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.

Nach einer abgeschlossenen Runde werden ggf. Punkte (durch Bestätigung eines anderen Teams) verteilt und dem jeweiligen Team zugeschrieben. Ein neues Team wird nun 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 somit viele Personen, von Schüler*innen bis zu Freund*innen bei einem gemütlichen Spieleabend. Also von Jung bis Alt.

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, Spielen beitreten, sie verlassen, würfeln und erklären. Er besitzt ein Userprofil mit zahlreichen Statistiken. Ein User kann (gegebenenfalls mehrere) Gastnutzer erstellen.

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 eingeloggt sein. In dem Browser des / der User ist eine Weboberfläche zu sehen, mit den relevanten Spielgeschehnissen (Punkteanzahl, …). Gegebenenfalls sind neben dem User andere User / Gastnutzer vorhanden.

> **Erklärendes / Ratendes Team**
> ist das Team, wovon ein Mitglied einen Begriff erklärt. Die anderen Teammitglieder versuchen diesen zu erraten.
>
> **Bestätigende Teams**
> sind alle Teams abgesehen vom erklärenden Team. Diese bestätigen einen erfolgreich erratenen Begriff und melden etwaige Regelbrüche.

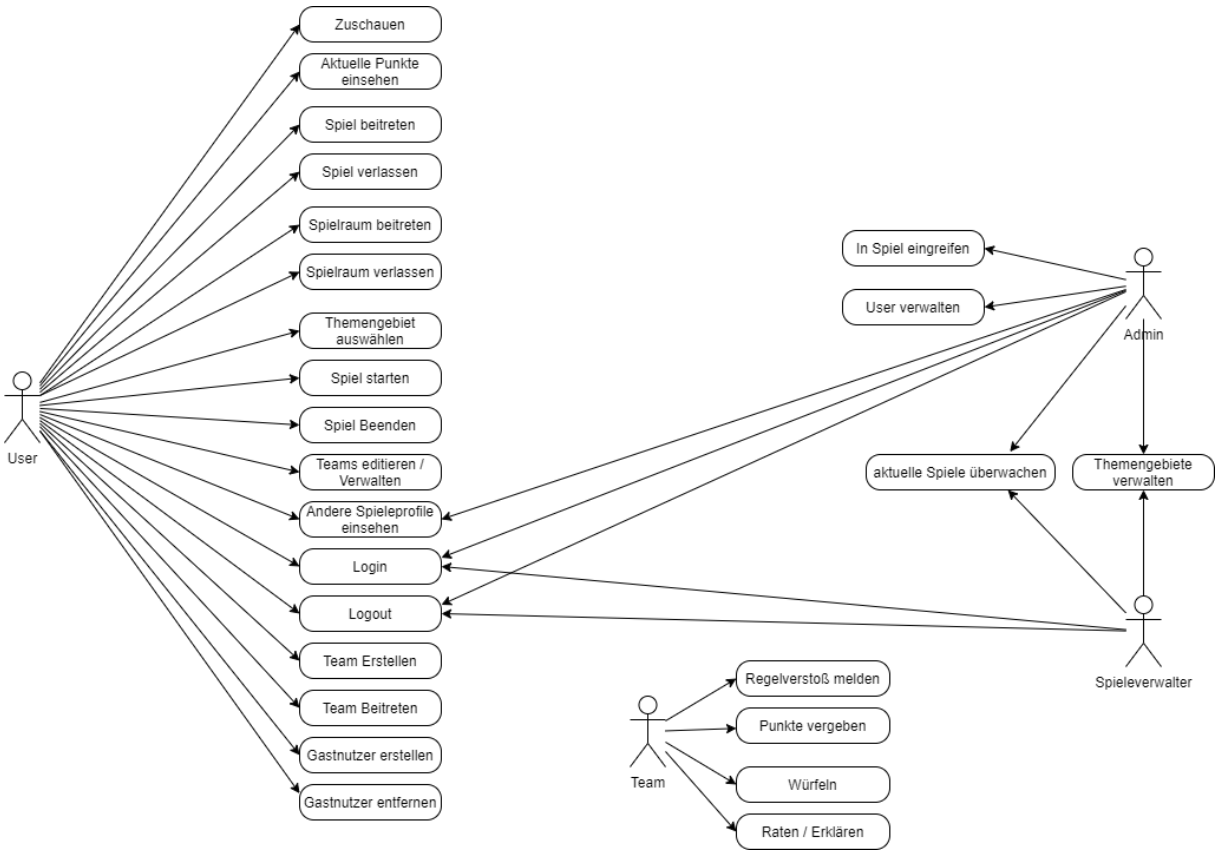
Spieleverwalter

Ein Spieleverwalter sieht alle aktuell laufenden Spiele und deren Zwischenstände. Ebenso darf dieser neue Themengebiete erfassen und erweitern.

Admin

Ein Admin entspricht einem Spieleverwalter mit mehr 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-Case Diagramm



2.3 Use-Cases

2.3.1 Akteur: User

Würfel Verbinden

- *Vorbedingung*: Es existiert ein TimeFlip Würfel und ein RaspberryPi, auf welchem das TimeCube Setup durchgeführt wurde
- *Ablauf:* Die Anwendung wird per Konsole gestartet, die Konfiguration wird korrekt ausgeführt
- *Erfolg:* In der Raum Auswahl kann man einen Würfel / RaspberryPi mit dem eingegebenen Namen auswählen und somit mit einem Raum verbinden.
- *Kein Erfolg*: Der RaspberryPi verbindet sich nicht mit dem Backend. (Dateneingabe falsch? Firewall offen?)
- *Involvierte Klassen*: Raum,

Register

- *Vorbedingung*: Der User hat noch keinen Account, das System läuft, man ist auf der Startseite (Spielauswahl).
- *Ablauf:* Die/Der Benutzer/in gibt Benutzernamen, Email-Adresse und Passwort ein, welches bestätigt werden muss daraufhin klickt er*sie auf „Benutzerkonto erstellen"

- *Erfolg:* Der User erhält eine Bestätigungsmeldung und wird zur Startseite weitergeleitet
- *Kein Erfolg*: Der User erhält eine Fehlermeldung (Validierung schlug fehl)
- *Involvierte Klassen*: User

Login

- *Vorbedingung*: Der User hat einen Account, das System läuft, man ist auf der Startseite (Spielauswahl), der anzumeldende Benutzer existiert.
- *Ablauf:* Die/Der Angestellte gibt seinen Benutzernamen und sein Passwort ein, daraufhin klickt er auf „Login"
- *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, das System läuft, man ist eingeloggt, der anzumeldende Benutzer existiert
- *Ablauf*: Der User wählt die Aktion Logout und bestätigt seine Aktion in einem separaten Popupfenster
- *Erfolg*: Der User ist abgemeldet zurück auf der Startseite
- *Kein Erfolg*: Der User erhält eine Fehlermeldung
- *Involvierte Klassen*: User

Raum erstellen

- *Vorbedingung:* Der Spieler ist eingeloggt, das System läuft, 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 Spielern und Konfigurationsoptionen angezeigt.
- *Kein Erfolg:* Der User erhält einer Fehlermeldung
- *Involvierte Klassen:* User, Raum, Würfel

Raum beitreten

- *Vorbedingung*: Der User ist eingeloggt, das System läuft, ein virtueller Spieleraum ist frei 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, Raum

Raum verlassen

- *Vorbedingung*: Der User ist eingeloggt, das System läuft, der User ist aktuell in einem Spielraum
- *Ablauf*: Der User wählt die Aktion “Spiel verlassen” und bestätigt seine Aktion in einem separaten Popupfenster
- *Erfolg*: Der User wird auf die Startoberfläche weitergeleitet
  - Host verlässt: Ein neuer Host wird ausgewählt
- *Kein Erfolg*: Der User erhält eine Fehlermeldung
- *Involvierte Klassen:* User, Raum

Raum schließen

- *Vorbedingung*: Der User ist eingeloggt, der User ist der Spielhost, der User leitet aktuell einen Spielraum
- *Ablauf*: Der User wählt die Aktion “Spiel beenden” und bestätigt seine Aktion in einem separaten Popupfenster
- *Erfolg:* Der User wird auf die Startoberfläche weitergeleitet und erhält die Erfolgsmeldung “Aktuelles Spiel erfolgreich beendet, Spielraum wurde geschlossen”, die restlichen Spieler in diesem Spielraum erhalten die Meldung “Spieleraum wurde vom Host geschlossen” und werden nach kurzer Zeit auf den Startbildschirm weitergeleitet. Der Raum ist nicht mehr verfügbar.
- *Kein Erfolg*: Der User erhält eine Fehlermeldung
- *Involvierte Klassen*: User, Raum

Team auswählen

- *Vorbedingung*: Der User ist eingeloggt, der User ist in einem aktiven Spielraum, Spiel läuft noch nicht.
- Ablauf: Der User wird zu Anfang zufällig einem Team zugelost. Dieses kann er auf Wunsch wechseln. (Der Host hat das letzte Wort, siehe “Spiel starten”)
- 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, Raum

Themengebiet auswählen

- *Vorbedingung:* Der User ist eingeloggt, das System läuft, der User ist Host eines Spieleraums, der Spielraum wurde noch nicht gestartet
- *Ablauf:* Der Host wähl auf einer Schaltfläche der Themengebiete ein Themengebiet aus.
- *Erfolg:* Allen Spielern im Raum wird das neue Themengebiet angezeigt. In der Lobbyübersicht wird das Thema des Raums aktualisiert.
- *Kein Erfolg:* Dem Host wird eine Fehlermeldung angezeigt.
- *Involvierte Klassen*: Themengebiet, User, Raum

Spiel starten

- *Vorbedingung:* Der Spieler ist eingeloggt, befindet sich in einem Spielraum und ist dessen Host, das Spiel ist noch nicht gestartet, es sind genügend Spieler vorhanden
- *Ablauf:* Der Host klickt auf "Spiel starten" und hat die Möglichkeit die Teamvorauswahl der Spieler zu überschreiben. Danach bestätigt er und startet damit das Spiel.
- *Erfolg:* Alle eingeloggten Spieler landen in der Spielansicht und das Spiel beginnt.
- Kein Erfolg: Je nach Fehler erhalten die betroffenen Spieler eine Fehlermeldung.
- *Involvierte Klassen*: Team, User, Raum, Themengebiet

Userprofile einsehen

- *Vorbedingung:* Das System läuft*, d*er gesuchte User existiert, der User wurde gefunden
- *Ablauf:* Der User klickt auf eine Schaltfläche "Profil ansehen"
- *Erfolg:* Der User sieht die Profilseite des gesuchten Users
- *Kein Erfolg:* Eine Fehlermeldung wird angezeigt.
- *Involvierte Klassen:* User

Aktuelle Punkte einsehen

- *Vorbedingung:* Man befindet sich in einem Raum;
- *Ablauf*: Jedes Mitglied hat zu jeder Zeit des Spiels Einsicht auf den aktuellen Punktestand (Anzeige in der web-app)
- *Erfolg*: Jedes Teammitglied sieht den aktuellen Punktestand
- *Kein Erfolg*: -
- *Involvierte Klassen*: Team, Raum

Zuschauen

- *Vorbedingung*: Ein Spiel läuft

- *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 etc)
- *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*: Raum, Team, User

2.3.2 Akteur: Teams

Regelverstoß melden

- *Vorbedingung*: Das Spiel läuft; man befindet sich in einem Raum; man befindet sich in der Rolle eines NICHT-ratenden Teams; ein RaspberryPi ist ausgewählt (somit ist der Würfel mit diesem verbunden)
- *Ablauf*:
    - Einer der Teammitglieder klickt auf den Menüpunkt “Regelverstoß melden”
    - Es wird nach einer erneuten Bestätigung gefragt, welche entweder mit “Regelverstoß senden” bestätigt wird oder mit “Abbrechen” abgebrochen wird.
- *Erfolg*: Der Regelverstoß wird bestätigt; Das ratende Team bekommt einen Punkt abgezogen.
- *Kein Erfolg*: Die Aktion wird abgebrochen. Es werden keine weiteren Veränderungen vorgenommen.
- *Involvierte Klassen*: Team, User

Punkte vergeben

- *Vorbedingung:* Das Spiel läuft; man befindet sich in einem Raum; man befindet sich in der Rolle des NICHT-ratenden Teams
- *Ablauf:*
    - Einer der Teammitglieder klickt auf den Menüpunkt “Punkte vergeben”
    - Die möglichen Optionen werden angezeigt
        - “Begriff wurde erraten” -> Punkte werden vergeben (abzulesen Anhand des Würfels)
        - “Begriff wurde NICHT erraten” -> Keine Punkte werden vergeben
- *Erfolg*: Der Punktestand wird gegebenenfalls angepasst.
- *Kein Erfolg*: Fehlermeldung wird angezeigt (z.B.: Lobby existiert nicht mehr...)
- *Involvierte Klassen*: Team, User, Raum

Würfeln

- *Vorbedingung*: Ein Spiel läuft; Man befindet sich in einem Raum; man befindet sich in der Rolle des ratenden Teams;

- *Ablauf*:
    - Der Würfel wird geworfen
    - Die nach oben zeigende Fläche wird erkannt und registriert (Punkte, Zeit, Art der Aktivität)
    - Die Informationen werden an den RaspberryPi gesendet
    - Die Informationen werden verarbeitet und dementsprechend auf den Bildschirmen der Teilnehmer angezeigt (z.B.: “Team 1 muss zeichnen, hat 1 Minute Zeit und kann 3 Punkte erzielen)
- *Erfolg*: Die Teilnehmer / Teams erfahren, was zu tun ist und erhalten Informationen
- *Kein Erfolg*: Fehlermeldung (z.B.: “Batterie ist leer”)
- *Involvierte Klassen*: Team, Raum, Würfel

Raten/Erklären

- *Vorbedingung:* Das Spiel läuft; man befindet sich in einem Raum; man befindet sich in der Rolle des ratenden Teams; Man hat bereits gewürfelt und einen Begriff / eine Aktivität zugeordnet bekommen
- *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)
    - Regelverstoß (siehe Regelverstoß)
- *Involvierte Klassen*: Team, Raum

2.3.2 Akteur: Admin

Spielaccounts verwalten

- *Vorbedingung*: Benutzer mit Recht “Admin” existiert, Ist eingeloggt
- *Ablauf*: Auf seinem Home-Screen 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*: Benutzer mit Recht “Admin” existiert, Ist eingeloggt
- *Ablauf*: Auf seinem Home-Screen 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, Raum, (User)

Spiel überwachen

- Wird von Spieleverwalter geerbt

Themengebiete verwalten

- Wird von Spieleverwalter geerbt

2.3.3 Akteur: Spieleverwalter

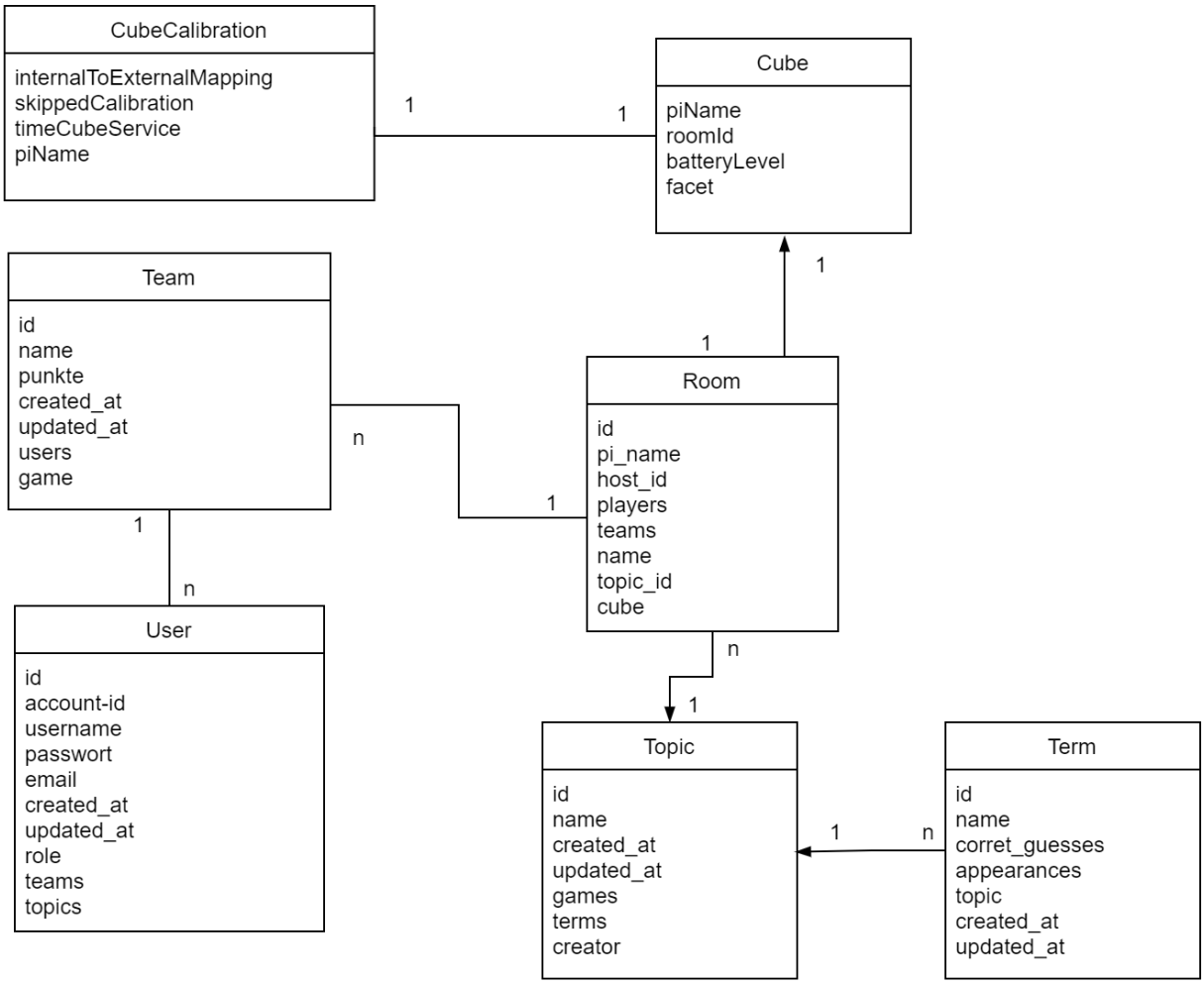
Spiel überwachen

- *Vorbedingung*: Benutzer mit Recht “Spieleverwalter” oder “Admin” existiert, Ist eingeloggt
- *Ablauf*: Auf seinem Home-Screen sieht er eine Liste der aktiven Spiele. Er kann auf ein Spiel klicken und sich die Details dieses Spieles anschauen.
- *Erfolg:* Er sieht die Details dieses Spiels
- *Kein Erfolg:* Fehlermeldung wird angezeigt. (z. B.: Spiel existiert nicht mehr.)
- *Involvierte Klassen:* Spieleverwalter, (Admin), Raum

Themengebiete verwalten

- *Vorbedingung:* Benutzer mit Recht “Spieleverwalter” oder “Admin” existiert, Ist eingeloggt
- *Ablauf:* Nutzer sieht 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), Themengebiet, Begriff

3. Klassendiagramm



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

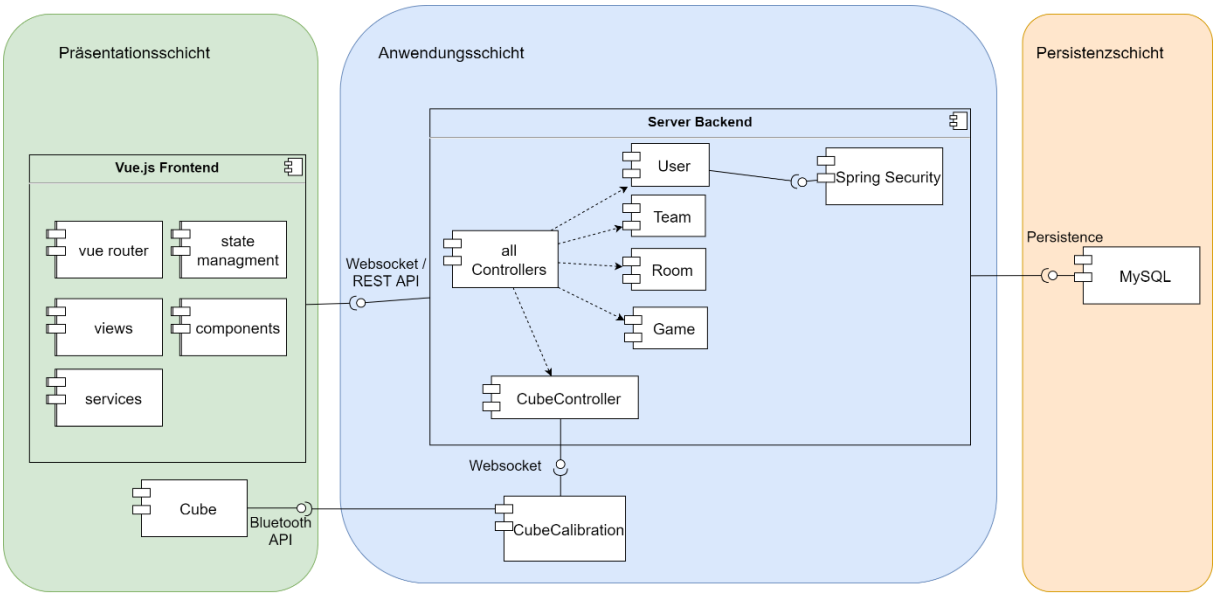
Kommunikationsschnittstelle zwischen Front- und Backend (auch RaspberryPi)

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 einen Websocket mit dem Server Backend. Die REST API dient zur einfachen Datenabfrage, während der Websocket den Echtzeitaustausch während 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).

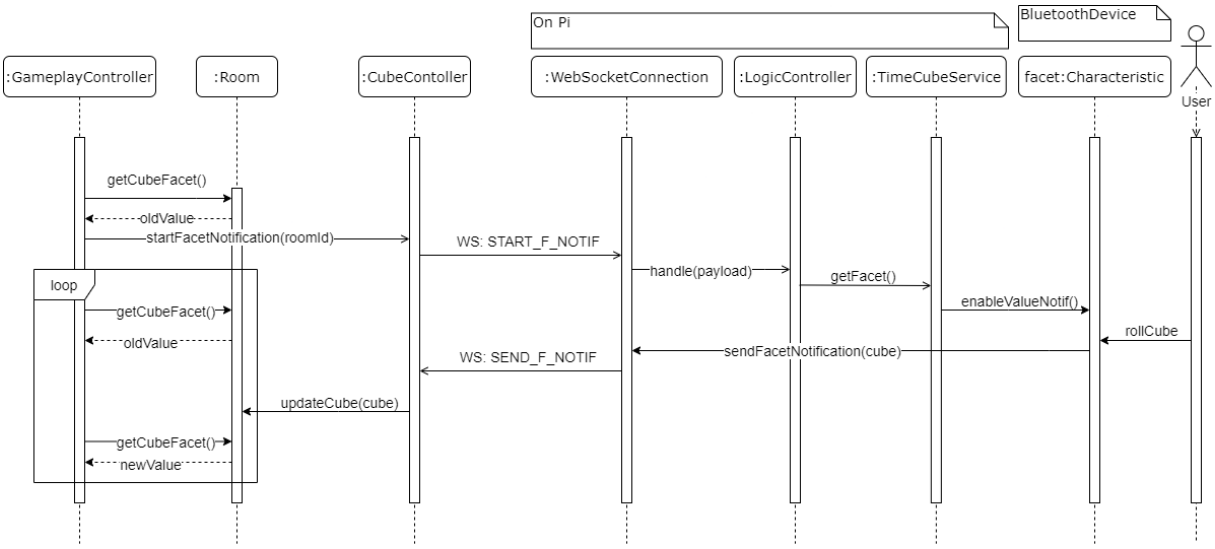
- Anwendungsschicht:
  Das Server Backend nutzt das Java Framework Spring. Über dieses werden die REST API sowie die Websockets für Frontend und RasberryPi implementiert. Die Spring Security verwaltet die Rollen. Das GameManagment definiert die Spiellogik, die Einstellungen für den Spieleverwalter und die Spielstatistiken.
  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 den Websocket des Server Backends weiter.

- Persistenzschicht:
  Als Datenbank wird eine MySQL Datenbank verwendet. Die Anbindung an das ServerBackend 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 Netzsteht).

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 RaumId bekannt. Hiermit wird gewährleistet, dass nur der aktuelle Raum die Benachrichtigung bekommt und nicht ein anderer.



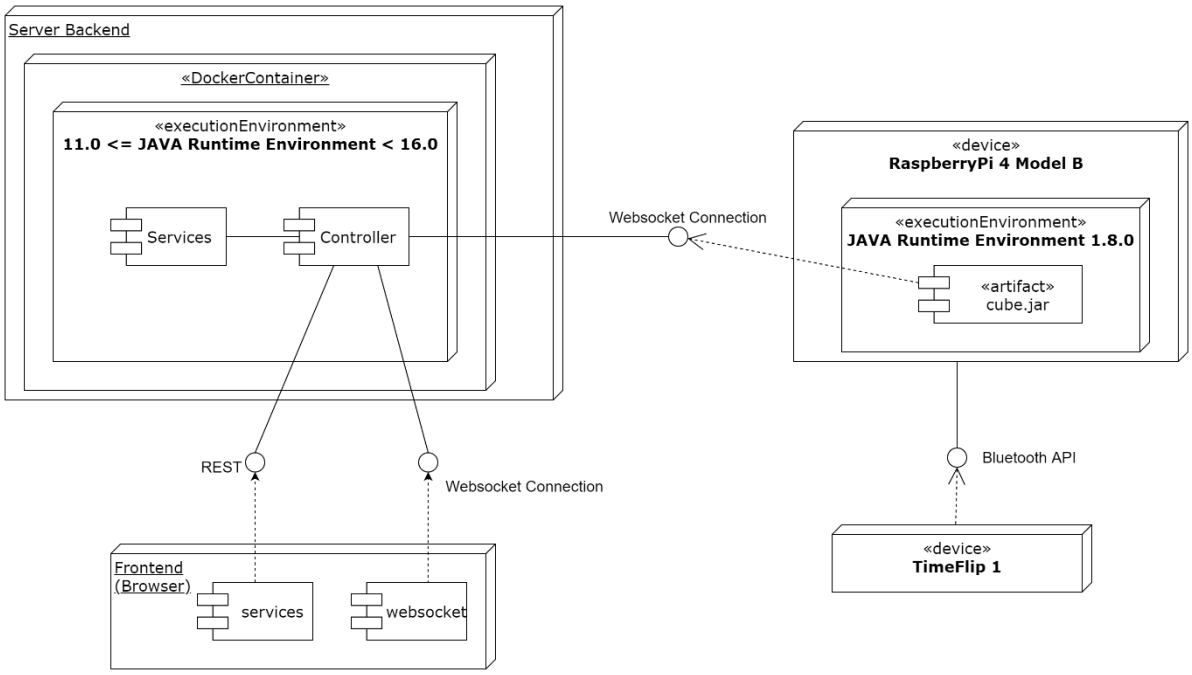
1Vereinfachte Darstellung der Kommunikation zwischen Backend und Würfel
Aus Platzgründgen ist das Wort „Notification“ oft mir „Notif“ abgekürzt.

Der GameplayController steuert die einzelnen Spiele. Nachdem das Spiel gestartet wurde, Teams und Personen zum Raten ausgewählt wurde, wird eine Websocket Nachricht an den Pi geschickt, 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 aus der Characteristic des Würfels. Danach wird die zuvor Durchgeführte WürfelKalibierung (hier nicht dargestellt) benutzt, um genau eine der 12 möglichen Würfelseiten zuzuweisen. Diese wird schließlich der WebsocketVerbindung gegeben, um ein WürfelObjekt an das Backend mittzuteilen.

Zwischen dem CubeController und dem Room liegt noch ein RoomService, welcher alle Roomobjekte verwaltet und anhand von 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

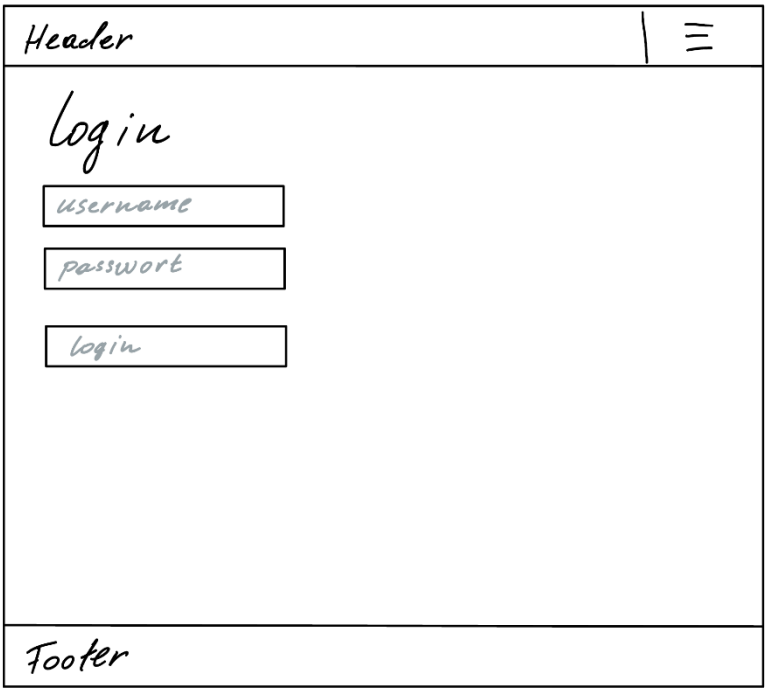
Grober Seitenaufbau

- 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).

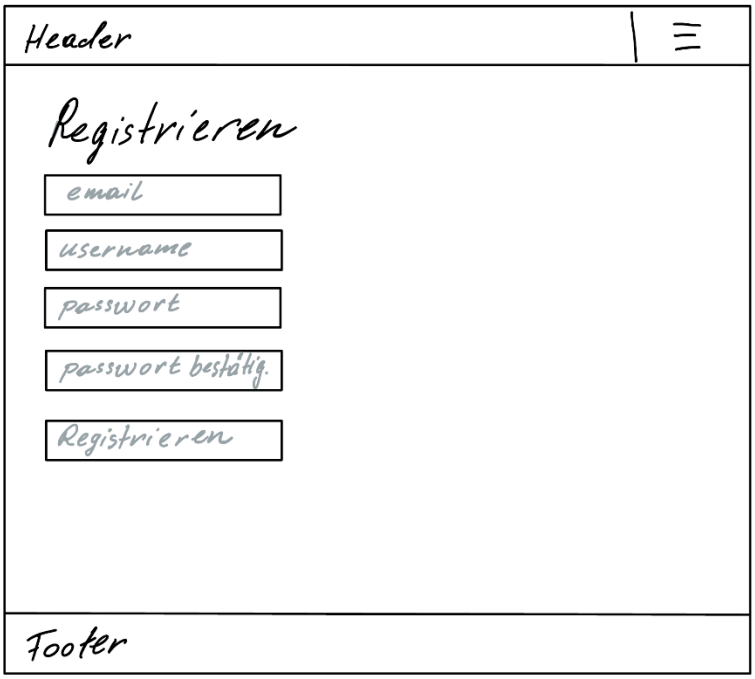
Login

Sonntag, 14. März 2021 14:41



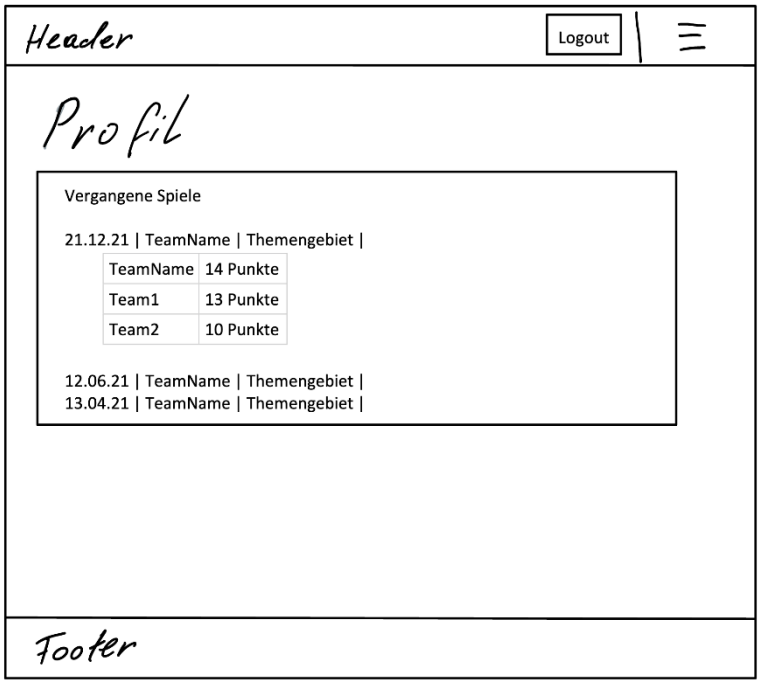
Register

Sonntag, 14. März 2021 14:41



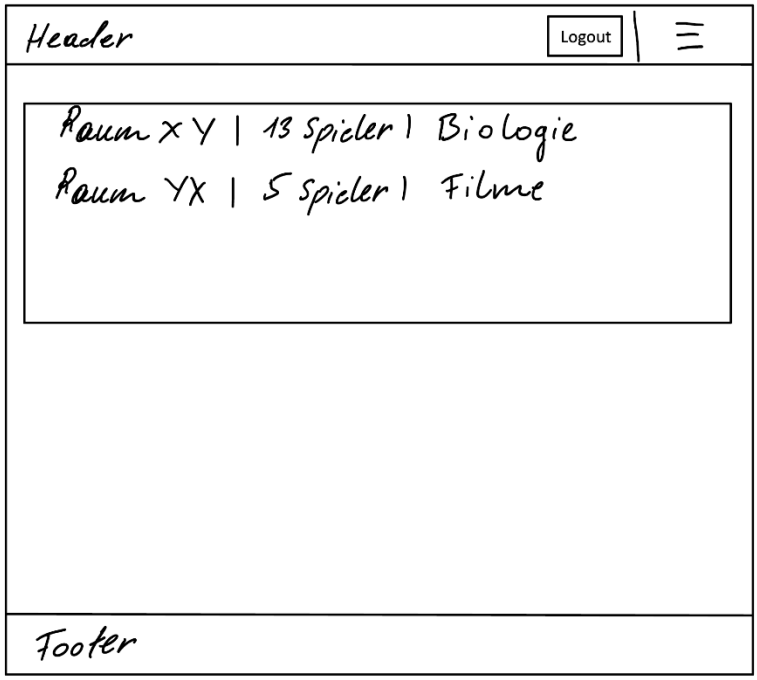
Profil

Sonntag, 14. März 2021 14:41



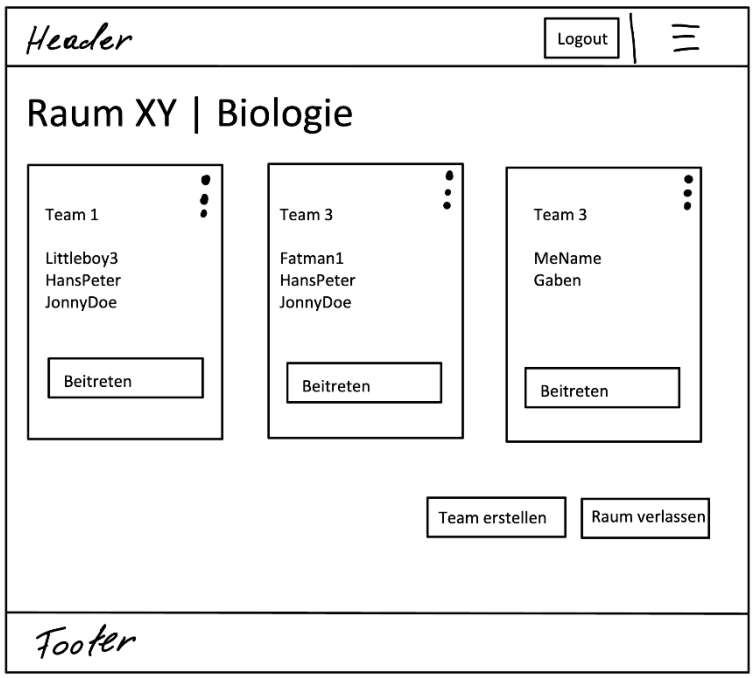
Overview

Sonntag, 14. März 2021 14:41



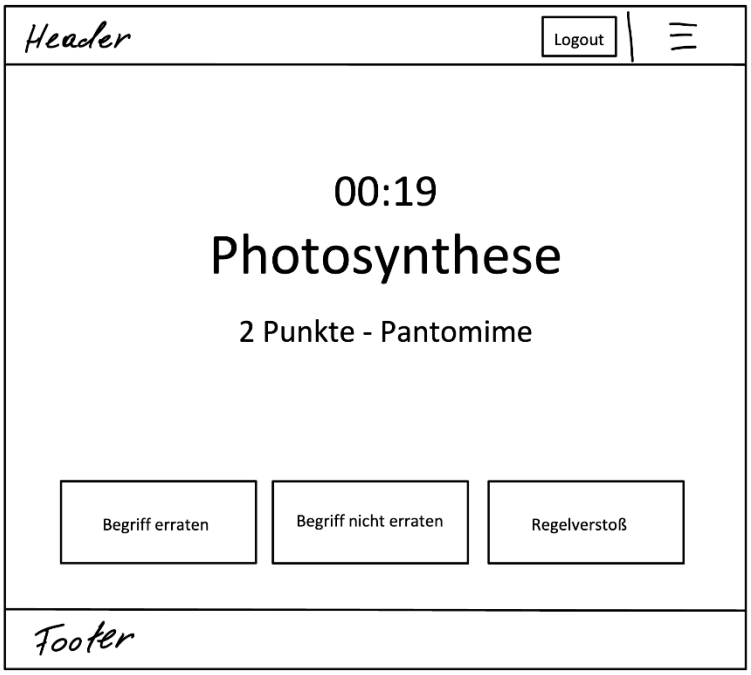
Raum

Sonntag, 14. März 2021 14:41



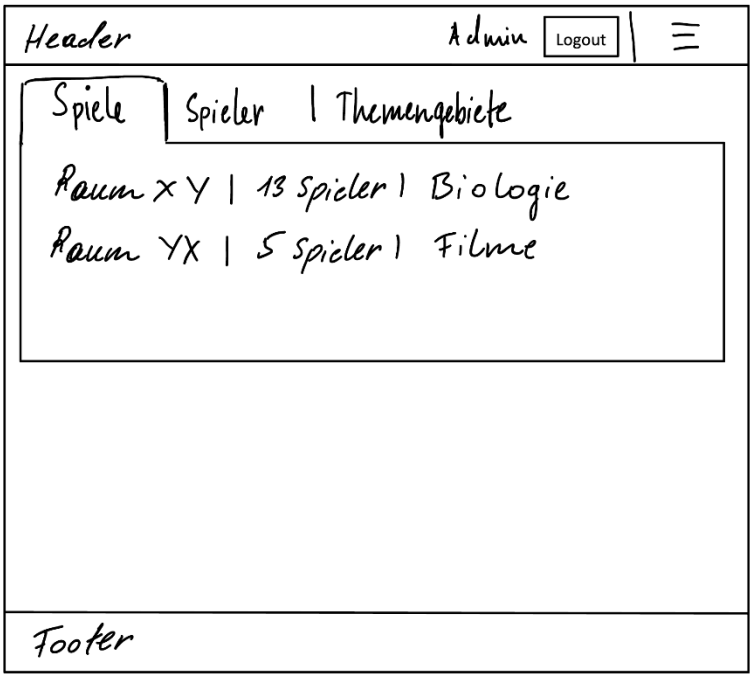
Spiel

Sonntag, 14. März 2021 14:41



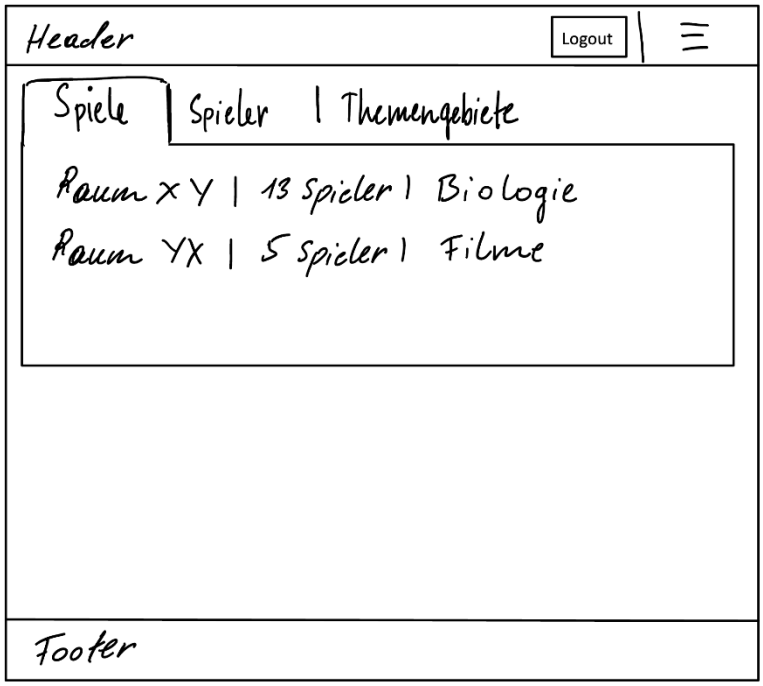
Dashboard(Admin)

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.