



AUGMENTED MULTI-PLAYER CARD GAME

Pflichtenheft zur Bachelor-Thesis

Studiengang:	Bsc Informatik
Autor:	Samuel Grimm
Betreuer:	Prof. Urs Künzler
Experte:	Dr. Federico Flueckiger
Datum:	20.01.2022
Version:	1.0

1	EINLEITUNG.....	4
2	SPIELBESCHRIEB.....	5
2.1	Kurzbeschrieb	5
2.2	Nebenrollen	6
2.3	Waffen.....	6
2.4	Aktionskarten.....	7
2.5	Weitere Gegenstandskarten	9
3	ARCHITEKTURBESCHREIBUNG/SCHNITTSTELLEN	11
3.1	Systemübersicht	11
3.2	Software-Architektur	12
3.2.1	Game.....	12
3.2.2	Finger-Tracking-Service.....	13
3.2.3	App	13
3.2.3.1	Flutter	13
3.2.4	Game States (DB).....	15
3.2.5	Authentification.....	15
3.2.6	Game Services	15
3.2.7	Software	15
4	ANFORDERUNGEN	16
4.1	Systemumfang.....	16
4.1.1	Muss-Kriterien	16
4.1.2	Soll-Kriterien	16
4.1.3	Optionale Kriterien	16
4.2	Systemfunktionalität.....	17
4.3	Akteure	19
4.4	Funktionale Anforderungen	20
4.4.1	Minimales Gameplay.....	20
4.4.1.1	Automatische Aktionen beim Spielstart	20
4.4.1.2	Spielzug.....	20
4.4.1.3	Grundlegende Karten und Spiellogik	21
4.4.2	Erweitertes Gameplay	23
4.4.2.1	Optional.....	23
4.4.3	Allgemeine Anforderungen an das System	24
4.4.3.1	Spiel-Teilnahme.....	24
4.4.3.2	Optional.....	25
4.5	Nicht funktionale Anforderungen.....	26
4.5.1	Hardware	26
4.5.2	Serverseitige Daten	26

4.6	Spießfluss	28
4.6.1	Fluss zwischen den einzelnen Screens (generell)	28
4.6.2	Fluss zwischen den einzelnen Screens während des Spieles	31
4.7	Einschränkungen	34
4.7.1	Abgrenzung	34
4.7.2	Einschränkungen	34
5	ABNAHMETESTS	34
5.1	Test der nichtfunktionalen Anforderungen	34
6	PROJEKTMANAGEMENT	35
6.1	Projektorganisation	35
6.2	Stakeholder und deren Aufgaben	35
6.3	Gantt-Diagramm	35
6.4	Meilensteine	36
7	LITERATURVERZEICHNIS	37
8	ABBILDUNGSVERZEICHNIS	37

1 Einleitung

Ein neues, komplexes Kartenspiel kennenzulernen oder jemandem beizubringen, ist manchmal ziemlich schwer und kann frustrierend enden, weil Gesagtes oft nicht auf Anhieb klar ist und es viele Fragen gibt, die nicht immer gestellt werden können.

In einer solchen Situation wäre es manchmal praktisch, wenn man als Spieler zu jeder Karte angezeigt bekommt, wie und ob diese gespielt werden kann.

Hier kommt das Augmented Card Game «ins Spiel». Es führt alle Anfänger unkompliziert durch das Spiel hindurch und nimmt z.B. die Arbeit des (Lebens)Punkte-Zählens ab. Ausserdem soll es möglich sein, mit den Liebsten «am gleichen Tisch» zu spielen, auch wenn jemand nicht physisch am gleichen Ort ist.

Um das Spielgeschehen möglichst authentisch zu machen, sitzen (fast) alle am gleichen Tisch und können mit den virtuellen Karten auf dem Tisch interagieren.

In diesem «Augmented Card Game» soll es möglich sein, das Spiel «Bang!» zu spielen. Dieses Projekt soll ein Prototyp von diesem «Augmented Card Game» liefern.

Dieses Dokument ist das Pflichtenheft zu diesem Projekt. Zuerst wird das Kartenspiel grundsätzlich erklärt, dann wie es architektonisch geplant ist, welche Anforderungen das System erfüllen sollte, wie der Spielfluss aussehen wird und wie das System getestet werden soll.

2 Spielbeschreibung

2.1 Kurzbeschreibung

Das Spiel «Bang!» von daVinci Games und Abacusspiele [1] ist ein rundenbasiertes Kartenspiel, das für 4 bis 7 Spieler geeignet ist. Jeder Spieler hat eine der folgenden Rollen:



Abbildung 1: Rollen des Spieles «Bang!».

Die einzig bekannte Rolle ist diejenige des Sherifs, alle anderen sind unbekannt bis zu dessen Tod. Das Ziel des Sherifs und der Hilssheriffs ist es, alle Banditen und den Gesetzlosen auszuschalten. Die Banditen wiederum wollen den Sheriff ausschalten. Der Gesetzlose ist auf keiner der beiden Seiten, er möchte der letzte Überlebende sein. Ein Spieler wurde eliminiert, wenn er/sie keine Lebenspunkte mehr hat.

Das Spielende ist erreicht, wenn eine der folgenden Situationen eintritt:

Bedingung	Gewonnen hat
▪ Der Sheriff wurde eliminiert	▪ Wenn nur noch der Gesetzlose lebt, dann er, ▪ ansonsten die Banditen
▪ Alle Banditen und der Gesetzlose wurden eliminiert	▪ Der Sheriff und alle Hilssheriffs

Das Spiel ist rundenbasiert. Konkret bedeutet das, dass jeweils nur ein Spieler an der Reihe ist und seinen Spielzug ausführen kann. Der Spielzug ist in drei Phasen unterteilt:

1. Phase: Der Spieler zieht 2 Karten vom Nachziehstapel.
2. Phase: Beliebige viele Karten spielen.
3. Phase: Überzählige Karten abspielen: Am Ende des Spielzuges darf der Spieler maximal so viele Karten auf der Hand haben wie er/sie Lebenspunkte hat.

Nach dem Spielzug kommt der nächste Spieler links an die Reihe.

2.2 Nebenrollen

Alle Spieler haben eine zweite Rolle. Am Anfang des Spieles werden jedem Spieler 2 Charakterkarten zugeteilt. Nur eine davon kann der Spieler ansehen. Ist er/sie mit der aufgedeckten Charakterkarte nicht zufrieden, so deckt er/sie die zweite Charakterkarte auf und die erste Charakterkarte gilt somit nicht mehr für ihn/sie. Ansonsten gilt die als erstes aufgedeckte Charakterkarte.

Die Nebenrollen/Charakterkarten bestimmen, wie viele Lebenspunkte ein Spieler maximal haben kann. Diese sind mit der Anzahl Patronen rechts auf der Karte gekennzeichnet. Der Sheriff hat einen Lebenspunkt mehr.



Abbildung 2: Charakterkarten, nicht vollständig.

Zusätzlich haben alle Charaktere eine Spezialfunktion, die auf der Karte beschrieben ist.

2.3 Waffen

Waffen-Karten werden während des Spielzuges ausgespielt. Alle Spieler dürfen maximal 1 Waffe vor sich spielen. Möchte der Spieler eine andere Waffe, so kann er/sie während des Spielzuges eine bereits vor sich gespielte Karte wegwerfen.

Die Waffen bestimmen die Reichweite, die der Spieler hat. Ohne Waffe hat der Spieler eine Reichweite von 1. Dies bedeutet, dass er/sie nur die Spieler, die direkt neben ihm/ihr sind, mit der Karte «Bang», treffen kann.

Eine spezielle Waffe ist die Volcanic: Hat ein Spieler diese Karte vor sich gespielt, so kann er beliebig viele «Bang»-Karten pro Spielzug spielen, ansonsten ist nur 1 «Bang» pro Spielzug erlaubt.

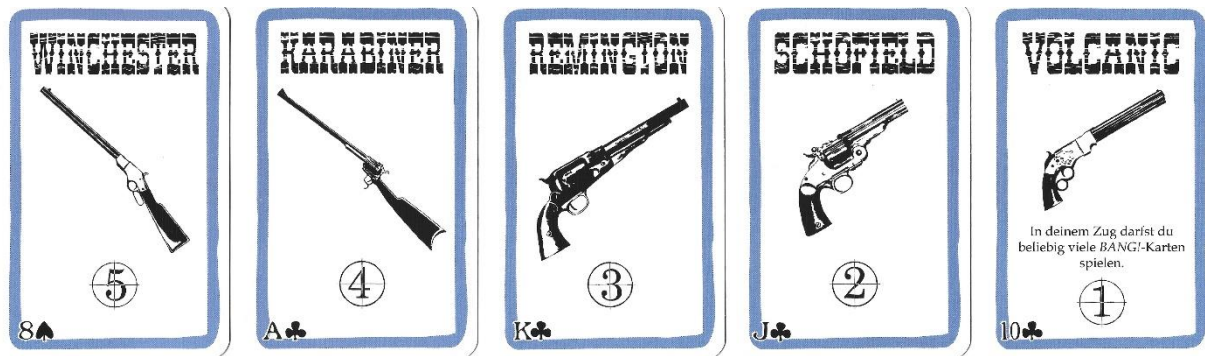


Abbildung 3: Waffenkarten

2.4 Aktionskarten

Die wichtigste Karte im Spiel ist die «Bang»-Karte. Diese Karte kann ein Spieler während seines Spielzuges gegen einen anderen Spieler spielen, der in seiner Reichweite liegt. Der angegriffene Spieler darf gegen eine «Bang»-Karte mit einem «Fehlschuss» reagieren (also von der Hand abwerfen). Tut er dies nicht, so verliert der angegriffene Spieler 1 Lebenspunkt.

Während des Spielzuges dürfen beliebig viele «Bier»-Karte gespielt werden. Pro solche Karte erhält dieser Spieler einen Lebenspunkt dazu. Die maximale Lebenspunkte dürfen aber nicht überschritten werden. Das Einzigartige dieser Karte ist, dass diese auch ausserhalb des Spielzuges von der Hand gespielt werden darf, wenn derjenige Spieler von einem Schuss getroffen wird.



Abbildung 4: Die wichtigsten Aktionskarten des Spieles

Die Symbole werden auf der Hilfskarte gut zusammengefasst:



Abbildung 5: Zusammenfassung der Symbole auf den Karten

Neben dem «Bang» und «Bie» gibt es weitere Möglichkeiten, anderen Lebenspunkte abzuziehen oder hinzuzufügen.

Der «Catling» zieht allen anderen Spielern 1 Lebenspunkt ab, die diesen Angriff nicht mit einem «Fehlschuss» abwehren. Der «Saloon» macht genau das Gegenteil: Alle Spieler erhalten einen Lebenspunkt zurück.

Wird der «Indianen» gespielt, so verlieren alle anderen Spieler einen Lebenspunkt, die nicht eine «Bang»-Karte spielen können.

Ein «Duell» wird hingegen gegen einen Spieler gespielt, den man auswählen kann. Anschließend werfen der angegriffene Spieler und der Spieler, der an der Reihe ist, abwechselungsweise eine «Bang»-Karte ab. Derjenige, der zuerst keine «Bang»-Karte mehr spielt, verliert einen Lebenspunkt.



Abbildung 6: Karten, mit welchen andere Lebenspunkte verlieren oder bekommen.

Mit einem «Wells Fargo» oder einer «Postkutsche» können 2 bzw. 3 Karten vom Nachziehstapel gezogen werden. Bei einem «Warenhaus» werden so viele Karten aufgedeckt, wie noch Spieler im Spiel sind. Anschliessen können alle Spieler beginnend beim Spieler, der an der Reihe ist, weiter in Spielrichtung eine Karte davon auf die Hand nehmen.

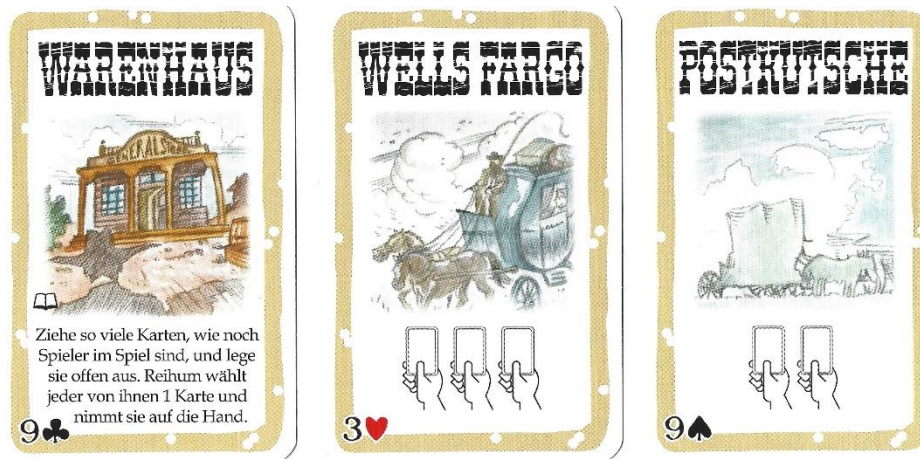


Abbildung 7: Karten, die bewirken, dass Karten gezogen werden können.

Mit dem «Cat Balou» kann der Spieler einen beliebigen anderen Spieler dazu zwingen, eine Handkarte abzuwerfen. Mit der «Panik»-Karte hingegen kann der Spieler eine beliebige Karte von einem anderen Spieler im Abstand 1 aufnehmen, die entweder vor dem anderen Spieler ausgespielt ist oder sich in der Hand des Spielers befindet.

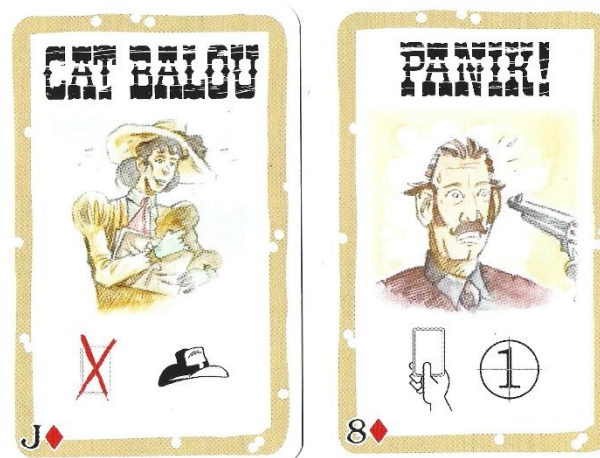


Abbildung 8: Karten, die eine Karte einem anderen Spieler entfernen.

2.5 Weitere Gegenstandskarten

So wie die Waffen werden diese blauen Karten vor einem Spieler gespielt.

Das «Dynamit» wird zuerst vor sich selbst abgespielt. Befindet sich diese Karte am Anfang des Spielzuges vor einen, muss man am Anfang des Spielzuges zuerst eine Karte aufziehen und wegwerfen. War unten links auf dieser Karte eine Schaufel, verliert dieser Spieler 3 Lebenspunkte. Ansonsten wandert diese Karte zum Spieler links von ihm/ihr.

Das «Gefängnis» kann vor einen beliebigen Spieler gespielt werden, der nicht Sheriff ist. Hat der Spieler am Anfang seines Zuges das Gefängnis vor sich, muss er zuerst eine Karte vom Stapel ziehen. Hatte diese Karte unten links ein Herz abgebildet, so kann er das Gefängnis wegwerfen und normal seinen Zug starten. Ansonsten kann er auch das Gefängnis wegwerfen, muss aber seinen Spielzug passen.

AUGMENTED MULTI-PLAYER CARD GAME

Mit dem «Zielfernrohr» oder «Mustang» sehen Spieler einen mit einem um 1 grösseren bzw. kleineren Abstand, was bedeutet, dass sich die Reichweite entsprechend um 1 verändert.

Das «Fass» kann der Spieler vor sich selbst spielen. Dies bewirkt, dass immer, wenn auf ihn geschossen wird, er eine Karte vom Stapel aufdecken kann. Falls unten links auf dieser Karte ein Herz war, so zählt diese Karte als «Fehlschuss». Das «Fass» bleibt aber bestehen nach dem Schuss.



Abbildung 9: weitere blaue Karten, die vor Spieler gespielt werden.

3 Architekturbeschreibung/Schnittstellen

3.1 Systemübersicht

Aus Hardware-Sicht ist das System so geplant:

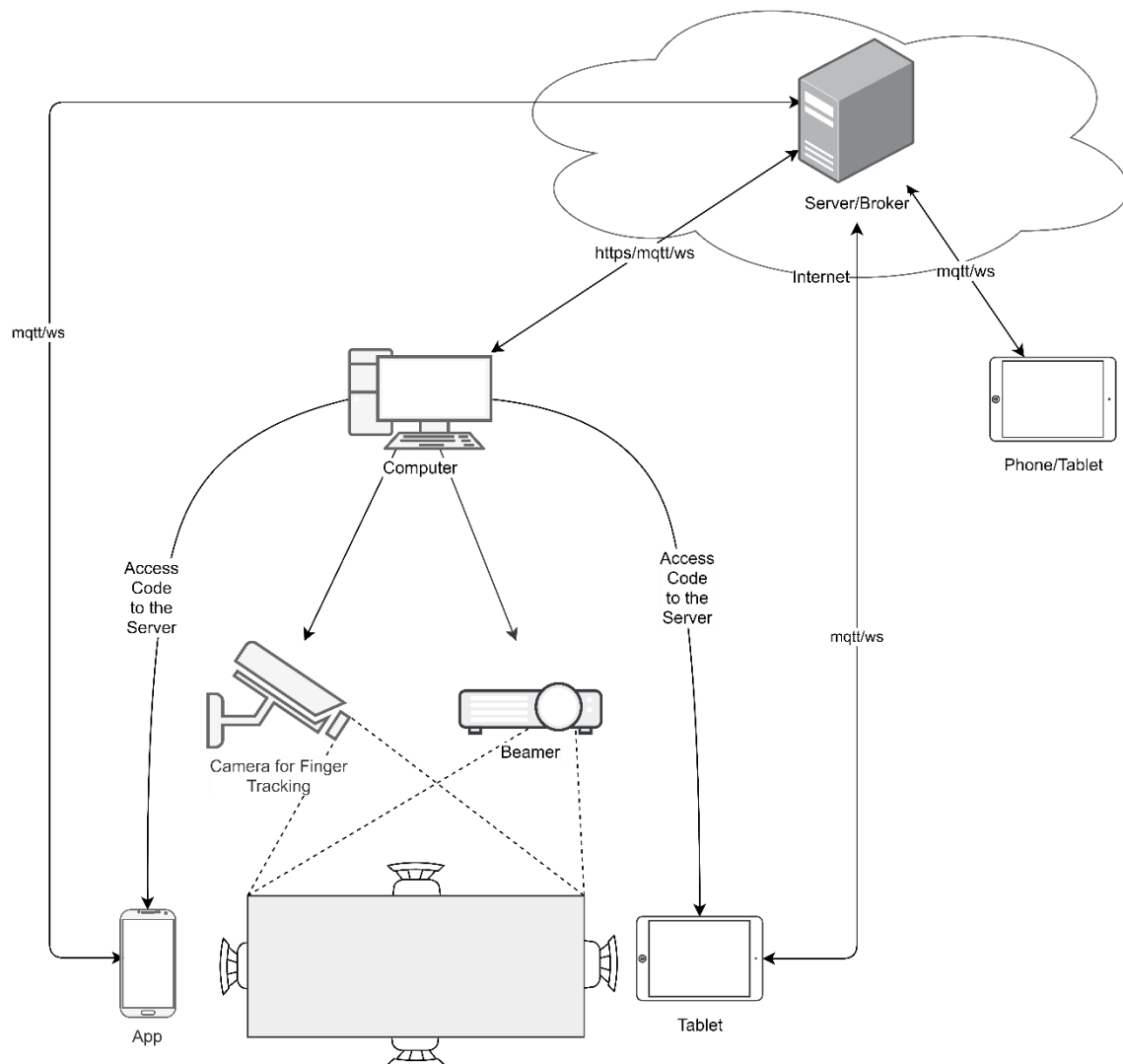


Abbildung 10: Hardware-Ansicht des Systems

Bis zu 4 Spieler sitzen an einem Spieltisch. Alle Spieler haben ihre Karten auf ihrem Smartphone oder Tablet. Auf dem Tisch vor ihnen wird das aktuelle Spielgeschehen mit einem Beamer projiziert. Wird von einem physisch am Tisch sitzenden Spieler eine Karte auf dem Spieltisch ausgewählt, so soll dies mit Handbewegung möglich sein. Hierfür kommt eine Kamera zum Einsatz, die die Handbewegungen der Spieler trackt.

Der Beamer und die Kamera für das Hand-Tracking sind mit einem Computer verbunden, auf dem das Spiel läuft.

Alle Tablets und Smartphones mit der App dieses Systems sind mit einem Server indirekt mit diesem Computer verbunden. Damit sich die Apps korrekt verbinden können, wird ein Code (QR-Code oder ein Text-/Zahlencode) auf den Tisch projiziert, der dann eingegeben bzw. gescannt werden kann. Dieser Code soll auch von solchen Spielern verwendet werden können, die nicht physisch am Tisch sitzen können.

3.2 Software-Architektur

Software-technisch ist folgende Architektur geplant:

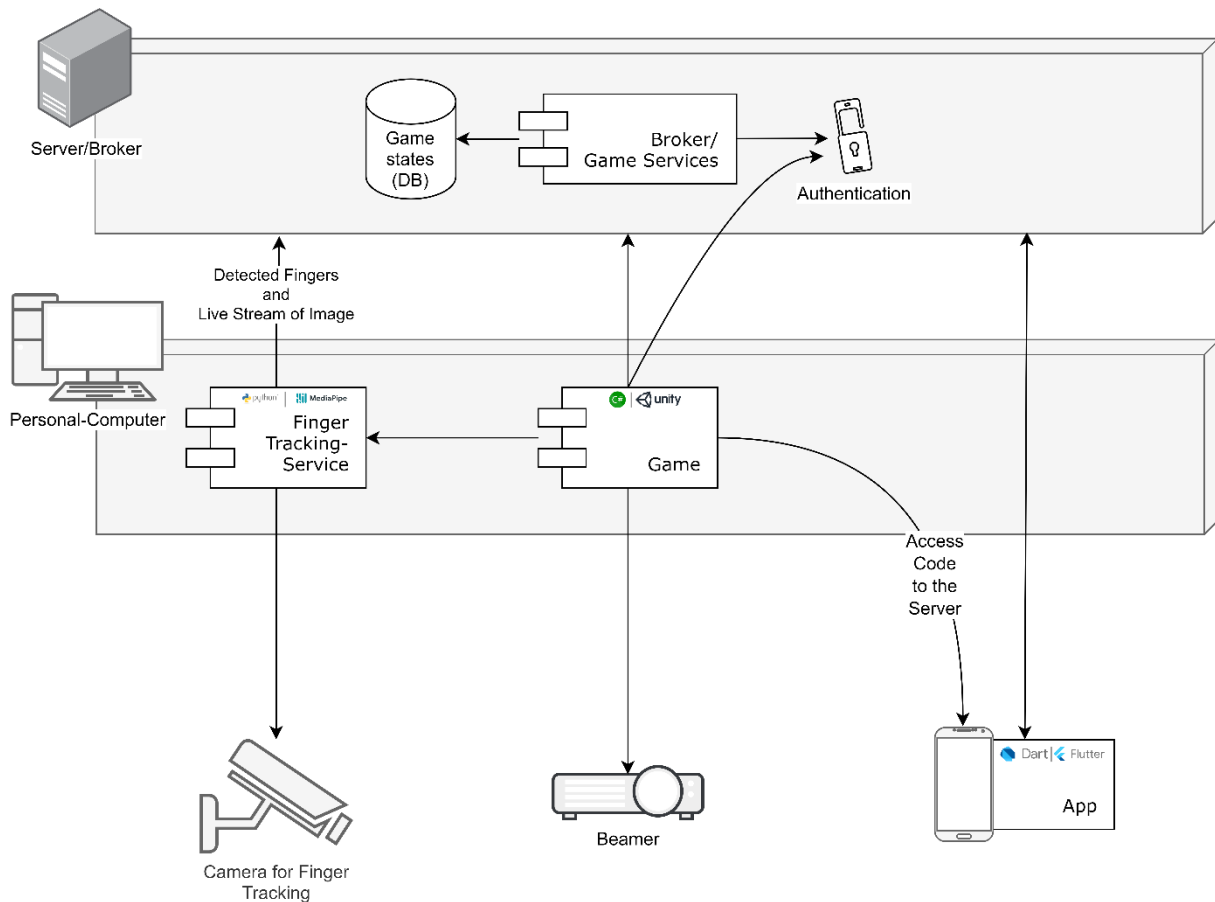


Abbildung 11: Software-Architektur des Systems

Das eigentliche Game läuft auf einem Computer, das mit der Kamera und dem Beamer verbunden ist. Das Game bekommt die Handtracking-Daten von einem «Finger Tracking»-Service. Das Game soll ausserdem bestimmte Parameter von diesem Service konfigurieren können. Das Game ist zusätzlich für die Spiellogik und das projizierte Bild verantwortlich.

Den aktuellen Spielstand muss auf einen Server aktualisiert werden, damit in der App der aktuelle Spielstand angezeigt werden kann.

In der Smartphone- und Tablet-App müssen die Handkarten des Spielers angezeigt werden. Des Weiteren wird angestrebt, dass alle Spielhandlungen auch mit der App durchgeführt werden können. Dies bedeutet, dass das projizierte Bild in einer ähnlichen Weise auch auf der App ersichtlich sein muss, und dass die Aktionen, die mit dem Finger-Tracking getätigt werden, auch mit der App ausführbar sein müssen.

In den folgenden Unter-Unter-Kapitel werden einzelne Software-Komponenten dieses Systems genauer erläutert.

3.2.1 Game

Diese Komponente ist das Herzstück des Systems. Dieses Unity-Spiel ist mindestens für folgendes verantwortlich:

- Rendern des Spieltisches, der mit dem Beamer projiziert wird

- Abfragen der Finger-Tracking-Informationen, um allfällige Aktionen im Spiel durchzuführen
- Festlegen von Parametern des Finger-Tracking-Services (z.B. Authentifizierung zu Servern, Korrekte Kamera, usw.)
- Bereitstellen eines Codes, damit sich App-Instanzen damit verbinden können.
- Wenn der Finger-Tracking-Service verwendet werden soll: Sicherstellen, dass dieser Service läuft

Unity wird in diesem Dokument nicht weiter erläutert, weil davon ausgegangen wird, dass diese Game-Engine den Lesern dieses Dokumentes vertraut sein sollte.

3.2.2 Finger-Tracking-Service

Um das Spiel mit Handbewegungen auf dem Tisch steuern zu können, sollen die Hände bzw. die Finger mit einer KI-basierten Bilderkennung erkannt werden. Aufgrund Erfahrungen wird dies mit der open-source und cross-platform Machine-Learning-Lösung *MediaPipe* von Google [2] versucht. *MediaPipe* unterstützt verschiedene Programmiersprachen wie C++, Python, JavaScript (im Browser) und Java (unter Android) und liefert auf einfache Art und Weise 3D-Kordinaten der Fingergelenke und -Spitzen.

Der zu entwickelnde Service soll für folgendes verantwortlich sein:

- Bereitstellen der Fingergelenk-Positionen
- Entgegennehmen von Konfigurations-Parametern (wie zu verwendende Kamera)
- Evtl. Kamera-Bild der App via Server zur Verfügung stellen

3.2.3 App

Die Smartphone- und Tablet-App soll den Spielern ihre Handkarten anzeigen. Spieler, die nicht physisch am Tisch sitzen (also per Remote mitspielen), sehen den aktuellen Spielscreen in der App und können dort auch Handlungen durchführen, die die Spieler sonst am Spieltisch z.B. mit Handgesten machen können.

Die App wird mit dem Framework «Flutter» und der Programmiersprache «Dart» entwickelt, welches im nächsten Unter-Unter-Unterkapitel vorgestellt wird.

3.2.3.1 Flutter

Flutter ist Googles open-source UI-Toolkit, um Cross-Platform-Apps mit nativer User Experience für Mobile (Android, iOS), Web, Desktop und Embedded-Devices mit nur einer Code-Base zu entwickeln. [3] Als Programmiersprache kommt das von Google entwickelt Dart zum Einsatz, welches eine moderne Alternative zu JavaScript darstellen soll. Dart bietet seit Kurzem *Sound Null Safety* an, welches per Default keine Null-Werte zulässt, während der Kompilier- sowie während der Laufzeit non-nullable Variablen während Zuweisungen auf 'null' überprüft und somit der häufigsten Fehlerursache (Nullpointer-Exception) entgegenwirkt. [4] Ein weiteres Key-Feature von Flutter (und Dart) ist *Hot Reload*, welches Code-Änderungen sehr schnell während der Laufzeit akzeptiert und die entsprechenden Klassen und Widgets direkt in der laufenden App aktualisiert. [5]

Alle UI-Elemente in Flutter sind Widgets (erben meistens direkt entweder von der Klasse *StatelessWidget* oder *StatefulWidget*), die ein in dieses Widget verschachteltes Widget rendern. Daraus resultiert ein sogenannter *WidgetTree*, der die Hierarchie der Widgets festhält. Soll sich ein bestimmtes Widget verändern, so wird dieses (und oft alle untergeordneten Widgets auch!) neu erstellt und im *WidgetTree* aktualisiert. Aus dem *WidgetTree* entnimmt Flutter die Information, welche nativen UI-Elemente gerendert

werden sollen. Flutter entscheidet im Normalfall selbst, welche nativen UI-Elemente neu erstellt und welche wiederverwendet werden sollen. Alle *StatefulWidget*s besitzen z.B. eine *setState(...)*-Methode, welche die *build(...)*-Methode ihres States aufruft und damit ihr Child-Widget (und somit den ganzen untergeordneten *WidgetTree*) neu erstellen lässt.

Um das dieses Konzept besser zu erklären, ist nachfolgend der komplette Dart-Source-Code einer App abgebildet, in welcher der Benutzer auf einen Button klicken kann und sich dann eine Zahl bei jedem Klick inkrementiert (Source-Code von [3]):

```
import 'package:flutter/material.dart';
class Counter extends StatefulWidget {
  @override
  _CounterState createState() => _CounterState();
}

class _CounterState extends State<Counter> {
  int val = 0;

  void change() {
    setState(() => val++);
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          children: <Widget>[
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Center(child: Text('$val'))),
            ElevatedButton(
              child: const Text('Add'),
              onPressed: change,
            ),
          ],
        ),
      ),
    );
  }
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Center(
        child: Counter(),
      ),
    );
  }
}

Future<void> main() async {
  runApp(MyApp());
}
```

Listing 1: Kompletter Dart-Code einer Counter-App mit Flutter

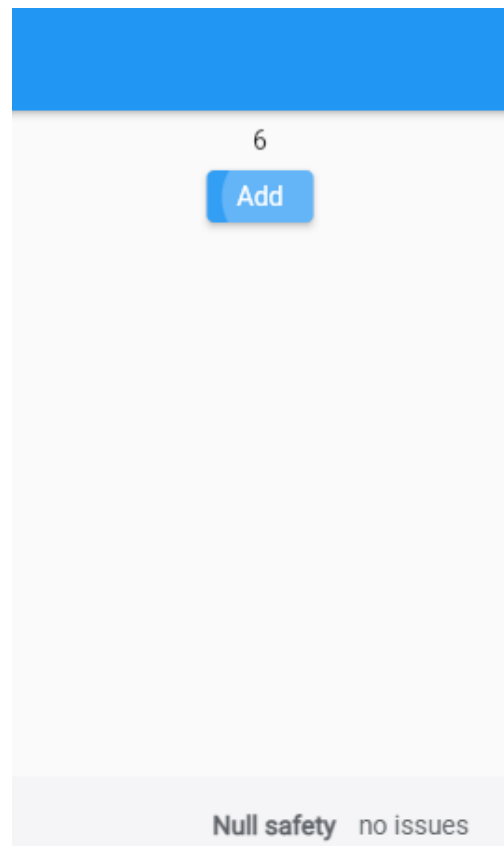


Abbildung 12: So sieht die Beispiel-Counter-App aus

Das Scaffold ist hier das Grundgerüst einer Standard-Material-App, bietet also eine AppBar (Titel-Bar) und einen Inhalt (Body) an. Das Center-Widget zentriert ihr Child, das Column-Widget ermöglicht mehrere Childs in vertikaler Richtung, das Padding-Widget fügt einen Abstand zwischen dem Widget und Child-Widget hinzu (auch «Padding» oder «Margin» genannt), der ElevatedButton stellt einen normalen Button dar und das Text-Widget stellt einen einfachen Text dar.

Flutter bietet sehr viele verschiedene Widgets an. Eine Übersicht ist im Flutter Widget catalog ersichtlich [6].

Das Widget «Counter» hat hier folgenden Widget-Tree:

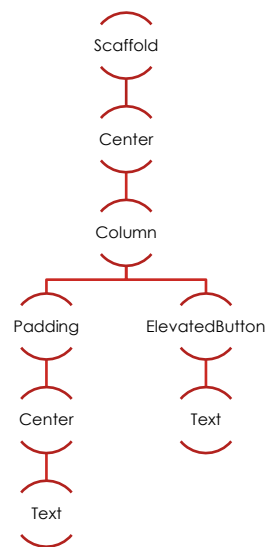


Abbildung 13: Widget-Tree der Beispiel-Applikation

3.2.4 Game States (DB)

Die «Game States» sollen eine Datenbank sein, die den aktuellen Spielstatus festhält, damit das projizierte Game und die App den gleichen Spielstatus haben.

3.2.5 Authentication

Alle Daten von einem bestimmten Spiel soll nur von Berechtigten (also Mitspieler oder dem Computer für den Spieltisch) gelesen oder bearbeitet werden können. Hierfür wird ein Token-basierter Software-as-a-Service-Dienst verwendet.

3.2.6 Game Services

Die Game Services beinhalten Dienste, die für die Spiel relevant sind, und nicht bereits durch das Unity-Game, die Game States (DB) oder den Authentication-Service abgedeckt sind. Möglicherweise wird dieser Service nicht benötigt.

3.2.7 Software

Bei den einzelnen Komponenten kommt folgende Software zum Einsatz:

KOMPONENTE	HARDWARE	SOFTWARE
GAME	Personal-Computer	Unity und C#
FINGER-TRACKING-SERVICE	Personal-Computer	Machine-Learning-Lösung MediaPipe Python
APP	Smartphone/Tablet	Flutter und Dart
GAME SERVICES	Cloud	NodeJS mit TypeScript
GAME STATES	Cloud (DB)	NoSQL Firebase Realtime DB (dokument-orientiert)
AUTHENTICATION	Cloud	Firebase Authentication (SaaS-Lösung)

4 Anforderungen

4.1 Systemumfang

4.1.1 Muss-Kriterien

- Das Spiel wird mit Beamer auf Tisch projizieren
- Der Spielstand wird via einen Server aktuell gehalten, damit die App den aktuellen Spielstand anzeigt.
- In der App müssen die Handkarten angezeigt werden und spielbar sein.
- Alle Aktionen, die mit Hand-Tracking möglich sein sollen, müssen auch mit der App ausführbar sein (z.B., wenn die Kamera nicht ordnungsgemäss funktioniert).
- Der verwendete Server gibt nur auf authentifizierten Anfragen Informationen zum Spiel heraus (anonyme Authentifizierung kann reichen).
- Ein minimales Gameplay (wie in den Anforderungen definiert) muss spielbar sein.
- Die Sitzpositionen am Tisch müssen definiert werden können.

4.1.2 Soll-Kriterien

- Die Spieler, die physikalisch vor dem Tisch sitzen, sollen bestimmte Handlungen mittels Handtracking mit Kamera auf dem Tisch direkt ausführen können.
- Der Hand-Tracking-Service soll konfigurierbar sein (z.B. Bildausschnitt definieren, Streamen des Bildes an-/ausschalten)
- Die App soll Hilfestellungen zu jeder Karte zur Verfügung stellen.

4.1.3 Optionale Kriterien

- Damit die Spieler ganz einfach dem Spiel beitreten können, könnte ein QR-Code projiziert werden, der mit der App gescannt wird.
- Bild-Übertragung der Smartphone- oder Tablet-Frontkamera durch die App und evtl. sogar Projektion mit Beamer auf den Spieltisch, damit diejenigen am Spieltisch die abwesenden sehen können und umgekehrt (nicht geplant im Rahmen der Bachelor-Thesis)
- Tutorial-Modus, damit die Spieler das Spiel einfacher lernen können.

4.2 Systemfunktionalität

Folgende Use Cases sollten vom System abgedeckt werden:

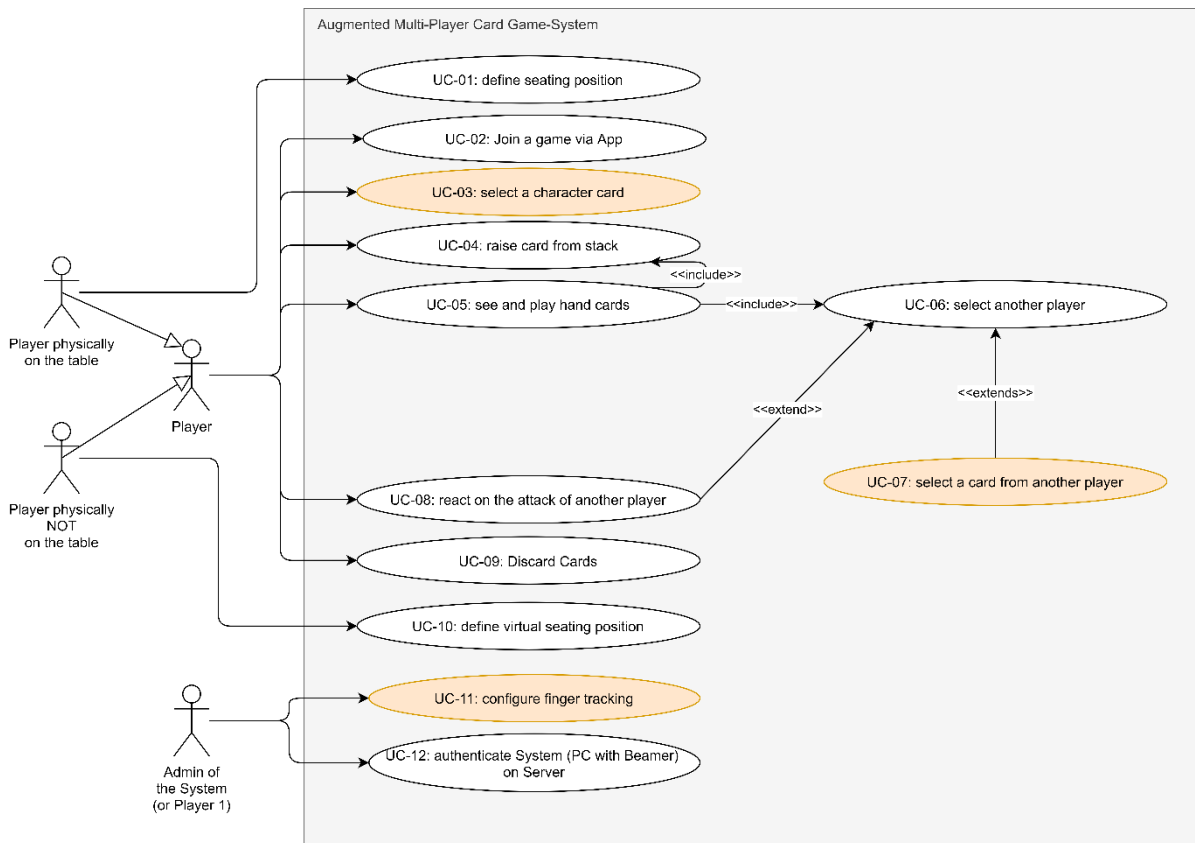


Abbildung 14: Anwendungsfälle des Systems

Die orangenen Use Cases sollen hierbei (teil-)optional sein. Folgendes sollen die einzelnen Use Cases beinhalten:

- UC-01: define seating position
Als Spieler, der physisch am Spieltisch sitzt, möchte ich dem System meine Sitzposition mitteilen können, damit das System die Informationen zu den Spielern am richtigen Platz anzeigen kann.
- UC-02: Join a game via App
Als Spieler möchte ich mich mit der dazugehörigen App mit dem Spiel verbinden können, damit ich meine Spielkarten auf dem Smartphone bzw. Tablet anschauen und spielen kann.
- UC-03: select a character card (optional)
Als Spieler möchte ich zum Spielbeginn meine Charakterkarte auswählen, damit ich eine bestimmte Zusatzfunktion oder eine bestimmte maximale Anzahl Lebenspunkte während dem Spiel habe.
- UC-04: raise card from stack
Als Spieler möchte ich am Anfang meines Spielzuges zwei Karten abheben können, damit ich Karten spielen kann.
- UC-05: see and play hand cards

Als Spieler möchte ich meine Handkarten nur in der App sehen, damit die anderen Spieler nicht wissen, welche Karte ich habe. Natürlich möchte ich diese Karten auch spielen können, um Einfluss auf das Spielgeschehen nehmen zu können.

- UC-06: select another player

Als Spieler, der einen anderen Spieler angreifen möchte, möchte ich den anderen Spieler auswählen können, den ich angreife.

- UC-07: select a card from another player (optional)

Als Spieler, der einen «Cat Balou» oder eine «Panik»-Karte gespielt hat, möchte ich eine bestimmte Karte von einem anderen Spieler auswählen können, damit ich diese eliminieren oder aufnehmen kann.

- UC-08: react on the attack of another player

Als Spieler, der Ziel eines «Bangs», «Indianers», «Duells» oder «Catlings» wurde, möchte diesem Angriff ausweichen können (z.B. mit «Bang» oder «Fehlschuss»), damit ich keinen Lebenspunkt verliere.

- UC-09: Discard Cards

Als Spieler möchte ich am Ende meines Spielzuges Handkarten abwerfen können, ohne dass diese einen Effekt haben, damit ich nicht mehr zu viele Handkarten habe und folglich meinen Spielzug beenden kann.

- UC-10: define virtual seating position

Als Spieler, der nicht physisch am Spieltisch sitzt, möchte ich die Gelegenheit haben, meine Sitzposition dennoch zu bestimmen, damit ich Einfluss auf meine direkten Sitznachbarn nehmen kann.

- UC-11: configure finger tracking (optional)

Als Administrator/Besitzer des Systems möchte ich Parameter für das Fingertracking setzen können, damit das Fingertracking am eigenen Spieltisch möglichst genau funktioniert.

- UC-12: authenticate System (PC with Beamer) on Server

Als Administrator möchte ich meinen PC, auf dem das projizierte Spiel läuft, sowie meine App-Instanz authentifizieren, damit nur ich mein System konfigurieren kann.

4.3 Akteure

Folgende Akteure sind für das Endprodukt relevant:

Stakeholder	Relevanz	Ziele und Interessen
Administrator/Besitzer des Systems	Hoch	Ein einfach konfigurierbares System, das bei Gebrauch verfügbar ist. Er/sie möchte mit Freunden und Bekannten das Kartenspiel «Bang» augmented spielen und Spass haben. Er möchte nicht viel Zeit mit Konfigurieren verbringen müssen.
Spieler am Tisch	Hoch	<p>Diese Spieler möchten schnell in das Spiel eintauchen können. Es wäre eine Zumutung, dass diese Spieler zuerst einen Benutzer-Account mit Verifizierung machen müssten oder vor dem Spiel Änderungen am Smartphone/Tablet (wie z.B. sich im gleichen WLAN befinden wie der Game-Computer) oder an der App machen müsste. Er/sie soll vor und während dem Spiel so wenig wie möglich falsch machen können.</p> <p>Möglicherweise kennt dieser Spieler das Spiel nicht. Deswegen soll das Spiel/System so selbsterklären wie möglich für ihn/sie sein.</p>
Spieler, der nicht physisch am Tisch sein kann	Mittel	<p>Gleiche Ziele wie der Spieler am Tisch.</p> <p>Im Gegensatz zu den anderen Spielern kann er/sie nicht direkt mit den anderen Spielern kommunizieren. Deswegen muss dieser Spieler so wenig Fehler machen können wie möglich.</p> <p>Möglicherweise (empfohlen) ist er/sie mit den anderen Spielern via Call (Skype, Discord, MS Teams, Zoom, etc.) verbunden. Vielleicht wünscht er sich, dass sich die Spieler gegenseitig sehen könnten. Aus diesem Grund kann es Sinn machen, das Webcam-Bild via App zu übertragen oder sogar sein eigenes Bild mit dem Beamer zu projizieren.</p>
Betreuer	Hoch	Fordert ein gut funktionierendes und dokumentiertes System.

4.4 Funktionale Anforderungen

4.4.1 Minimales Gameplay

4.4.1.1 Automatische Aktionen beim Spielstart

4.4.1.1.1 FA-01 – distribute roles

Ziel	Das System weist den Spielern eine der folgenden Rollen zu: Sherif, Hilfssherif, Bandit oder Outlaw. Alle Spieler sehen ihre Rolle auf ihrem Smartphone/Tablet.		
Use Case	(keiner, da dies automatisch passiert)		
Begründung	Im Spiel müssen alle Teilnehmenden eine Rolle haben, damit diese ihren Auftrag erfüllen können.		
Priorität	MUSS	Version	1.0

4.4.1.1.2 FA-02 show who's the sherif

Ziel	Auf dem projizierten Bild ist ersichtlich, welcher Spieler der Sherif ist.		
Use Case	(keiner, da dies automatisch passiert)		
Begründung	Die Rolle des Sherifs muss bekannt sein.		
Priorität	MUSS	Version	1.0

4.4.1.1.3 FA-04 receive life points in the beginning of the game

Ziel	Am Anfang des Spieles erhalten alle Spieler so viele Lebenspunkte, wie es ihre Zusatzrolle zulässt. Der Sherif erhält einen mehr. Die Anzahl Punkte werden auf dem projizierten Bild angezeigt.		
Use Case	UC-03: select a character card		
Begründung	Die Lebenspunkte sind ein zentrales Element des Spieles. Alle Spieler müssen sehen können, welcher Spieler wie viele Lebenspunkte noch besitzen.		
Priorität	MUSS	Version	1.0

4.4.1.2 Spielzug

4.4.1.2.1 FA-05 automatically raise cards in the beginning of the game

Ziel	Allen Spielern werden automatisch so viele Handkarten verteilt, wie Lebenspunkte sie besitzen.		
Use Case	(keiner, da dies automatisch passiert)		
Begründung	Vor dem ersten Spielzug müssen alle Spieler bereits Karten auf der Hand haben.		
Priorität	MUSS	Version	1.0

4.4.1.2.2 FA-06 raise cards in the beginning of the player's move

Ziel	Am Anfang jedes Spielzuges kann (und muss) der Spieler, der an der Reihe ist, 2 Karten aufziehen.		
Use Case	UC-04: raise card from stack		
Begründung	Vorgabe des Spieles.		
Priorität	MUSS	Version	1.0

4.4.1.2.3 FA-07 select and play hand cards

Ziel	Während des eigenen Spielzuges kann der Spieler auf seinem Gerät spielbare Karten auswählen und spielen.		
Use Case	UC-05: see and play hand cards		
Begründung	Alle Spieler müssen Aktionen ausführen können.		
Priorität	MUSS	Version	1.0

4.4.1.2.4 FA-08 discard hand cards

Ziel	Der Spieler, der an der Reihe ist, kann beliebig viele Karten auf seiner Hand wegwerfen.		
Use Case	UC-09: discard cards		
Begründung	Alle Spieler dürfen am Ende ihres Spielzuges nur so viele Karten auf der Hand haben, wie Lebenspunkte sie besitzen.		
Priorität	MUSS	Version	1.0

4.4.1.2.5 FA-09 max amount of hand cards

Ziel	Der Spieler, der an der Reihe ist, kann seinen Spielzug erst beenden, sobald er/sie maximal so viele Handkarten wie Lebenspunkte besitzt.		
Use Case	UC-09: discard cards		
Begründung	Alle Spieler dürfen am Ende ihres Spielzuges maximal so viele Handkarten besitzen, wie sie Lebenspunkte haben.		
Priorität	MUSS	Version	1.0

4.4.1.2.6 FA-13 automatically discard played cards

Ziel	Jede Karte, die gespielt wird, wird sichtbar auf den Stapel der gespielten Karte abgelegt. Diese Karte befindet sich ab diesem Zeitpunkt nicht mehr auf der Hand des Spielers.		
Use Case	(keiner, da dies automatisch passiert)		
Begründung	Gespielte Karten sollen nicht mehrmals von einem Spieler gespielt werden können.		
Priorität	MUSS	Version	1.0

4.4.1.3 Grundlegende Karten und Spiellogik

4.4.1.3.1 FA-10 play weapons

Ziel	Während des Spielzuges kann der Spieler Waffenkarten spielen. Diese Karte wird dann vor dem Spieler angezeigt. Die Waffenkarte erhöht die Reichweite des Spielers entsprechend.		
Use Case	UC-05: see and play hand cards		
Begründung	Waffen werden benötigt, um weiter entfernte Spieler erreichen zu können.		
Priorität	MUSS	Version	1.0

4.4.1.3.2 FA-11 play «Bang!»

Ziel	Spielt der Spieler eine «Bang!»-Karte, so startet ein Modus, in welchem er/sie einen anderen Spieler auswählt. Kann dieser ausgewählte Spieler nicht mit einem Fehlschuss den Angriff abwehren, verliert dieser einen Lebenspunkt.		
Use Cases	<ul style="list-style-type: none"> ▪ UC-05: see and play hand cards ▪ UC-06: select another player ▪ UC-08: react on the attack of another player 		
Begründung	Elementare Karte des Spieles.		
Priorität	MUSS	Version	1.0

4.4.1.3.3 FA-12 react with «Fehlschuss» on «Bang» or «Catling»

Ziel	Alle Spieler, auf welche mit einer «Bang»- oder «Catling»-Karte geschossen wurde, haben die Möglichkeit, diesen Schuss mit einer «Fehlschuss»-Karte abzuwehren. Hierfür wird ein Modus gestartet, in welchem der/die im Beschuss stehende(n) Spieler aufgefordert werden, entweder eine «Fehlschuss»-Karte zu spielen oder ein Lebenspunkt abzugeben.		
Use Case	UC-08: react on the attack of another player		
Begründung	Alle Spieler sollten die Möglichkeit haben, auf eine «Bang»- oder «Catling»-Karte zu reagieren		
Priorität	MUSS	Version	1.0

4.4.1.3.4 FA-14 die (Sterben)

Ziel	Sobald ein Spieler keine Lebenspunkte mehr besitzt, kann dieser nicht mehr mitspielen. Des Weiteren wird auf dem projizierten Bild sowie auf seinem Gerät angezeigt, dass dieser Spieler ausgeschieden ist.		
Use Case	(keiner, da dies automatisch passiert)		
Begründung	Vorgabe des Spiels.		
Priorität	MUSS	Version	1.0

4.4.1.3.5 FA-15 check if and who has won the game

Ziel	Immer, wenn ein Spieler im Spiel stirbt, wird evaluiert, ob das Spiel bereits beendet ist. Dies ist der Fall, wenn entweder <ul style="list-style-type: none"> • Der Sherif getötet wurde. Dann haben die Banditen gewonnen (und der Outlaw, falls er noch lebt.) • Oder alle Banditen und Outlaws getötet wurden. In diesem Fall haben der Sherif und seine Hilfssherifen gewonnen. Es wird angezeigt, wer gewonnen hat.		
Use Case	(keiner, da dies automatisch passiert)		
Begründung	Das Spiel muss ein Ende mit Gewinnern und Verlierern finden können.		
Priorität	MUSS	Version	1.0

4.4.2 Erweitertes Gameplay

4.4.2.1 Optional

4.4.2.1.1 FA-16 play «Been»

Ziel	Wird eine «Bienen»-Karte während des Spielzuges gespielt, so erhält der Spieler 1 Lebenspunkt. Besitzt der Spieler bereits die maximale Anzahl an Lebenspunkten, so hat diese Karte keine Funktion.		
Use Case	UC-05: see and play hand cards		
Begründung	Wichtige Karte des Spieles		
Priorität	OPTIONAL	Version	1.0

4.4.2.1.2 FA-17 play «Catling»

Ziel	Wird während des Spielzuges ein «Catling» gespielt, so wird automatisch auf alle Spieler geschossen. Alle Spieler haben die Gelegenheit, einen «Fehlschuss» zu spielen, ansonsten verlieren sie einen Lebenspunkt.		
Use Case	UC-05: see and play hand cards		
Begründung	Wichtige Karte des Spieles		
Priorität	OPTIONAL	Version	1.0

4.4.2.1.3 FA-18 play «Postkutsche» and raise two cards

Ziel	Wird eine «Postkutsche» während des Spielzuges gespielt, so kann der Spieler 2 Karten vom Stapel ziehen.		
Use Case	<ul style="list-style-type: none"> ▪ UC-05: see and play hand cards ▪ UC-04: raise card from stack 		
Begründung	Wichtige Karte des Spieles		
Priorität	OPTIONAL	Version	1.0

4.4.2.1.4 FA-19 play «Indianen»

Ziel	Wird die «Indianen»-Karte während des Spielzuges gespielt, so wird ein Modus gestartet, in welchem alle anderen Spieler eine «Bang»-Karte spielen können. Alle Spieler, die keine «Bang»-Karte spielen konnten, verlieren einen Lebenspunkt.		
Use Case	<ul style="list-style-type: none"> ▪ UC-05: see and play hand cards ▪ UC-08: react on the attack of another player 		
Begründung	Elementare Karte des Spieles.		
Priorität	OPTIONAL	Version	1.0

4.4.2.1.5 FA-20 play «Duell»

Ziel	Wird eine «Duell»-Karte während des Spielzuges gespielt, wird ein Modus gestartet, in welchem der aktuelle Spieler einen anderen Spieler auswählen kann. Dies wiederum führt zu einem Modus, in welchem der aktuelle und der ausgewählte Spieler abwechselungsweise eine «Bang»-Karte von der Hand spielen können. Der ausgewählte Spieler muss beginnen. Derjenige Spieler, der als erstes keine «Bang»-Karte mehr spielt, verliert einen Lebenspunkt.		
Use Cases	<ul style="list-style-type: none"> ▪ UC-05: see and play hand cards ▪ UC-06: select another player ▪ UC-08: react on the attack of another player 		

Begründung	Elementare Karte des Spieles.		
Priorität	OPTIONAL	Version	1.0

4.4.2.1.6 FA-03 distribute and select character card

Ziel	Jedem Spieler werden zwei zufällige Charakterkarten verteilt. Nur eine davon wird dem Spieler sichtbar gezeigt. Von beiden kann jeder Spieler eine davon auswählen. Die ausgewählte Charakterkarte wird bei jedem Spieler sichtbar für die anderen angezeigt.		
Use Case	UC-03: select a character card		
Begründung	Alle Spieler müssen eine Zusatzrolle haben, die bestimmt, wie viele Lebenspunkte sie maximal haben können.		
Priorität	OPTIONAL	Version	1.0

4.4.2.1.7 FA-21 Character card with specific function

Ziel	Die Funktionen der Charakterkarte werden angewendet.		
Use Case	UC-03: select a character card		
Begründung	Die Funktionen der Zusatzrollen bringen mehr Abwechslung ins Gameplay.		
Priorität	OPTIONAL	Version	1.0

4.4.3 Allgemeine Anforderungen an das System

4.4.3.1 Spiel-Teilnahme

4.4.3.1.1 FA-22 login/register playing table on the server

Ziel	Mit dem Computer, der das Spiel projizieren soll, kann sich der Besitzer beim Server anmelden und registrieren.		
Use Case	UC-12: authenticate System (PC with Beamer)		
Begründung	Nur angemeldete Personen sollen Zugriff auf die Daten des Systems haben.		
Priorität	MUSS	Version	1.0

4.4.3.1.2 FA-23 join game via app

Ziel	Die Spieler können mittels der App einem Spiel beitreten. Hierfür soll der Spieler in der App einen Code eingeben/einscannen, damit er sich automatisch direkt mit dem richtigen Spieltisch verbinden kann.		
Use Case	UC-02: Join a game via App		
Begründung	Die Spieler müssen sich via Smartphone oder Tablet mit dem System verbinden können.		
Priorität	MUSS	Version	1.0

4.4.3.1.3 FA-24 enter user nickname

Ziel	In der App soll es eine Möglichkeit geben, dass jeder Spieler einen Nickname eingeben kann, damit dieser z.B. auf dem Spieltisch auch angezeigt werden kann.		
Use Case	(noch nicht vorhanden)		
Begründung	Es kann praktisch sein, wenn man als Spieler auf dem Tablet oder Tisch sieht, wer an einer bestimmten Position sitzt.		
Priorität	SOLL	Version	1.0

4.4.3.1.4 FA-25 select camera for finger tracking

Ziel	Wenn Fingertracking eingesetzt wird, soll der Admin/Spieler 1 die Möglichkeit haben, die richtige Kamera hierfür einzusetzen.		
Use Case	UC-11: configure finger tracking		
Begründung	Die meisten Computer haben mehrere Kameras. Deswegen soll es möglich sein, die richtige Kamera auszuwählen.		
Priorität	SOLL	Version	1.0

4.4.3.1.5 FA-27 define seating position

Ziel	In der App müssen alle Spieler die Möglichkeit haben, ihre Sitzposition zu bestimmen.		
Use Case	<ul style="list-style-type: none"> ▪ UC-01: define seating position ▪ UC-10: define virtual seating position 		
Begründung	Damit die Karten beim am richtigen Ort projiziert und die Finger der richtigen Person zugeordnet werden können, muss das System wissen, wo sich welche Person befindet.		
Priorität	MUSS	Version	1.0

4.4.3.2 Optional

4.4.3.2.1 FA-26 crop the playing field for more accurate finger tracking position

Ziel	Wenn Fingertracking eingesetzt wird, soll der Admin/Spieler 1 die Möglichkeit haben, mit der App anzugeben, wo auf der Kamera sich das projizierte Spielfeld befindet. Hiermit soll die Genauigkeit der errechneten Finger-Positionen verbessert werden.		
Use Case	UC-11: configure finger tracking		
Begründung	Meistens ist die Kamera nicht 100%ig senkrecht auf den Spieltisch gerichtet. U.a. deswegen kommt es zu kleinen Verzerrungen. Diese sollen hiermit ein bisschen korrigiert werden.		
Priorität	OPTIONAL	Version	1.0

4.4.3.2.2 FA-28 show hand position on table (basic finger tracking)

Ziel	Auf dem Spieltisch soll auf Wunsch angezeigt werden können, wo auf dem Tisch die Hände erkannt wurden.		
Use Case	Erweitert folgende Use Cases: <ul style="list-style-type: none"> ▪ UC-04: raise card from stack ▪ UC-05: see and play hand cards ▪ UC-06: select another player ▪ UC-07: select a card from another player Ist notwendig für: <ul style="list-style-type: none"> ▪ UC-11: configure finger tracking 		
Begründung	Während des Entwickelns kann es praktisch sein zu sehen, wo Finger erkannt wurden.		
Priorität	OPTIONAL	Version	1.0

4.5 Nicht funktionale Anforderungen

4.5.1 Hardware

Grundsätzlich läuft das System auf drei verschiedenen Computer-Typen:

- Die Spiel-Logik ('Game') sowie der Hand-Tracking-Service soll auf einem normalen Personal-Computer/Laptop (Windows 11) laufen. Für das Handtracking ist eine (USB-)Kamera mit dem Computer verbunden. Das Game (bzw. das Spielfeld) wird auf dem Computer gerendert und mittels angeschlossenen Beamer auf einen Tisch projiziert.
- Die Smartphone- und Tablet-App, die unter anderem die Handkarten den Spielern anzeigt, soll auf Windows-Tablets und Android-Smartphones und -Tablets laufen. Um einen QR-Code scannen zu können, wird von den Geräten eine festverbaute und von Android erkannte Kamera vorausgesetzt. Für das Windows-Tablet wird dies nicht nötig sein, weil dort wahrscheinlich auf diese Funktion verzichtet wird (QR-Bibliotheken für Flutter unterstützen nur Android und iOS).
- Die Server werden voraussichtlich in einer Cloud gehostet.

4.5.2 Serverseitige Daten

Folgende Daten werden von den Benutzern langfristig auf einem Server gespeichert:

ID	Bezeichnung	Dauer	Konkrete Daten
DA-01	Login-Informationen	langfristig	<ul style="list-style-type: none"> ▪ Benutzername/-ID ▪ Passwort (verschlüsselt) oder Token

Kurzfristig – also während des Spiels – müssen sicher folgende Daten auf einem Server zwischengespeichert werden:

ID	Bezeichnung	Dauer	Konkrete Daten
DA-02	Spieler in einem Spiel	kurzfristig	<ul style="list-style-type: none"> ▪ ID des Benutzers ▪ Nickname ▪ Authentifizierungs-Token
DA-03	Karten und Spielinformationen	kurzfristig	<ul style="list-style-type: none"> ▪ Alle Rollen aller Spieler ▪ Karten jedes Spielers auf der Hand und vor sich ▪ Letzte Karte(n) auf dem Wegwerf-Stapel ▪ Lebenspunkte aller Spieler ▪ Vor die Spieler gespielten Karten
DA-04	Aktueller Spielmodus	kurzfristig	<ul style="list-style-type: none"> ▪ Welche Spielphase gerade läuft (also z.B. Anmeldephase für Smartphone, Rollenverteilung oder reguläres rundenbasiertes Spiel) ▪ Welcher Spieler an der Reihe ist ▪ Ob gerade ein spezieller Modus (wie «Duell» oder «Indianen») gestartet wurde und wer bereits wie reagiert hat. ▪ Welche(r) Spieler involviert sind

DA-05	Kamera-Bild	kurzfristig	<ul style="list-style-type: none"> ▪ Bild der Kamera ▪ Ob das Bild gestreamt wird
DA-06	Erkannte Finger	kurzfristig	<ul style="list-style-type: none"> ▪ Erkannte Finger und deren Koordinaten
DA-07	Einstellungen	Kurzfristig	<ul style="list-style-type: none"> ▪ Beim PC vorhandene Kameras ▪ Die Kamera, die gerade verwendet wird ▪ Evtl. die Koordinaten, wo auf dem Kamerabild sich die Spielfläche befindet

Die Daten werden voraussichtlich in einem JSON-ähnlichen Format in einer Firebase-Realtime-Db gespeichert. Die Datenstruktur könnte diese Form annehmen:

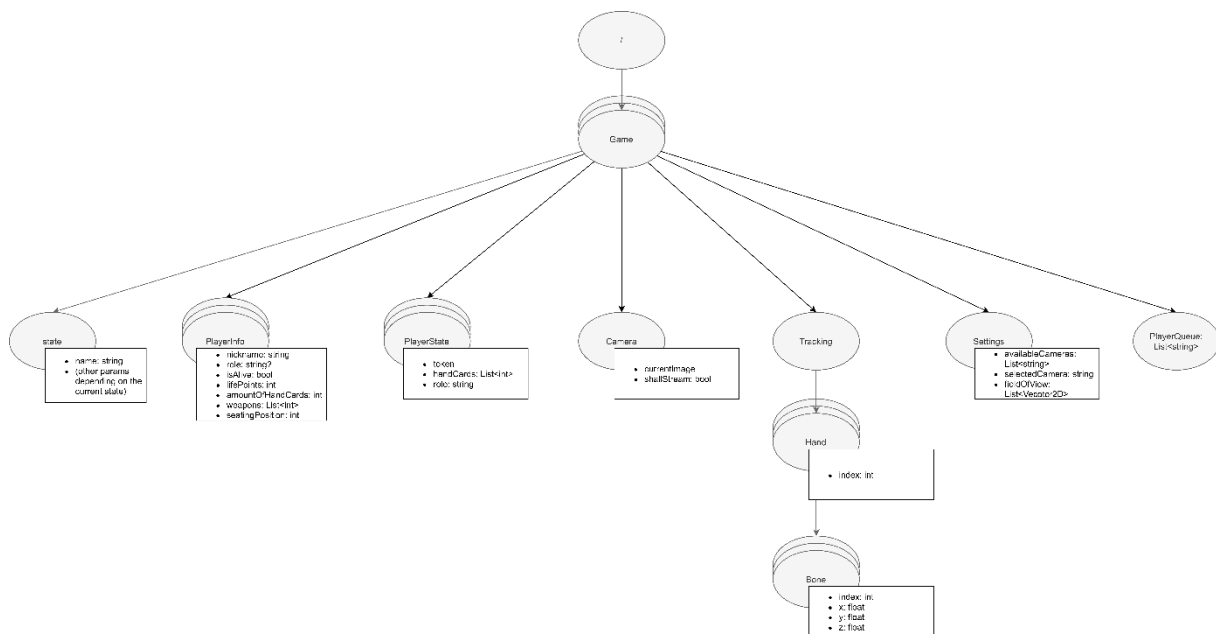


Abbildung 15: voraussichtliche Datenstruktur in der Datenbank

Ein Oval soll ein Objekt darstellen. Sind zwei Ovale unter diesem abgebildet, so ist es eine Liste von Objekten. Die Rechtecke beschreiben, welche Felder direkt auf dem Objekt enthalten sind.

Hier wurde eine Unterscheidung zwischen *PlayerInfo* und *PlayerState* gemacht. *PlayerState* soll Daten enthalten, die für einen bestimmten Spieler, aber nicht für andere sichtbar sein soll. *PlayerInfo* enthält Informationen, welche andere Spieler/Apps lesen dürfen. Das *state*-Objekt soll die Information enthalten, in welchem Modus das Spiel gerade ist, wie z.B. in einem «Duell»- oder «Bang»-Modus. Dieses Objekt kann seine Struktur je nach Modus anpassen, das Feld «name» soll aber immer vorhanden sein. Die *PlayerQueue* soll eine Art Warteschlange sein, damit sich Apps mit dem Spiel verbinden können, wobei sich das Game bemüht, diese Liste zu lesen und Apps in das Spiel einzulassen und sie auf die richtigen Objekte zu berechtigen.

Login-Daten werden in einer anderen Datenbank gespeichert, die von Firebase Authentication verwaltet wird.

4.6 Spielfluss

4.6.1 Fluss zwischen den einzelnen Screens (generell)

Bevor das eigentliche Spiel starten kann, kommen auf dem PC ungefähr diese Screens vor:

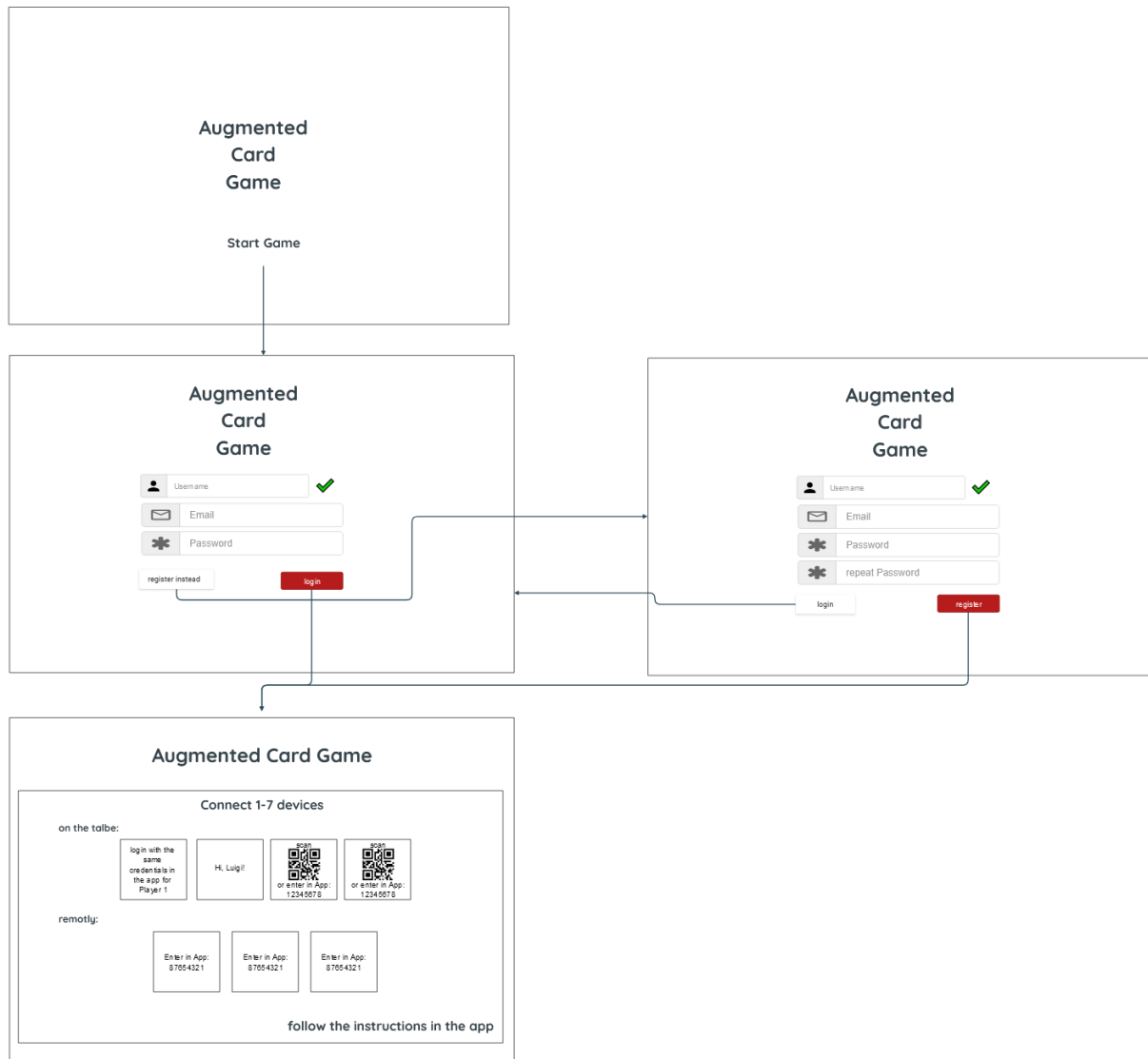


Abbildung 16: Screens des PCs vor dem Spielstart

Im folgenden Flussdiagramm sind die notwendigen Screens des Games (ohne App) als abgerundetes Viereck ersichtlich:

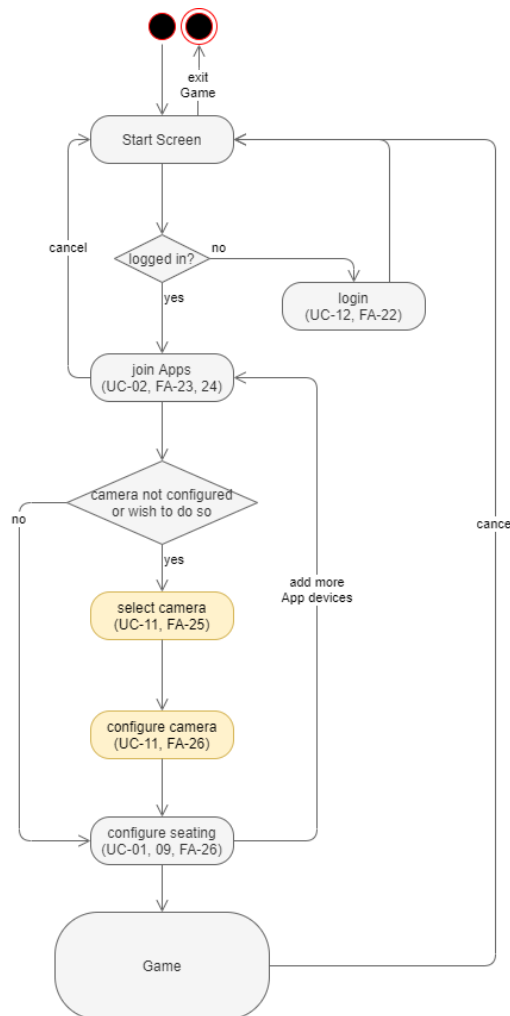


Abbildung 17: Fluss der einzelnen Screens (ohne spezifisches Spiel)

Das Spiel (welches auf dem Computer läuft, welcher mit dem Beamer verbunden ist) startet immer im Start Screen. Sollte der Computer nicht beim Server eingeloggt sein, so erscheint eine Aufforderung zum Einloggen oder Registrieren. Der Benutzer loggt sich hierbei noch gewöhnlich mit Maus und Tastatur ein.

Anschliessend erscheint ein Screen, der es erlaubt, Smartphones und Tablets via App diesem Spieltisch anzuschliessen. Dies geschieht, indem in der App ein Code eingegeben/eingescannt wird, der vom Computer stammt. Die erste App ist der Admin. In der App kann dann z.B. der eigene Nickname angegeben werden.

Wenn der Admin via App mit dem Game verbunden ist, kann er in der App ein Setup durchführen, um die Kamera zuerst auszuwählen und anschliessend zu konfigurieren. Mit Konfigurieren ist gemeint, dass er einstellen kann, welcher Ausschnitt des Kamerabildes der Spielfläche entspricht.

Anschliessend könnte das Spiel gestartet werden. Sind noch zu wenige Spieler anwesend, müssen sich noch weitere anmelden mit der App. Wenn alle Spieler in der App bekannt gegeben haben, dass sie bereit sind, können sie in der App angeben, wo sie sitzen und dann wird das Spiel gestartet.

AUGMENTED MULTI-PLAYER CARD GAME

Die grafische Umsetzung in der App könnte so aussehen:

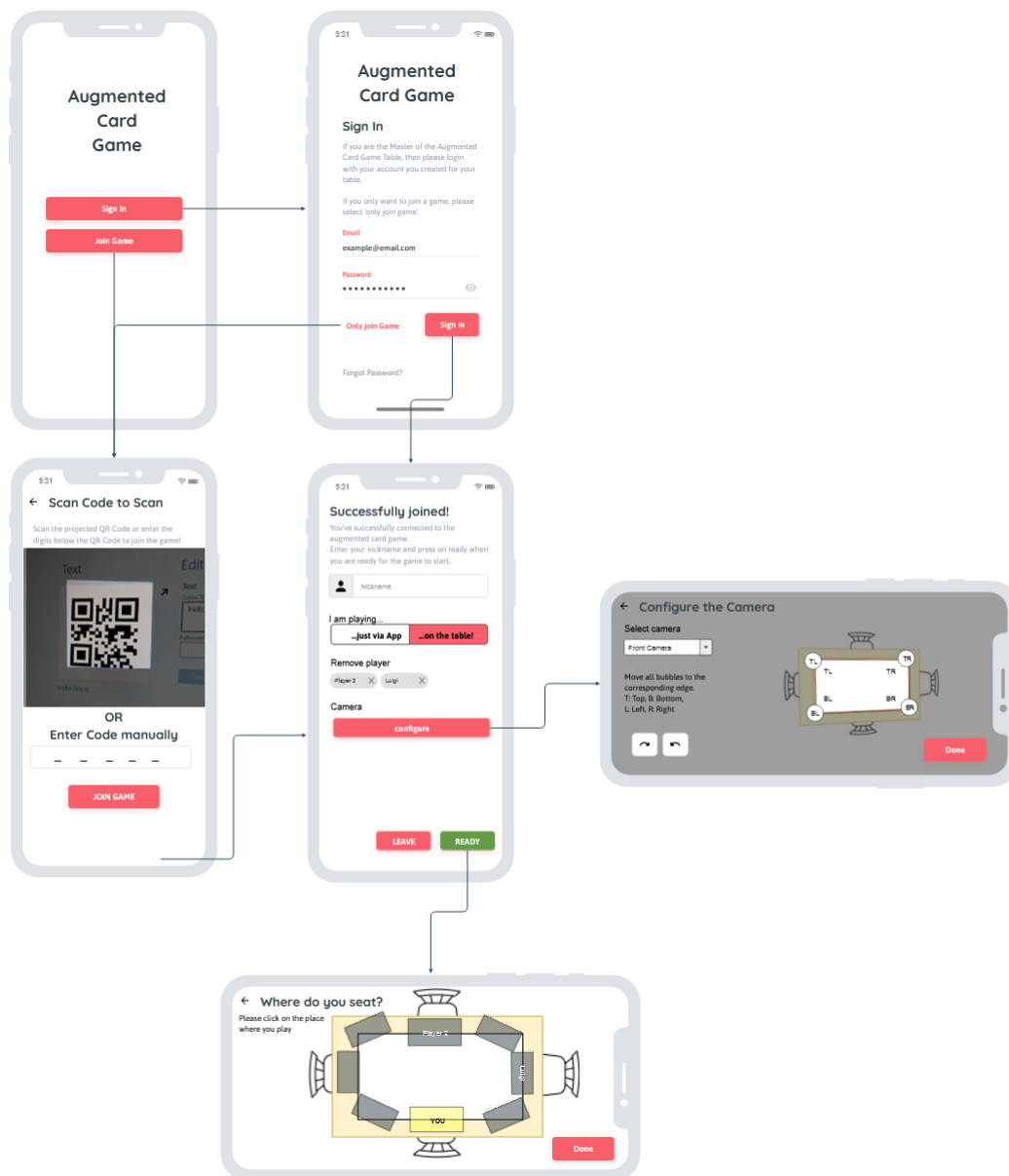


Abbildung 18: Screens vor dem Spielstart in der App

4.6.2 Fluss zwischen den einzelnen Screens während des Spieles

Nachdem das Game gestartet wurde, wird folgender Spielfluss durchlaufen. Die einzelnen abgerundeten Rechtecke entsprechen verschiedene Screens am Computer (also auf dem Spieltisch):

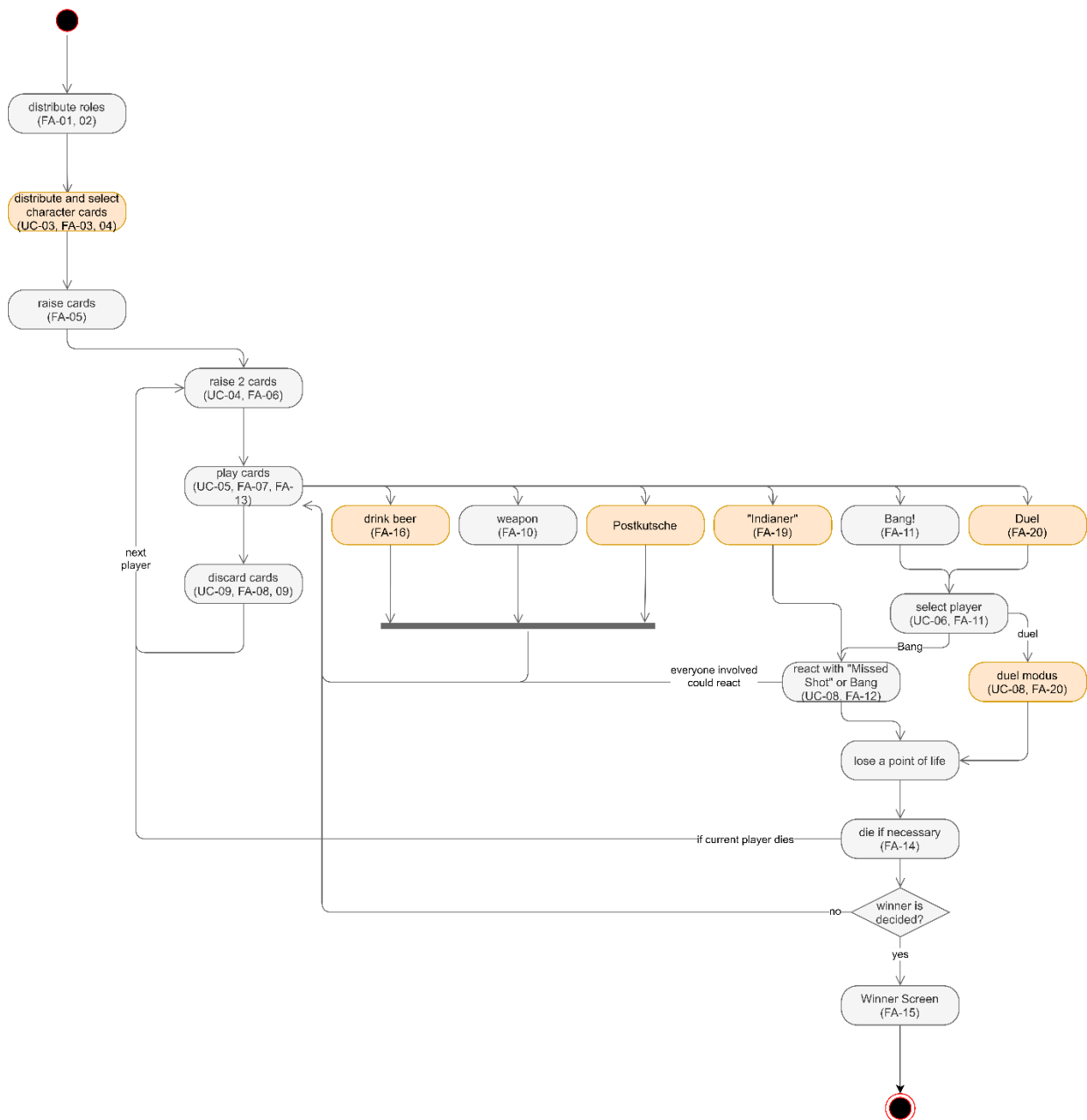


Abbildung 19: Die notwendigen Screens für die minimale Version des Bang!-Spieles

1. Zuerst werden die Rollen verteilt und in der App angezeigt. Der Sheriff wird offen auf dem Spieltisch angezeigt.
2. Anschliessend werden allen Spielern zwei zufällige Charakterkarten ausgespielt, wobei nur die erste sichtbar ist. Alle Spieler können in der App auswählen, ob sie die aufgedeckte Charakterkarte haben möchten oder die andere. Möchte der Spieler die zweite, so wird diese aufgedeckt und die nicht gewollte Karte wird weggeworfen. Die gewählte Charakterkarte ist für alle sichtbar. Entsprechend den Anzahl Maximal-Lebenspunkte werden die Lebenspunkte verteilt, wobei der Sheriff ein Maximal-Lebenspunkt mehr erhält.

3. Automatisch werden allen Spielern so viele Handkarten verteilt, wie sie Lebenspunkte besitzen. Die Lebenspunkte werden bei allen Spielern angezeigt auf dem Spieltisch.
4. Anschliessend beginnt der Spieler, der an der Reihe ist, seinen Spielzug, indem er/sie zwei Karten vom Stapel zieht. Dies geschieht, in dem er/sie auf den Stapel klickt (in der App) oder mit der Hand den Stapel berührt (auf dem Spieltisch).
5. Anschliessend kann der Spieler in der App eine Karte auswählen, die er/sie spielen möchte. Diese gespielte Karte wird auch kurz auf dem Spieltisch angezeigt. Danach wird die aktuelle Karte ausgeführt und evtl. kartenspezifische Screens angezeigt.
 - a. Bei einem «Bang» oder «Duell» wird zuerst ein Screen angezeigt, der es erlaubt, einen anderen Spieler auszuwählen. In der App kann man einen Spieler anklicken. Auf dem Spieltisch genügt es, wenn man die Hand genügend lange vor einem Spieler hält. Visuell soll dann ein Feld vor dem ausgewählten Spieler aufleuchten, um zu zeigen, dass gerade dieser Spieler ausgewählt wird. Diese Aktion soll in der App bestätigt oder abgelehnt werden können.
 - b. Wurde eine «Duell»-Karte gespielt, so wird ein entsprechender Screen angezeigt, der abwechselungsweise den angegriffenen Spieler und den Spieler, der an der Reihe ist, auffordert, eine «Bang»-Karte zu spielen. Auf dem Tablet sind bei beiden Spielern die Handkarten zu sehen, wobei alle nicht «Bang»-Karten deaktiviert sind. Diese Spieler können dann entweder eine «Bang»-Karte spielen oder «aufgeben». Weil es in diesem Modus einen Verlierer geben muss, wird ein Spieler einen Lebenspunkt verlieren. Hierfür wird zu einem entsprechenden Screen gewechselt, in der das angezeigt wird.
 - c. Bei einem «Bang» (nach Spielerauswahl) oder «Indianen» wird ein Screen gezeigt, der Anzeigt, welcher Angriff ausgeführt wurde. In der App werden bei allen beteiligten Spielern alle nicht «Bang»- bzw. «Fehlschuss»-Karten deaktiviert. Wie beim «Duell» gibt es auch die Möglichkeit zu passen in der App. Bei denjenigen Spielern, die getroffen wurden, wird ein Lebenspunktabzug angezeigt. Dann kann der Spieler, der an der Reihe ist, weitere Karten spielen.
6. Am Ende des Spielzuges darf der Spieler maximal so viele Karten auf der Hand haben, wie er/sie Lebenspunkte besitzt. Hierfür kann er/sie in der App Karten auswählen, die er/sie wegwerfen möchte. Die entsprechenden Karten landen auf dem Tisch auf den Wegwerf-Stapel.
7. Wird ein Spieler eliminiert, so wird rasch ein Screen mit einer entsprechenden Meldung angezeigt und anschliessen steht beim eliminierten Spieler ein entsprechender Text.
8. Sobald das Spiel entschieden ist, wird zu einem entsprechenden Screen gewechselt. In diesem Screen wird bekanntgegeben, wer gewonnen hat. Die entsprechenden Spieler werden auf dem Spieltisch aufgeleuchtet. In der App erscheint auch eine entsprechende Meldung.

AUGMENTED MULTI-PLAYER CARD GAME

Nachfolgend ist abgebildet, was ungefähr projiziert und was in der App sichtbar sein soll. Dabei werden zuerst die Rollen verteilt, anschliessen beginnt der erste Spielzug mit 2 Karten aufnehmen, dann mit dem Spielen von Karten, wie z.B. eines «Bangs!», welches auf den letzten beiden Stufen angezeigt wird.

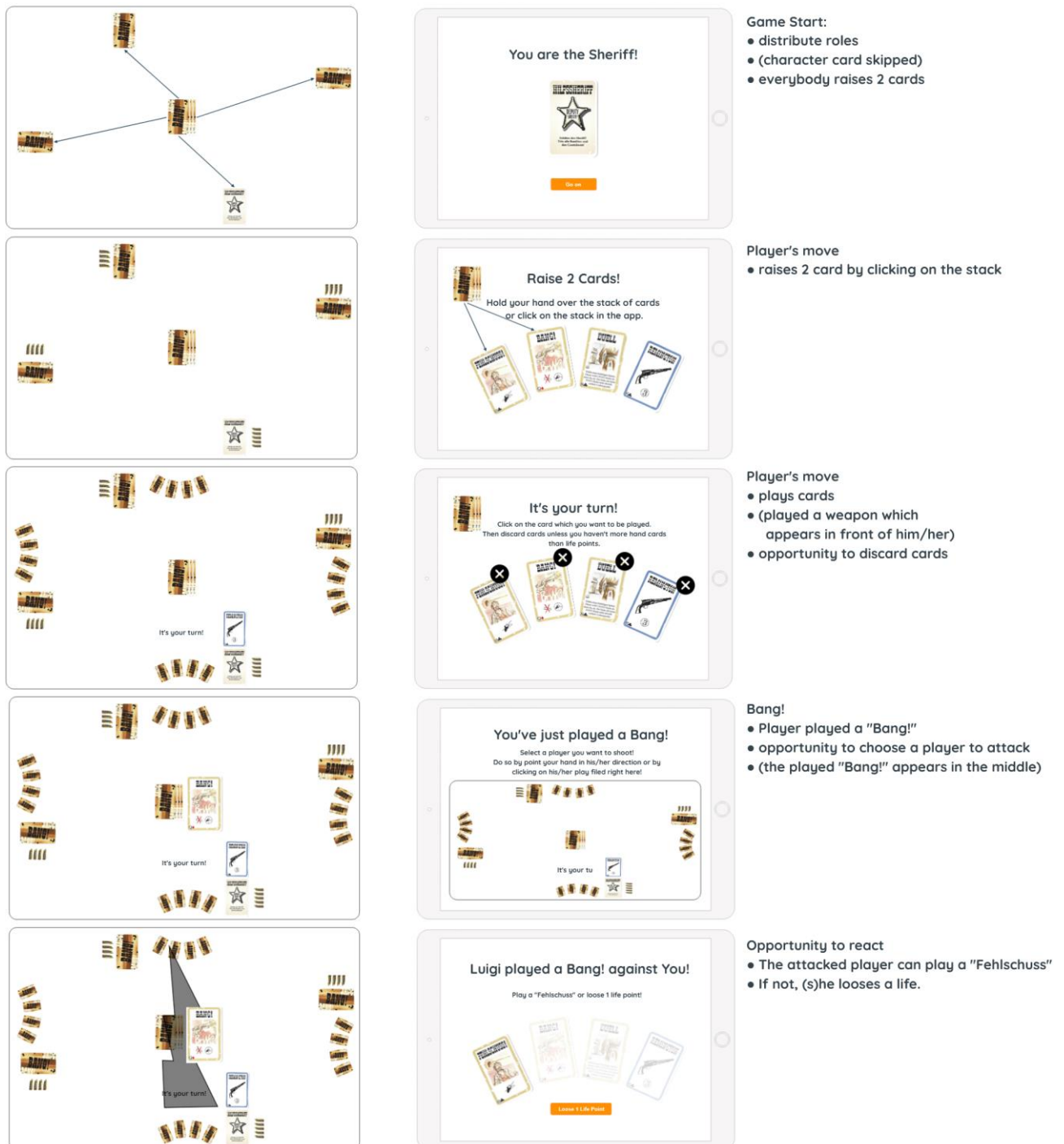


Abbildung 20: Mockups zum Spielfluss von Spielbeginn über eine «Bang!»-Angriffe

4.7 Einschränkungen

4.7.1 Abgrenzung

Diese Arbeit soll primär ein Prototyp für ein Augmented Card Game als Endprodukt liefern. Es soll lediglich das Spiel «Bang!» unterstützen, andere Spiele sind im Rahmen dieser Arbeit nicht geplant. Ausserdem soll das Spiel keine Spiel-Extensions unterstützen.

4.7.2 Einschränkungen

- Das Spiel wurde von daVinci Games und Abacusspiele herausgeben und ist urheberrechtlich geschützt. Für diese Arbeit sind keine Lizenzen ausgestellt worden. Aus diesem Grund darf diese Bachelor-Thesis nicht kommerziell verwendet werden.
- Aus Platzgründen und wegen dem Fingertracking können maximal 4 Personen physisch am Spieltisch spielen. Weitere Spieler können nicht am Spieltisch sitzen.
- Es können maximal 7 Spieler mitspielen (Limitation des Spieles «Bang!»). Im Rahmen dieser Arbeit wird das Spielen aber nur mit max. 4 Personen getestet.
- Die optionale QR-Code-Scannen-Funktion wird unter Windows wahrscheinlich nicht implementiert, weil für Flutter diese Funktion nur unter Android und iOS verfügbar ist.
- Die Login-Funktion ist lediglich dafür da, um die Daten von Unbefugten zu schützen. Bei der Login-Funktion ist weder eine «Passwort vergessen»- noch eine «Email-Adresse bestätigen»-Funktion geplant.

5 Abnahmetests

Die spezifischen Tests sind noch nicht Teil des Pflichtenheftes. Vor der Abgabe des Systems wird mindestens 1 Testfall pro Use Case definiert.

Die Tests sollen mindestens folgende Beschreibungen enthalten:

- Vor- und Nachbedingung
- Das erwartete Resultat und das zu vermeidende Resultat
- Ablauf des Tests

Die App soll auf Windows sowie auf einem Android-Smartphone und -Tablet getestet werden.

5.1 Test der nichtfunktionalen Anforderungen

Um herauszufinden, ob die App sowie der Spieltisch einfach zu bedienen ist, soll ein Usability-Test mit mindestens 2 Freiwilligen gemacht werden, um zu überprüfen, ob die Anwendungen selbsterklärend und einfach zu bedienen sind. Der genaue Testfall hierzu wird später definiert.

6 Projektmanagement

6.1 Projektorganisation

Für die Thesis kommt keine spezielle Projektmanagement-Methode wie Kanban oder Scrum zum Einsatz. Am Anfang der Thesis wird ein Pflichtenheft definiert und die entsprechenden Features werden dann Woche für Woche implementiert. In jeder Woche ist ein kurzes Austausch-Meeting zwischen dem Betreuer und dem Studierenden geplant, wo die aktuelle und folgende Arbeit besprochen wird.

6.2 Stakeholder und deren Aufgaben

Prof. Urs Künzler (Betreuer)

Er ist die Haupt-Ansprechperson für den Studierenden und bewertet die Bachelor-Thesis.


Dr. Federico Flueckiger (Experte)

Neben dem Betreuer bewertet auch der Experte die Endarbeit.

Samuel Grimm (Studierender)

Ist verantwortlich für das Projektmanagement der Thesis und für das Umsetzen des Projektes.

6.3 Gantt-Diagramm

Die Umsetzung der einzelnen Anforderungen ist zeitlich wie folgt geplant. Die Felder, welche ein «» enthalten, symbolisieren einen Meilenstein, der im nächsten Unterkapitel beschrieben wird.

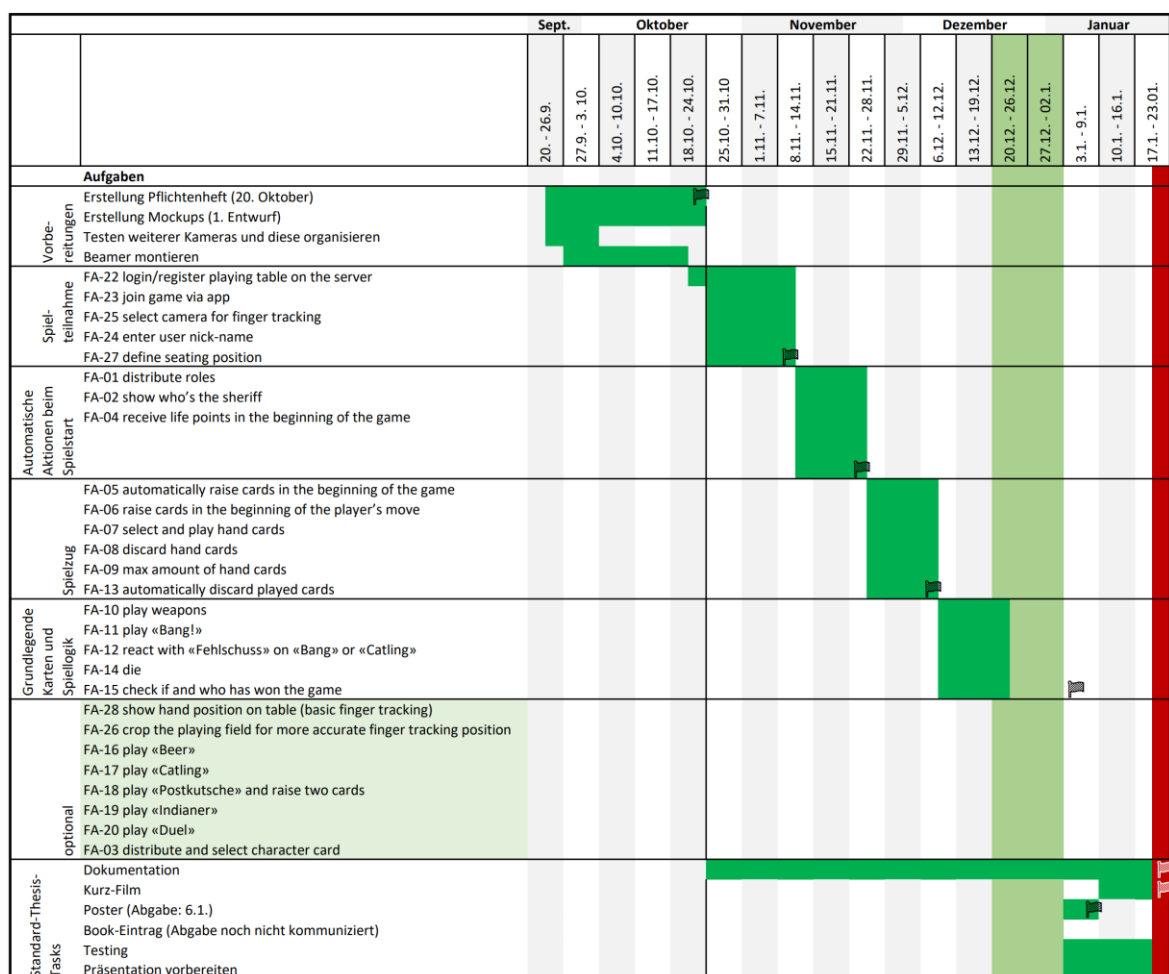


Abbildung 21: Zeitplan für die Thesis

6.4 Meilensteine

Die Meilensteine bestehen aus Paketen von Anforderungen. Folgende werden definiert:

NAME DES MEILENSTEINS	ERKLÄRUNG
SPIELTEILNAHME	Im Game für den Spieltisch kann sich der Benutzer anmelden und anschliessend können sich App-Benutzer damit verbinden. Das Game zeigt die verbundenen Benutzer an.
AUTOMATISCHE AKTIONEN BEIM SPIELSTART	Die automatischen Aktionen am Spielstart werden automatisch durchlaufen und angezeigt.
SPIELZUG	Die Spieler können normal einen Spielzug mit Karten legen bzw. ausführen. Die Karten haben aber noch keine Wirkung.
GRUNDLEGENDE KARTEN UND SPIELLOGIK	Die wichtigsten Karten aus dem Spiel können gespielt werden. Ausserdem ist es möglich zu sterben und zu gewinnen.

Tabelle 1: Meilensteine der Thesis

7 Literaturverzeichnis

- [1] Abacusspiele, „BANG! 4. Edition,“ Abacusspiele, [Online]. Available: <https://abacusspiele.de/produkt/bang-4-edition/>. [Zugriff am 19 10 2021].
- [2] Google LLC, „MediaPipe,“ Google LLC, 2020. [Online]. Available: <https://mediapipe.dev/>. [Zugriff am 13 10 2021].
- [3] Google, „Flutter,“ Google, [Online]. Available: <https://flutter.dev/>. [Zugriff am 17 10 2021].
- [4] Flutter, „Sound null safety,“ Google, 2020. [Online]. Available: <https://dart.dev/null-safety>. [Zugriff am 19 10 2021].
- [5] Flutter, „Hot Reload,“ Flutter, [Online]. Available: <https://flutter.dev/docs/development/tools/hot-reload>. [Zugriff am 19 10 2021].
- [6] Flutter, „Widget catalog,“ Google, [Online]. Available: <https://flutter.dev/docs/development/ui/widgets>. [Zugriff am 19 10 2021].

8 Abbildungsverzeichnis

Abbildung 1: Rollen des Spieles «Bang!».....	5
Abbildung 2: Charakterkarten, nicht vollständig.....	6
Abbildung 3: Waffenkarten.....	7
Abbildung 4: Die wichtigsten Aktionskarten des Spieles.....	7
Abbildung 5: Zusammenfassung der Symbole auf den Karten	8
Abbildung 6: Karten, mit welchen andere Lebenspunkte verlieren oder bekommen.....	8
Abbildung 7: Karten, die bewirken, dass Karten gezogen werden können.....	9
Abbildung 8: Karten, die eine Karte einem anderen Spieler entfernen.....	9
Abbildung 9: weitere blaue Karten, die vor Spieler gespielt werden.....	10
Abbildung 10: Hardware-Ansicht des Systems	11
Abbildung 11: Software-Architektur des Systems	12
Abbildung 12: So sieht die Beispiel-Counter-App aus.....	14
Abbildung 13: Widget-Tree der Beispiel-Applikation.....	15
Abbildung 14: Anwendungsfälle des Systems	17
Abbildung 15: voraussichtliche Datenstruktur in der Datenbank	27
Abbildung 16: Screens des PCs vor dem Spielstart	28
Abbildung 17: Fluss der einzelnen Screens (ohne spezifisches Spiel)	29
Abbildung 18: Screens vor dem Spielstart in der App.....	30
Abbildung 19: Die notwendigen Screens für die minimale Version des Bang!-Spieles.....	31
Abbildung 20: Mockups zum Spielfluss von Spielbeginn über eine «Bang!»-Attacke.....	33
Abbildung 21: Zeitplan für die Thesis	35