Konzeptbeschreibung

**Team: <SpeziRangers/NR.3>**

Mitglied 1: (Bernhard Astl, 01319305)

Mitglied 2: (Martin Beyer, 11909749)

Mitglied 3: (Nicolas Marte, 11909113)

Mitglied 4: (Islam Mechtjev, 11910366)

Mitglied 5: (Martin Neuner, 11917314)

Mitglied 6: (Clemens Prosser, 11907449)

**Proseminargruppe: <6>**

**Datum: <18.03.2021>**

# 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.   
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. 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.

# 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 Userprofile mit zahlreichen Statistiken.

**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, …)

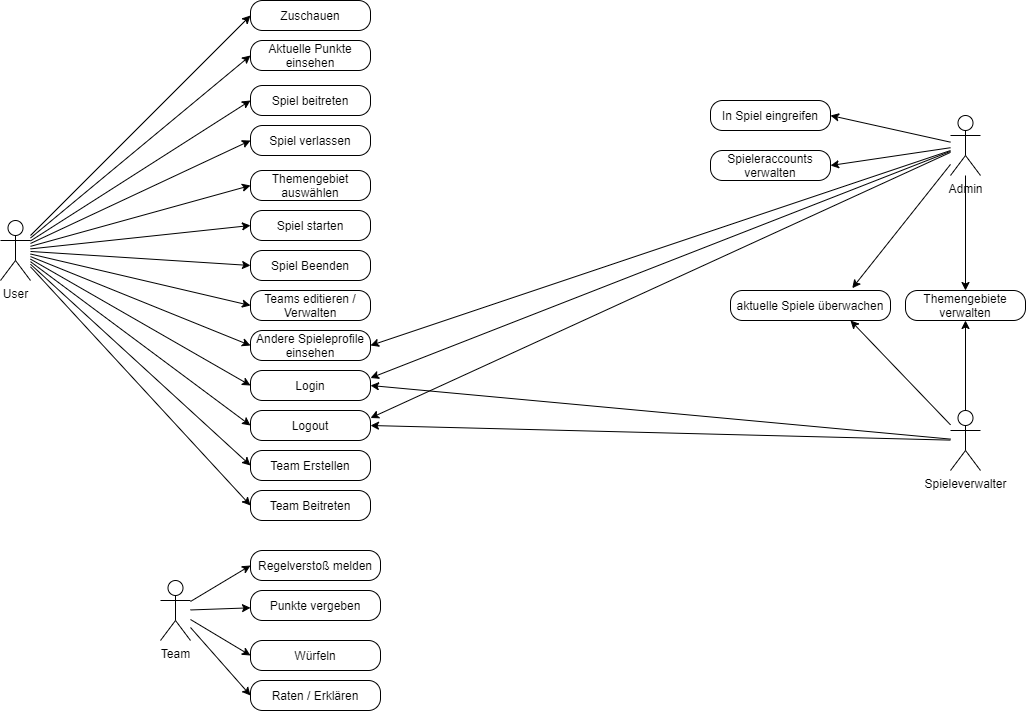
Erklärendes / Ratendes Team  
Ist ein Team, wovon ein Mitglied einen Begriff erklärt. Die anderen Teammitglieder versuchen diesen zu erraten.

Bestätigendes Team  
Ist ein Team, welches antworten von Gegnern bestätigt und etwaige Regelbrüche meldet.  
Erklären ist immer auf (mündlich, pantomimisch, zeichnerisch) bezogen.

**Admin**  
Ein Admin verwaltet User und Themengebiete, kann neue hinzufügen, hat in diese Einsicht.

**Spieleverwalter**  
Ein Spieleverwalter sieht alle aktuell laufenden Spiele und deren Zwischenstände. Weiter hat er auch die Möglichkeit Spielebegriffe zu importieren.

## 2.2 Use-Case Diagramm



2.3 Use-Case

### 2.3.1 Akteur: User

**Login**

* *Vorbedingung*: Der User hat einen Account, das System läuft, man ist auf der Startseite (Lobby), 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.
* *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

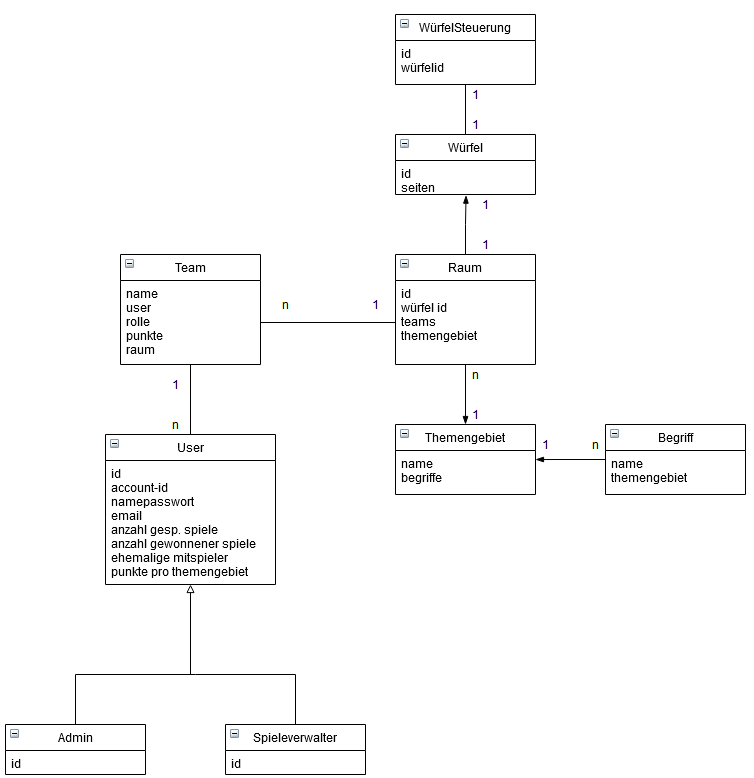
**Spiel überwachen**

* *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

# Klassendiagramm



## 3.1 Backend

### 3.1.1 Entities

**User**

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

**Admin**

Erbt von User. Ein Admin ist ein User mit den höchsten Berechtigungen, wie z. B. Userverwaltung und das Verwalten von Themengebieten. Kann auch in Spiele eingreifen.

**Spieleverwalter**

Erbt von User. Kann Spielebegriffe importieren, Teams ändern

**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 von der Würfelsteuerung erfasst und verarbeitet.

**Würfelsteuerung**

Diese Klasse hat 2 Funktionen. Einerseits wird stetig überprüft ob und welche TimeFlip-Würfel derzeit mit dem Pi in Verbindung stehen. Andererseits wartet diese Klasse auf Nachrichten einzelner Würfel. Geschieht dies, werden mithilfe des REST-Services Daten Anfragen an den zugeteilten Server übermittelt.

### 3.1.2 Zukünftig geplant

**Repositories**

Zu jeder Entity-Klasse wird ein Repository angelegt. Dieses enthält SQL-Queries welche sich auf Suchanfragen im Entsprechenden Table beschränken.

**Services**

Klasse welche als Schnittstelle zwischen der Java Anwendung und der Datenbank dient. Verwendet Funktionen aus Repositories wie z.B. Löschen und Erstellen von User.

**Controller**

Schnittstelle zwischen Front- und Backend.

**RESTService**

Dient zur Erstellung und Übermittlung von REST-Anfragen.

**Config**

Config-Datei auf einem Rasperry PI welche bei Installation des Programmes auf dem PI erstellt wird. Enthält Informationen zu Raum, Würfel, User, Team und Themengebiete.

# SW-Architektur

## 4.1 Komponentendiagramm

Die Architektur ist in drei Schichten aufgeteilt:

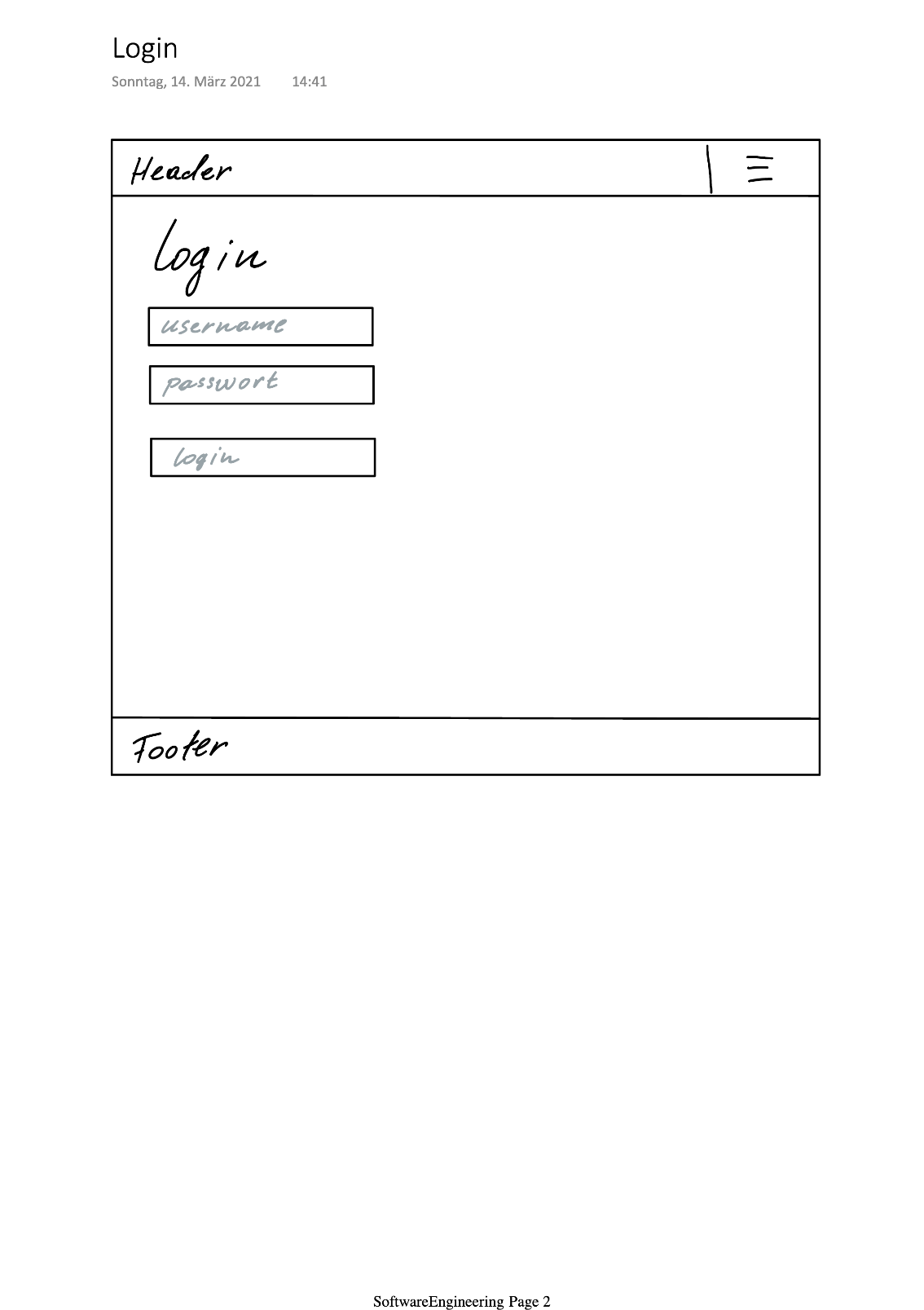
* Präsentationsschicht  
  Diese beinhaltet alle Module, mit denen der User direkt interagiert.  
  Das Frontend läuft im Browser des Users und interagiert teils über eine REST API und teils über einen Websocket mit dem Server Backend. Die REST API dient zur einfachen Datenabfrage, während der Websocket 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.

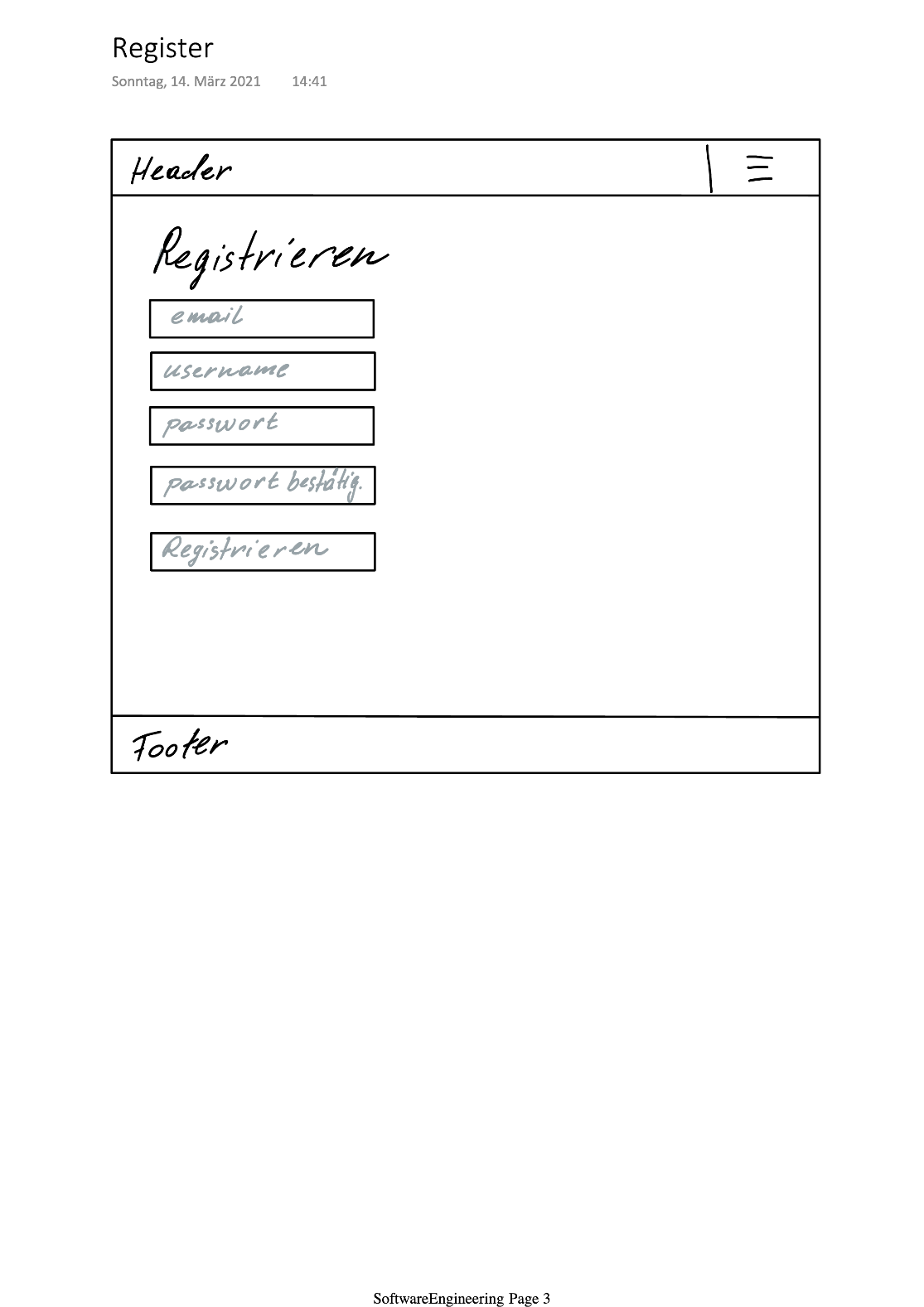
# GUI Prototyp

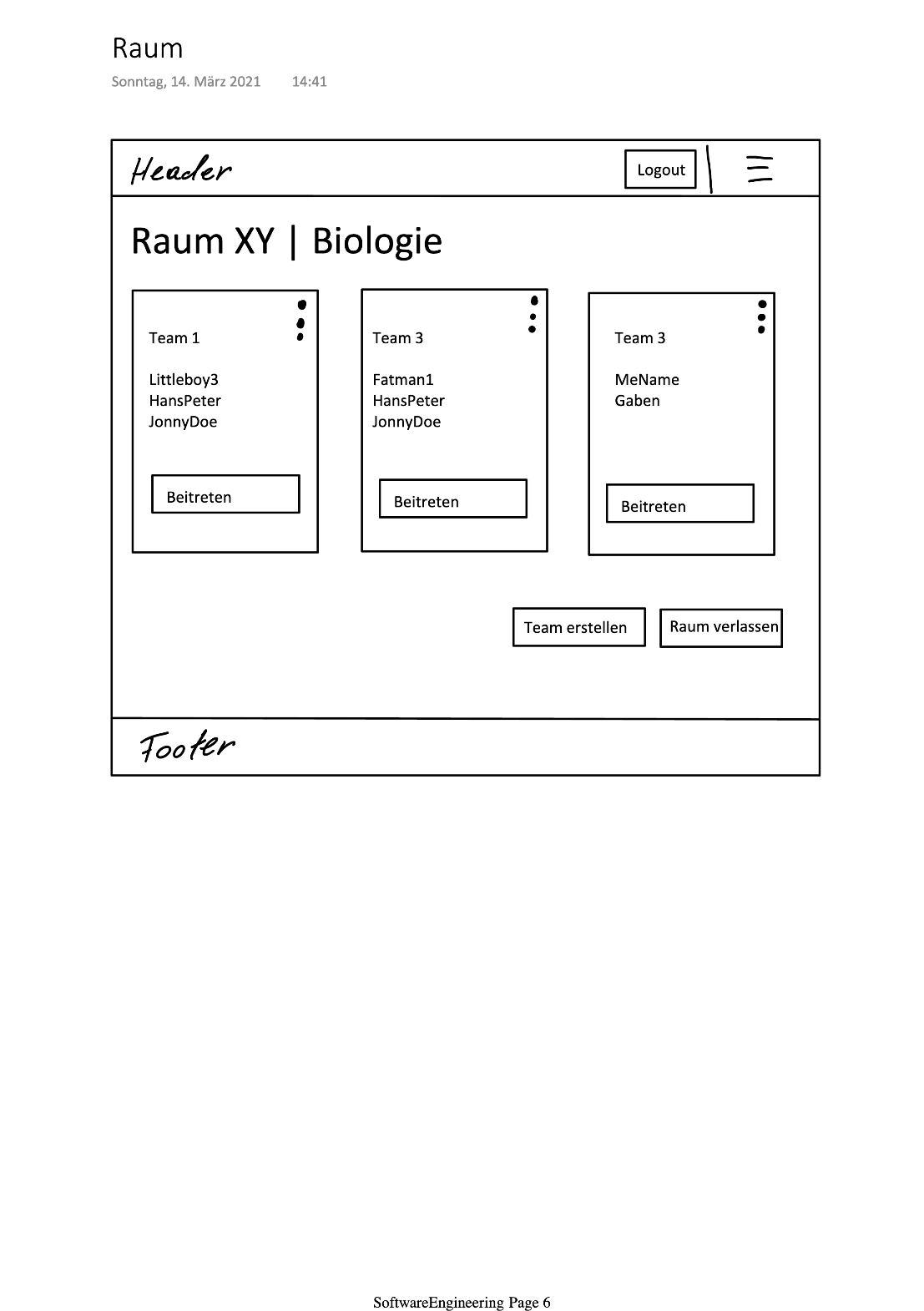
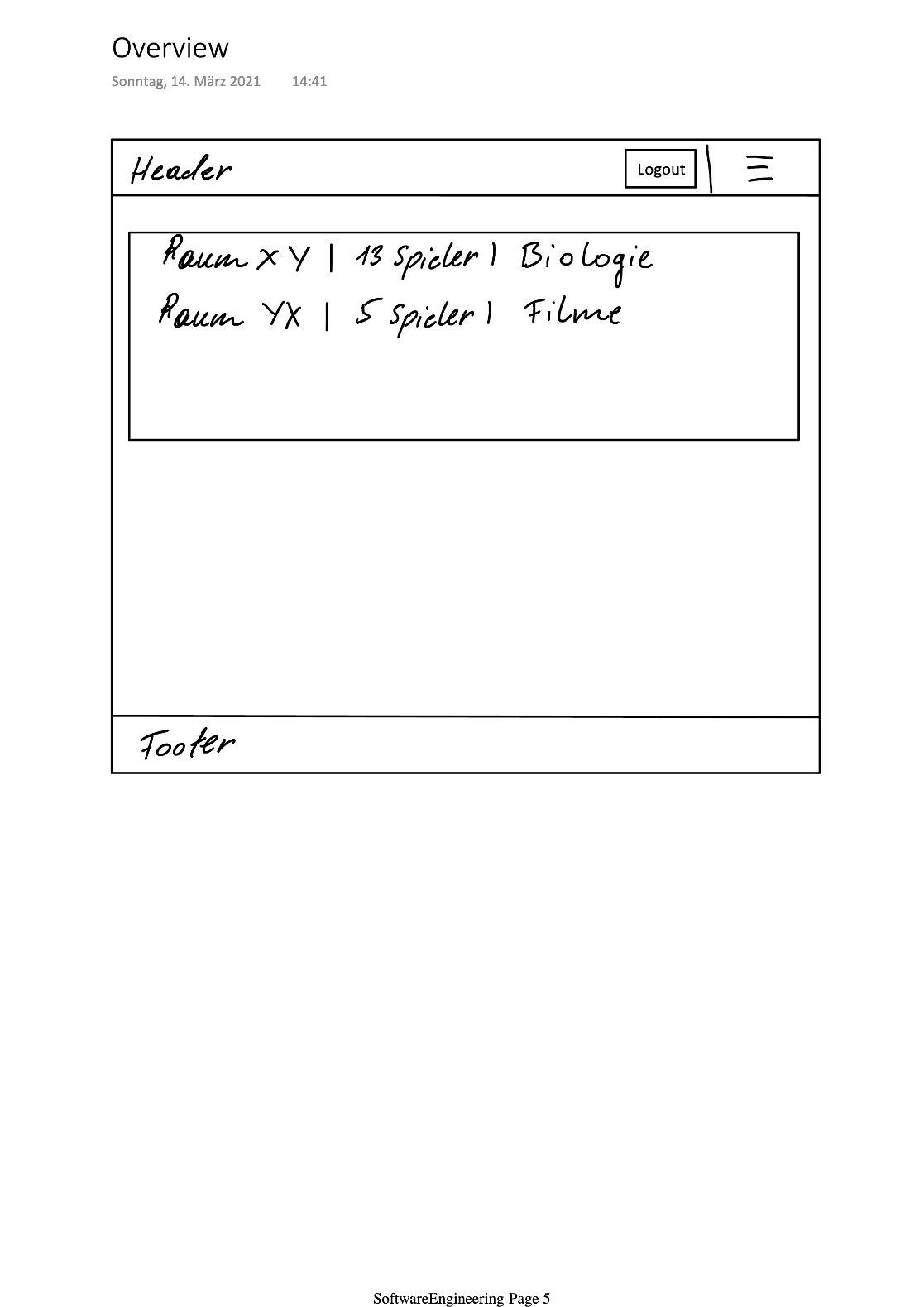
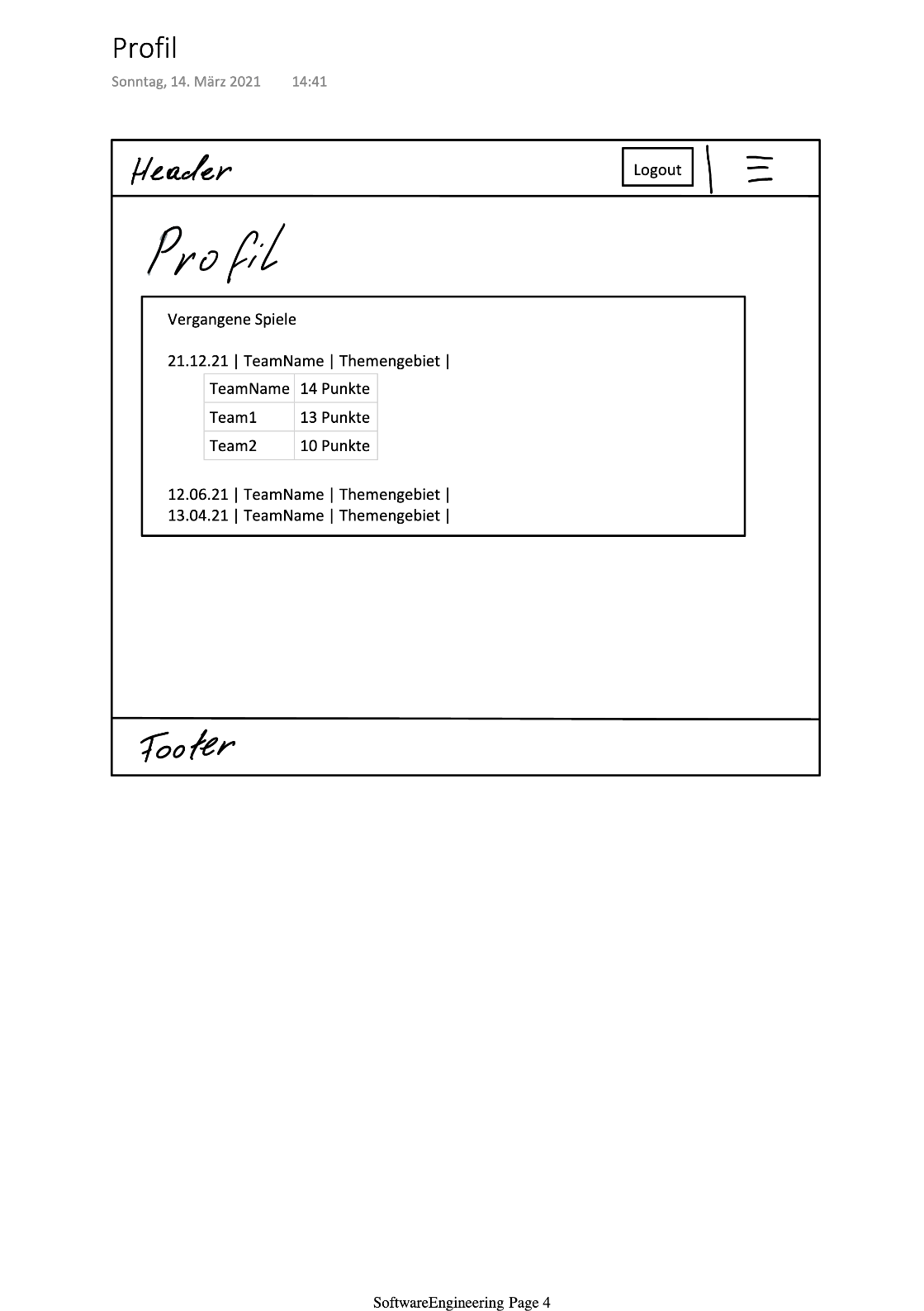
Grober Seitenaufbau

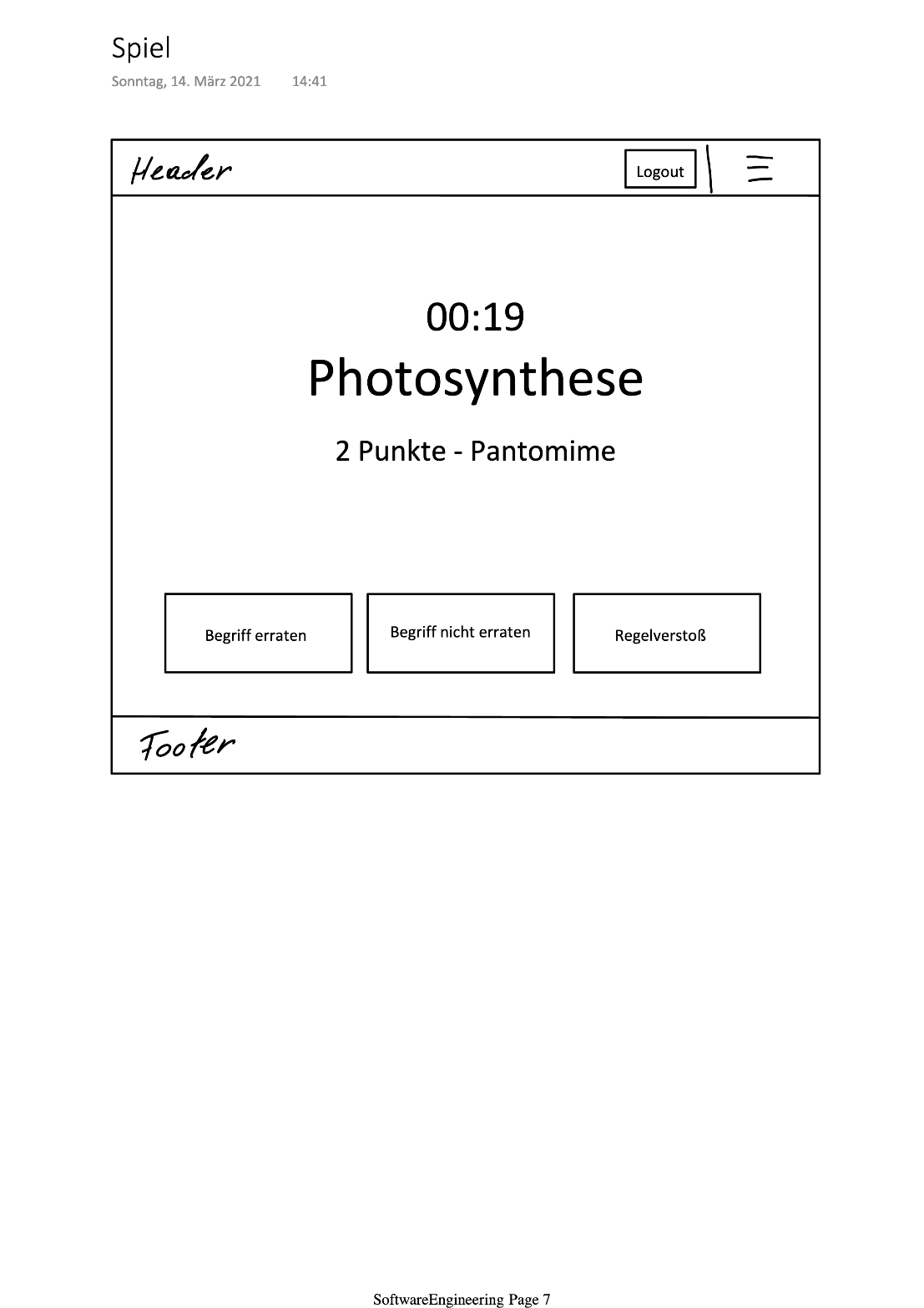
* Login/ Registrieren
* Profile
  + Statistiken
    - Zuletzt gespielte Spiele inkl. Spieler
  + Logout
* Overview
  + Verfügbare Räume
* Raum
  + Teams
  + Beigetretene Spieler
  + Teamverwaltung
  + Teamauswahl
  + Themengebiet
    - Anzahl der Begriffe
    - Schwierigkeit
* Spiel
  + Ratenden Spieler 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

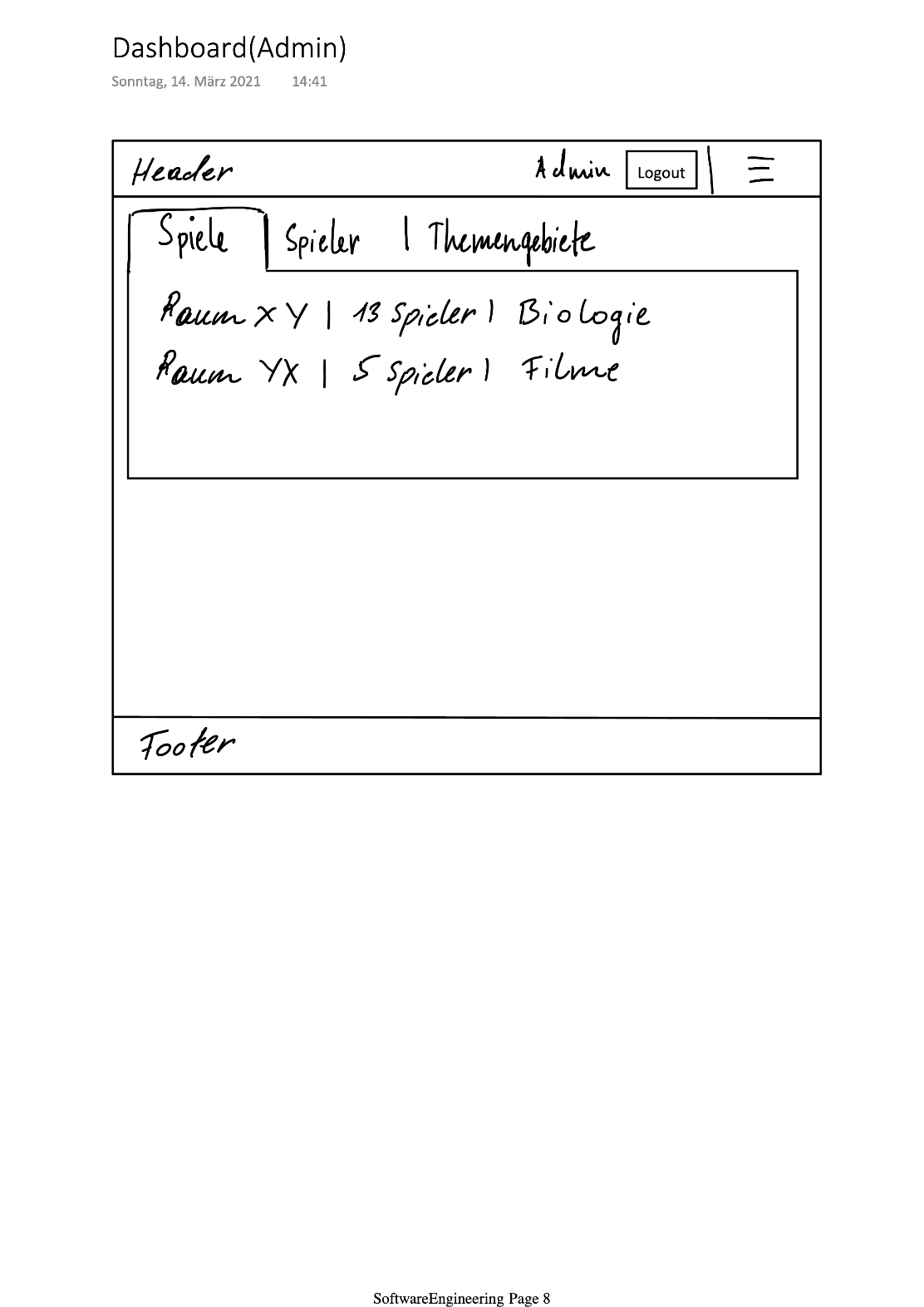
Da es sich um eine Single-Page Applikation handelt, konnten animierte Elemente in dieser statischen Ansicht nicht wiedergegeben werden (Dropdown, Popup, Tooltips).

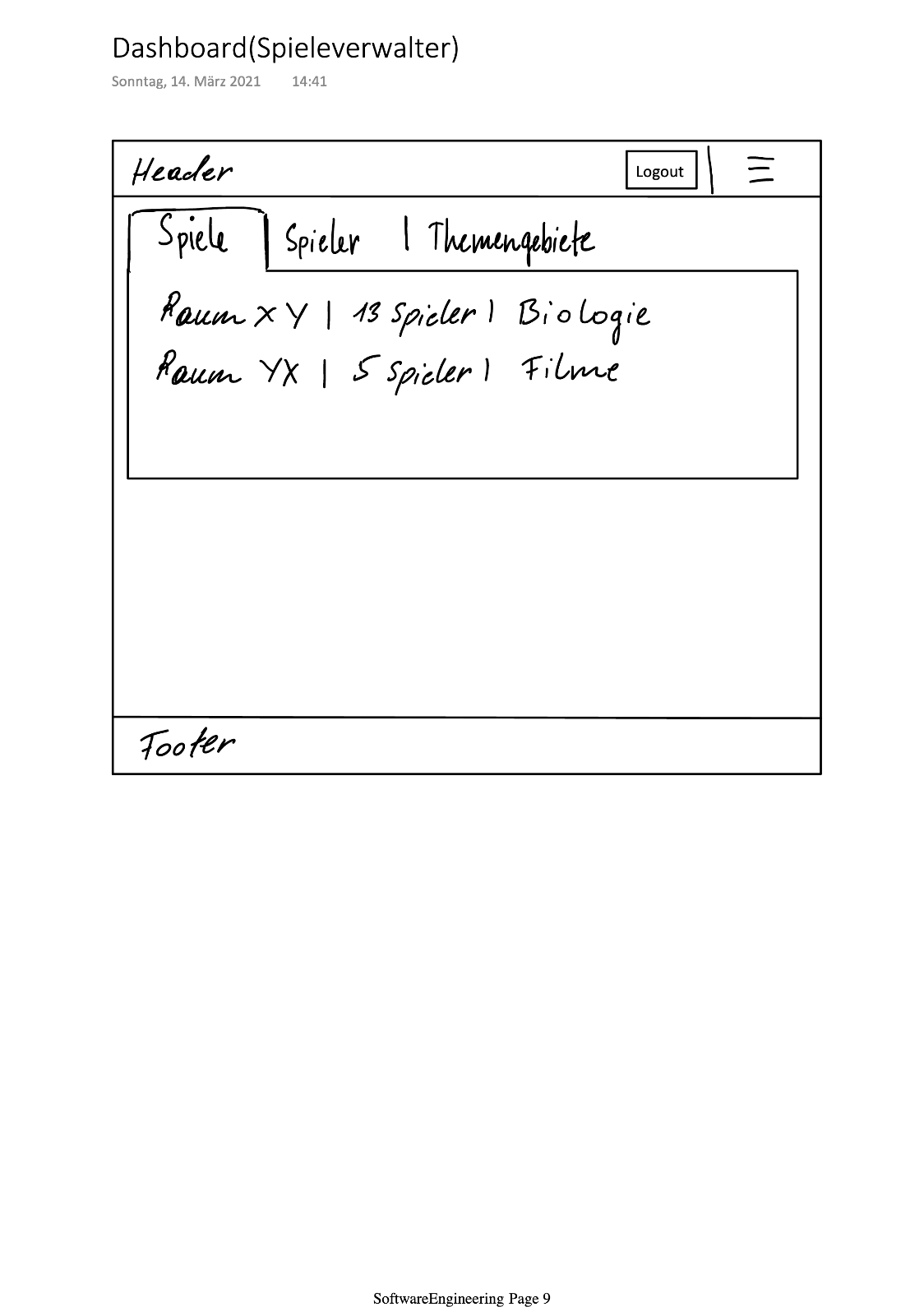












# Projektplan

**Rolleneinteilung**

* Hardware / Bluetooth Architekt: Martin Beyer, Martin Neuner
* Frontend Architekt: Clemens Prosser, Islam Mechtijev
* Backend Architekt: Nicolas Marte, Bernhard Astl

**Zeitplan**

* Wöchentliche JourFixe
* Inkremente:

|  |  |  |
| --- | --- | --- |
| Nummer | Datum | Bezeichnung |
| 1 | 18.03.21 | Fertigstellung Konzept |
| 2 | 28.03.21 | Project Setup / Konfiguration |
| 3 | 11.04.21 | Meilenstein 1 |
| 4 | 25.04.21 | 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; UI), arc42 Dokumentation (Diagramme)
* Fertigstellung aller Features:  
  Feinschliff, Bugtesting