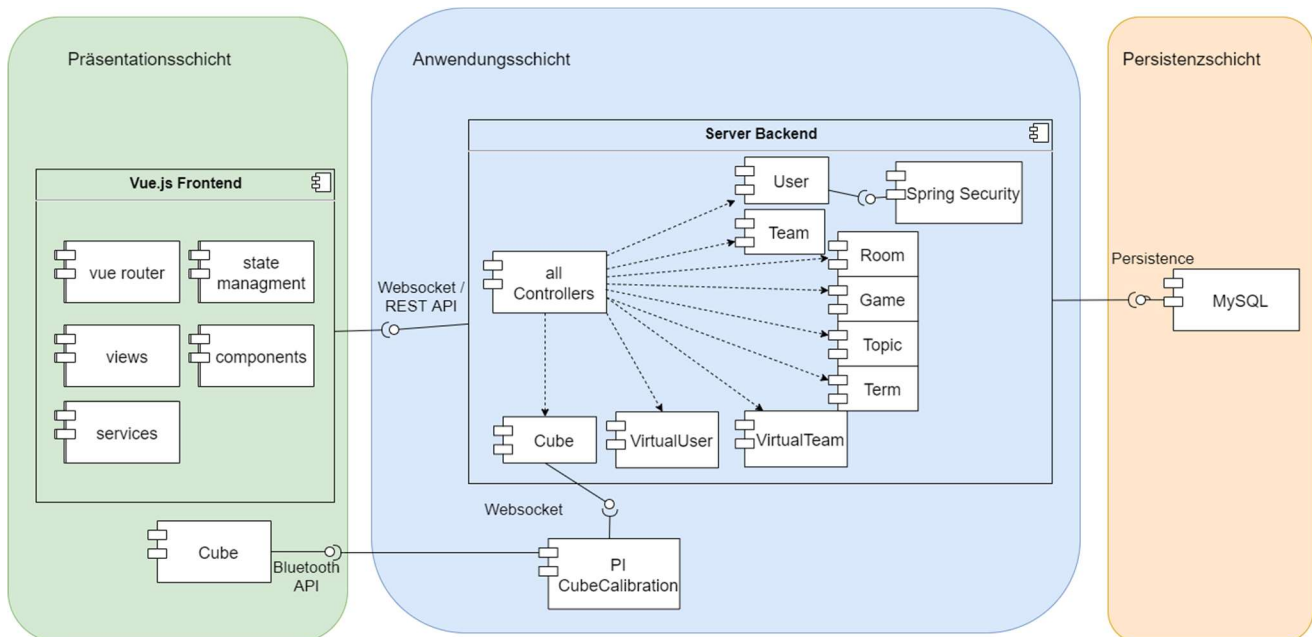
3.2.1 Entities (Models)

Cube

Jeder nutzbare Cube ist mit einem **RaspberryPI** verbunden. Dieses Gerät verbindet sich anschließend mit dem Backend und registriert einen Cube. Informationen dazu (Aktuelle Seite, Batterie, ...) finden sich in dieser Klasse. Hier wird kein DTO benötigt. Hierbei handelt es sich um ein Modell, welches ausschließlich **in-memory** verwendet wird.

4. SW-Architektur

4.1 Komponentendiagramm



Die Architektur ist in drei Schichten unterteilt:

- **Präsentationsschicht**

Diese Schicht beinhaltet alle Module, mit denen der User direkt interagiert. Als Frontend versteht man die Weboberfläche, welche im Browser des Nutzers angezeigt wird. Aktionen werden via REST-API angestoßen. Zusätzlich zu den Antworten der REST-API erhält das Frontend laufend WebSocket Nachrichten. Diese Nachrichten ermöglichen einen Echtzeitaustausch von Informationen, welcher für ein digitales Spiel essentiell ist.

Um eine gute User Experience zu ermöglichen, wird JavaScript eingesetzt – hier explizit das JavaScript Framework Vue.js und die dazugehörige Komponenten-Architektur.

Ebenso zur Präsentationsschicht zählt der Würfel. Dieser liegt bei den Spielern und kommuniziert über die Bluetooth Schnittstelle mit einem dazugehörigen RaspberryPI. Über diesen Würfel können Aktionen angestoßen werden (Bsp.: Begriff erraten).

- **Anwendungsschicht:**

Im Backend wird das Java Framework Spring verwendet. Dieses beinhaltet den Großteil der Logik und steht im ständigen Informationsaustausch mit Dritten. Komplexe Sicherheitsmechanismen sind vorhanden, um einen unbefugten Zugriff zu unterbinden. Für die Kommunikation stehen REST-API und WebSockets zur Verfügung.

Der RaspberryPI zählt zum Backend – für diese steht ein eigenes Unterprojekt zur Verfügung, hier wird ebenso das Java Framework Spring verwendet. Die Kommunikation erfolgt via Bluetooth beziehungsweise WebSockets.

- **Persistenzschicht:**

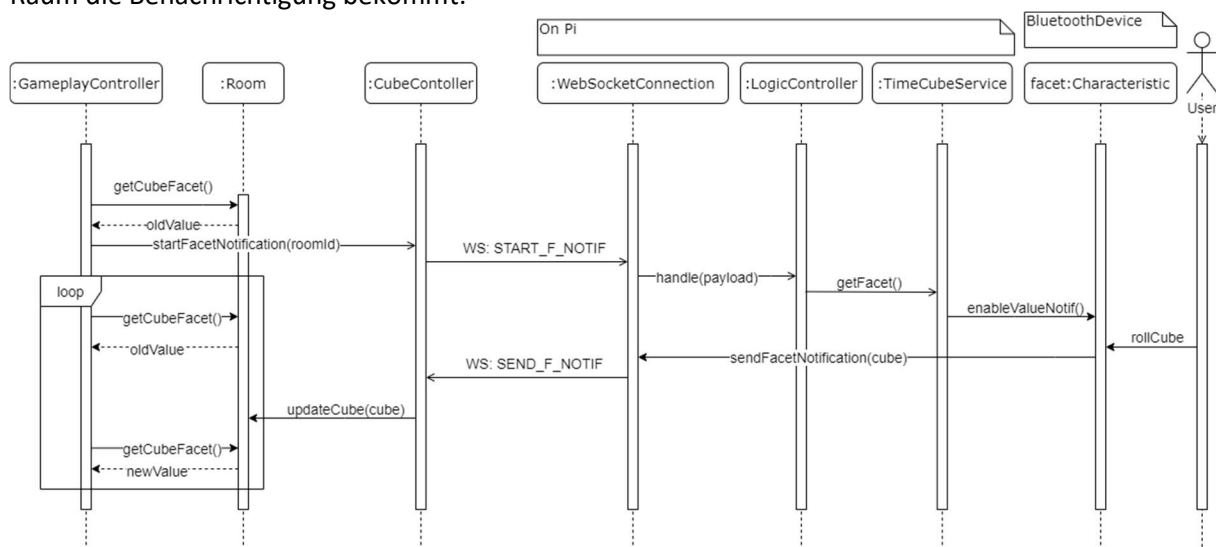
Daten müssen persistent gespeichert werden, da diese ansonsten spätestens nach einer Beendigung der Anwendung nicht mehr vorhanden wären. Alle persistenten Daten werden zentral beim Backend in einer dazugehörigen MySQL Datenbank verwaltet. Die MySQL

Datenbank muss nicht zwingend auf demselben Server laufen. Zur Anbindung von Spring wird die Spring Data JPA verwendet (Hibernate).

4.2 Laufzeitsicht

Um eine Würfelseite des Dodekaeders abzufragen, wird ein MiniComputer mit Bluetooth Schnittstelle benötigt. Da ein RaspberryPi benutzt wird, wird hier einfach nur „Pi“ als Bezeichnung dieses MiniComputers benutzt. Der Pi braucht ebenfalls eine funktionierende Internetanbindung (Intranet, wenn das Backend im lokalen Netzwerk liegt).

In nachfolgende Sequenzdiagramm wird erklärt, wie das Gameplay die Benachrichtigung einer Würfelseitenänderung aktiviert und diese schließlich empfängt. Da der Pi hier schon mit einem Raum verbunden ist, ist ihm schon die *room_id* bekannt. Hiermit wird gewährleistet, dass nur der aktuelle Raum die Benachrichtigung bekommt.

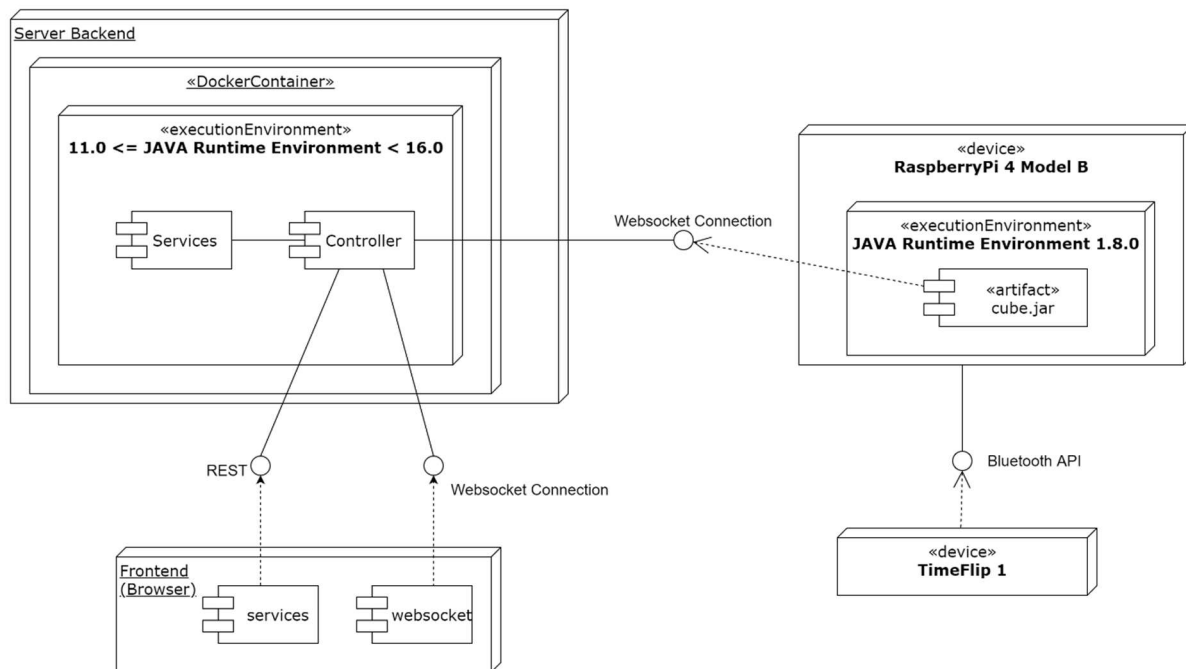


Vereinfachte Darstellung der Kommunikation zwischen Backend und Würfel
 Aus Platzgründen ist das Wort „Notification“ gegebenenfalls mit „NOTIF“ abgekürzt.

Der GameplayController steuert die einzelnen Spiele. Nachdem das Spiel gestartet wurde sowie ein Team zum Raten ausgewählt wurde, wird eine WebSocket Message an den Pi geschickt. Diese aktiviert die Benachrichtigungen beim Pi. Alle WebSocket Messages werden durch den LogicController initial verarbeitet.

Wenn nun eine Änderung der Würfelseite erfolgt, sendet der Würfel an den Pi eine Benachrichtigung via Bluetooth. Der Pi liest die aktuelle Seite des Würfels aus. Danach wird die zuvor durchgeführte Würfelkalibrierung (hier nicht dargestellt) benutzt, um genau eine der 12 möglichen Würfelseiten zuzuweisen. Diese Information wird im Anschluss an das Backend via WebSocket übermittelt.

4.3 Verteilungssicht



Das System besteht aus drei Geräten. Der Würfel, ein RaspberryPi und ein Server. Auf dem Server ist ein Docker Container mit der Anwendung in einer Java Laufzeitumgebung. Die Datenbank befindet sich ebenfalls in dem Container. Die Services realisieren die Transaktionen zwischen der Datenbank und der Anwendung. Die Controller stellen die Schnittstellen für die Kommunikation mit den anderen Geräten.

Das Frontend wird vom Backend via Tomcat ausgehändigt. Mithilfe einer REST-API (erkennbar an der Naming Convention) können Daten abgefragt und Aktionen angestoßen werden. Zusätzlich werden Informationen via WebSocket dem Frontend zur Verfügung gestellt.

Der CubeController bietet einen Full-Duplex WebSocket Channel an, welcher für die Kommunikation mit dem Pi dient. Auf dem Pi läuft ebenfalls eine Java Laufzeitumgebung, jedoch auf Java 1.8.0 (Die Funktionalität vom Bluetooth Adapter tinyb wird hiermit gewährleistet).

5. GUI Prototyp

5.1 Beschreibung und Struktur

Das GUI wird in Form einer modernen Single Page Application auf Basis von Vue.js realisiert. Ein typisches Konzept zur Realisierung von Webseiten ist **Mobile First**. Dabei realisiert man Webseiten zuerst für mobile Endgeräte – diese weisen in der Regel wesentlich weniger Platz auf. Das vereinfacht den Prozess sehr, da man so nicht zwingend Elemente entfernen muss (um von einer Desktop Ansicht auf eine mobile Ansicht zu optimieren). Dieses Konzept wurde hier angewendet, wird allerdings nicht angeführt.

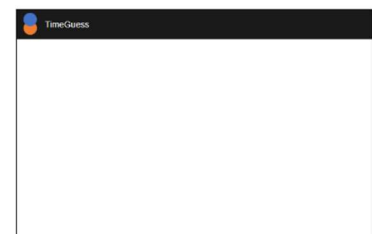
Grundsätzlich wird die nachfolgende Seitenstruktur verwendet:

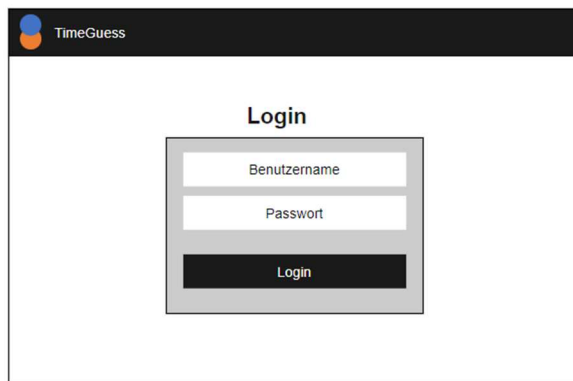
- **Übersicht**
Hier werden die verfügbaren Räume gelistet
- **Authentifizierung**
 - **Signup**
Hier kann man sich registrieren
 - **Signin**
Hier kann man sich anmelden
- **Profil(e)**
Hier können sowohl das eigene Profil, als auch Nutzerprofile anderer Nutzer eingesehen werden.
- **Dashboard (Spieleverwalter / Admin)**
Hier sieht man eine erweiterte Liste der Räume (inklusive Punktestand und Themengebiet), eine Liste der Themengebiete (inkl. Aktionen) sowie eine Liste aller Nutzer (inkl. Aktionen).
- **Raum**
Hier können beigetretene Spieler eingesehen werden und Teams verwaltet werden. Sofern man am Spiel teilnimmt, wechselt die Ansicht in den Spielmodus (Aktueller Spieler, Timer, Begriff, ... wird angezeigt)
- **Error**
Hier finden sich mehrere Fehlerbildschirme (Error 404 und 500)
- **WS Debug**
Hier kann ein Debugtool für WebSocket Verbindungen gefunden werden (Hinweis: In einem Production Build ist diese Seite deaktiviert).

Animierte Elemente (Dropdowns, Tooltips, Toasts, ...) können in nachfolgenden Bildern nicht/schwer dargestellt werden. Manche der oben angeführten Seiten können nur aufgerufen werden, sofern die dazugehörigen Bedingungen erfüllt sind (Bsp.: Nutzerrolle).

5.2 GUI Entwurf

Nachfolgend befindet sich der GUI Entwurf. Alle Seiten werden dargestellt. Die jeweilige Funktionalität kann den Use Cases entnommen werden.





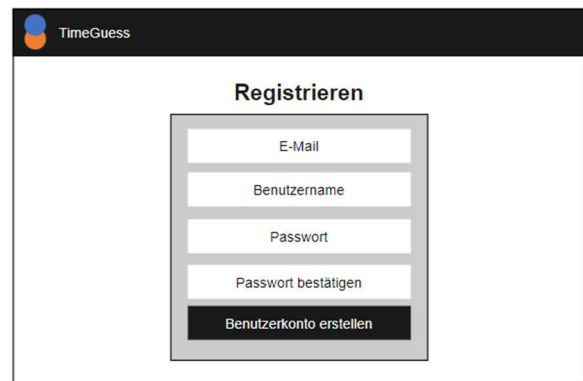
TimeGuess

Login

Benutzername

Passwort

Login



TimeGuess

Registrieren

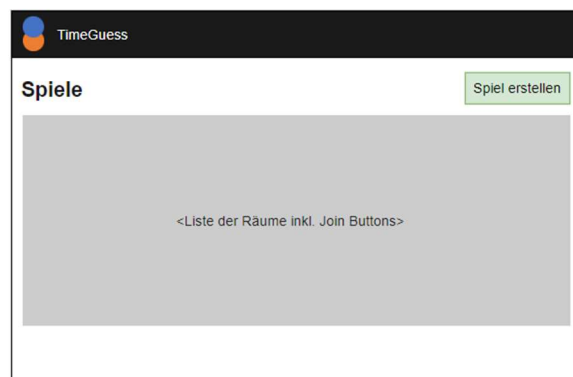
E-Mail

Benutzername

Passwort

Passwort bestätigen

Benutzerkonto erstellen

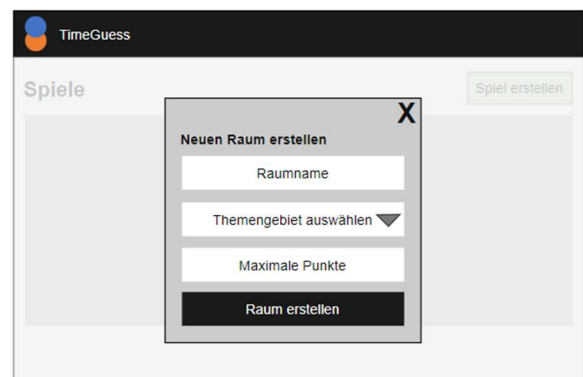


TimeGuess

Spiele

Spiel erstellen

<Liste der Räume inkl. Join Buttons>



TimeGuess

Spiele

Spiel erstellen

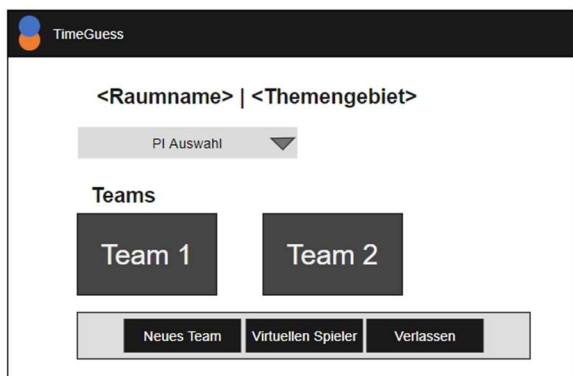
Neuen Raum erstellen

Raumname

Themengebiet auswählen

Maximale Punkte

Raum erstellen



TimeGuess

<Raumname> | <Themengebiet>

PI Auswahl

Teams

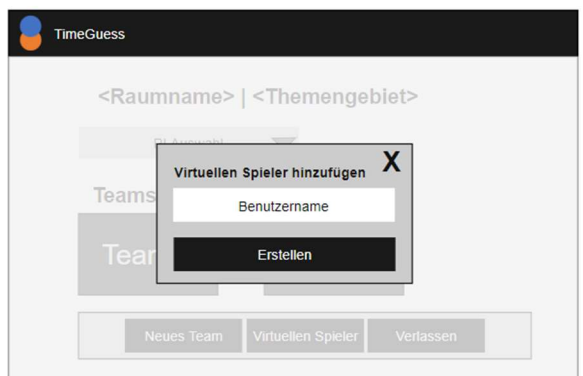
Team 1

Team 2

Neues Team

Virtuellen Spieler

Verlassen



TimeGuess

<Raumname> | <Themengebiet>

PI Auswahl

Teams

Team 1

Team 2

Virtuellen Spieler hinzufügen

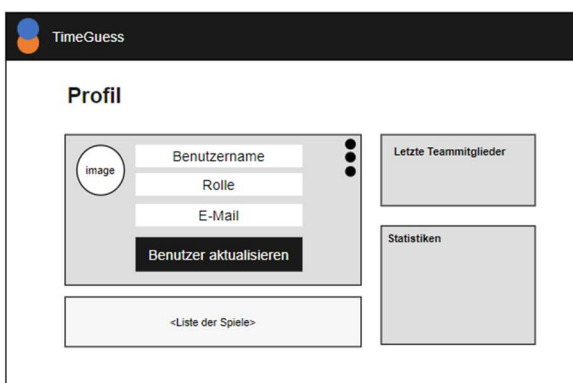
Benutzername

Erstellen

Neues Team

Virtuellen Spieler

Verlassen



TimeGuess

Profil

image

Benutzername

Rolle

E-Mail

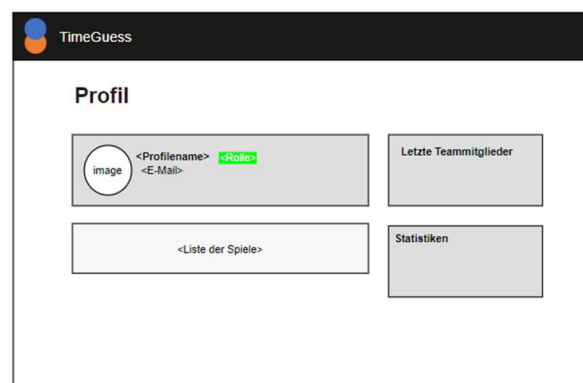
Benutzer aktualisieren

Letzte Teammitglieder

Statistiken

<Liste der Spiele>

Mit Klick auf die drei Punkte kann dieses Inline-Formular zur Bearbeitung der Nutzerdaten angezeigt werden.



TimeGuess

Profil

image

<Profilname>

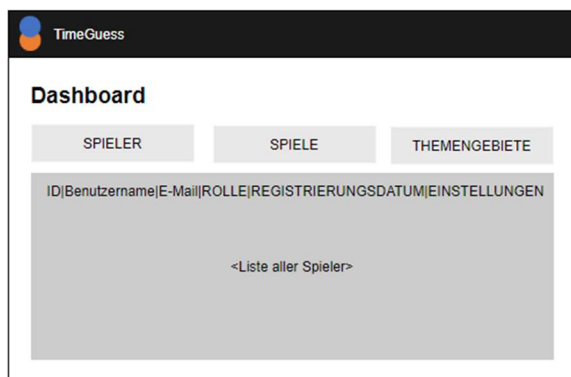
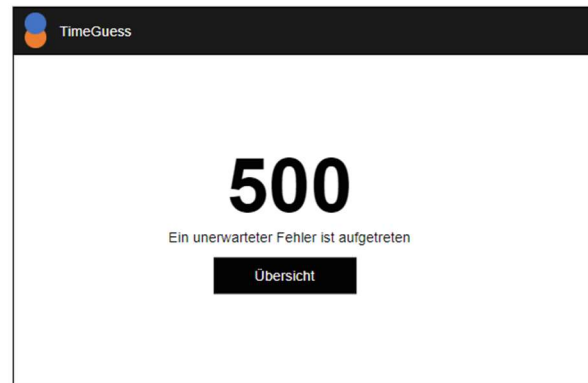
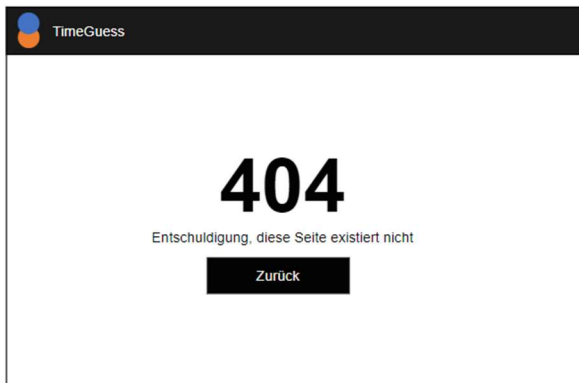
<E-Mail>

<Rolle>

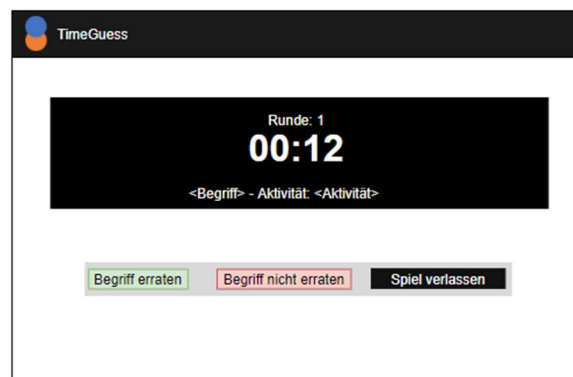
Letzte Teammitglieder

Statistiken

<Liste der Spiele>



Je nach ausgewählter Sektion (Spieler, Spiele, Themengebiete) wird die jeweilige Liste und dazugehörige Aktionen angezeigt. In der Sektion Themengebiete existiert zusätzlich der Begriff-Importer zum Importieren einer JSON Datei.



6. Projektplan

6.1 Rolleneinteilung

Nachfolgende Rolleneinteilung würde für das Projekt definiert:

- **Hardware / Bluetooth Architekt:** Martin Johannes Beyer
- **Frontend Architekt:** Islam Mechtijev
- **Backend Architekt:** Martin Fritz Neuner
- **GIT Architekt:** Clemens Prosser

Die jeweiligen Architekten sind für die zeitliche Koordination von Issues sowie Kommunikation und Management hinsichtlich des jeweiligen Aufgabenbereichs zuständig. Sie stellen den jeweiligen ersten Ansprechpartner dar.

6.2 Zeitplan

Damit alle Projektmitglieder stets über den jeweiligen Status informiert sind, finden wöchentliche JourFixe Montags (19:00) sowie Freitags (16:00) statt.

Der Projektverlauf wird mithilfe von Meilensteinen geplant. Sofern alle Issues eines Meilensteins erledigt sind, ist der Meilenstein erledigt. Bevorzugt soll an Issues des momentan aktiven Meilensteins gearbeitet werden – man kann allerdings auch vorarbeiten.

Nummer	Datum	Bezeichnung
1	18.03.21	Fertigstellung Konzept
2	28.03.21	Project Setup / Konfiguration
3	11.04.21	Meilenstein 1
4	02.05.21	Meilenstein 2
5	16.05.21	Meilenstein 3
6	23.05.21	Fertigstellung aller Features
7	13.06.21	Projektabschluss

Die angeführten Meilensteine umfassen nachfolgende Aufgaben:

- **Project Setup / Konfiguration:**
ER-Diagramm, Glossar, Issues einpflegen, git workflow, Dummy Projekt, Frontend Build Integration, API Planung
- **Meilenstein 1:**
Kommunikation RaspberryPi und TimeCube, REST und Websocket Kommunikation, REST Endpoint Implementierung, Datenbank Setup, Authentifizierung und Autorisierung
- **Meilenstein 2:**
Frontend Views (GUI Implementierung), Websocket Implementierung, Docker Setup
- **Meilenstein 3:**
Frontend Views (Adminoberfläche), Erweitertes Testen (Testabdeckung), arc42 Dokumentation (Diagramme)
- **Fertigstellung aller Features:**
Feinschliff, Bugtesting

6.2 Daily Updates

Um zusätzlich zu den JourFixe Terminen über den Status anderer Branches / Issues informiert zu sein, wird der Status der Entwicklung von jeder Person manuell mittels Checkin - Checkout Messages (in diesem Fall über Slack) festgehalten. Andere Teammitglieder müssen so nicht explizit nachfragen, zusätzlich erhält man wesentlich öfter Statusupdates (als lediglich mit Jour Fixe).

6.3 Releases und Feature Freeze

An jedem Sonntag (bis spätestens Montag 08:00) wird ein Release angelegt. Dazu werden gegebenenfalls noch offene Merge Requests nach Dev gemerged und anschließend der Release inklusive Tag angelegt. Damit noch Zeit besteht, Merge Conflicts o. ä. zu beheben, existiert ein **Feature Freeze** (Sonntag 21:00). Änderungen, welche nach Feature Freeze eingereicht werden, können für den derzeitigen Release nicht berücksichtigt werden