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: 18.03.2021

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 registrieren 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.

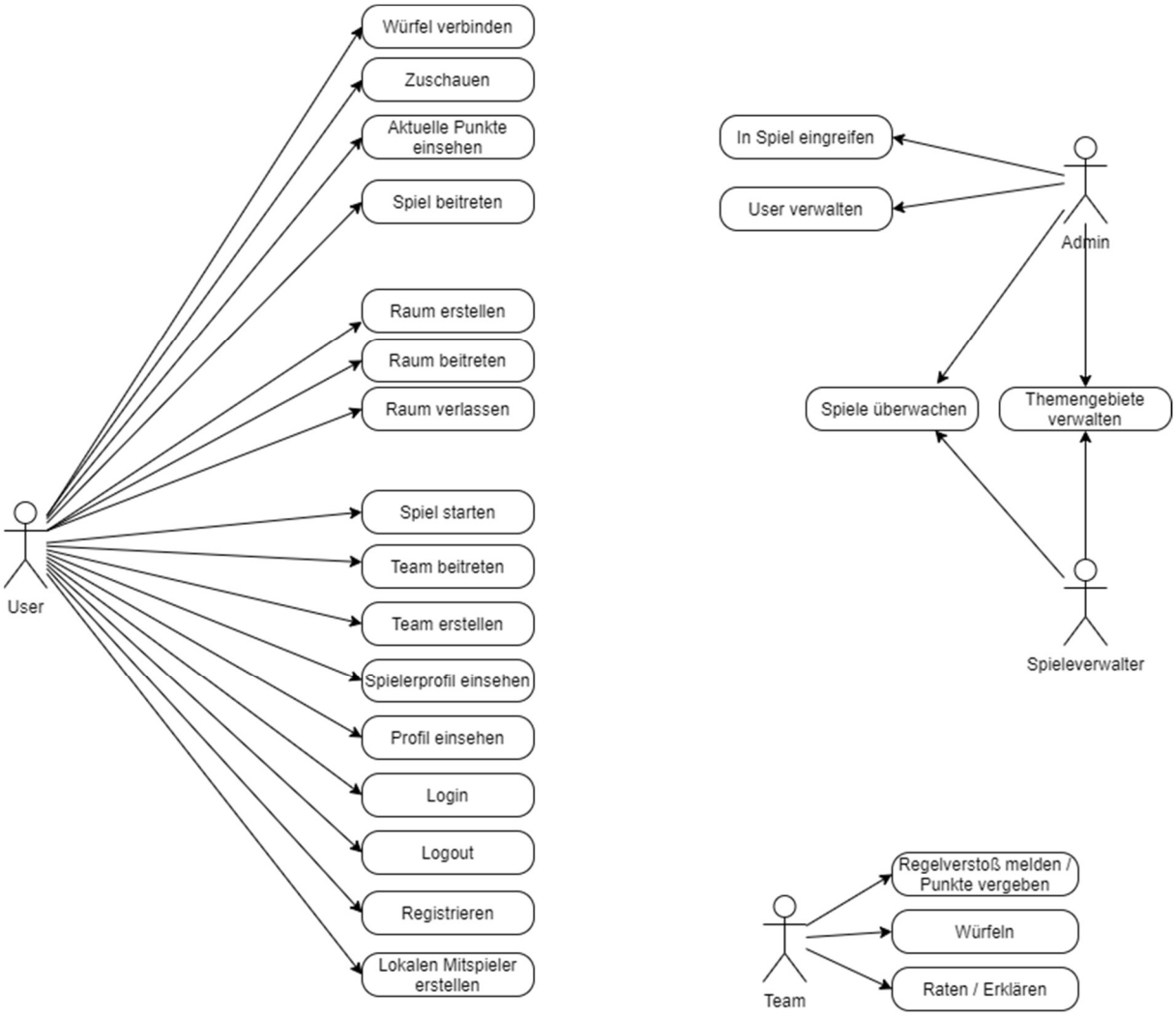
Spieleverwalter
Ein Spieleverwalter stellt einen Nutzer mit erhöhten Berechtigungen dar (er kann somit in der Theorie genauso an einem Spiel regulär teilnehmen; es muss kein neuer User erstellt werden). Dieser sieht alle aktuell laufenden Spiele und deren Zwischenstände. Ebenso darf dieser neue Themengebiete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

Admin
Ein Admin entspricht einem Spieleverwalter mit erhöhten Berechtigungen. Er kann zusätzlich alle Spieler verwalten (Anlegen, Bearbeiten, Löschen) und Berechtigungen vergeben (Spieleverwalter/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 ⊆ Spieleverwalter ⊆ 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

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 einer 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 Spieleraumes 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

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

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

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

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. 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 Spieleverwalter geerbt

Themengebiete verwalten

- Wird von Spieleverwalter geerbt

2.3.3 Akteur: Spieleverwalter

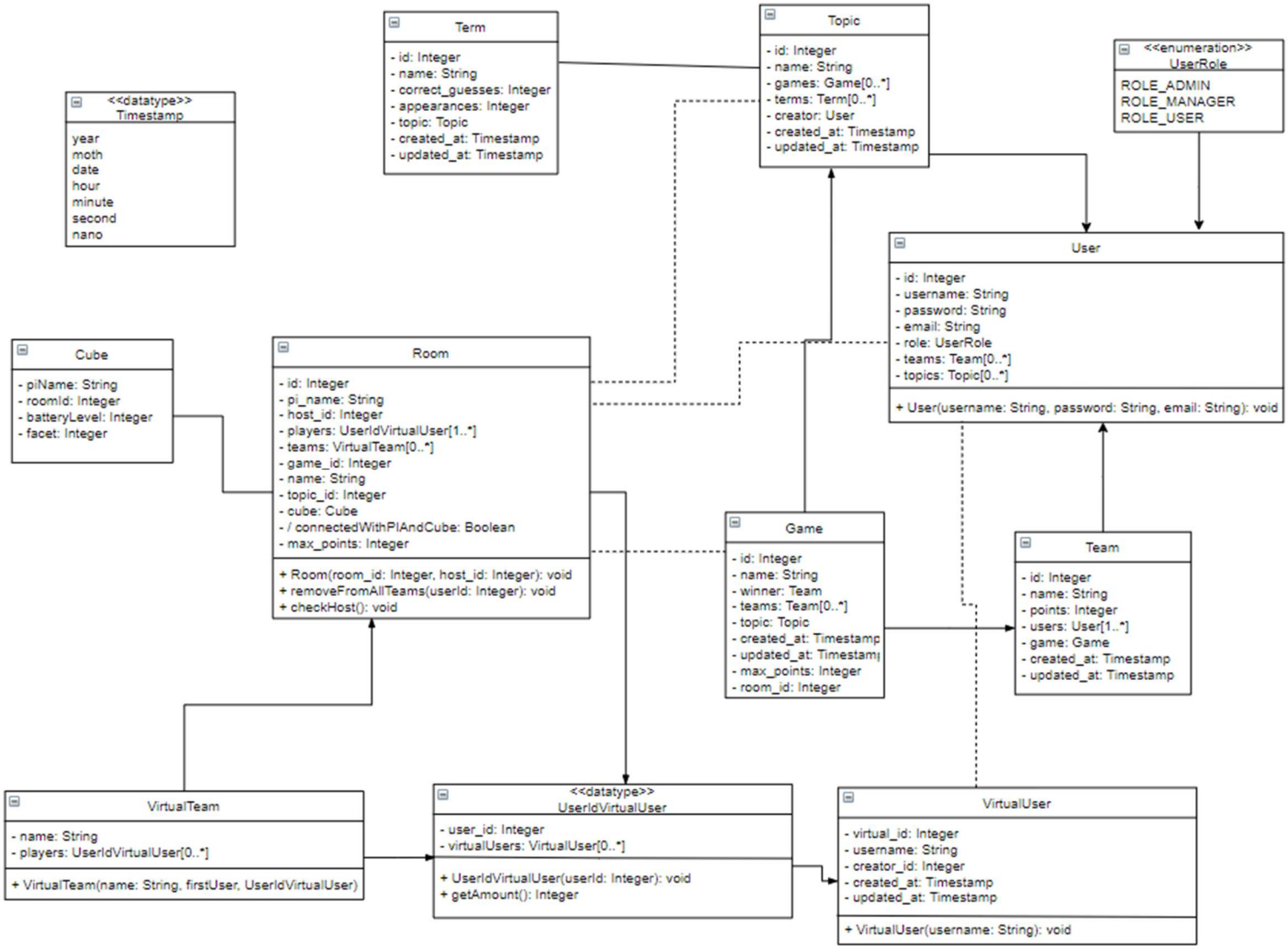
Spiel überwachen

- *Vorbedingung*: Das System läuft und User mit Berechtigung „Spieleverwalter“ 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. Klassendiagram



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)

User

Speichert alle wichtigen Daten die normale Spieler betreffen (Benutzername, Passwort, Email, Spieldaten, etc...)

Team

Team speichert alle relevanten Daten wie den Punktestand am Ende eines Spieles, verwendeter Raum, Rolle, Teamnamen und User eines Teams.

Raum

Enthält die ID des zugehörigen TimeFlip-Würfels, Daten über die sich darin befindlichen Teams, sowie Daten bezüglich des Themengebietes.

Themengebiet

Ist eine Sammlung aus Begriffen zu einem Thema.

Begriff

Begriffe bestehen aus einem Wort und werden verschiedenen Themengebieten zugeordnet. Im Standardfall wird ein Begriff nur einem Themengebiet zugeordnet.

Würfel

Jeder TimeFlip Würfel hat eine eindeutige ID, sowie 12 Seiten für 12 verschiedene Funktionen. Diese Funktionen werden in der Würfel Kalibrierung erfasst und verarbeitet.

Würfel Kalibrierung

Die Würfel Kalibrierung bietet die Möglichkeit ein Würfelseite mit einer Fläche aus der Angabe (Punkte, Aktivität, Zeit) zu verbinden.

Repositories

Zu jeder Entity-Klasse (Model) gibt es ein Repository. Dieses enthält SQL-Queries welche sich auf Suchanfragen im Entsprechenden Table beschränken.

Services

Klassen welche als Schnittstelle zwischen der Java Anwendung und der Datenbank dienten. Verwenden Funktionen aus Repositories wie z. B. Löschen und Erstellen von User.

Controller

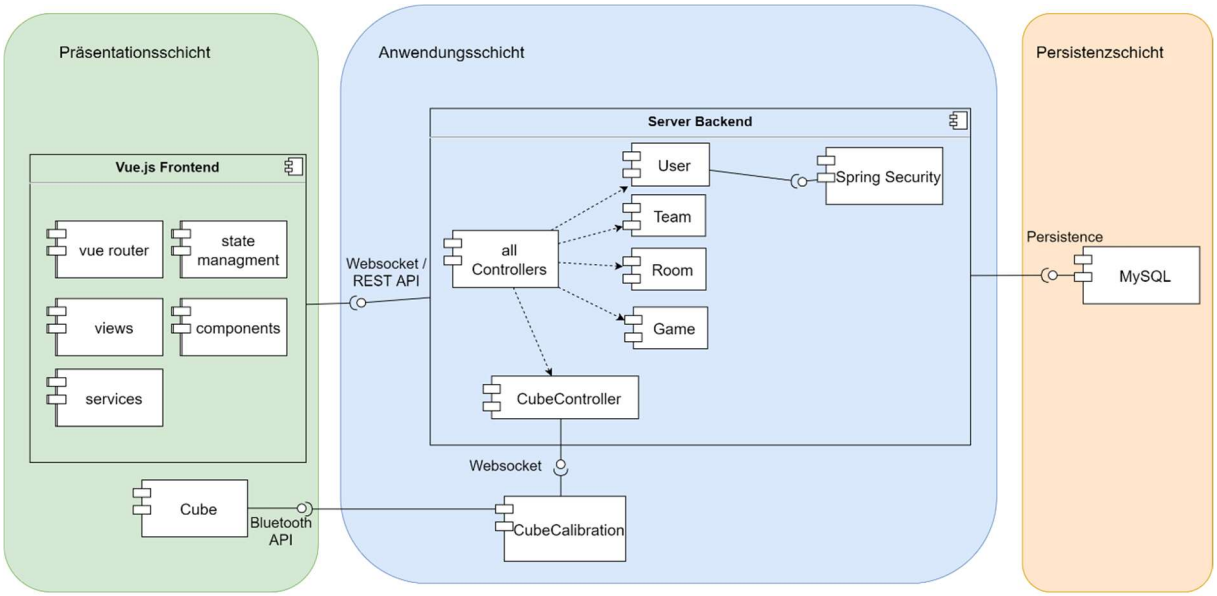
Kommunikationsschnittstelle zwischen Front- und Backend (auch RaspberryPi)

RESTService

Dient zur Erstellung und Übermittlung von REST-Anfragen.

4. SW-Architektur

Komponentendiagramm



Die Architektur ist in drei Schichten aufgeteilt:

- Präsentationsschicht
  Diese beinhaltet alle Module, mit denen der User direkt interagiert.
  Das Frontend läuft im Browser des Users und interagiert teils über eine REST API und teils über half-duplex websockets mit dem Server-backend. Die REST API dient zur einfachen Datenabfrage, während die websockets den Echtzeitaustausch eines Spiels übernimmt. Das Frontend nutzt das JavaScript Framework Vue.js und dessen Komponenten-Architektur.
  Der Würfel (Cube) liegt bei den Spielern und kommuniziert über eine Bluetooth Schnittstelle mit dem RasberryPi (Controller).

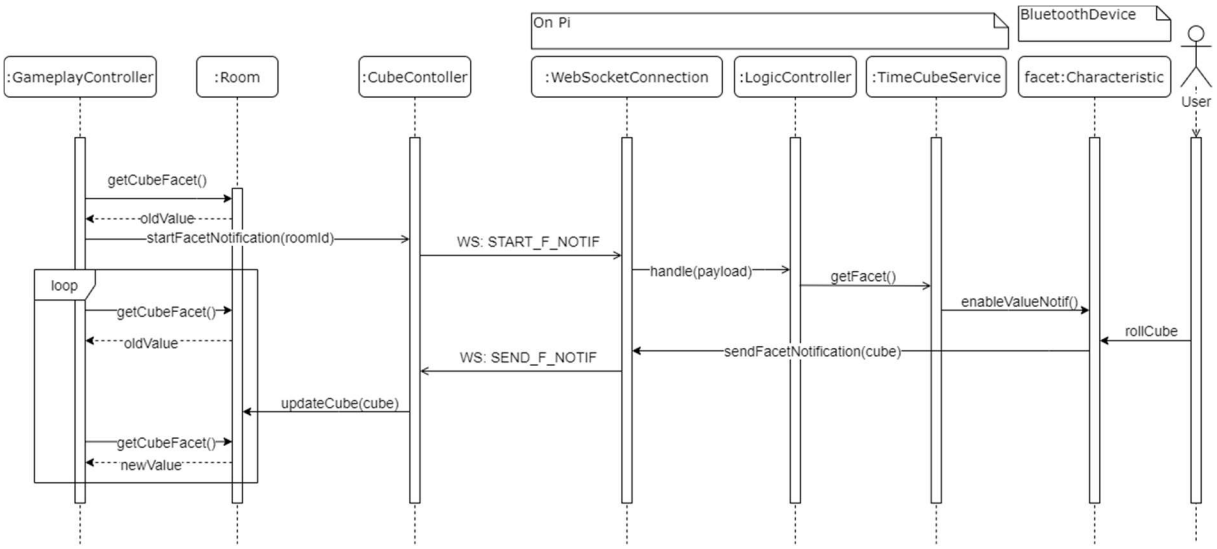
- Anwendungsschicht:
  Das Server-backend nutzt das Java Framework Spring. Über dieses werden die REST API sowie die Websockets für Frontend und RasberryPi implementiert. Die Spring Security verwaltet die Rollen.
  Der Controller ist ein externes Modul, welches auf einem RaspberryPi bei den Spielern liegt. Es bekommt Daten über die Bluetooth Schnittstelle des Würfels und leitet diese an die Websockets des Server-backends weiter.

- Persistenzschicht:
  Als Datenbank wird eine MySQL Datenbank verwendet. Die Anbindung an das Server-backend erfolgt über die Spring Data JPA.

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 Netz steht).

In folgenden Sequenzdiagramm wird erklärt, wie das Gameplay die Benachrichtigung einer Würfelseitenänderung einschaltet und diese bekommt. Da der Pi hier schon mit einem Raum verbunden ist, ist dem MiniComputer somit schon die Raum_id bekannt. Hiermit wird gewährleistet, dass nur der aktuelle Raum die Benachrichtigung bekommt.



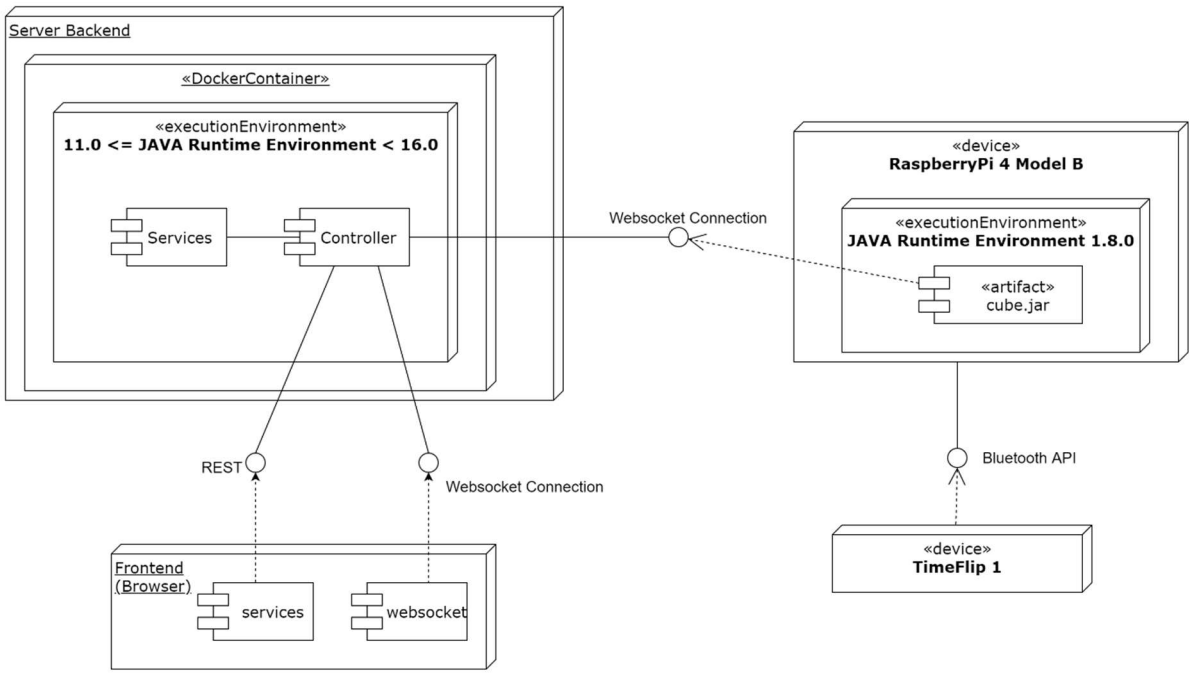
1Vereinfachte Darstellung der Kommunikation zwischen Backend und Würfel
Aus Platzgründgen ist das Wort „Notification“ oft mir „Notif“ abgekürzt.

Der GameplayController steuert die einzelnen Spiele. Nachdem das Spiel gestartet wurde, Teams und Personen zum Raten ausgewählt wurden, wird eine Websocket Nachricht an den Pi geschickt, aus welcher man entnehmen kann, dass er den Raum zu jeder Änderung der Würfelseite benachrichtigen soll. Beim Pi, bearbeitet der LogicController alle Websocket Anfragen, in diesem Fall, wird der (zuvor verbundene) TimeCubeService benutzt, um die FacetValueNotification einzuschalten.

Wenn nun eine Änderung der Würfelseite erfolgt, sendet der Würfel an den Pi eine Benachrichtigung. 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 wird schließlich der WebsocketVerbindung gegeben, um ein WürfelObjekt an das Backend zu senden.

Zwischen dem CubeController und dem Room liegt noch ein RoomService, welcher alle Room Objekte verwaltet und anhand der Id den entsprechenden Room zurückgibt.

Verteilungssicht



Das System besteht aus drei Geräten. Der Würfel, ein RaspberryPi 4 Model B und ein Server. Auf dem Server ist ein Dockercontainer 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.

Zum einen greifen User des Systems auf das Frontend zu. Dieses läuft im Browser und benutzt die Rest Schnittstelle und Websockets, welche von den Controllern im Backend zur Verfügung gestellt werden.
Der CubeController bietet einen zweiten Websocket Chanel an, welcher die Kommunikation mit dem Pi übernimmt. Auf dem Pi läuft ebenfalls eine Java Laufzeitumgebung, jedoch auf Java 1.8.0 damit der Bluetooth Adapter tinyb korrekt läuft.

5. GUI Prototyp

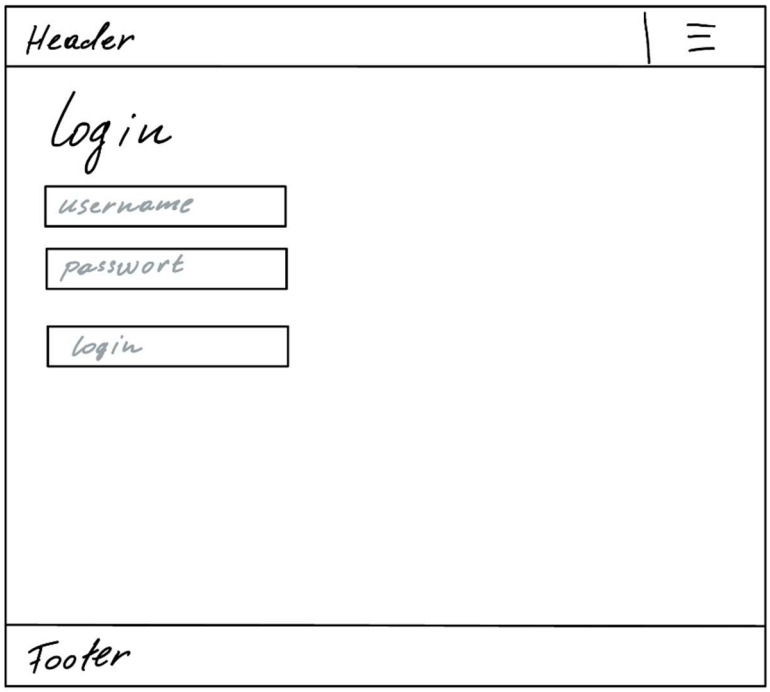
Grober Seitenaufbau

- Login/ Registrieren
- Profile
  - Statistiken
    - Zuletzt gespielte Spiele inkl. Spieler
  - Logout
- Overview
  - Verfügbare Räume
- Raum
  - Teams
  - Beigetretene Spieler
  - Teamverwaltung
  - Teamauswahl
  - Themengebiet
    - Anzahl der Begriffe
    - Schwierigkeit
- Spiel
  - Ratenden Spieler anzeigen
  - Begriffauswahl
    - 3 Zufällige Begriffe aus Themengebiet
  - Simple Würfeln anzeigen
  - Begriff wird angezeigt
    - Ratender Spieler -> Nur Begriff
    - Nicht-ratendes Team -> Begriff, Punktevergabe und Regelverstoß
    - Zuschauer -> Begriff
  - Punkte
  - Restzeit des jetzigen Begriffes
- Dashboard (Spielverwalter)
  - Liste der Räume
    - Punktestand, Themengebiet
  - Liste Themengebiete
    - Optionen zum Verwalten
  - Liste aller Spieler
    - Einsicht auf die Spielerprofile
- Dashboard (Admin)
  - Erbt vom Spieleverwalter, nur mehr Optionen
  - Spieleraccounts anlegen
  - Rollen vergeben

Da es sich um eine Single-Page Applikation handelt, konnten animierte Elemente in dieser statischen Ansicht nicht wiedergegeben werden (Dropdown, Popup, Tooltips).

Login

Sonntag, 14. März 2021 14:41

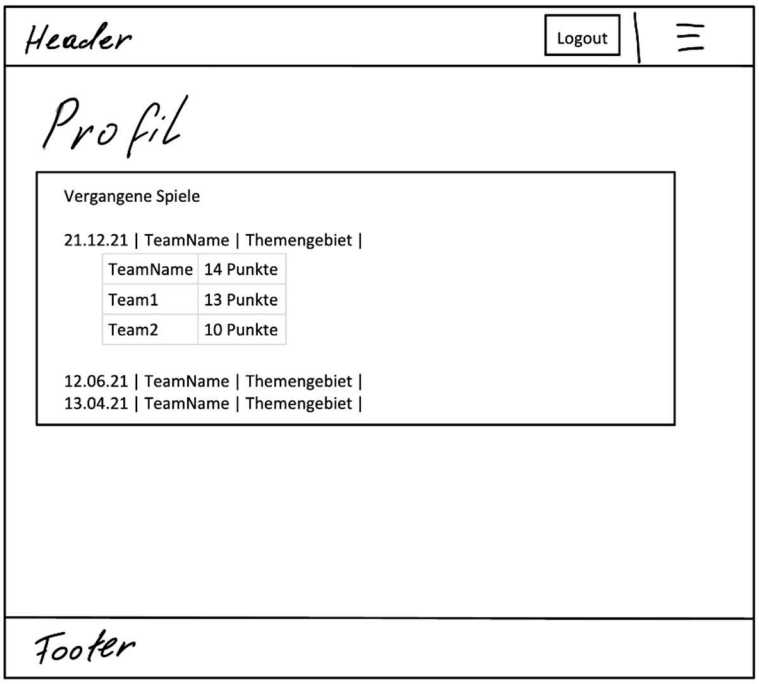


Register

Sonntag, 14. März 2021 14:41

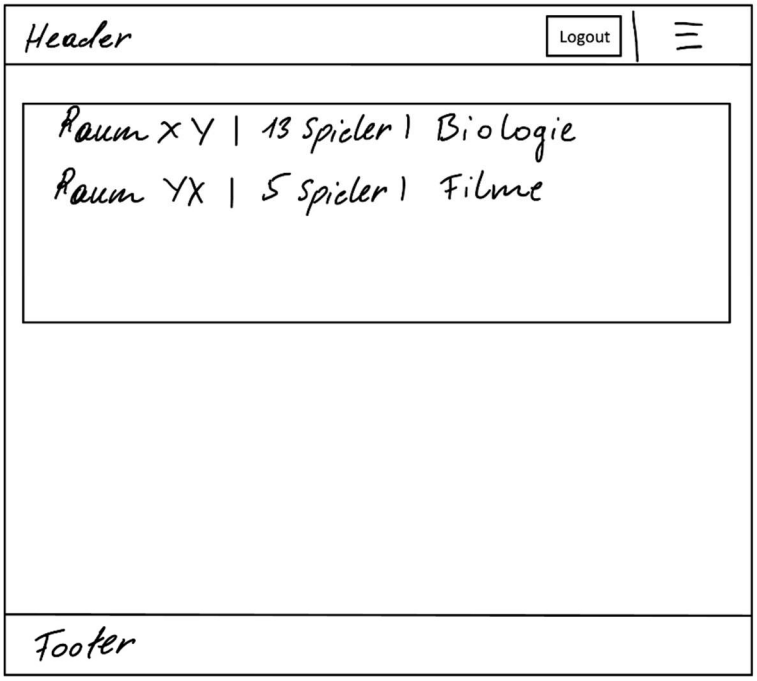
Profil

Sonntag, 14. März 2021 14:41



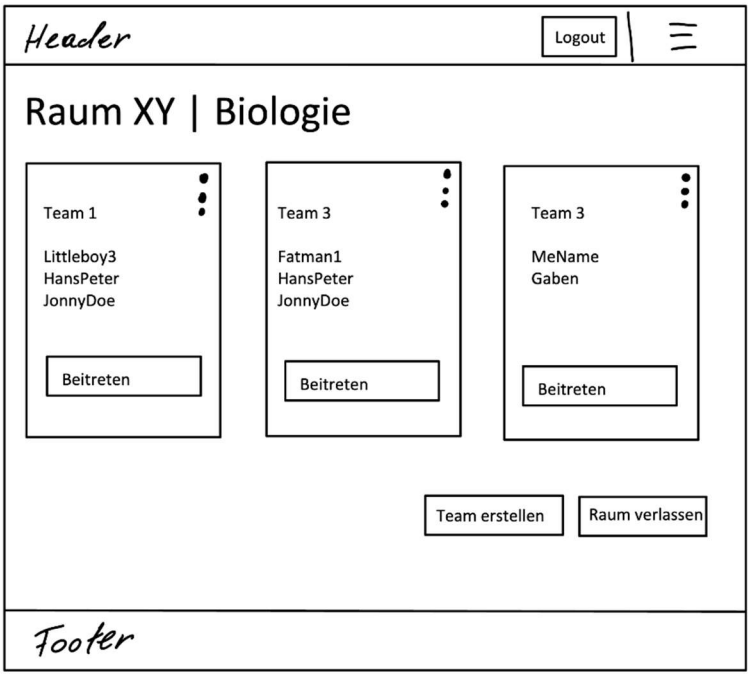
Overview

Sonntag, 14. März 2021 14:41



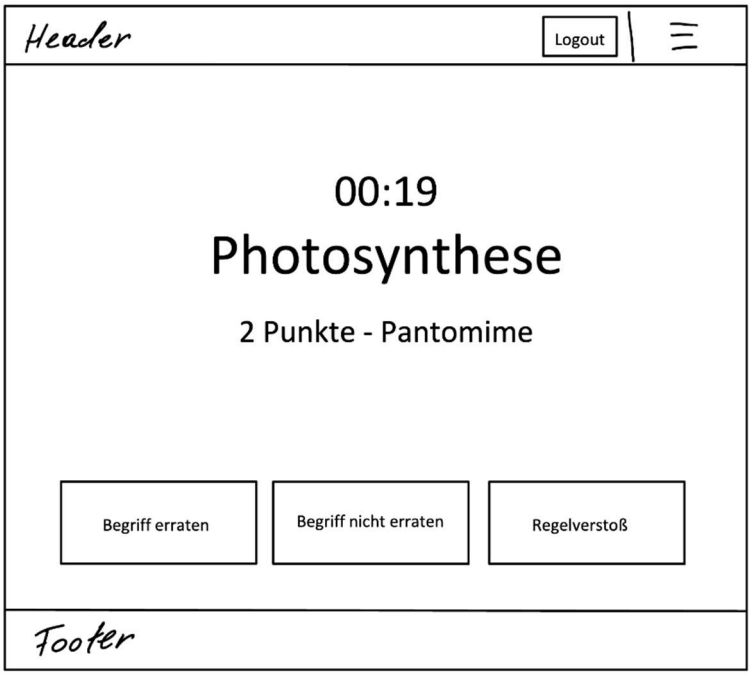
Raum

Sonntag, 14. März 2021 14:41



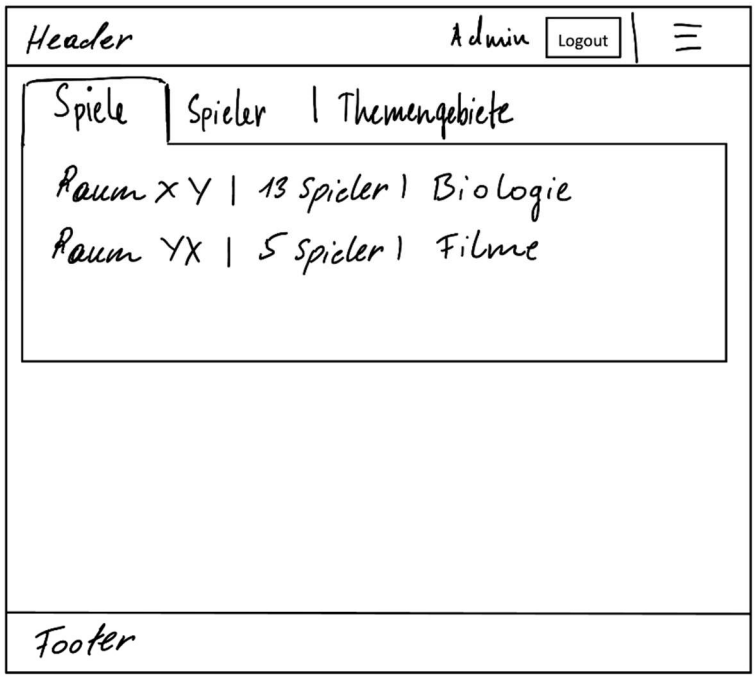
Spiel

Sonntag, 14. März 2021 14:41



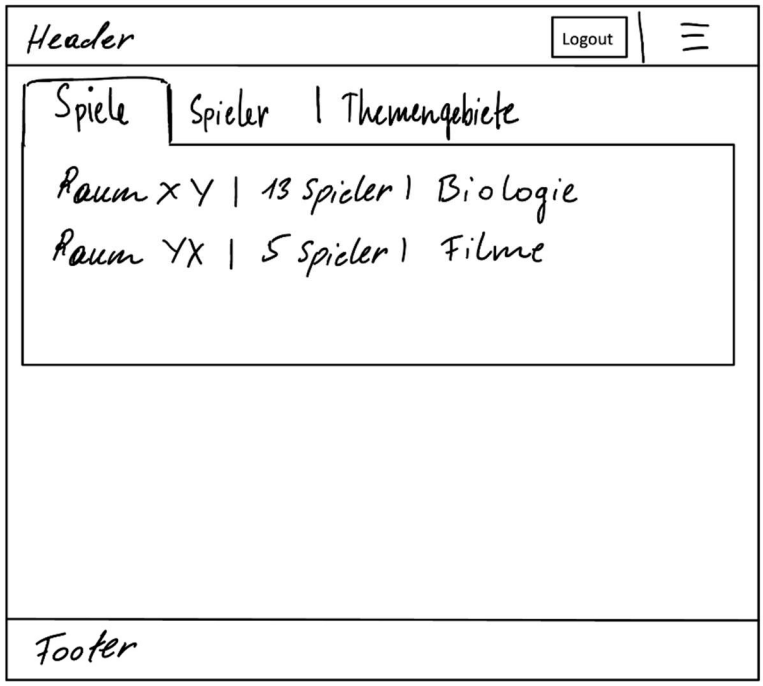
Dashboard(Admin)

Sonntag, 14. März 2021 14:41



Dashboard(Spieleverwalter)

Sonntag, 14. März 2021 14:41



6. Projektplan

Rolleneinteilung

- 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.

Zeitplan

- Wöchentliche JourFixe am Montag 19:00 sowie optionalen Termin Freitag 16:00
- Inkremente:

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

- Project Setup / Konfiguration:
  ER-Diagramm, Glossar, Issues einpflegen, git workflow, Dummy Projekt, Frontend Build Integration, Swagger 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

Daily Updates

Um stets 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).

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, welcher auf Sonntag 21:00 Uhr gesetzt wurde. Danach dürfen keine neuen Features dem nächstfolgenden Release hinzugefügt werden.