

Dokumentation der Projektarbeit zur Veranstaltung

**Web Engineering**

im

SS 2018

**The Challenger**

Vorgelegt von   
  
**Nils Nappenfeld 018100666**

**Leon Nappenfeld 018100667**

**Lisa-Marie Häfner**

**Marcel Brockskothen**

am   
  
30.06.2018

**Inhaltsverzeichnis**

[1 Aufgabenstellung 3](#_Toc491086878)

[Muss-Kriterien 3](#_Toc491086879)

[2 Eingesetzte Hard- und Software 3](#_Toc491086880)

[2.1 Softwareumgebung 3](#_Toc491086881)

[2.2 Hardware 3](#_Toc491086882)

[3 Problemlösung 4](#_Toc491086883)

[3.1 Umgebung des Systems (SystemKontext) 4](#_Toc491086884)

[3.2 Anforderungsanalyse 4](#_Toc491086885)

[3.2.1 Kartenspielbeschreibung 4](#_Toc491086886)

[3.2.2 Use-Case Analyse 5](#_Toc491086887)

[3.2.3 Infrastruktur 7](#_Toc491086888)

[3.3 Objektorientierte Analyse (OOA): 8](#_Toc491086889)

[3.3.1 Use-Case bezogene Klassendiagramme 8](#_Toc491086890)

[3.3.2 Use-Case bezogene Sequenzdiagramme 9](#_Toc491086891)

[3.3.3 Fachkonzeptklassendiagramm 10](#_Toc491086892)

[3.4 Objektorientiertes Design (OOD) 11](#_Toc491086893)

[3.4.1 Komponentendiagramm 11](#_Toc491086894)

[3.4.2 Zustandsdiagramm (Client) 12](#_Toc491086895)

[3.4.3 Zustandsdiagramm (Server) 13](#_Toc491086896)

[3.4.4 Aktivitätsdiagramm 15](#_Toc491086897)

[3.4.5 Verteilungsdiagramm 16](#_Toc491086898)

[3.4.6 Grundsatzentscheidungen 17](#_Toc491086899)

[3.4.7 Klassendiagramm 18](#_Toc491086900)

[3.5 Objektorientierte Programmierung (OOP) 19](#_Toc491086901)

[4 Tests, Ergebnisse und Diskussion 20](#_Toc491086902)

[4.1 Testfälle Spielablauf 20](#_Toc491086903)

[4.2 Testfälle Spielelogik 21](#_Toc491086904)

[5 Zusammenfassung 23](#_Toc491086905)

[Anhang 23](#_Toc491086906)

[A.1 Programmstart-Erklärung für den Anwender 23](#_Toc491086907)

[A.2 Dateien 23](#_Toc491086908)

[A.3 Quellcode 24](#_Toc491086909)

# Aufgabenstellung

Es soll eine Smartphone-Applikation (App) entwickelt werden, die dem Benutzer die Möglichkeit gibt sich mit anderen Benutzer in bestimmten Disziplinen zu messen. Die App soll ermitteln, wie hoch und wie weit der Anwender springen und um wie viel Grad sich der Anwender mit einem Sprung um die eigene Achse drehen kann.

## Muss-Kriterien

* Erfassung von definierten Sensordaten
* Eingabe der gewünschten Disziplin des Benutzers
* Position- und Lagebestimmung des Smartphones
* Ausgabe der vom Benutzer erreichten Leistungen

## Kann-Kriterien

* Counter-Funktion, welche zählt wie oft das Smartphone auf dem Boden gefallen ist
* Erstellung eines Benutzerprofils
* Festhaltung des Erfolgsfortschritts und der Rekorde

# Eingesetzte Hard- und Software

## Softwareumgebung

Betriebssystem: Microsoft Windows 7 und Microsoft Windows 10

Entwicklungsumgebung: Android Studio Version 3.2.1

Programmiersprache: Java Version 1.8.0\_40

Diagrammerstellung: UMLet Version 14.1.1

Projektdokumentation: Microsoft Word

## Hardware

Hardware: PC/Notebook

# Problemlösung

## Umgebung des Systems (SystemKontext)

Abb. 1: Systemkontextdiagramm

## Anforderungsanalyse

### *Anwendungsbeschreibung*

### *Use-Case Analyse*

**Name des AWF:** Karte setzen

**Akteure:** 4 verschiedene Spieler

**Vorbedingungen:** Es müssen 4 Spieler angemeldet sein und das System muss den Spielern jeweils 8 Karten ausgeteilt bzw. zugewiesen haben.

**Auslöser:** Das System fordert einen Spieler auf und sendet jedes Mal dem Spieler sein aktuelles Kartenblatt, welches in seiner Konsole erscheint!

Dieser Spieler hat entweder die letzte Runde gewonnen, hat zuvor ungültige Karten gelegt oder er startet das Spiel mit der Karo 7.

**Nachbedingungen:**

**Ergebnisse:** Die gelegte(n) Karte(n) ist/sind gültig und wird/werden bei den anderen Spielern auf dem Display angezeigt. Das System fordert dann den nächsten Spieler auf, seine Karten zu legen.

Die gelegte(n) Karte(n) ist/sind ungültig und das System fordert den Spieler erneut auf Karten zu setzen. Außerdem erscheint in seinem Terminal eine entsprechende Fehlermeldung.

**Invarianten:** Alle 4 Spieler müssen verbunden sein!

**Ablaufbeschreibung:**

1. Spieler 1 mit der Karo 7 muss beginnen.
   * Diese Karte muss gespielt werden, evtl. mit weiteren Siebenen, wenn möglich.
2. Spieler 2 kann mit derselben Anzahl Karten drüber gehen, allerdings müssen die Karten vom Zahlenwert höher sein als die des ersten Spielers. Wenn Spieler 2 nicht drüber kann/will, hat er die Möglichkeit zu passen.
3. Anschließend wird auch so mit den weiteren Spielern 3 und 4 fortgefahren.
4. Passen in einer Runde alle Spieler, ist die Runde beendet und der Spieler, der die letzte(n) Karte(n) gelegt hat, darf die nächste Runde mit belieben Karten beginnen. Jedoch müssen die Karten, wenn mehrere gelegt werden, denselben Zahlenwert haben.
5. Wird ein oder mehrere Asse in einer Runde gespielt, darf derjenige Spieler die neue Runde beginnen, welcher das/die Ass(e) gespielt hat.
6. Das Spiel ist beendet wenn alle Spieler, bis auf einer, keine Karten mehr haben.

**Ausnahmen:** Wenn der aktuelle Spieler nicht über die bereits

Gelegten Karten drüber gehen will, hat er die Möglichkeit zu passen und der nächste Spieler ist an der Reihe und wird aufgefordert, Karten zu legen oder zu passen.

**Variationen:** Man kann eine Karte aufspielen, man hat aber auch die Möglichkeit 2x, 3x oder 4x die gleichen Karten aufzuspielen. Die nachfolgen Spieler müssen dann mit der gleichen Anzahl an Karten, aber mit einem höheren Kartenwert drüber gehen, oder passen.

**Geschäftsregeln: -**die erste Runde eines Spieles, muss der Spieler mit der Karo 7 beginnen.

**-**Der nächste Spieler muss eine Karte mit einem höheren Kartenwert aufspielen, oder passen.

-Wenn mehrere Karten mit demselben Kartenwert aufgespielt werden, muss der nächste Spieler dieselbe Anzahl an Karten spielen, allerdings mit einem höheren Kartenwert. Wenn er dies nicht kann, muss er passen.

**Allgemeine Systemdienste:** TCP/IP

**Änderungshistorie:** 11.07.2017 H. Schönmuth

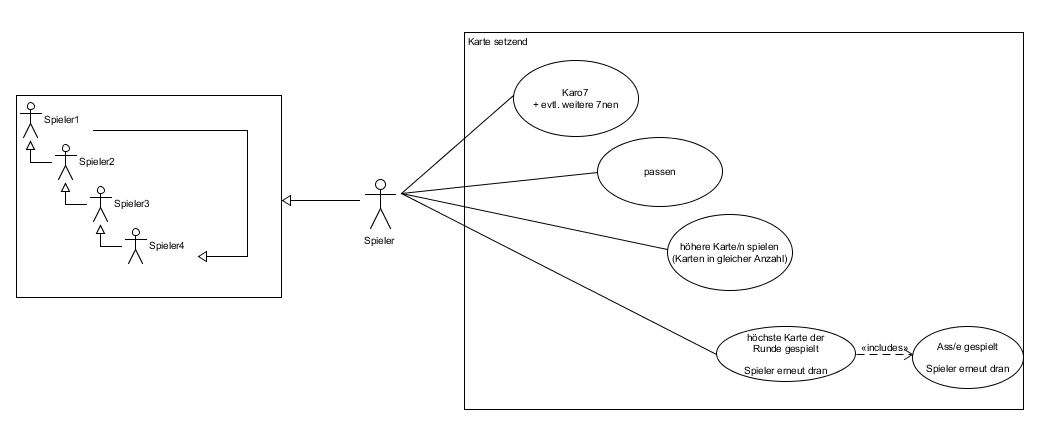


Abb. 2: USE-CASE-Diagramm „Karte setzen“

### *Infrastruktur*

## Objektorientierte Analyse (OOA):

### *Use-Case bezogene Klassendiagramme*

### *Use-Case bezogene Sequenzdiagramme*

### *Fachkonzeptklassendiagramm*

## Objektorientiertes Design (OOD)

### *Komponentendiagramm*

### *Aktivitätsdiagramm*

### *Verteilungsdiagramm*

### *Grundsatzentscheidungen*

### *Klassendiagramm*

## Objektorientierte Programmierung (OOP)

# Tests, Ergebnisse und Diskussion

## Testfälle

# Zusammenfassung