Schulprojekt Winter 2018

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur schulischen Projektarbeit

Entwicklung einer Android Applikation

Entwicklung des Spiels „Dino Jousting“ als Prototyp für zukünftige Projekte

Abgabedatum: 21.12.2018

**Schüler:**

Gar Georg <Georg.Gar@cerner.com>

Fedorak Wilhelm <Willi.Fedorak@cerner.com>

**Ausbildungsbetrieb:**

Cerner Health Services Deutschland GmbH

Hadersberg 1

84427 St. Wolfgang



Inhaltsverzeichnis

[1. Projektauftrag 1](#_Toc533070593)

[1.1. Projektumfeld 2](#_Toc533070594)

[1.2. Projektziel 2](#_Toc533070595)

[1.3. Projektabgrenzung 2](#_Toc533070596)

[2. Projektplanung 3](#_Toc533070597)

[2.1. Beschreibung des Ist-Zustandes 3](#_Toc533070598)

[2.2. Beschreibung des Soll-Konzepts 3](#_Toc533070599)

[2.3. Vorgehensmodel 4](#_Toc533070600)

[2.4. Zeitplanung 4](#_Toc533070601)

[2.4.1. 4](#_Toc533070602)

[3. Projektdurchführung 4](#_Toc533070603)

[3.1. Entwurfsphase 4](#_Toc533070604)

[3.1.1. Spielkonzept 5](#_Toc533070605)

[3.1.2. User Interface 5](#_Toc533070606)

[3.1.3. Datenbankmodell 6](#_Toc533070607)

[3.1.4. Client-Server-Architektur 6](#_Toc533070608)

[3.2. Implementierung 6](#_Toc533070609)

[3.2.1. Datenbank 6](#_Toc533070610)

[3.2.2. Clientseite 6](#_Toc533070611)

[3.2.3. 6](#_Toc533070612)

[3.2.4. Serverseite 6](#_Toc533070613)

[3.2.5. Soll-Ist-Vergleich 6](#_Toc533070614)

[4. Anhang 8](#_Toc533070615)

[4.1.1. 8](#_Toc533070616)

[4.1.2. 8](#_Toc533070617)

# Projektauftrag

## Projektumfeld

Im Rahmen eines Schulprojektes müssen die Auszubildenden der Berufsschule Freising in Zweiergruppen eine App bzw. einen Prototyp programmieren. Dabei dürfen sie selbst entscheiden, ob dies ein Spiel ist, oder nicht.

Für dieses Projekt haben die Schüler Georg Gar und Wilhelm Fedorak sich entschieden, den Prototypen für ein Spiel in Android Studio zu entwickeln. Dieser soll als Basis für zukünftige Entwicklung dienen.

Außerdem soll dadurch weiteres Wissen über die Entwicklung von Android Apps und deren Client-Server-Architektur erlangt werden. Das Projekt wird von Herrn Pfleiderer begleitet und bewertet.

In der [Tabelle 1](#_Tabelle_1) können genaue Details zu den genutzten Technologien und deren Versionen eingesehen werden.

## Projektziel

Das Ziel dieses Projektes ist es, ein funktionsfähiges Android-Spiel zu erstellen, welches von zwei Spielern an verschiedenen Orten gegeneinander gespielt werden kann. Außerdem soll die Möglichkeit geboten werden, das Spiel auch alleine gegen einen programmierten Gegner zu spielen. Dabei soll vor allem Erfahrung über das Programmieren im Bereich Android gesammelt werden. Das Spiel selbst dient dabei als Basis für zukünftige Ideen und hat daher den Fokus nicht direkt auf das Spielerlebnis. Die App soll einen Rahmen liefern, auf dessen Grundlage mit wenig Aufwand weitere Spielinhalte hinzugefügt werden können. Das Projekt soll alle Abläufe einer gebrauchsüblichen App enthalten. Dazu gehören User-Management mit einer Datenbank, ein autoritärer Game-Server, der den Hauptspielablauf berechnet und client-seitige Darstellung des Spielgeschehens.

## Projektabgrenzung

Der Zeitrahmen für das Projekt läuft vom 14. September 2018 bis zum 21. Dezember 2018.

Am Ende dieses Rahmens soll das Projekt und alle seine Ziele abgeschlossen und dokumentiert sein. Zur besseren Verfolgung des Projektzustandes sollen Milestones mit einem bedingten Abschlussdatum angelegt werden. Diese sind möglichst einzuhalten, um einen erfolgreichen Abschluss des Projektes zu gewährleisten. In [Tabelle 2](#_Tabelle_2) können alle Ziele und ihr geplantes Abschlussdatum eingesehen werden.

Dieses Projekt ist vorrangig ein technischer Prototyp und soll die Funktionsweisen einer Android-App aufweisen. Dabei wird nicht zu sehr auf das Game-Design geachtet, weshalb Spaß-Faktoren wie etwa Herausforderung und Variation des Spiel-Erlebnisses nicht im Vordergrund stehen. Diese Faktoren sollen in Zukunft jedoch ohne großen Aufwand implementierbar sein, weshalb ein Rahmen für leichte Erweiterbarkeit geschaffen werden soll.

# Projektplanung

## Beschreibung des Ist-Zustandes

Dieses Projekt ist der erste Prototyp zum Thema Android-Entwicklung. Es gibt daher keine Basis, auf der es aufbauen kann. Außerdem ist noch keine Erfahrung zu diesem Thema vorhanden. Es ist bereits Erfahrung in den Bereichen Datenbanken, Programmieren in Java und JavaScript vorhanden.

## Beschreibung des Soll-Konzepts

Das Spiel soll einen personalisierten Login und eine Registrierungsmöglichkeit für neue Benutzer bereitstellen. Dabei muss sichergestellt sein, dass ein Benutzer sich nicht mehrfach anmelden kann. Es muss die Möglichkeit bereitgestellt werden, alleine gegen den Computer oder gegen einen anderen Spieler zu spielen. Dabei sollen alle erforderlichen Daten wie Benutzernamen, Passwörter und Sieger-Statistiken in einer Datenbank abgespeichert werden. Auf diese soll über ein „Data Access Object“ (DAO) an nur einer Stelle im Code zugegriffen werden.

Es besteht die Möglichkeit, das Spiel so zu implementieren, dass beide Clients direkt miteinander über Peer-to-Peer kommunizieren und für das Abspeichern von Daten direkten Zugriff auf eine Datenbank haben. Eine weiter Möglichkeit ist es, jeden Client nur mit einem autoritären Server zu verbinden, welcher das Spielgeschehen kontrolliert und alle Datenbankaufrufe durchführt.

Erstere Möglichkeit bietet eine einfache Implementierung und benötigt keine Server-Architektur. Sie ist jedoch anfällig für Manipulation der Spieldaten. Außerdem ist es unumgänglich, dass hierbei die Datenbankzugriffe auf jedem Gerät vorhanden sind, wodurch ein Sicherheitsrisiko für die Datenbank besteht.

Die zweite Möglichkeit bietet mehr Sicherheit der Spieldaten wie auch der Datenbank, da die Clients über diese Daten keine Autorität haben. Sie erhalten lediglich das berechnete Spielgeschehen vom Server und schicken ihre Inputs zu diesem. Dafür muss jedoch ein eigener Web-Server implementiert werden, was mit zusätzlichem Aufwand verbunden ist und dadurch zu Zeitmangel führen könnte.

Da die erste Möglichkeit zu viele Sicherheitsrisiken birgt, wird für das Projekt ein autoritärer Server implementiert. Sowohl das Benutzermanagement als auch das Spielgeschehen werden auf diesem ablaufen.

Aufgrund der vorhandenen Erfahrung wurde entschieden, einen auf Node.js basierenden Web-Server mit einer MySQL-Datenbank zu verwenden.

## Vorgehensmodel

Für dieses Projekt ist das inkrementelle Vorgehensmodell vorgeschrieben. Dies hat die Strategie Systeme nach einem gewissen Ablaufplan iterativ zu integrieren. Dabei sollen agil Entscheidungen getroffen werden können. So können Erfahrungen, die während dem Entwicklungsprozess gemacht werden, direkt angewandt werden und Probleme flexibel gelöst werden. Die einzelnen Schritte des Ablaufs bestehen aus Implementieren, Testen, Abwägen der Ergebnisse und Planen der nächsten Schritte. Diese werden wiederholt, bis das gewünschte Ergebnis erreicht ist.

Dieses Modell bietet den Vorteil, dass Änderungen schnell getestet werden können und dadurch schnelle Entscheidungen bei der Entwicklung getroffen werden können. So ist es möglich, Technologien anzuwenden und, je nach Entscheidung zu nutzen oder zu verwerfen. Dadurch wird ein großer Überblick über die Android-Entwicklung geschaffen.

## Zeitplanung

Für das Projekt stehen 70 Stunden zur Verfügung. Diese wurden auf die verschiedenen Phasen der Projektentwicklung aufgeteilt. Die Aufteilung dieser Zeit kann [Tabelle 3](#_Tabelle_3) entnommen werden.

In der Entwurfsphase wird das Spielkonzept ausgearbeitet. Es werden die Haupt-Mechaniken diskutiert und die notwendigen und möglichen Technologien abgewogen.

Darauf folgt die Implementierungsphase, in der nach dem inkrementellen Modell versucht wird, den im Projektziel beschriebenen Zustand zu erreichen.

Letztendlich wird zum Projekt die Dokumentation angelegt.

|  |  |
| --- | --- |
| **Projektphase** | **Dauer** |
| Entwurfsphase | 8 h |
| Implementierungsphase | 52 h |
| Erstellen der Dokumentation | 10 h |
| **Gesamt** | **70 h** |

#### Tabelle 3

# Projektdurchführung

## Entwurfsphase

### Spielkonzept

Das Kernprinzip des Spieles ist an das mittelalterliche Tjosten angelehnt. Zwei Ritter stehen sich dabei auf einer Zwei-Dimensionalen Ebene gegenüber. Während sie sich aufeinander zubewegen müssen sie versuchen, ihre Lanze so anzuheben, dass sie den Gegner trifft. Sie beschleunigen während dem Vorgang auf die maximale Geschwindigkeit des gewählten Reittieres. Dieser Ablauf stellt eine Runde dar. Es wird so lang gespielt, bis die Lebenspunkte eines Spielers auf null sinken.

Der gesamte Ablauf wird in 3 Phasen unterteilt:

1. Start:

Beide Spieler stehen auf der x-Achse mit einem gewissen Abstand voneinander entfernt, dieser wird zufällig aus einem Spektrum gewählt. Ein Countdown läuft ab, nach dem die nächste Phase startet.

1. Bewegung:

Beide Spieler starten zeitgleich zu beschleunigen. Dabei hängt Maximalgeschwindigkeit und Beschleunigung vom gewählten Reittier ab. In dieser Phase können die Spieler ihre Lanze durch drücken des Bildschirms anheben. Wird dies nicht gemacht, fällt die Lanze langsam. Es soll versucht werden, den Gegner zu treffen.

1. Treffpunkt:

Die Spieler treffen sich. Abhängig von der Position der Lanze wird das Rundenergebnis berechnet. Dabei ist auch die Höhe der Reittiere der Spieler zu beachten. Es gibt drei Trefferzonen.

1. Kein Treffer – Die Lanze verfehlt und der Gegner verliert keine Lebenspunkte
2. Körpertreffer – Die Lanze trifft den Körper und zieht einen kleinen Teil der Lebenspunkte des Gegners ab
3. Kopftreffer – Die Lanze trifft den Kopf und zieht einen großen Teil der Lebenspunkte des Gegners ab

Wenn die Lebenspunkte eines Spielers auf null sinken, endet das Spiel und der Sieger bekommt einen Sieg eingetragen. Im Falle eines unentschieden wird beiden Spielern ein Sieg eingetragen.

Außerhalb des Spiels kann man sein Reittier wählen. Außerdem kann das Spiel leicht um die Möglichkeit erweitert werden, weitere Lanzen zur Auswahl zur Verfügung zu stellen.

### User Interface

Das Spiel enthält während der Kernphase eine „minimap“, auf der die Position beider Spieler angezeigt wird. Außerdem gibt es zwei Lebenspunkteanzeigen, welche den aktuellen Zustand beider Spieler anzeigen.

Bei der Auswahl eines Reittieres wird eine Vorschau sowie die Eigenschaften des Tieres angezeigt.

### Datenbankmodell

Da alle relevanten Daten in der Datenbank abgespeichert werden sollen, wurde ein Entity-Relationship-Modell entworfen. Dabei wurde darauf geachtet, dass Lanzen und Reittiere leicht erweiterbar sind. In [Diagramm 1](#_Diagramm_1) kann das Datenbankmodell eingesehen werden. Die Passwörter von Benutzern werden als Hash abgespeichert, um die Sicherheit der Passwörter zu gewährleisten.

### Client-Server-Architektur

Für das Projekt ist serverseitige Berechnungen und clientseitige Darstellung des Spieles geplant. Deshalb wurde ein Modell für die Kommunikation zwischen Server und Client erstellt, welches einen möglichen, jedoch nicht endgültigen Prozessablauf diesbezüglich darstellt. Dieses kann in [Diagramm 2](#_Diagramm_2) eingesehen werden. Dabei ist zu beachten, dass Inputs vom Client und Updates vom Server asynchron ablaufen, wobei sie bis zur Durchführung abgespeichert werden. Dies ermöglicht ein flüssiges Spielerlebnis ohne Pausen.

## Implementierung

Das Projekt wird in Client, Server und Datenbank aufgeteilt. Dabei werden auch verschiedene Sprachen und Technologien verwendet, auf die nun eingegangen wird.

### Datenbank

Auf Basis des in Kap. 4.1.3: [Datenbankmodell](#_Datenbankmodell) beschriebenen Entity-Relationship-Modell wurde eine MySQL-Datenbank erstellt und die nötigen Tabellen mit dem Command Line Tool angelegt.

Erwähnenswert ist hierbei die Spalte „token“ in der Tabelle „users“. Dieser ist der Authentifizierungstoken des Nutzers, dessen Funktion in Kap. 4.2.2: [Serverseite](#_Serverseite_1) erläutert wird.

Somit entspricht die Datenbank den in der Entwurfsphase definierten Anforderungen.

### Serverseite

### Clientseite

### Soll-Ist-Vergleich

#### Sollzustand

Das Spiel soll einen personalisierten Login und eine Registrierungsmöglichkeit für neue Benutzer bereitstellen. Dabei muss sichergestellt sein, dass ein Benutzer sich nicht mehrfach anmelden kann.

Sowohl das Benutzermanagement als auch die Spielmechaniken sollen bestenfalls über einen Web-Server implementiert werden.

Es muss die Möglichkeit bereitgestellt werden, alleine gegen den Computer oder gegen einen anderen Spieler zu spielen.

Dabei sollen alle erforderlichen Daten wie Benutzernamen, Passwörter und Sieger-Statistiken in einer Datenbank abgespeichert werden. Auf diese soll über ein „Data Access Object“ (DAO) an nur einer Stelle im Code zugegriffen werden.

#### Istzustand

Beim starten des Spieles werden Nutzerdaten über eine Login-Ansicht abgefragt, wenn noch keine Login-Daten abgespeichert sind. Es wird außerdem ein Button angeboten, der zu einer Registrations-Ansicht führt. Dort kann dann ein neuer Benutzer angelegt werden. Dafür sind ein nicht benutzter Name und ein mindestens vierstelliges Passwort nötig. Beim Anmelden wird dem Nutzer ein eindeutiger Token zugewiesen, den dieser für die Authentifizierung beim Server benutzt. Außerdem wird dieser abgespeichert, sodass kein erneuter Login notwendig ist. Beim Ausloggen wird der Token gelöscht.

Über diesen Token wird außerdem sichergestellt, dass jeder Nutzer nur einmal angemeldet sein kann. Loggt sich jemand anderes mit gleichen Daten ein, wird der Token des ersten Nutzers ungültig und der Nutzer damit abgemeldet.

Nach der Authentifizierung gelangt man in ein Hauptmenü, in dem der Nutzer ein Einzel- und Mehrspieler Spiel starten kann. Außerdem kann er dort seine Siege einsehen und sich ausloggen.

Für jede dieser Optionen erfolgt ein Datenaustausch mit einem Server. Dabei ist eine dauerhafte „Socket“-Verbindung für bidirektionale Kommunikation zuständig. Die Spieledaten werden auf dem Server verarbeitet, und auf dem Client dargestellt.

Für die Datenspeicherung wird eine „MySQL“-Datenbank genutzt, deren Steuerung über ein DAO auf dem Server erfolgt.

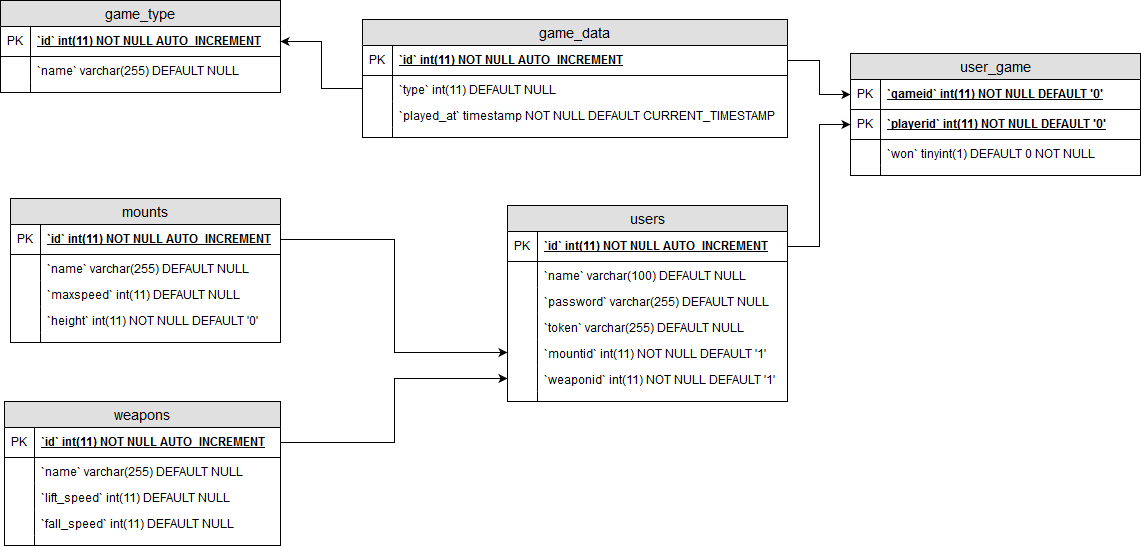
# Anhang

|  |  |
| --- | --- |
| **Technologie** | **Version** |
| Android Compiler (Client, IDE: Android Studio) | API 28: Android 9.0 (Pie) |
| Min. benötigte SDK-Version | API 25: Android 7.1.1 (Nougat) |
| Node.js (Server, IDE: Visual Studio Code) | 9.4.0 |
| Typescript (Server Programmiersprache) | 3.2.1 |
| HAXM (Hardware-Beschleunigung) | 7.3.2 |
|  |  |
|  |  |
|  |  |
|  |  |

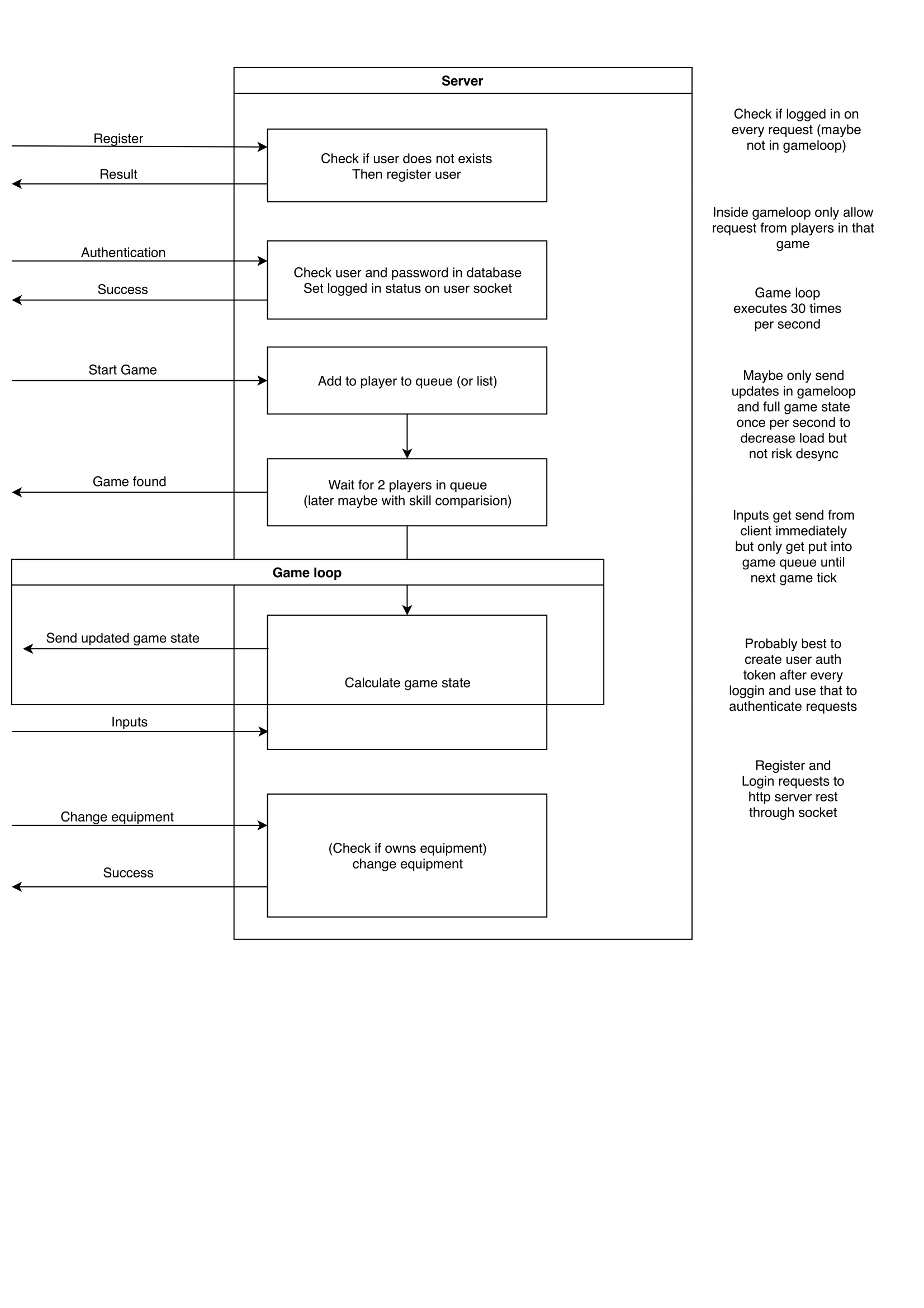
#### Tabelle 1

|  |  |  |
| --- | --- | --- |
| Fälligkeitsdatum | Was | Fertig |
| 14.09.2018 | Anlegen des Projektes | Fertig |
| 14.09.2018 | Lauffähige App am Handy starten | Fertig |
| 14.09.2018 | Einbinden des Projektes in einem GitHub Branch | Fertig |
| Gängiger Prozess, etwa bis 07.12.2018 | Starten des Entwicklungsprozesses, ausarbeiten des GDD | Fertig |
| 05.10.2018 | Darstellen von Elementen in der App | Fertig |
| 05.10.2018 | Server Aufsetzen, Branch dafür anlegen | Fertig |
| 19.10.2018 | User-Management über Server regeln | Fertig |
| 26.10.2018 | Umschreiben der Logik: Client -> Server | Fertig |
| 02.11.2018 | Gegnerlogik erstellen | Fertig |
| 09.11.2018 | Darstellung des Spiels am Klienten | Fertig |
| 16.11.2018 | Kollisionen | Fertig |
| 23.11.2018 | Kernelemente des Spiels ausarbeiten | Fertig |
| 30.11.2018 | Rahmen: Optionen, Equipment ausarbeiten | Fertig |
| 07.12.2018 | Refactoring des Codes | Fertig |
| 20.12.2018 | Dokumentation anlegen |  |

#### Tabelle 2



#### Diagramm 1



#### Diagramm 2