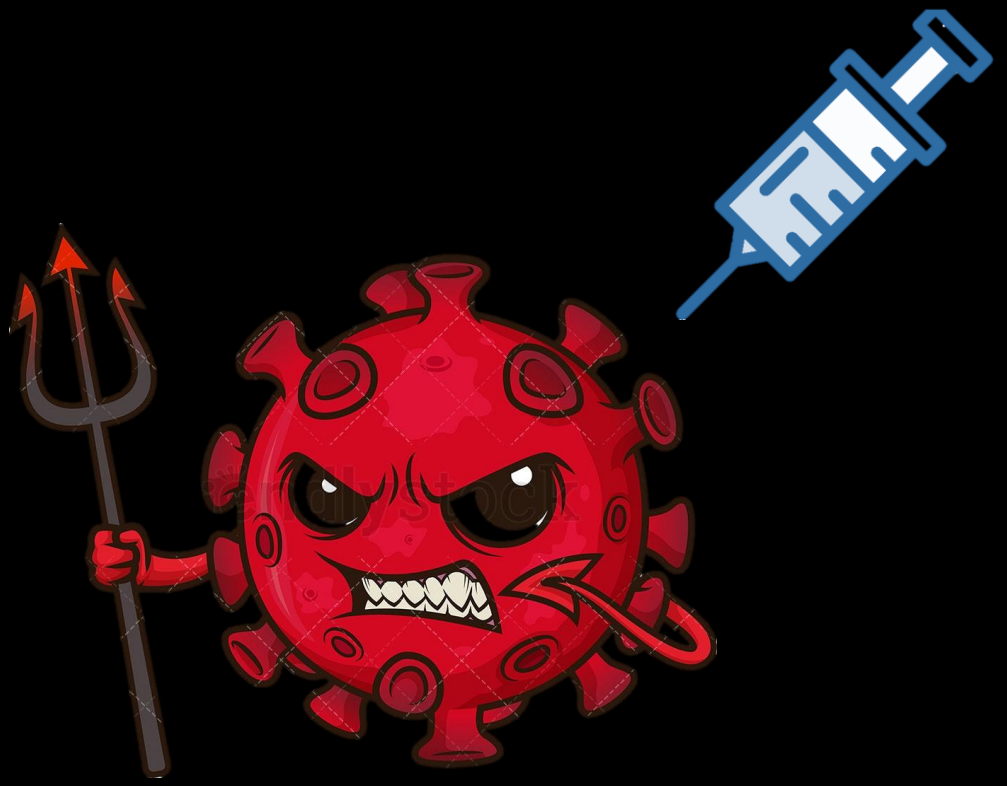
FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI



Coronattack

Infiziere die Menschen des Gegners und heile sie danach wieder!

Autoren: Theologos Baxevanos und Chantale Gihara
BSC_INF_WEB_E
Dozent: Heinrich Zimmermann

Inhaltsverzeichnis

1 Ausgangslage

1.1 Projektauftrag

Im Modul WebE haben wir den Auftrag erhalten ein Game mit folgenden Vorgaben zu entwickeln und dabei eine Woche vor jeder PVA den jeweiligen Meilenstein abzugeben.

Entwicklung eins Spiels mittels Web-Technologien vom folgenden Typ:

1. Runden-basiert oder Educational, oder Datensammler
2. Das Spiel muss eine Client/Server Architektur haben
3. Der Server und die Clients kommunizieren über ein Text-basiertes Protokoll. Das Protokoll muss lesbar sein.
4. Die Server-Funktionalität ist wie folgt definiert:
    a. Er verwaltet den Spielverlauf (überprüft und stellt sicher, dass alle Spielzüge regelkonform sind, erkennt das Ende des Spiels, zählt Punkte, etc.)
    b. Wenn alle Spieler das Spiel verlassen, dann beendet der Server das Spiel.
5. Ein Client hat folgende Eigenschaften:
    a. Er nimmt Benutzereingaben durch eine grafische Schnittstelle (graphical User Interface, GUI) entgegen
    b. Er gleicht den lokalen Status eines Spiels mit dem Status des Servers ab (Synchronisation)
    c. Er erlaubt den Spielern eines Spiels zu chatten.
6. Folgende Aspekte sollen beachtet werden: Internationalisierung, Usability, Accessibility, Levels (das Spiel muss mind. 3 Levels haben), Responsiveness
   Am Ende des Projekts muss eine komplette Distribution des Spiels abgegeben werden (lauffähiges Spiel inklusive Quellcode, Installationsanleitung, Handbuch)

1.2 Projektidee

Wir haben uns für das Spiel «Coronattack» entschieden, dieses basiert auf einem Ping-Pong-Spiel. Es ist ein Dualplayer - Spiel und kann auf verschiedenen Schwierigkeitsgraden (unterschiedlich schneller Ball) gespielt werden.

Dabei sollte es zwei Modi geben:

1. mit dem Virus (Ball) müssen die Menschen des Gegenspielers getroffen werden, wer alle Menschen getroffen hat gewinnt.
2. mit der Impfung (Ball) müssen die Menschen wieder genesen werden, wer zuerst alle Menschen wieder genesen hat, hat gewonnen.

Hier eine kleine Skizze wie das Spiel ungefähr aussehen soll:

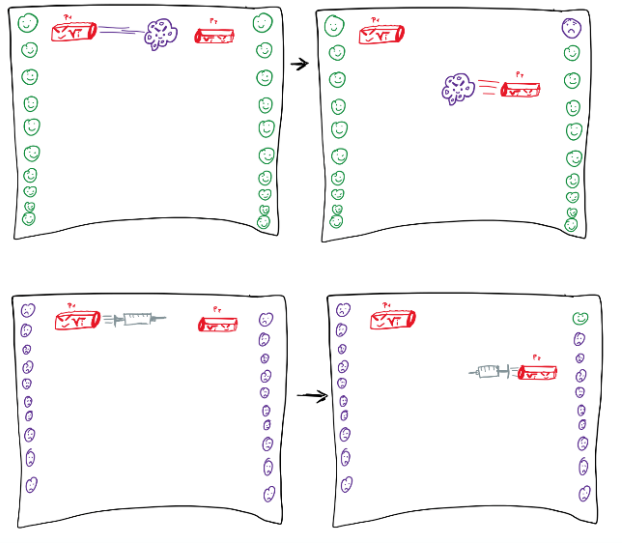


Abbildung 1: Skizze «Coronattack» Spiel

1.3 Spielregeln

Jeder Spieler muss sich zuerst registrieren und erhält dabei eine ID (diese ist nicht sichtbar), die UserId für den Benutzer ist die E-Mail-Adresse.

Beim Spielstart muss er den Namen und das Level des Spiels eintragen.

Der Spieler kann entweder ein Spiel eröffnen oder einem Spiel beitreten. Falls er ein Spiel eröffnet hat, muss er darauf warten, dass ein Spieler dem Spiel beitritt, dabei kann er den Spieler zulassen oder zurückweisen. Wenn der Spieler einem Spiel beitritt, muss er auf die Zulassung des Spielers, welcher das Spiel eröffnet hat, warten:

Spiel 1:

- Zuerst wird das Spiel «infiziere» aufgeschaltet.
- Dabei müssen die Spielgegner ihre Menschen vor dem Virus (Ball) mit einer fahrenden Maske schützen.
- Konnte der Virus (Ball) den Gegner nicht aufhalten, wird der Menschenkopf infiziert.
- Sind alle Menschenköpfe des Gegenspielers oder des Spielers infiziert ist das Spiel beendet
- Dem Verlierer werden seine erhaltenen Punkte zugeteilt
- Dem Gewinner werde die Punkte im Gesamtscore zugeteilt

Spiel 2:

- Ist Spiel 1 beendet geht es in Runde 2
- Die Spieler müssen jetzt sicherstellen, dass die Menschen krank bleiben.
- Der Spieler, welcher beim Gegenspieler zuerst alle Menschen geheilt hat, hat gewonnen
- Dem Gewinner werden die Punkte im Gesamtscore zugeteilt.

Falls es unentschieden steht, geht das Spiel weiter, bis jemand gewinnt. Am Ende wird eine Rangliste mit den Gesamtpunktzahlen der Spieler sichtbar.

1.4 Ziele

- Alle Meilensteine werden komplett und zeitgerecht abgegeben
- Das Spiel ist vor der letzten Präsenzveranstaltung umgesetzt und spielbereit
- Alle Voraussetzungen gemäss Projektauftrag 1.1 werden umgesetzt
- Spieler können sich registrieren und erhalten eine ID (diese ist nicht sichtbar)
- Spieler können sich den Gegner in der Liste aussuchen.
- Spieler kann sich sein Schwierigkeitslevel (Geschwindigkeit) selbst aussuchen

1.5 Abgrenzung

Anhand folgender zeitlicher, sachlicher und sozialer Abgrenzungspunkte wird das Projekt definiert:

Zeitliche Abgrenzung:

Das Projekt wurde vom Dozenten, Herr Dr. Heinrich Zimmermann freigegeben und endet mit der Abgabe der Game- Software am 26.11.2021.

Sachliche Abgrenzung:

Das Projekt hat zum Ziel, ein Spiel gemäss Auftrag 1.1 zu realisieren, welche es ermöglicht, Spielern sich zu registrieren, einzuloggen, zu Spielen und den High-Score einzusehen.

Soziale Abgrenzung:

Zum Projektteam gehören Chantale Gihara und Theologos Baxevanos.

Nicht zum Projekt gehören:

- Die Wartung des Spieles nach der Abgabe.
- Die Sicherheit und Verschlüsselung der Kundendaten. Die Kunden sind selbst verantwortlich für die Sicherheit ihrer Daten, sowie für eine Firewall- und Port-Konfiguration.

1.6 Information und Kommunikation

Die Kommunikation im Team erfolgt nach Absprache und übers Trello Board: https://trello.com/b/7HNVSgbQ/kanban-webe-coronattack oder wenn möglich vor Ort, 2 wöchentlich Starbucks Zürich Oerlikon Bahnhof.

Während den Meetings wird von einem der Mitglieder ein Protokoll geführt, welches gleich hier im Annex 3.1 nachgeführt wird.

Dazu haben wir auch eine WhatsApp, in der wir auch ausserhalb des zwei wöchentlichen Meetings miteinander kommunizieren.

Die geplanten Stunden und Wochen pro Arbeitspaket werden in der Board Card im Trello eingeplant. Nachdem das Arbeitspaket abgeschlossen wurde, werden die effektiv verbrauchten Stunden und Wochen eingetragen. So kann der Aufwand gut überwacht werden und falls nötig angepasst werden.

Informationen zum Projekt werden an nachfolgenden Stellen gepflegt bzw. sind dort verfügbar:

- Projektdokument «Coronattack»: Wird in einem Word-Dokument geführt und wird auf One-Drive abgelegt, welches für beide von uns zugänglich ist. Dieses Dokument wird von uns beiden aktualisiert und gepflegt.
- Meeting-Protokolle werden wie oben erwähnt im Annex 3.1 aufgeführt und gepflegt.
- Zentraler Zugriffspunkt für die Dokumentation zum Projekt: Die Projektseite auf OneDrive.
- Zentraler Zugriffspunkt für die Arbeitspakete und Zeitplaner zum Projekt: Trello Board
- Code Repository: Auf GitLab, gehostet von der FFHS. Alle Beide pflegen das Repository. https://git.ffhs.ch/chantale.gihara/coronattack
- Die Visualisierung der Aufgaben mittels Kanban: Das Board wird auf Trello von beiden gepflegt und fortlaufend aktualisiert. Darunter fällt auch die Zeiterfassung auf den Board-Karten. https://trello.com/b/7HNVSgbQ/kanban-webe-coronattack

1.7 Iterationsmodell

Wir werden im Projekt gemäss den 4 vergebenen Meilenstein iterativ aufteilen:

Dabei werden die Arbeitspakete gemäss den gewünschten Vorgaben in der Iteration berücksichtigt. Siehe Projektstrukturplan 1.8

1.8 Projektstrukturplan

Der Projektstrukturplan wurde gemäss iterativem Vorgehen entwickelt. Am Ende jeder Iteration gilt der der Meilenstein als abgeschlossen. Ausser bei der Iteration 2. 3 Arbeitspakete «2.11 Spiel 1», «Spiel 2», «Show High Score», werden in der Iteration 2 gestartet, werden aber erst in der Iteration 3 als abgeschlossen gelten.

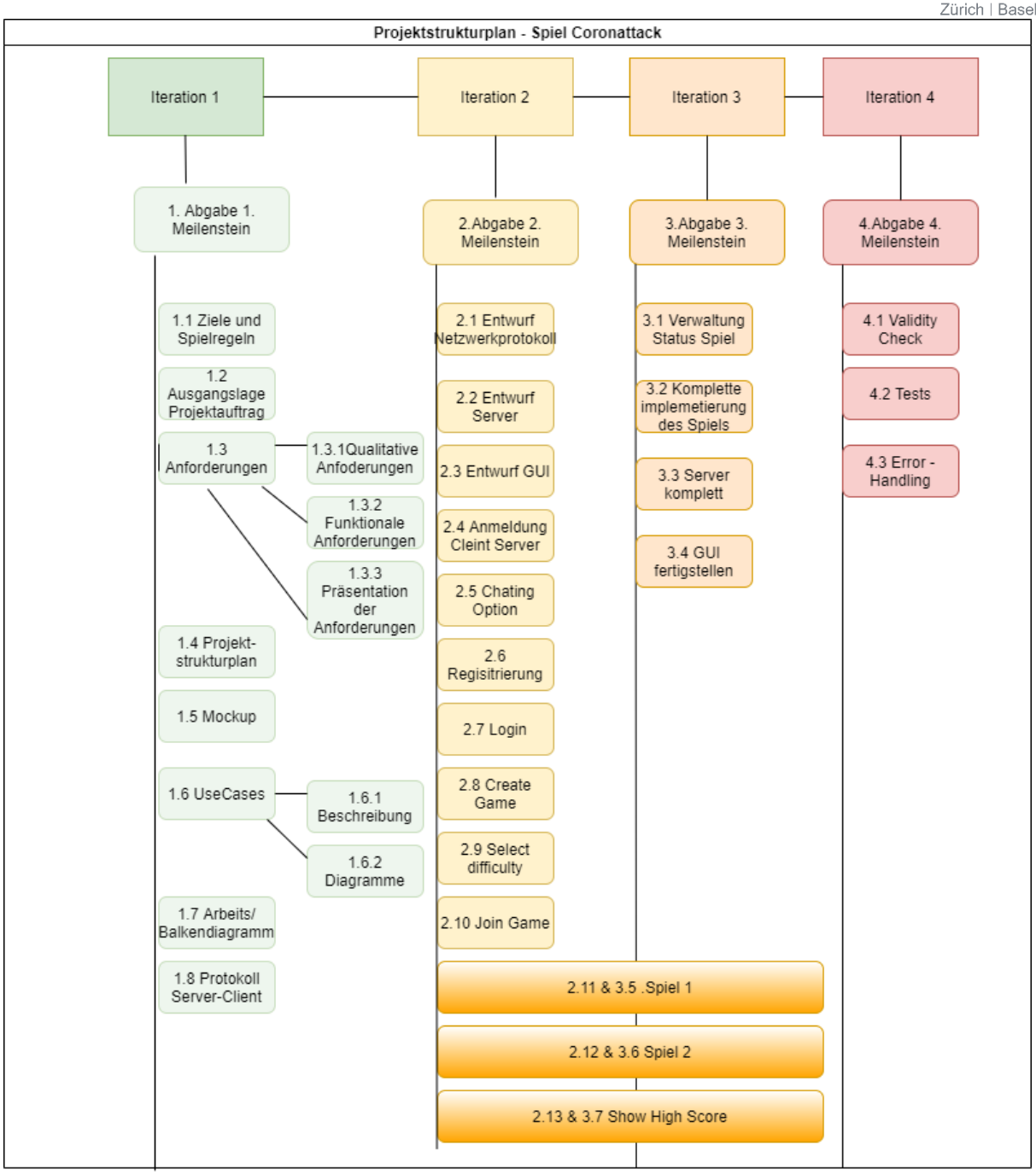


Abbildung 2:Projektstrukturplan

1.9 Arbeitsplan / Balkendiagramm

Die Arbeitsbalken und Timeline sind in Trello ersichtlich: https://trello.com/b/7HNVSgbQ/kanban-webe-coronattack/timeline

1.9.1 Meilenstein 1 / Iteration 1

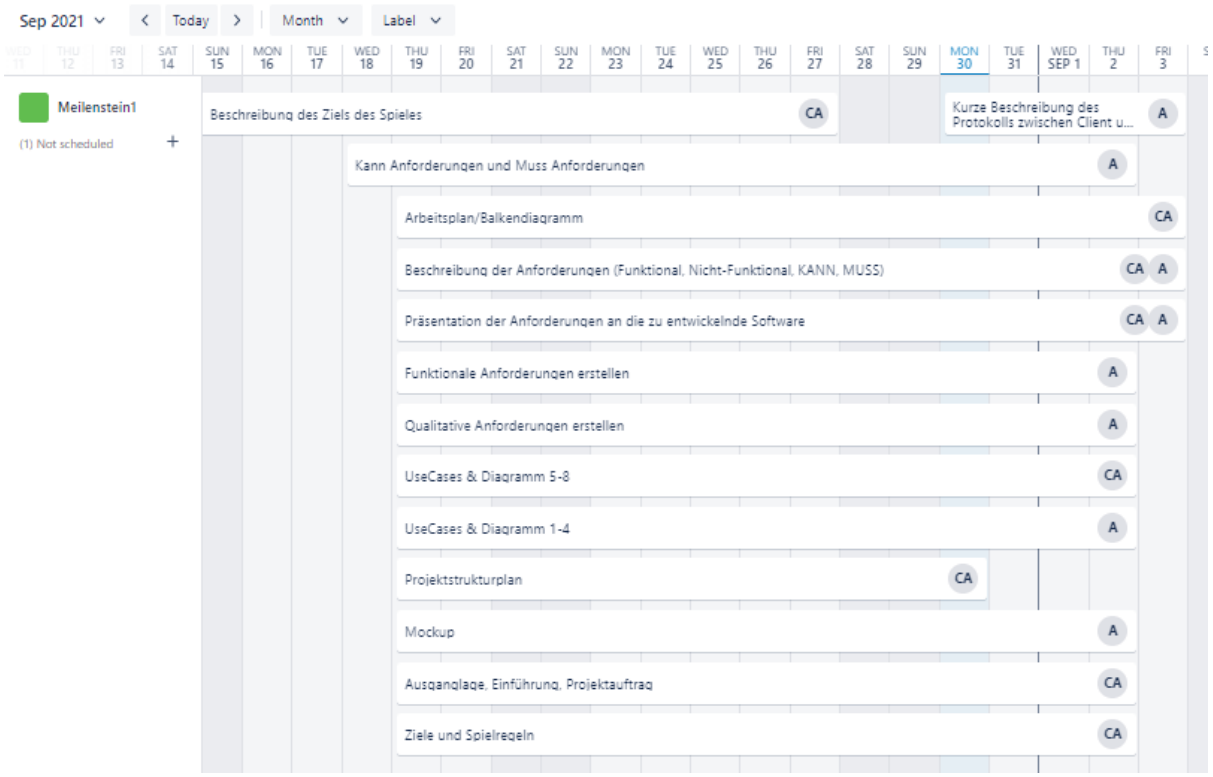


Abbildung 3: Arbeitsplan/Balkendiagramm - Meilenstein 1

1.9.2 Meilenstein 2 / Iteration 2

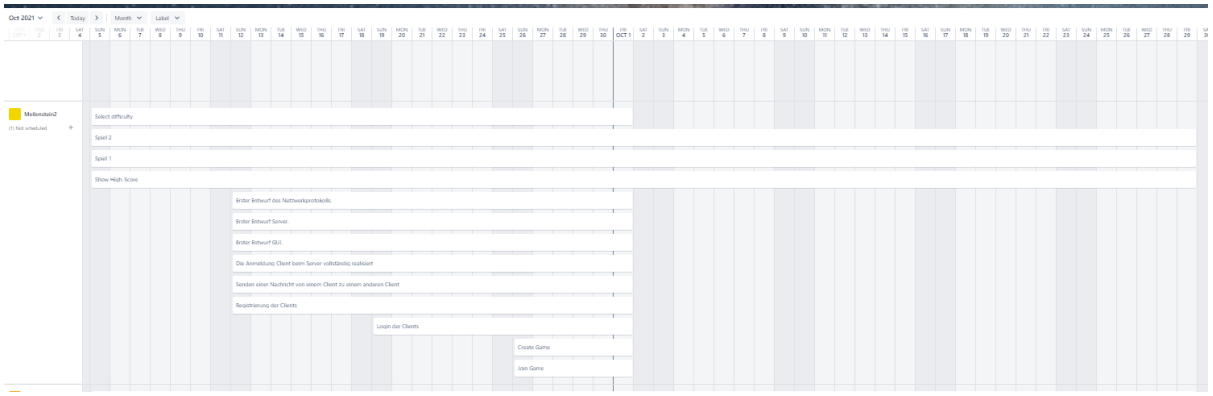


Abbildung 4: Arbeitsplan/Balkendiagramm - 2. Meilenstein

1.9.3 Meilenstein 3 / Iteration 3

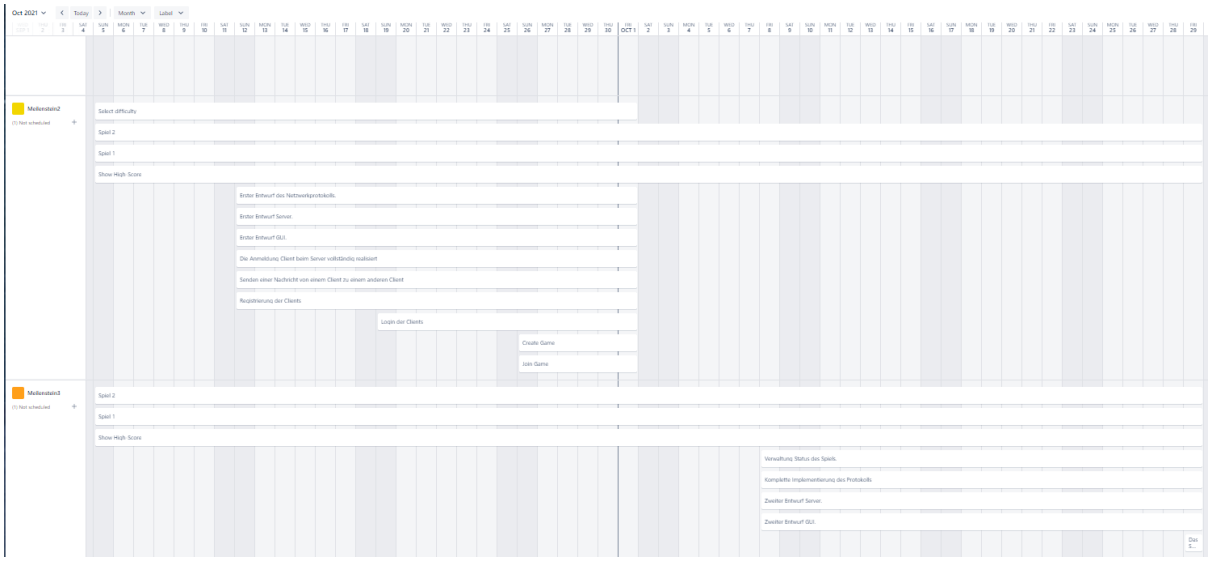


Abbildung 5:Arbeitsplan/Balkendiagramm - 3. Meilenstein

1.9.4 Meilenstein 4 / Iteration 4

Abbildung 6: Arbeitsplan/Balkendiagramm - 4. Meilenstein

2 Softwareengineering

2.1 Anforderungen

Wir habe die Vorlage für die Anforderungen von (Ludewig, 2013) übernommen:

2.1.1 Funktionale Anforderungen

Tabelle 1: FR - 001

ID	FR-001
Typ	Funktionale Anforderung
Titel	Plattformunabhängigkeit

Aussage	Das Spiel kann auf der Endanwenderseite Plattform- und Gerätunabhängig verwendet werden.
Begründung	Heutzutage spielt man webbasierte Spiele nicht nur am PC, sondern auch auf dem Handy bzw. Tablet. Das Betriebssystem variiert auch oft und ein vorausgesetztes Betriebssystem soll nicht verlangt werden. Somit soll die Applikation plattform- und Gerätunabhängig sein.
Verbindlichkeit	Pflicht
Priorität	Hoch
Abnahmekriterien	ID: AC1 Kriterium: Anwendung ist Gerätunabhängig OK-Entscheid: Die Anwendung kann auf verschieden Geräten und Browser abgerufen werden
Version	1.0
Änderungsdatum	02.09.2021
Autor	Theologos Baxevanos
Zustand	genehmigt

Tabelle 2: FR – 002

ID	FR-002
Typ	Funktionale Anforderung
Titel	Bedienfreundliche Oberfläche
Aussage	Das Spiel verfügt über eine bedienerfreundliche, grafische Oberfläche.
Begründung	Das Zielpublikum für das Spiel ist nicht nur ICT-Affin Personen. Somit soll die Oberfläche bedienerfreundliche und verständlich sein.
Verbindlichkeit	Pflicht
Priorität	Hoch
Abnahmekriterien	ID: AC2 Kriterium: Oberfläche ist bedienerfreundlich. OK-Entscheid: UX-Tester geben Feedback, dass es benutzerfreundlich ist,
Version	1.0
Änderungsdatum	
Autor	Theologos Baxevanos
Zustand	genehmigt

Tabelle 3: FR – 003

ID	FR-003
Typ	Funktionale Anforderung
Titel	Multiplayer
Aussage	Das Spiel kann ohne einen Gegner nicht gestartet werden
Begründung	“Coronattack” besitzt keine Single-Player Funktion, somit muss ein Gegner dem Spielraum betreten, sonst soll das Spiel nicht gestartet werden können.
Verbindlichkeit	Pflicht
Priorität	Hoch
Abnahmekriterien	ID: AC3

	Kriterium	Ein Spieler allein kann das Spiel nicht starten
	OK-Entscheid	Spiel mit einem Spieler startet nicht.
Version	1.0	
Änderungsdatum	02.09.2021	
Autor	Theologos Baxevanos / Chantale Gihara	
Zustand	genehmigt	

Tabelle 4: FR – 004

ID	FR-004	
Typ	Funktionale Anforderung	
Titel	Registrierung	
Aussage	Nur registrierte Spieler können ins Spiel einloggen und ein neues Spiel starten bzw. einem Spielraum betreten.	
Begründung	Die Zug-Punkte werden in einer Tabelle “High Scores” eingetragen. Somit müssen alle Spieler identifizierbar sein.	
Verbindlichkeit	Pflicht	
Priorität	Hoch	
Abnahmekriterien	ID	AC4
	Kriterium	Nur registrierte Spieler können dem Spiel beitreten
	OK-Entscheid	Registrierte Spieler können dem Spiel beitreten. Nicht registrierter kann kein Spiel eröffnen oder beitreten.
Version	1.0	
Änderungsdatum	02.09.2021	
Autor	Theologos Baxevanos / Chantale Gihara	
Zustand	Genehmigt	

Tabelle 5: FR – 005

ID	FR-005	
Typ	Funktionale Anforderung	
Titel	Chat Funktion	
Aussage	Eine Chat Funktion steht den Spielern zur Verfügung, mit der sie während des Spiels miteinander kommunizieren können.	
Begründung	Es geht um ein interaktives online Spiel, wo der Spieler gegen einen echten Menschen spielt, somit soll eine Chat Funktion vorhanden sein.	
Verbindlichkeit	Pflicht	
Priorität	Hoch	
Abnahmekriterien	ID	AC5
	Kriterium	Eine Chatfunktion ist zur Verfügung
	OK-Entscheid	Spieler sendet eine Message zu einem anderen Spieler und die Nachricht kommt an und vice versa auch.
Version	1.0	
Änderungsdatum	02.02.2021	

Autor	Chantale Gihara / Theologos Baxevanos
Zustand	Genehmigt

Tabelle 6: FR – 006

ID	FR-006
Typ	Funktionale Anforderung
Titel	Schwierigkeitsauswahl
Aussage	Die Schwierigkeitsauswahl steht für die Spieler zur Verfügung. Die Schwierigkeit hängt mit der Geschwindigkeit des Balles (Corona) zusammen. Je höher die Schwierigkeit, je schneller geht der Ball (Corona) hin und her. Es sollte min 3 Stufen geben. Easy, medium, hard.
Begründung	Es war eine Vorgabe des Modules Schwierigkeiten einzubauen.
Verbindlichkeit	Pflicht
Priorität	Hoch
Abnahmekriterien	ID: AC6 Kriterium: Schwierigkeiten hard, medium, hard können ausgewählt werden OK-Entscheid: Die Schwierigkeiten können frei gewählt werden bei einer Spiel Eröffnung. Simple: das langsamste Medium: Schneller als Simple, aber langsamer als hard Hard: ist das schnellste der 3 Schwierigkeitsgraden.
Version	1.0
Änderungsdatum	02.09.2021
Autor	Chantale Gihara / Theologos Baxevanos
Zustand	Genehmigt

2.1.2 Qualitätsanforderung (Nicht funktionale Anforderungen)

Tabelle 7: QR – 001

ID	QR-001
Typ	Qualitative Anforderung
Titel	JavaScript Front & Backend
Aussage	Das Spiel soll mit JavaScript (Front & Backend) entwickelt werden.
Begründung	Beide Autoren haben geringe Erfahrung mit der Programmierungssprache “JavaScript” und möchten die Gelegenheit nutzen, das Spiel vollständig in JS zu programmieren, um Ihre JS Kenntnisse zu vertiefen.
Verbindlichkeit	Von Vorteil
Priorität	Mittel
Abnahmekriterien	ID: AC7 Kriterium: Frontend ist in JavaScript geschrieben OK-Entscheid: JavaScript ist im Frontend erkennbar
Version	1.0
Änderungsdatum	02.09.2021
Autor	Chantale Gihara / Theologos Baxevanos

Zustand	genehmigt

Tabelle 8: QR - 002

ID	QR-002
Typ	Qualitative Anforderung
Titel	Mehrsprachigkeit
Aussage	Der Spieler kann die Spielanzeigesprache umstellen
Begründung	Es soll möglich sein, die Sprache der Benutzeroberfläche auf Deutsch umstellen zu können, da wir das Spiel auf Englisch programmieren werden, jedoch unsere ersten Spieler sich in der Schweiz befinden werden.
Verbindlichkeit	Von Vorteil
Priorität	Mittel
Abnahmekriterien	ID: AC8 Kriterium: Sprache ist auf Deutsch und Englisch umstellbar OK-Entscheid: Sprachwechsel funktioniert überall und ist vorhanden.
Version	1.0
Änderungsdatum	02.09.2021
Autor	Chantale Gihara / Theologos Baxevanos
Zustand	genehmigt

Tabelle 9: QR - 003

ID	QR-003
Typ	Qualitative Anforderung
Titel	Anpassbare Benutzeroberfläche
Aussage	Die Benutzeroberfläche ist vom Benutzer anpassbar
Begründung	Dem Benutzer stehen verschiedene Optionen für die “in-game" Figuren zur Verfügung und er kann sie vor dem Beginn des Spiels ändern (den Ball, die Schläger und die “Smilies”). Somit spielt er immer mit Figuren seines Geschmacks.
Verbindlichkeit	Von Vorteil
Priorität	Mittel
Abnahmekriterien	ID: AC9 Kriterium: Verschiedene Graphische Icons für die Auswahl der Avatare steht zur Verfügung. OK-Entscheid: Avatare und Icons sind wählbar und kann hin und her gewechselt werden und wird übernommen.
Version	1.0
Änderungsdatum	02.09.2021
Autor	Chantale Gihara / Theologos Baxevanos
Zustand	genehmigt

(Ludewig, 2013)

2.2 Use Cases

Wir haben unser Spiel in folgende Use Cases unterteilt

- Registrierung
- Login
- Create Game
- Select Difficulty
- Join Game
- Spiel 1
- Spiel 2
- Scoring

Die Use Case Diagramme habe wir nach (Seidl, 2012) und (creately) erstellt.

2.2.1 Use Case 1: Registrierung

Tabelle 10: Registrierung

Merkmal	Beschreibung
Use Case Nr.	1
Bezeichnung	Registrierung
Kurzbeschreibung	Der Akteur (Spieler 1) registriert sich, um das Spiel zu spielen. Mit dem Registrieren wird ein Benutzeraccount erstellt. Er muss eine E-Mail-Adresse und ein Passwort eingeben. Mit diesen Zugangsdaten kann er sich ins Spiel einloggen.
Auslösendes Ereignis	Webseite des Spieles aufrufen
Akteur	Der Spieler 1
Vorbedingung	Zugriff auf die Webseite des Spieles
Nachbedingung	Der Benutzer meldet sich mit seinen Zugangsdaten an
Ergebnis	Benutzeraccount wurde erstellt und der Spieler ist eingeloggt
Szenarios	1) Der Spieler 1 registriert sich 2) Der Spieler loggt sich ein

2.2.2 Use Case 2: Login

Tabelle 11: Login

Merkmal	Beschreibung
Use Case Nr.	2
Bezeichnung	Login
Kurzbeschreibung	Der Akteur (Spieler) loggt sich ein, um das Spiel zu spielen. Mit dem Einloggen werden dem Spieler zwei Optionen angezeigt, entweder ein Spiel erzeugen oder einen Spielraum beizutreten.
Auslösendes Ereignis	Webseite des Spieles aufrufen
Akteur	Der Spieler
Vorbedingung	Zugriff auf die Webseite des Spieles
Nachbedingung	Der Spieler wählt, ob er ein Spiel erzeugen oder einen Spielraum beitreten möchte.
Ergebnis	Der Spieler ist mit seinem einzigartigen Account eingeloggt
Szenarios	1) Der Spieler hat sich bereits registriert 2) Der Spieler meldet sich an

2.2.3 Use Case 3: Create Game

Tabelle 12: Create Game

Merkmal	Beschreibung
Use Case Nr.	3

Bezeichnung	Create Game
Kurzbeschreibung	Der Akteur (Spieler 1) wählt die Option ein neues Spiel zu erzeugen aus, gibt das Spiel ein Name und setzt wie hoch die Schwierigkeit sein soll.
Auslösendes Ereignis	Ein neues Spiel wird erzeugt
Akteur	Der Spieler 1 (Spiel Erzeuger)
Vorbedingung	Zugriff auf die Webseite des Spieles, registrierter Spieler
Nachbedingung	Der Spieler 1 wartet auf die Anfrage eines weiteren Spielers, der den Spielraum beintreten möchte,
Ergebnis	Das Spiel wurde erzeugt
Szenarios	1) Der Spieler 1 hat sich bereits registriert 2) Der Spieler 1 meldet sich an 3) Der Spieler 1 wählt die Option ein neues Spiel zu erzeugen 4) Der Spieler 1 gibt einen Namen für das Spiel ein 5) Der Spieler 1 setzt den Grad der Schwierigkeit des Spiels 6) Der Spieler 1 wartet auf die Anfrage eines anderen Spielers (Spieler 2)

2.2.4 Use Case 4: Schwierigkeit setzen

Tabelle 13: Select Difficulty

Merkmal	Beschreibung
Use Case Nr.	4
Bezeichnung	Select Difficulty
Kurzbeschreibung	Der Akteur (Spieler 1) wählt den Grad der Schwierigkeit des Spiels an.
Auslösendes Ereignis	Der Spieler hat angewählt ein neues Spiel zu erzeugen
Akteur	Der Spieler 1 (Spiel Erzeuger)
Vorbedingung	Zugriff auf die Webseite des Spieles, registrierter Spieler
Nachbedingung	Die Schwierigkeit wird gesetzt und der Spieler 1 darf einen Spielnamen eingeben und das Spiel starten
Ergebnis	Anhand der ausgewählten Schwierigkeit wird die Geschwindigkeit des Spielballes definiert
Szenarios	1) Der Spieler 1 hat sich bereits registriert 2) Der Spieler 1 meldet sich an 3) Der Spieler 1 wählt die Option ein neues Spiel zu erzeugen 4) Der Spieler 1 setzt den Grad der Schwierigkeit des Spiels 5) Der Spieler 1 gibt einen Namen für das Spiel ein und darf das Spiel starten

2.2.5 Use Case 5: Join Game

Tabelle 14: Join Game

Merkmal	Beschreibung
Use Case Nr.	5
Bezeichnung	Join Game
Kurzbeschreibung	Der Spieler möchte sich in ein schon eröffnetes Spiel einloggen und der Spieler, welcher das Spiel eröffnet hat, akzeptiert die Anfrage des neuen Gegenspielers
Auslösendes Ereignis	Auswahl Join Game
Akteur	Spieler 1 und Spieler 2
Vorbedingung	Ein Spieler hat schon ein Spiel eröffnet (Create Game)
Nachbedingung	Spiel1 startet
Ergebnis	Spieler, welcher das Spiel eröffnet hat, kann die Anfrage annehmen oder ablehnen, falls es angenommen wird, wird das Spiel 1 gestartet.
Szenarios	1) Spieler2 macht eine Anfrage für in einem Spiel beizutreten 2) Spieler1 akzeptiert die Anfrage 3) Spieler1 lehnt die Anfrage ab und wartet auf einen anderen Gegenspieler.

Abbildung 7: Use Case Diagramm 5 «Join Game»

2.2.6 Use Case 6: Spiel 1

Tabelle 15: Spiel 1

Merkmal	Beschreibung
Use Case Nr.	6
Bezeichnung	Spiel 1
Kurzbeschreibung	Spiel 1 (Runde 1) versucht der Spieler die Menschen des Spielgegners mit Corona zu infizieren.
Auslösendes Ereignis	Join Game oder Create Game
Akteur	Spieler, Spiel
Vorbedingung	2 Spieler sind im Game einer vom Create Game oder vom Join Game
Nachbedingung	Ein Spieler hat gewonnen, alle Menschen des Gegners infiziert
Ergebnis	Spiel 1 startet, Spieler gewinnt oder verliert, Spiel 1 wird beendet
Szenarios	1. Spieler versucht mit der Maske seine Menschen zu beschützen 2. Spieler versucht mit Maske den Corona zu werfen, dass er die Menschen des Spielgegners trifft.

2.2.7 Use Case 7: Spiel 2

Tabelle 16: Spiel 2

Merkmal	Beschreibung
Use Case Nr.	7
Bezeichnung	Spiel 2
Kurzbeschreibung	Im Spiel 2 (Runde 2) müssen die Menschen des Gegners wieder geheilt werden.
Auslösendes Ereignis	Spiel 1 ist beendet, ein Spieler hat alle Menschen mit dem Corona infiziert
Akteur	Spieler, Spiel
Vorbedingung	Spiel 1 wurde beendet
Nachbedingung	High Score der Gesamtrangliste wird angezeigt
Ergebnis	Spiel 2 startet
Szenarios	1. Spieler versucht mit einer fliegenden Spritze die Menschen wieder gesund zu treffen 2. Spieler versucht seine Menschen von der Spritze zu beschützen

2.2.8 Use Case 8: High Score anzeigen

Tabelle 17: High Score

Merkmal	Beschreibung
Use Case Nr.	8
Bezeichnung	High Score anzeigen
Kurzbeschreibung	Die High Scores werden angezeigt
Auslösendes Ereignis	Durch das Beenden von Spiel 2 oder durch das Anwählen «High Score» wird die Seite mit den High Score angezeigt
Akteur	Die Spieler (beenden des Spieles, anwählen des Spieles)
Vorbedingung	Spiele müssen beendet sein oder Spieler wählen die Ansicht «High Score»
Nachbedingung	Man kann ins Menu zurück navigieren oder man schliesst die Seite
Ergebnis	High Score der Spieler (Rangliste) wird angezeigt
Szenarios	1. Spiel wird beenden

	2. Ansicht «High Scores» ausgewählt

2.3 Mockups

Mockups werden hier zur Präsentation und Qualitätskontrolle eingesetzt. Sie dienen dazu, Vorstellungen und Anforderungen an die Benutzeroberfläche bezüglich Grundfunktionen, Navigation, Inhaltsarchitektur und Design mit dem Kunden abzustimmen.
Die Mockups wurden mit dem Webbasierten Prototypentool “Figma” erstellt. Der Prototyp kann unter https://www.figma.com/proto/GZGqOcwK1Uol5fH2fiIhUA/coronattack?node-id=2%3A2&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=2%3A2 aufgerufen werden.

2.3.1 Startseite

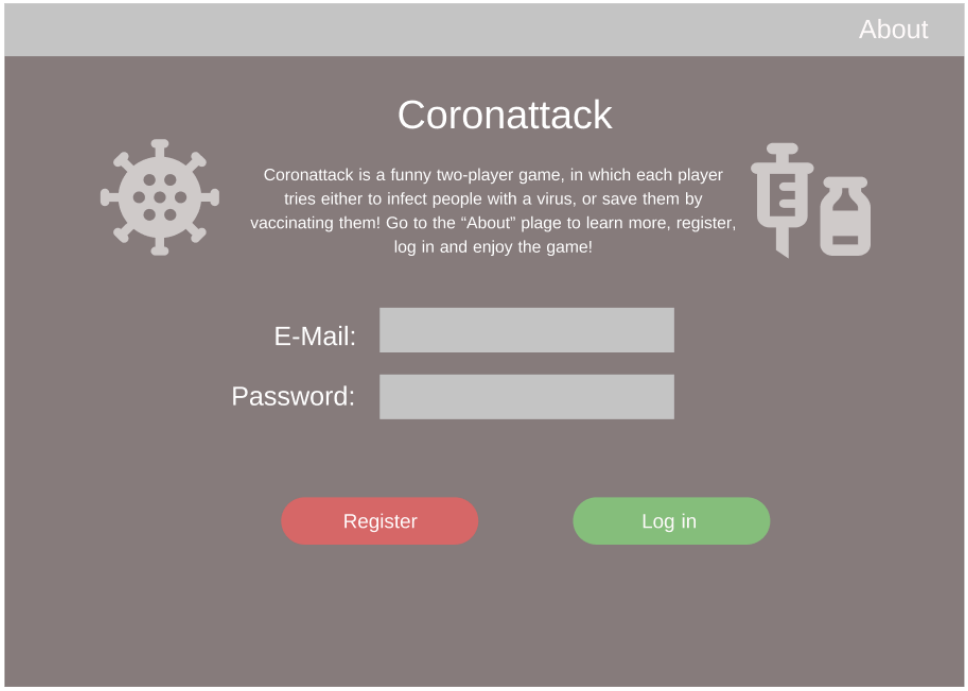


Abbildung 2: Startseite

Die Startseite erklärt in wenigen Worten das Spiel. Der Benutzer kann detaillierte Informationen lesen, wenn er die “About” Seite aufruft, sich registrieren bzw. einloggen.

2.3.2 About Seite

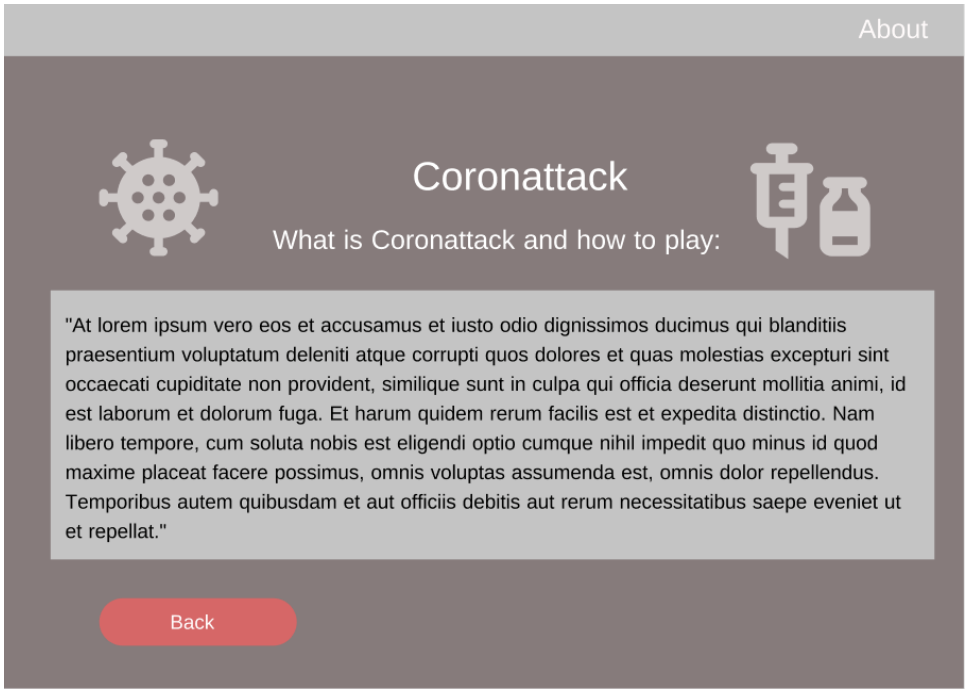


Abbildung 3: About Seite

In der About Seite wird das Spiel in Details erklärt, wie man sich registrieren, einloggen und spielen kann und einige Wörter über die Entwickler des Spiels.

2.3.3 Registrierung

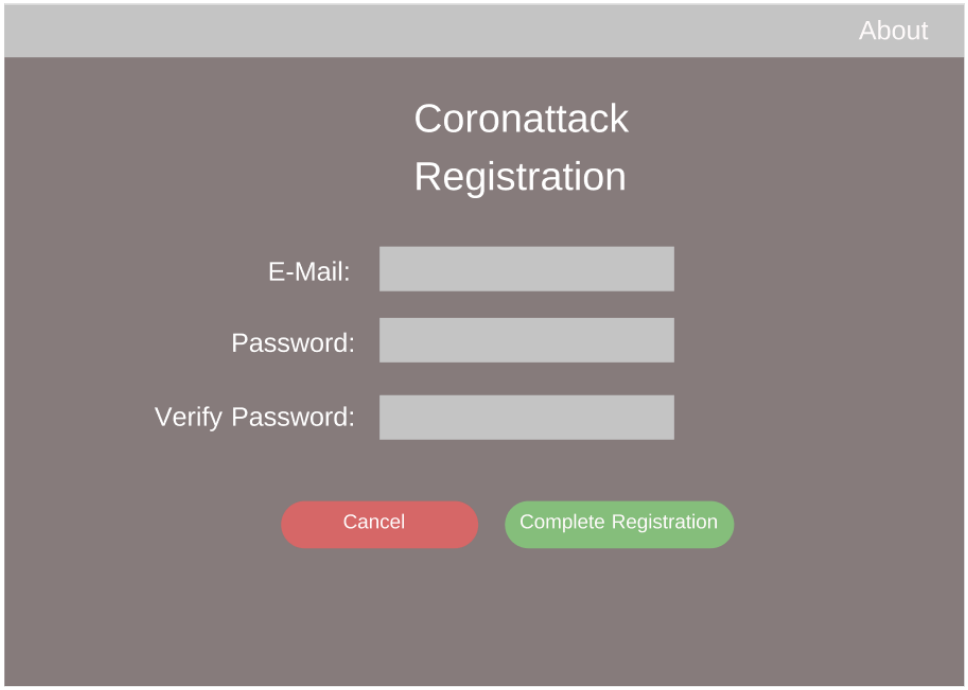


Abbildung 4: Registrierung

Auf dieser Seite kann der Spieler sich registrieren. Er muss eine E-Mail-Adresse und ein Passwort eingeben. Die E-Mail-Adresse dient als Benutzername.

2.3.4 Spiel erzeugen / beitreten

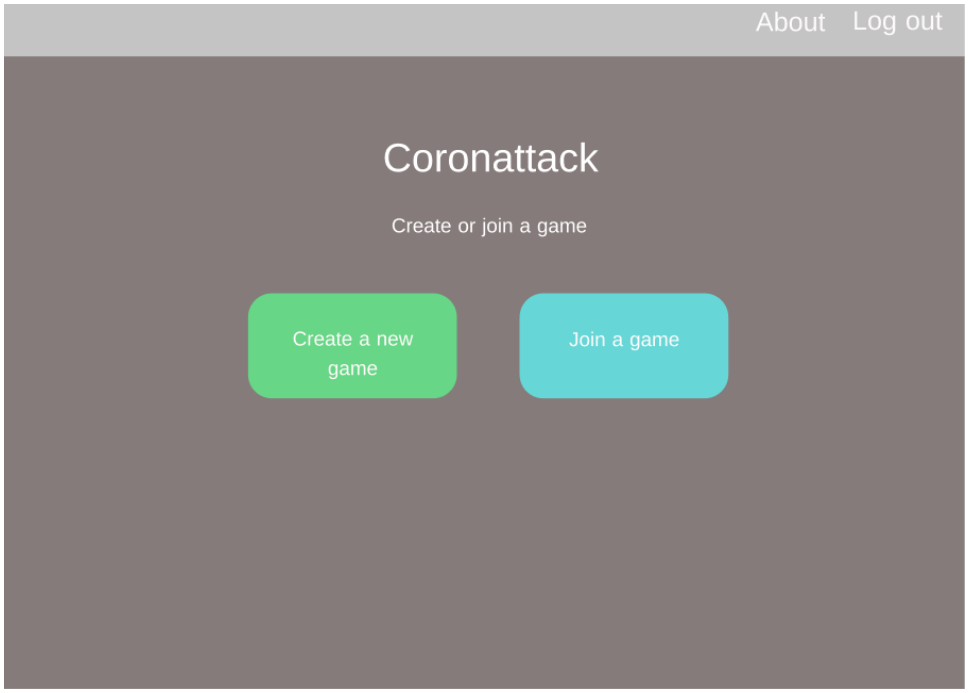


Abbildung 5: Spiel erzeugen oder beitreten

Auf dieser Seite kann der Spieler eine der zwei Optionen auswählen. Entweder kann er ein Spiel erzeugen oder einem existierenden Spielraum beitreten.

2.3.5 Spielraum Beitreten

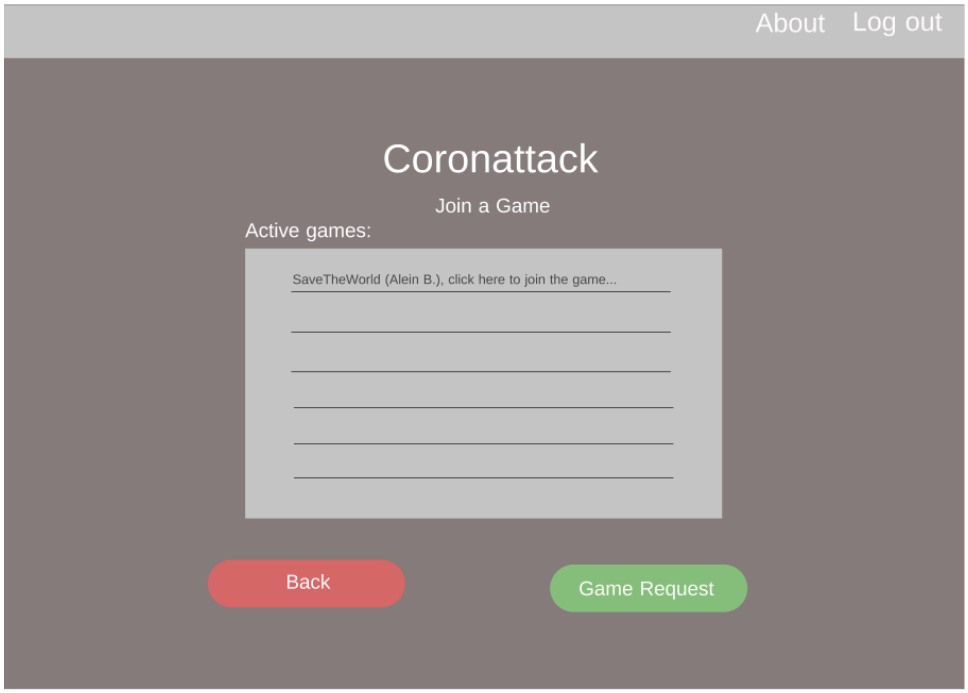


Abbildung 6: Spielraum beitreten

Der Spieler kann einem existierenden Spielraum beitreten. Wenn er auf den Button “Game request” drückt, wird dem anderen Spieler eine Anfrage geschickt. Der andere Spieler (der, der den Spielraum erzeugt hat) kann diese Anfrage entweder ablehnen oder annehmen. Der Spieler hat auf dieser Seite auch die Option eine Seite zurückzugehen, sich auszuloggen oder die About Seite aufzurufen.

2.3.6 Spielanfrage

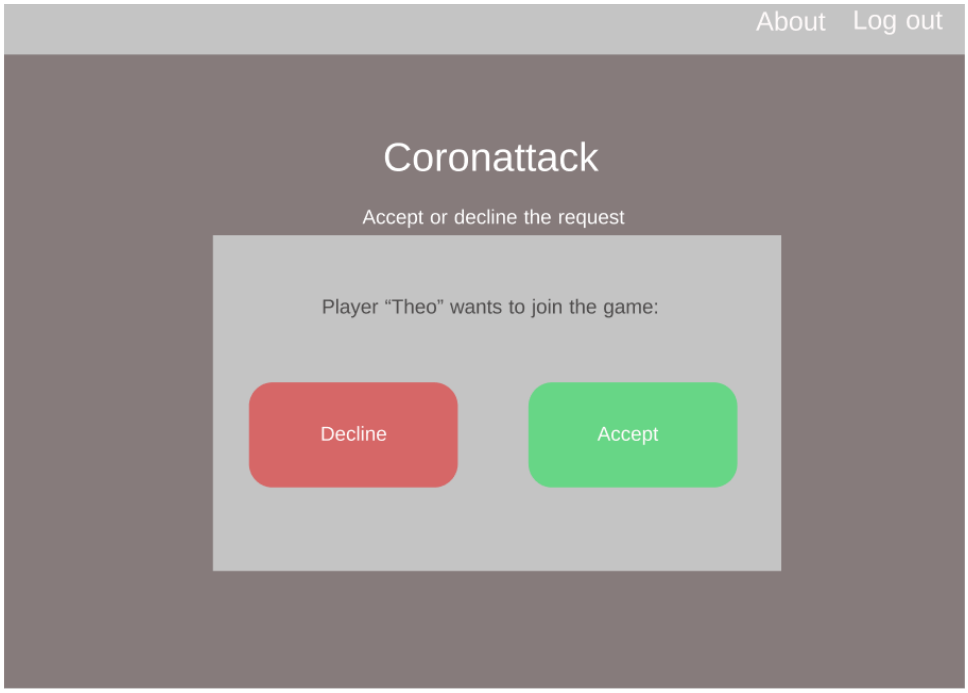


Abbildung 7: Spielanfrage

Das ist ein Pop-up Dialog, der dem Erzeuger des Spiels angezeigt wird, wenn ein anderer Spieler dem Spielraum beitreten möchte. Der Spielerzeuger hat die Möglichkeit die Anfrage abzulehnen oder anzunehmen.

2.3.7 Spiel erzeugen

Abbildung 8: Spiel erzeugen

Auf dieser Seite kann der Spieler ein Spiel erzeugen. Für da Spiel muss ein Name und die Schwierigkeit eingegeben werden.

Bei der Schwierigkeit “easy” wird die Ballgeschwindigkeit auf “5” und die Anzahl der infizierten/geheilten Personen auf “10” gesetzt.

Bei der Schwierigkeit “medium” wird die Ballgeschwindigkeit auf “10” und die Anzahl der infizierten/geheilten Personen auf “5” gesetzt.

Bei der Schwierigkeit “hard” wird die Ballgeschwindigkeit auf “15” und die Anzahl der infizierten/geheilten Personen auf “2” gesetzt.

2.3.8 Der Spielraum

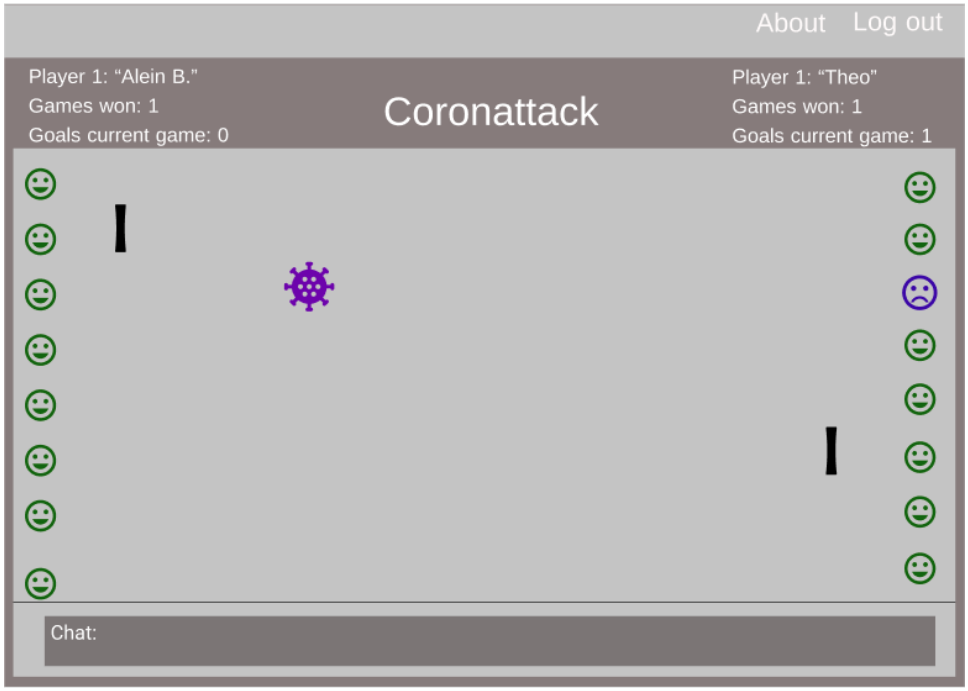


Abbildung 9: Der Spielraum

Auf dieser Seite wird das Spielbrett dargestellt. Oben links und rechts werden die Punkte und die Namen der Spieler angezeigt. Im unteren Bereich gibt es ein Chatraum, wo sich die zwei Spielern “live” miteinander kommunizieren können. Die Spielregeln werden im Abschnitt 1.3 beschrieben.

2.4 Protokolle

2.4.1 Protokoll Client-Server

Im Rahmen dieser Semesterarbeit wird ein Spiel mit einer Client-Server Architektur entwickelt. Die Gliederung der Architektur richtet sich nach Arc42 (https://arc42.org/). Die Client-Server-Architektur ist ein Computermodell, bei dem der Server die meisten vom Client zu verbrauchenden Ressourcen und Dienste hostet, bereitstellt und verwaltet. Bei dieser Art von Architektur sind ein oder mehrere Client-Computer über eine Netzwerk- oder Internetverbindung mit einem zentralen Server verbunden.

Diese Kommunikation zwischen Client und Server erfolgt mit Hilfe eines Protokolls. Das Protokoll, an das wir gedacht haben, ist das “HTTP” Protokoll.

(HyperText Transfer Protocol) ist ein Kommunikationsprotokoll und seine primäre Funktion ist eine Verbindung mit dem Server aufzubauen und HTML-Seiten an den Browser des Benutzers zurückzusenden.

Da wir aber ein Spiel entwickeln möchten, was eine ständig offene Verbindung zwischen Client und Server verlangt, haben wir ein Problem.

HTTP ist ein "zustandsloses" (stateless) Anfrage-/Antwortsystem. Die Verbindung zwischen Client und Server wird nur für die sofortige Anforderung aufrechterhalten und die Verbindung wird geschlossen. Nachdem der HTTP-Client eine TCP-Verbindung mit dem Server aufgebaut und ihm einen Anforderungsbefehl gesendet hat, sendet der Server seine Antwort zurück und schließt die Verbindung.

Die Lösung in diesem Problem ist ein modernes Kommunikationsprotokoll zu verwenden, und zwar “Websockets”.

WebSocket ist ein Computer-Kommunikationsprotokoll, das Vollduplex-Kommunikationskanäle über eine einzige TCP-Verbindung bereitstellt. Das WebSocket-Protokoll wurde 2011 von der IETF als RFC 6455 standardisiert, und die WebSocket-API in Web IDL wird vom W3C standardisiert.

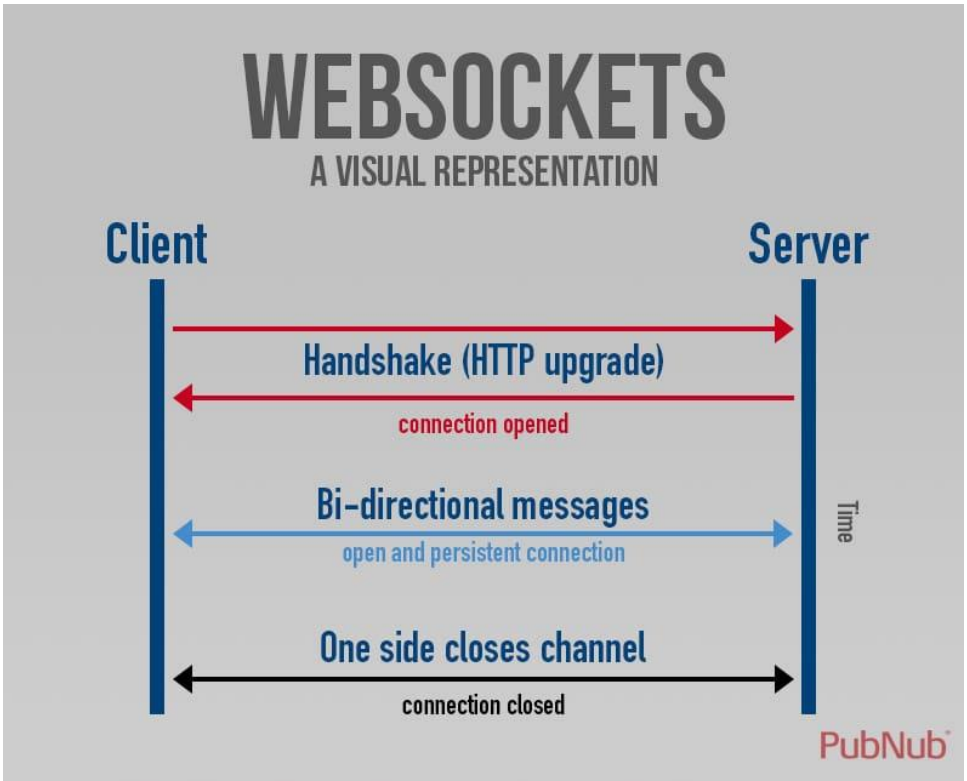


Abbildung 10: Websockets Kommunikation, https://images.ctfassets.net/3prze68gbwl1/6gIRdHedHRLNmco97gFajb/2d36a5ddfc47831ca737bbcf24e31d7c/WebSockets2.jpg

Das WebSocket-Protokoll ermöglicht die Interaktion zwischen einem Webbrowser (oder einer anderen Clientanwendung) und einem Webserver mit geringerem Overhead als Halbduplex-Alternativen, wie HTTP-Polling, wodurch die Datenübertragung in Echtzeit vom und zum Server erleichtert wird. Dies wird ermöglicht, indem dem Server ein standardisierter Weg zur Verfügung gestellt wird, um Inhalte an den Client zu senden, ohne zuvor vom Client angefordert zu werden, und es ermöglicht, Nachrichten hin und her zu übergeben, während die Verbindung geöffnet bleibt. Auf diese Weise kann zwischen dem Client und dem Server eine laufende Konversation in beide Richtungen stattfinden.

Somit lösen wir mit Websockets unser Problem und können sicherstellen, dass es immer eine offene Verbindung für das Spiel gibt.

Die Objekte, die über dieses Protokoll übermittelt werden, werden ein JSON Format haben. JSON (JavaScript Object Notation) ist ein offenes Standarddateiformat und Datenaustauschformat, das lesbaren Text zum Speichern und Übertragen von Datenobjekten verwendet, die aus Attribut-Wert-Paaren und Arrays (oder anderen serialisierbaren Werten) bestehen.

2.4.1.1 Datentypen

Wir möchten folgende Datentypen verwenden, jedoch kommen bestimmt während der Programmierung noch weitere Typen dazu.

Tabelle 18: Typen Datenaustausch

Datentype	Beschreibung
username	Der Benutzername des Spielers
data	Objekt welches benötigt wird für den Austausch der Informationen zwischen Server und Client

gameId	UUID des Spiels auch als gameroom bekannt.
timestamp	Der Zeitstempel des Servers
userId	UUID, um den Spieler zu identifizieren für den Server und Client
email	Ist für den Benutzer die userId für sein Login Identifikation
gamestate	Der Status des Spieles, Spiel 1 Start oder Spiel2start oder Spiel1 beendet oder Spiel2 beendet.
userscore	Ist dem User sein high score
highscore	High Score ist eine Liste von den Spielers Scores
rank	Das ist der Rang des Spielers auf der Spielerrangriste
level	LevelId definiert die Schwierigkeit und die Geschwindigkeit des Spieles
message	Nachrichten werden versendet

2.4.2 Netzwerkprotokolle

2.4.2.1 Protokoll - Schnittstellen Registrierung

Tabelle 19: Protokoll – Schnittstelle Registrierung

Sender	Client
Receiver	Server
Type	register
Methode	POST

Data	“type”: “register”,
	“data”: {
	“username”: “[username]”,
	“email”: “[email]”,
	“password”: “[password]”,
	“password”: “[password]”
	}

Code	201 created

2.4.2.2 Protokoll – Schnittstelle Login

Tabelle 20: Protokoll – Schnittstelle Login

Sender	Client
Receiver	Server
Type	login
Methode	GET

Data	"type": "",
	"data": {
	“email”: “[email]”,
	“password”: “[password]”
	“password”: “[password]”
	}

Code	202 accepted

2.4.2.3 Protokoll – Schnittstelle Create Game

Tabelle 21: Protokoll – Schnittstelle Create Game

Sender	Client
Receiver	Server
Type	createGame
Methode	POST
Data	“type”: “createGame”, “data”: { “gameId”: “[gameId]”, “username”: “[username]”, “userId”: “[userId]”, “levelId”: “[levelId]”, “gamestate”: “[gamestate]”,
Code	201 created

2.4.2.4 Protokoll – Schnittstelle Select level

Tabelle 22: Protokoll – Schnittstelle Select Level

Sender	Client
Receiver	Server
Type	level
Methode	GET
Data	"type": "level", "data": { "levelId": "[levelId]", "gameId": "[gameId]", "userId": "[userId]", "games": [/* Liste aller aktiven Spieler, deren high score und level des Spieles.*/]
Code	202 accepted

2.4.2.5 Protokoll – Schnittstelle Join Game

Tabelle 23: Protokoll – Schnittstelle Join Game

Sender	Server
Receiver	Client
Type	gameJoin
Methode	GET

Data	"type": "gameJoin",
	"data": {
	"gameId": "[gameId]",
	"levelId": "[levelId]"
	"username": "[username]",
	"gamestate": "[gamestate]",
	"userId": "[userId]"
	}

Code	202 accepted

2.4.2.6 Protokoll – Schnittstelle Spiel 1start

Tabelle 24: Protokoll – Schnittstelle Spiel 1 startet – Spieler1 / Server

Sender	Client[1]
Receiver	Server
Type	gameStart1
Methode	GET

Data	{
	“type”: “gameStart”,
	“data”: {
	“username”: “[username]”,
	“gamestate”: “[gamestate]”,
	“userId”: “[userId]”,
	}
	}

Code	200 OK

Tabelle 25: Protokoll – Schnittstelle Spiel 1 startet – Server / Spieler2

Sender	Server
Receiver	Client[2]
Type	gameStart
Methode	GET

Data	{
	"type": "gameStart",
	"data": {
	"username": "[username]",
	"gameId": "[gameId]",
	"gamestate": "[gamestate]",
	"message": "[message]",
	}
	}

Code	200 ok

Tabelle 26: Protokoll – Schnittstelle Spiel 1 startet – Spieler2 /Server

Sender	Client[2]
Receiver	Server
Type	gameStart
Methode	GET

Data	{
	“type”: “gameStart”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

Tabelle 27: Protokoll – Schnittstelle Spiel 1 startet – Server / Spieler1

Sender	Server
Receiver	Client[1]
Type	gameStart
Methode	GET

Data	{
	"type": "gameStart”
	"data": {
	"username": "[username]",
	"gameId": "[gameId]",
	"message": "[message]",
	}
	}

Code	200 ok

2.4.2.7 Protokoll – Schnittstelle Spiel 1 beendet

Tabelle 28: Protokoll – Schnittstelle Spiel 1 beendet – Server / Spieler1

Sender	Server
Receiver	Client[1]
Type	gameEnd
Methode	GET

Data	{
	“type”: “gameEnd”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“userscore”: “[highscore]”
	“message”: “[message]”,
	}
	}

Code	200 ok

Tabelle 29: Protokoll – Schnittstelle Spiel 1 beendet – Server / Spieler2

Sender	Server
Receiver	Client[2]
Type	gameEnd
Methode	GET

Data	{
	“type”: “gameEnd”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“userscore”: “[highscore]”
	“gamestate”: “[gamestate]”,

	} }
Code	200 ok

Tabelle 30: Protokoll – Schnittstelle Spiel 1 – Server / Spieler2

Sender	Server
Receiver	Client[2]
Type	gameEnd
Methode	GET

Data	{
	“type”: “gameEnd”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“userscore”: “[userscore]”
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

2.4.2.8 Protokoll – Schnittstelle Spiel 2 startet

Tabelle 31: Protokoll – Schnittstelle Spiel 2 startet – Spieler1 / Server

Sender	Client[1]
Receiver	Server
Type	gameStart
Methode	GET

Data	{
	“type”: “gameStart”,
	“data”: {
	“username”: “[username]”,
	“userId”: “[userId]”,
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

Tabelle 32: Protokoll – Schnittstelle Spiel 2 startet – Server / Spieler2

Sender	Server
Receiver	Client[2]
Type	gameStart
Methode	GET

Data	{
	“type”: “gameStart”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

Tabelle 33: Protokoll – Schnittstelle Spiel 2 startet – Spieler2 /Server

Sender	Client[2]
Receiver	Server
Type	gameStart
Methode	GET
Data	{

	“type”: “gamestate”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

Tabelle 34: Protokoll – Schnittstelle Spiel 2 startet – Server / Spieler1

Sender	Server
Receiver	Client[1]
Type	gameStart
Methode	GET

Data	{
	“type”: “gameStart”
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

2.4.2.9 Protokoll – Schnittstelle Spiel 2 beendet

Tabelle 35: Protokoll – Schnittstelle Spiel 1 beendet – Server / Spieler1

Sender	Server
Receiver	Client[1]
Type	gameEnd
Methode	GET

Data	{
	“type”: “gameEnd”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“userscore”: “[userscore]”
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

Tabelle 36: Protokoll – Schnittstelle Spiel 1 beendet – Server / Spieler2

Sender	Server
Receiver	Client[2]
Type	gameEnd
Methode	GET

Data	{
	“type”: “gameEnd”,
	“data”: {
	“username”: “[username]”,
	“gameId”: “[gameId]”,
	“userscore”: “[userscore]”
	“gamestate”: “[gamestate]”,
	}
	}

Code	200 ok

2.4.2.10 Protokoll – Schnittstelle High Score

Tabelle 37: Protokoll – Schnittstelle High Score – Server / Client

Sender	Server
Receiver	Client[1]
Type	highScore
Methode	POST
Data	{ “username”:”[username]” “userscore”: “[userscore]”}
Code	201 created

Tabelle 38: Protokoll – Schnittstelle High Score – Server / Client

Sender	Server
Receiver	Client[1]
Type	highScore
Methode	GET
Data	{ “username”:”[username]” “userscore”: “[userscore]”, “rank”: “[rank]”, “highscores”: [{ “rank”: “{rank}”, “username”: “{username}”, “userId” : “{userId}”, “userscore” : “{Integer}” }]}
Code	200 ok

2.4.2.11 Protokoll – Schnittstelle Chat

Tabelle 39: Protokoll – Schnittstelle Chat – Client/Server

Sender	Client[i]
Receiver	Server
Type	chat
Methode	POST

	{
	“type”: “chat”,
	“data”: {
	“timestamp”: “[timestamp | YYYY-MM-DDThh:mm:ss]”,
	“username”: “[username]”,
	“userId”: “[userId]”,
	“message”: “[message]”
	}
	}

Code	202 accepted

Tabelle 40: Protokoll – Schnittstelle Chat – Server/Client

Sender	Server
Receiver	Client[i]
Type	chat

	{
	“type”: “chat”,
	“data”: {
	“timestamp”: “[timestamp | YYYY-MM-DDThh:mm:ss]”,
	“username”: “[username]”,

	“userId”: “[userId]”, “message”: “[message]” } }
Code	201 accepted

3 Annexes

3.1 Protokolle

3.1.1 Protokoll vom 02.09.2021

Teilnehmer	Theologos Baxevanos
	Chantale Gihara
Datum Ort	02.09.2021 Starbucks Zürich Oerlikon
Informationen	• Use Case Diagramme besprechen • Mock up wird abgenommen • Anforderungen werden zusammen reviewt und validiert. • Abgabe wird besprochen
Diskussionen	• Protokoll und Schnittstelle sind wir nicht ganz sicher. Wir wünschen uns gerne ein Feedback des Dozenten nach unserer Abgabe. • Das Schnittstellen Protokoll ist in Meilenstein 1 noch nicht gefordert, wir möchten es aber schon verstehen, deshalb implementieren wir es schon in die Arbeit, damit wir so rasch als möglich ein Feedback dazu haben.
Meilensteine	• Meilenstein 1 abgeschlossen somit Iteration 1 abgeschlossen • Meilenstein 2 kann nun gestartet werden • Zeitplanung der Meilensteine und der Arbeitspaket sind im Kanban Board von Trello siehe Link next Steps ersichtlich.
Next Steps	• https://trello.com/b/7HNVSgbQ/kanban-webe-coronattack • Chantale startet mich der ersten Phase der Programmierung • Theo zieht in Meilenstein 2 seine Karten, verfeinert die Userstory mit Checklist.
Nächstes Meeting	• 11.09.2021 vor Ort PVA2 WebE

3.1.2 Protokoll vom 21.08.2021

Teilnehmer	Theologos Baxevanos
	Chantale Gihara
Datum Ort	21.08.2021 Starbucks, Zürich Oerlikon Bahnhof
Informationen	• Arbeitsaufteilung • Meilensteine noch einmal revidieren. • Anforderungen werden zusammen reviewt und validiert. • Bis wann muss abgegeben werden • Arbeitspakete und Projektstrukturplan, wie Balkendiagramm wird von Chantale übernommen • Jede Arbeit wird bei Beendung
Diskussionen	• Wie sollte das Spiel aussehen? Theo macht eine Skizze, sendet Chantale das Mock up asap zu. • Arbeitsaufteilung • Use Cases werden zusammen erstellt, dabei werden die Beschreibungen und die Diagramme aufgeteilt. • Server/Client Protokoll wird von Theo übernommen.
Meilensteine	• Meilenstein 1 in Bearbeitung • Kanban Board wird erstellt mit Arbeitspaketen, damit die Meilensteine überwacht werden können. Dafür werden wir Trello nehmen.
Next Steps	• Kanban Board folgen nach Erstellung • Jeder schaut was zu reviewen ist, wenn der andere das Arbeitspaket zur Review schiebt. • Schauen das wir alles bis zum nächsten Meeting fertigstellen, damit wir bereit sind fürs
Nächstes Meeting	• 02.09.2021, Starbucks, Zürich Oerlikon Bahnhof

3.1.3 Protokoll vom 04.09.2021

Teilnehmer	Theologos Baxevanos
	Chantale Gihara
Datum Ort	04.09.2021 Welle 7, Bern
Informationen	• Projektstruktur wird angeschaut • Erster Draft React erster Versuch wird angeschaut • Protokolle und deren Schnittstellen wurde beschrieben
Diskussionen	• Wie sollte das Spiel aussehen? Theo macht eine Skizze, sendet Chantale das Mock up asap zu. • Arbeitsaufteilung • Use Cases werden zusammen erstellt, dabei werden die Beschreibungen und die Diagramme aufgeteilt. • Server/Client Protokoll wird von Theo übernommen.
Meilensteine	• Meilenstein 2 in Bearbeitung • Kanban Board wird erstellt mit Arbeitspaketen, damit die Meilensteine überwacht werden können. Dafür werden wir Trello nehmen. • Meilenstein 1 wurde zeitgerecht eingereicht
Next Steps	• Docker Container erstellen • Registrierung und Login fertigstellen • Chat Funktion erstellen
Nächstes Meeting	• 27.09.2021, Welle 7, Bern

3.1.4 Protokoll vom 27.09.2021

Teilnehmer	Theologos Baxevanos
	Chantale Gihara
Datum Ort	27.09.2021 Teams, online
Informationen	• Überprüfung was gemacht werden muss • Stand Austausch
Diskussionen	• Protokoll anschauen • Use Case Diagramme entfernen und zusammenfassen • Server/Client Protokoll wird von Theo übernommen. • Registration abschliessen und mit Protokoll abgleichen
Meilensteine	• Meilenstein 2 in Bearbeitung • Abgabe vom 01. Oktober im Auge behalten
Next Steps	• Server aufsetzen • Client aufsetzten • Docker fixieren • Registrierung und Login fertigstellen • Chat Funktion erstellen
Nächstes Meeting	• 30.09.2021, Teams online

3.1.5 Protokoll vom 30.09.2021

Teilnehmer	Theologos Baxevanos
	Chantale Gihara
Datum Ort	30.09.2021 Teams, online
Informationen	• Überprüfung Server, Client, Chat, Registration • Protokoll Besprechung • Stand Austausch
Diskussionen	• Was wird abgeben und wer • Wie funktionieren wir weiter
Meilensteine	• Meilenstein 2 in Bearbeitung • Meilenstein 2 Abgabe morgen 01.10.2021
Next Steps	• Server verbessern • Client verbessern • Daten in DB speichern • Spiel aufsetzten

Nächstes Meeting	• 08.10.2021, vor Ort PVA3 WebE => Welle7, Bern

Tabellen

Abbildungen

Quellen

Bloch, J. (2001). Writing Effective Java. In Addison-Wesley, & 1. edition (Ed.), *Writing Effective Java.*

Brumfit, J. (2002). Java coding standard and guidelines for the Herschel common science system. In J. Brumfit, *Java coding standard and guidelines for the Herschel common science system.* ESTEC.

Control, E. B. (2004). Java coding standards. In E. B. Control, & ESA (Ed.), *Java coding standards.* ESA Board for Software Standardization and Control. Retrieved from Online verfügbar:http://www.rssd.esa.int/llink/livelink/Java coding standards.pdf?func=doc.Fetch&nodeId=5

creately. (n.d.). *Use Case Diagram Tutorial (Guide with examples).* creately. Retrieved 08 25, 2021, from https://creately.com/blog/diagrams/use-case-diagram-tutorial/

Dieter Kunz, N. S. (2018). Handbuch WBT Projektmanagement. In N. S. Dieter Kunz, *Handbuch WBT Projektmanagement* (Vol. 5. Auflage). ConPlus Gunten + Partner.

G. Steele J. Gosling, B. J. (2000). The Java Language Specification. In Addison-Wesley, *The Java Language Specification* (Vol. 2nd edition).

Inden, M. (2017). Der Weg zum Java-Profi. In M. Inden, *Der Weg zum Java-Profi* (Vol. 4. Auflage).

Ludewig, J. u. (2013). Software Engineering. In *Software Engineering* (Vol. 3. Auflage). Horst.

Martin, R. C. (2009). Clean Code. In R. C. Martin, *Clean Code.*

Microsystems, S. (1999). Code conventions for the java programming language. In S. Microsystems, *Code conventions for the java programming language.* Sun Microsystems. Retrieved from http://java.sun.com/docs/codeconv

Microsystems, S. (1999). Java look and feel design guidelines. In S. Microsystems, *Java look and feel design guidelines.* Sun. Retrieved from http://java.sun.com/products/jlf/dg/index.html

Microsystems, S. (2000). How to write doc comments for Javadoc. In S. Microsystems, *How to write doc comments for Javadoc.* Sun. Retrieved from http://java.sun.com/products/jdk/javadoc/writingdoccomments/index.html

Pohl, K. u. (2015). Basiswissen Requirements Engineering. In *Basiswissen Requirements Engineering* (Vol. 4. Auflage).

Seidl, M. u. (2012). UML@Classroom. In M. u. Seidl, *UML@Classroom* (Vol. 1. Auflage).

Starke, G. (2015). Effektive Software-Architekturen. In G. Starke, *Effektive Software-Architekturen* (Vol. 7. Auflage).

Zörner, S. (n.d.). DokChess nach arc 42. In S. Zörner, *DokChess nach arc 42.* Retrieved 02 22, 2021, from https://www.dokchess.de