# 



**Pflichtenheft**

Im Rahmen des Softwarepraktikums 2020

**Team „Drunken Slot“**

Minden, den 03. Mai 2020

Überarbeitet: Minden, den 30. Juni 2020

**Autoren:**

Dominik Haacke

(Jonas vom Braucke) (ausgestiegen)

<https://github.com/DMKxD/drunkenslot>

Inhalt



[1 Einführung 3](#_Toc44424963)

[1.1 Beschreibung (Ursprünglich) 3](#_Toc44424964)

[1.1.1 Änderungen 3](#_Toc44424965)

[1.2 Ziele 4](#_Toc44424966)

[2 Anforderungen 4](#_Toc44424967)

[2.1 Stakeholder 4](#_Toc44424968)

[2.2 Funktionale Anforderungen 4](#_Toc44424969)

[2.3 Nicht-funktionale Anforderungen 4](#_Toc44424970)

[2.3.1 Rahmenbedingungen 4](#_Toc44424971)

[2.3.2 Betriebsbedingungen 5](#_Toc44424972)

[2.3.3 Qualitätsmerkmale 5](#_Toc44424973)

[2.4 Graphische Benutzerschnittstelle 6](#_Toc44424974)

[Startscreen 6](#_Toc44424975)

[Host 7](#_Toc44424976)

[Join 8](#_Toc44424977)

[Game 10](#_Toc44424978)

[Zwischenstand 12](#_Toc44424979)

[Endscreen 13](#_Toc44424980)

[Zustandsdiagramm 13](#_Toc44424981)

[2.5 Anforderungen im Detail 14](#_Toc44424982)

[3 Technische Beschreibung 14](#_Toc44424983)

[3.1 Systemübersicht 14](#_Toc44424984)

[3.2 Softwarearchitektur 15](#_Toc44424985)

[3.3 Schnittstellen 15](#_Toc44424986)

[3.4 Datenmodell 16](#_Toc44424987)

[3.5 Abläufe 16](#_Toc44424988)

[3.6 Entwurf 19](#_Toc44424989)

[3.7 Fehlerbehandlung 19](#_Toc44424990)

[3.8 Validierung 19](#_Toc44424991)

[4 Projektorganisation 20](#_Toc44424992)

[4.1 Annahmen 20](#_Toc44424993)

[4.2 Verantwortlichkeiten 20](#_Toc44424994)

[4.2.1 Rollen 20](#_Toc44424995)

[4.3 Grober Projektplan 20](#_Toc44424996)

[5 Anhänge 21](#_Toc44424997)

[5.1 Glossar 21](#_Toc44424998)

[5.2 Referenzen 21](#_Toc44424999)

[5.3 Index 21](#_Toc44425000)

# 1 Einführung

## Beschreibung (Ursprünglich)

Unser Projekt „Drunken Slot“ ist ein verteiltes Trinkspiel, das sich sowohl lokal, als auch Online, spielen lässt. Gerade in Zeiten des Corona Virus sind Trinkspiele die man Online zusammen spielen kann eine gute Ablenkung zum sonst sehr eingeschränkten Unterhaltungsprogramm.

Ein vergleichbares Online Trinkspiel existiert derzeit nicht und es ist somit eine neue innovative Idee, wir haben uns dabei von einem alten „Xbox Live Arcade“ Spiel inspirieren lassen.

Wir entwickeln hier eine „Slot Maschine“, wo man sich vorab mit bis zu (? Maximale Spieleranzahl) Leuten zusammen finden kann. Jeder Spieler bekommt ein „Wild-Symbol“ zugewiesen. Pro Spielrunde ist jeder Spieler einmal dran mit drehen. Die Slot Maschine hat fünf Walzen mit drei Reihen an Symbolen und neun Gewinnlinien. Es gibt fünf normale Symbole, „Shot Verteilen“, Schluck Verteilen“, „Shot Trinken“, „Schluck Trinken“ und Regel. Es gibt drei Arten von Spezialsymbolen, ein „Wild-Symbol“ der alle Spieler betrifft, jeweils ein „Wild-Symbol“ für jeden Spieler und ein „Scatter-Symbol“. Bei jedem Dreh wird festgelegt welches Gewinnbild ausgewählt wird und dem entsprechend erscheinen die Symbole auf den Walzen, sobald diese anhalten. Damit dies nicht zu selten vorkommt, kann man die Gewinnchancen vorab einstellen. Ab drei Symbolen in einer Gewinnreihe kommt es zu einem Gewinn, der sich auf bis zu fünf Symbolen erweitern kann. Bei 3 Symbolen kann man die jeweilige Aktion einmal ausführen, bei vier Symbolen zwei Mal und bei fünf Symbolen drei Mal, zum Beispiel bei vier Mal „Shot Verteilen“ kann der aktuelle Spieler zwei Shots an die Mitspieler verteilen. Kommt in der Gewinnreihe ein Wild Symbol eines Spielers vor, darf dieser anstelle des aktuellen Spielers die Aktion ausführen, bei einer vollen Wild Reihe des gleichen Spielers zählt dies als drei Shots verteilen. Kommt ein Wild Symbol das alle Spieler betrifft dürfen dann alle Spieler diese Aktion ausführen. Bei drei Skatter Symbolen irgendwo auf den Walzen (pro Walze maximal ein Skatter Symbol) werden die „Freispiele“ ausgelöst. Hierbei kann der aktuelle Spieler fünf weitere Male drehen und die Gewinne können nur noch die anderen Spieler negativ beeinflussen. Bei fünf Regel Symbolen kann sich ein Spieler eine Regel ausdenken, die man so lange befolgen muss, bis ein anderer Spieler eine Regel festlegt.

Das Projekt ist so ausgelegt, dass wir bei der Nutzung maximal 10 Spieler pro Lobby behandeln.

An dem Projekt arbeiten Jonas vom Braucke und Dominik Haacke.

Da unsere Gruppe nur aus zwei Personen besteht, wird der Fokus auf dem Java PC-Client und dem Server liegen. Die Android App wird erstmal zweitrangig sein und eher zum Ende des Projekts in Angriff genommen.

### Änderungen

Da mein Teampartner Jonas vom Braucke kurzfristig aus dem Projekt ausgestiegen ist, habe ich es ohne seine Hilfe entwickelt und implementiert. Dadurch hat sich einiges verändert. DrunkenSlot ist jetzt eine Java Standalone Application und läuft sowohl rein in „Command Line“ als auch als Java Swing Client. Die Client-Server Komponente ist leider aus Zeitgründen weggefallen. Die folgenden Grafiken, Diagramme und Texte sind noch auf das Client-Server Prinzip ausgelegt, statt dem Server kommt jetzt an seiner Stelle die Engine zum Einsatz. Die Abläufe sind aber sonst im groben gleich geblieben.

## Ziele

In der heutigen Zeit gibt es eine immer größer werdende Nachfrage nach Videospielen und auch der Markt für Android Apps befindet sich im stetigen Wachstum. Aus diesem Grund haben wir uns entschlossen unser Spiel „Drunken Slot“ Plattformübergreifend zu entwickeln. Mittlerweile hat sich ebenfalls flächendeckend der Standard entwickelt, dass Multiplayerspiele ebenfalls über eine Onlinefunktion verfügen, deshalb wird unser Spiel zusätzlich zu einem lokalen Modus ebenfalls als eine Client-Server Applikation entwickelt. Das fertige Spiel besteht aus drei Komponenten, einem PC-Client, einem Server und einer Android App. Der Server verwaltet das Spiel und fordert die Spieler auf die Aktionen und Züge auszuführen. Die Handlungen der Spieler werden auf allen Clients angezeigt und mit jeweiligen Animation visualisiert.

Die Zielgruppe sind vor allem alle partybegeisterten Leute, die gerne auch mal Online ein Trinkspiel spielen wollen. Da es sich um ein Trinkspiel handelt, in dem es auch darum geht hochprozentigeren Alkohol zu konsumierten ist es natürlich nur für Personen über 18 Jahren geeignet und soll auch nicht zum zu extremen Alkoholkonsum anregen.

# 2 Anforderungen

## 2.1 Stakeholder

|  |  |  |
| --- | --- | --- |
| **Funktion/Relevanz** | **Name** | **Interessen/Ziele** |
| Leiter/Entwickler | Dominik Haake | Fertiges Produkt & Dokumentation & Lernerfolg |
| (Leiter/Entwickler) | (Jonas vom Braucke) | (Fertiges Produkt & Dokumentation & Lernerfolg) |
| Auftraggeber | Prof. Dr. rer. nat.  Jörg Brunsmann | Fertiges Produkt & Dokumentation |

## 2.2 Funktionale Anforderungen

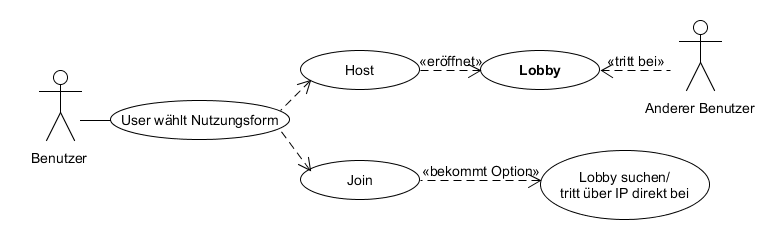


Abbildung : Use-Case-Diagramm

## 2.3 Nicht-funktionale Anforderungen

### 2.3.1 Rahmenbedingungen

#### PC-Anforderungen

* Prozessor: Intel Pentium Prozessor mit 1.5 GHz
* Arbeitsspeicher: 6 GB RAM
* Festplattenspeicher: 1GB
* Bildschirmauflösung: 1024x768

### 2.3.2 Betriebsbedingungen

**PC-Software:**

* Betriebssystem: Microsoft Windows 8 oder neuer
* Eclipse
* Java SE 8

### 2.3.3 Qualitätsmerkmale

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Qualitätsmerkmal** | **Sehr gut** | **Gut** | **Normal** | **Nicht relevant** |
| **Zuverlässigkeit** |  |  |  |  |
| Fehlertoleranz | X |  |  |  |
| Wiederherstellbarkeit |  |  |  | X |
| Ordnungsmäßigkeit | X |  |  |  |
| Richtigkeit | X |  |  |  |
| Konformität | X |  |  |  |
| **Benutzerfreundlichkeit** |  |  |  |  |
| Installierbarkeit | X |  |  |  |
| Verständlichkeit |  | x |  |  |
| Erlernbarkeit | X |  |  |  |
| Bedienbarkeit | X |  |  |  |
| **Performance** |  |  |  |  |
| Zeitverhalten |  |  | x |  |
| Effizienz |  |  |  | X |
| **Sicherheit** |  |  |  |  |
| Analysierbarkeit |  |  | x |  |
| Modifizierbarkeit |  |  | x |  |
| Stabilität |  | x |  |  |
| Prüfbarkeit |  | x |  |  |

## 2.4 Graphische Benutzerschnittstelle

Hier sehen sie den groben Entwurf für das Layout unseres Programms. Die Entwürfe gelten analog für das Smartphone.

### Startscreen

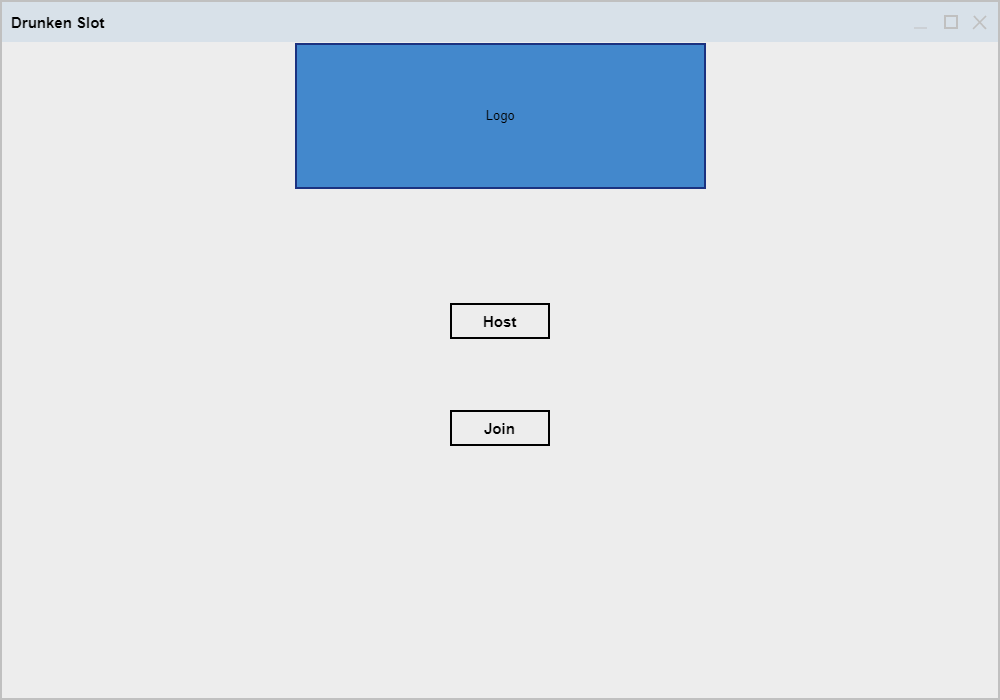


Abbildung : Startscreen Mokup

### Host

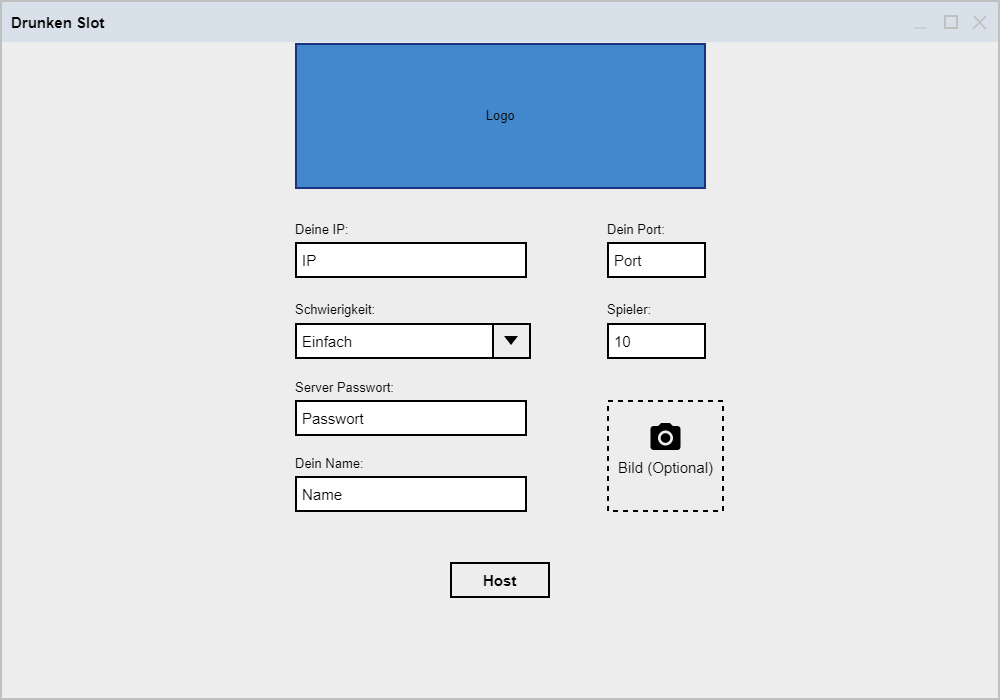


Abbildung : Hostscreen Mokup

### Join

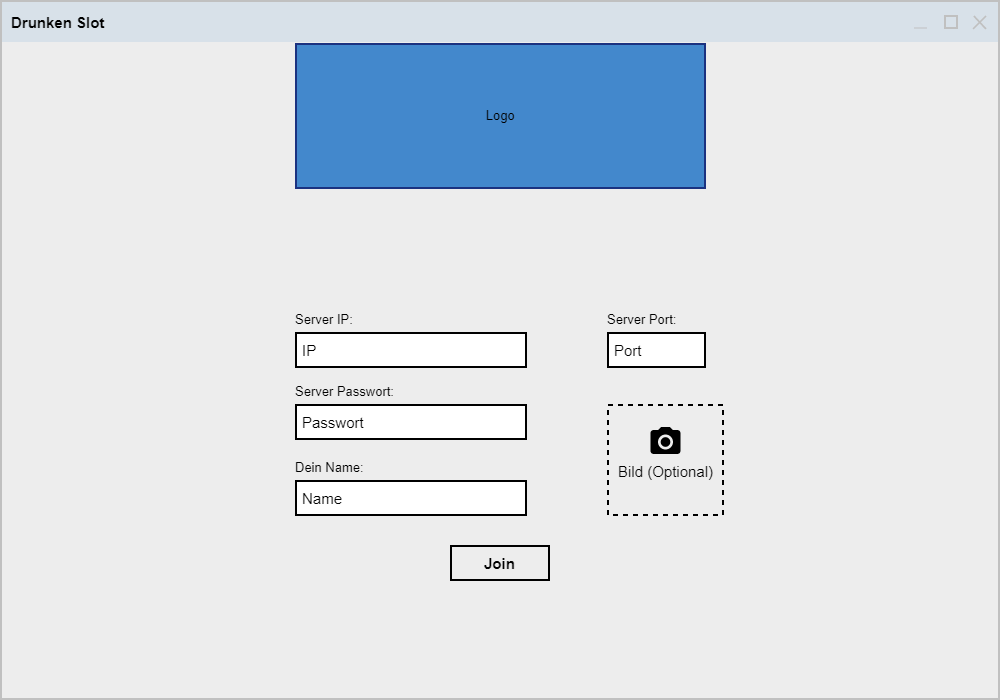


Abbildung : Join Mokup

**Lobby**

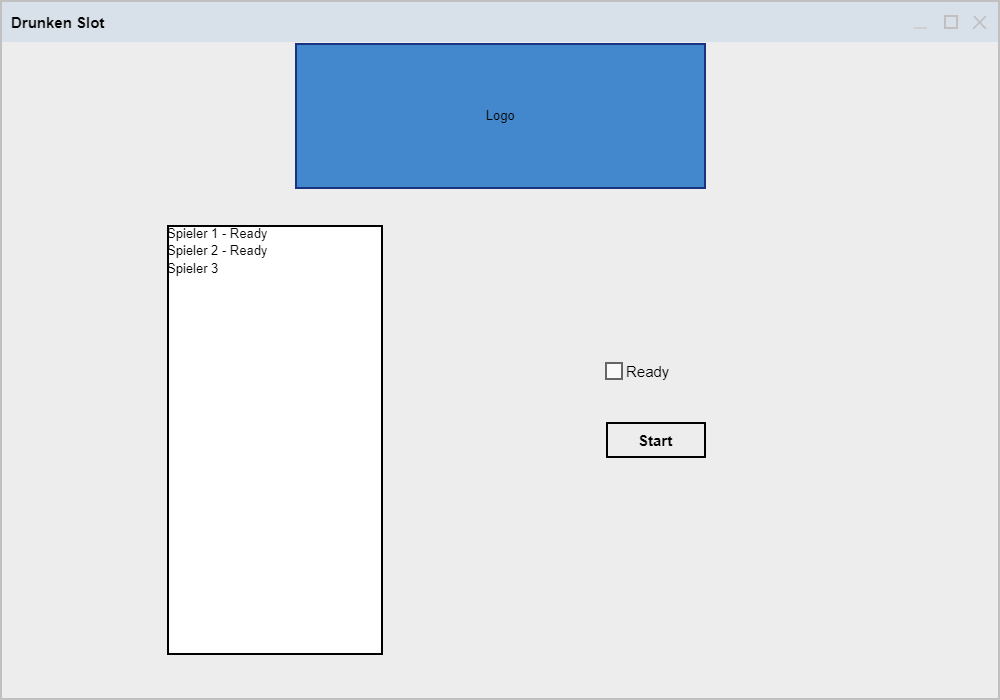


Abbildung : Lobbyscreen Mokup

### Game

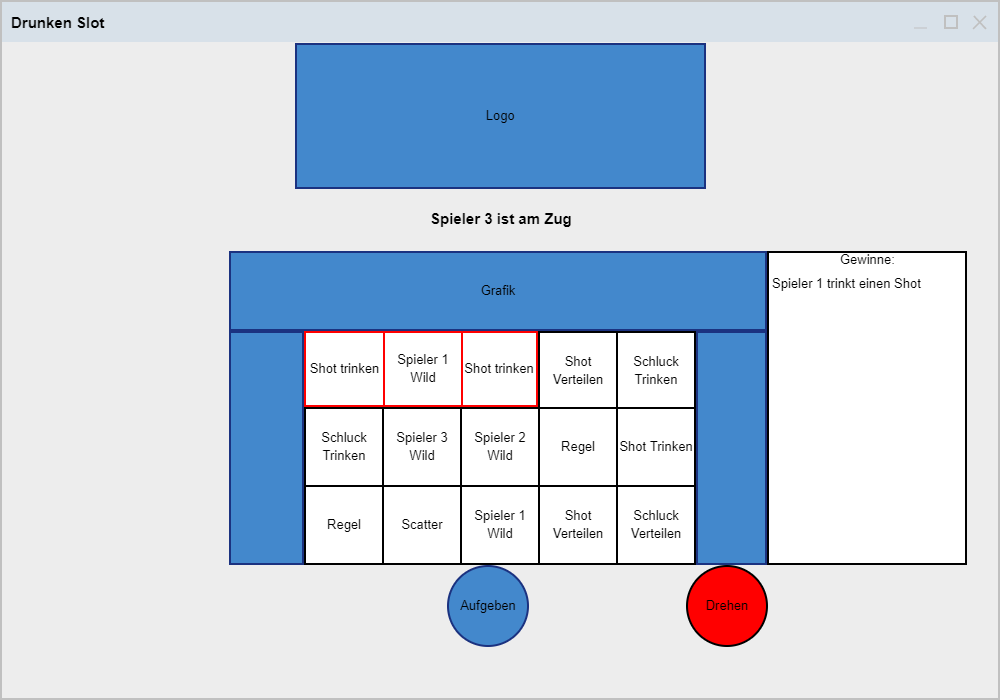


Abbildung : Gamescreen Mokup

**Freispiele**

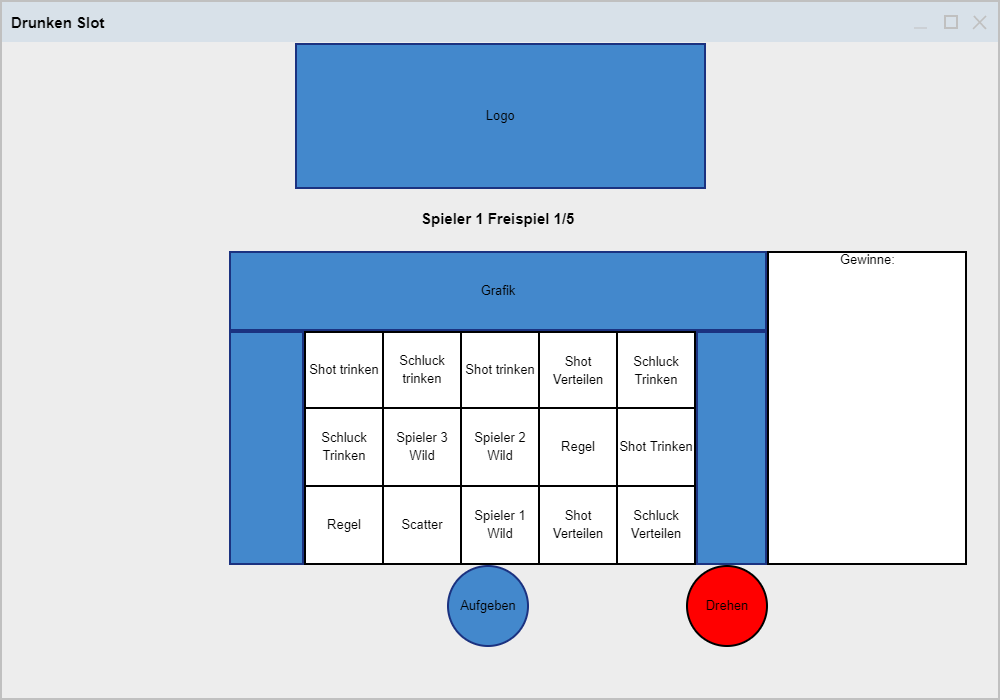


Abbildung : Freispielescreen Mokup

### Zwischenstand

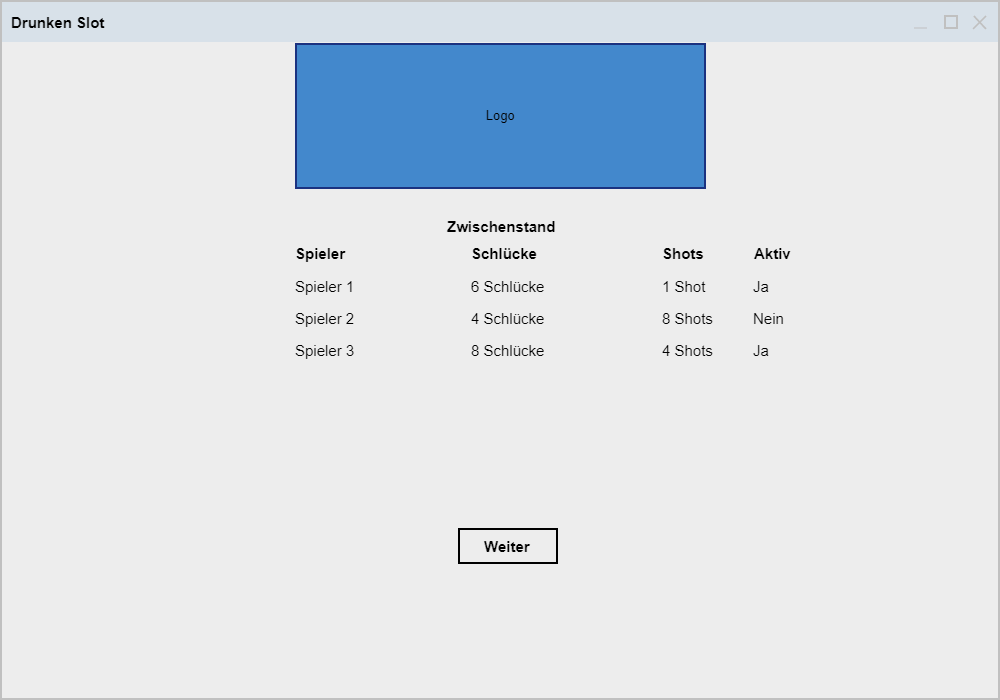


Abbildung : Zwischenstands-Screen Mokup

### Endscreen

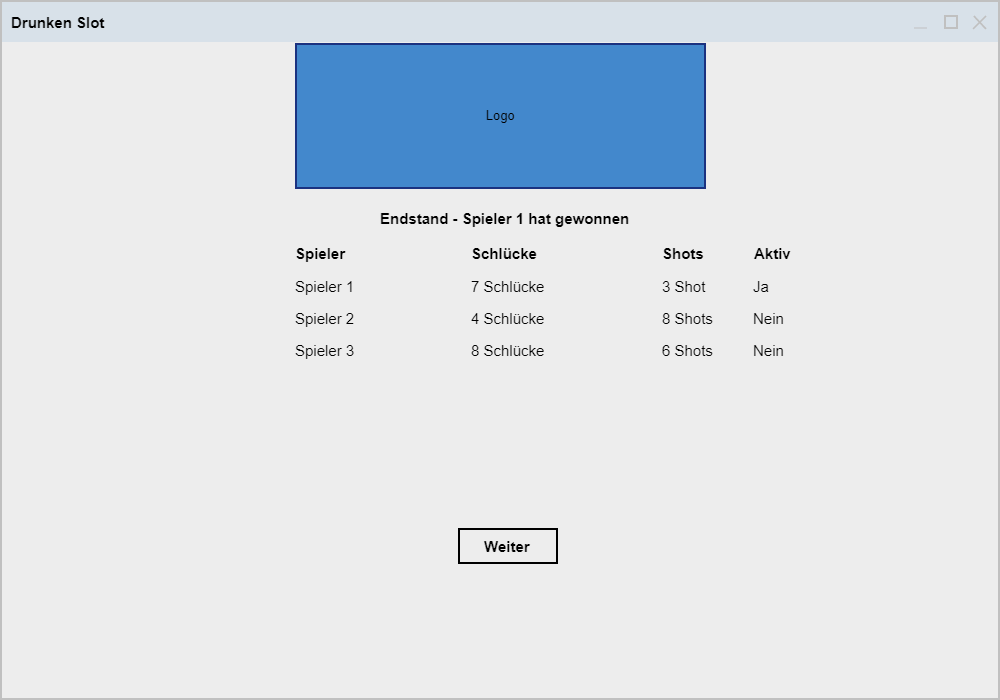


Abbildung : Endstand Mokup

### Zustandsdiagramm

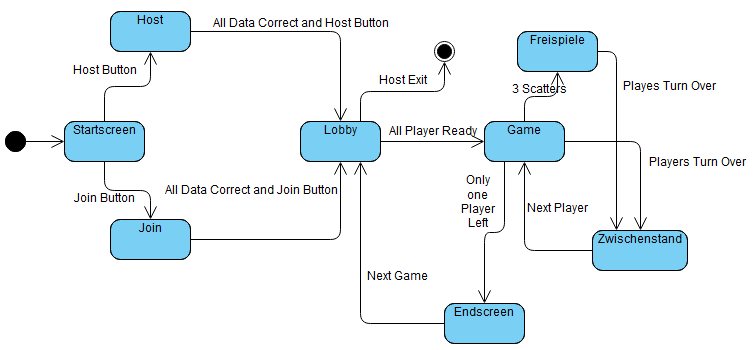


Abbildung : Navigation zwischen den Screens

## 2.5 Anforderungen im Detail

|  |  |  |  |
| --- | --- | --- | --- |
| **Als** | **möchte ich** | **so dass** | **Akzeptanz** |
| Benutzer | ein Spiel hosten/leiten können | andere mit mir Spielen können | Spiel kann von mir gehostet werden |
| Benutzer | einem Spiel beitreten können | ich mit anderen spielen kann | Ich kann einem Host joinen |
| Host (jemand der ein Spiel initialisiert) | meine IP, den Port und ein Passwort einstellen können | ich sicher gehen kann, dass die richtigen Leute beitreten | IP, Port und Passwort sind einstellbar |
| Host | die Schwierigkeit einstellen können | ich die Wahrscheinlichkeit zu regeln | Ich kann die Schwierigkeit einstellen |
| Client (jemand der einem Spiel beitritt) | die IP, den Port und das Passwort eines Hosts eingeben können | einem bestimmten Spiel beitreten kann | Ich kann IP, Port und Passwort eingeben |
| Benutzer | einen Namen und ein Anzeigebild einstellen können | andere Nutzer im Spiel mich erkennen / unterscheiden können | Ich kann einen Namen eingeben und ein Bild setzen |
| Benutzer | einsehen können wer mit mir in einer Lobby ist | ich weiß wer mit mir zusammen spielt | Ich habe eine Anzeige für Spieler in der Lobby |

# 3 Technische Beschreibung

## 3.1 Systemübersicht

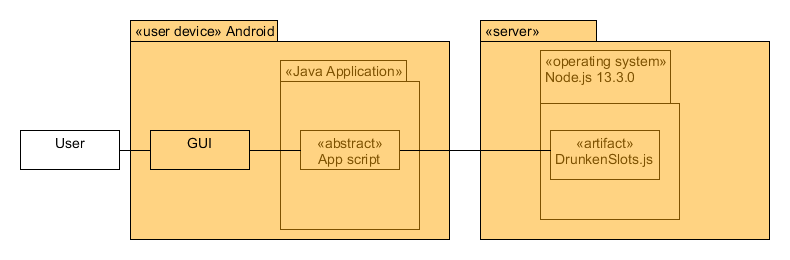


Abbildung : Systemarchitektur Diagram

## 3.2 Softwarearchitektur

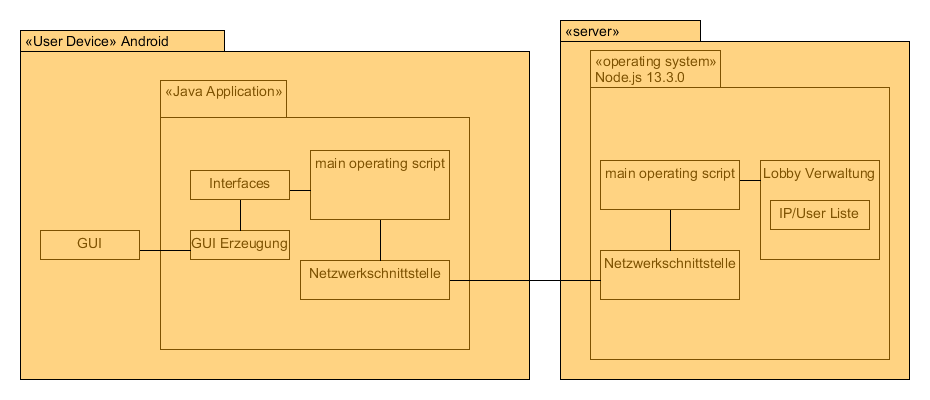


Abbildung : Softwarebausteine

## 3.3 Schnittstellen

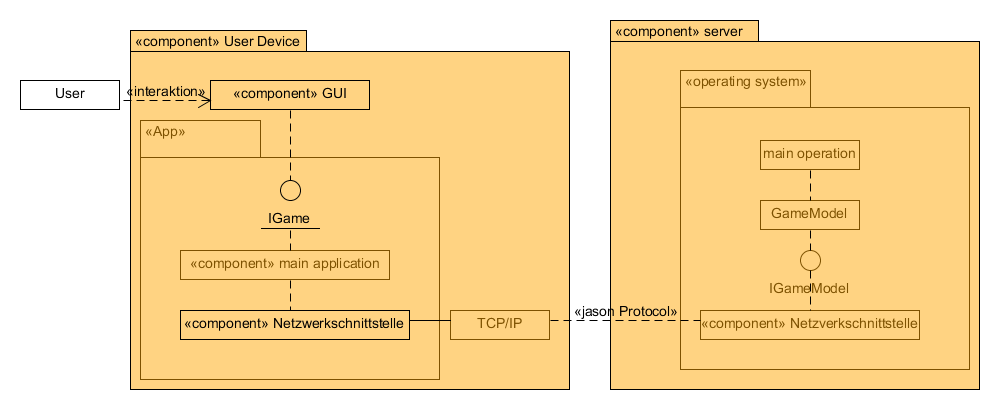


Abbildung : Softwareschnittstellen

## 3.4 Datenmodell

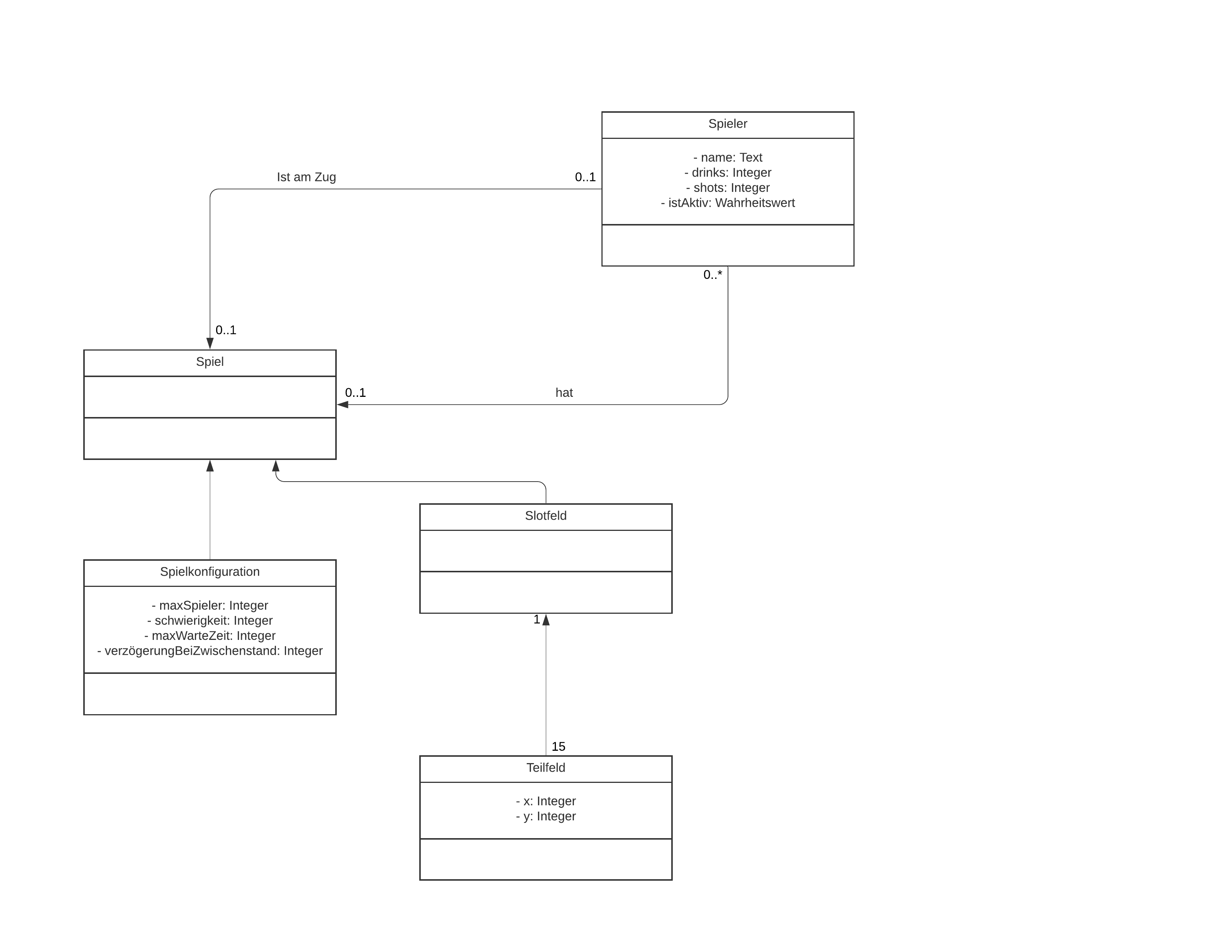


Abbildung : Konzeptionelles Analyseklassendiagramm

## 3.5 Abläufe

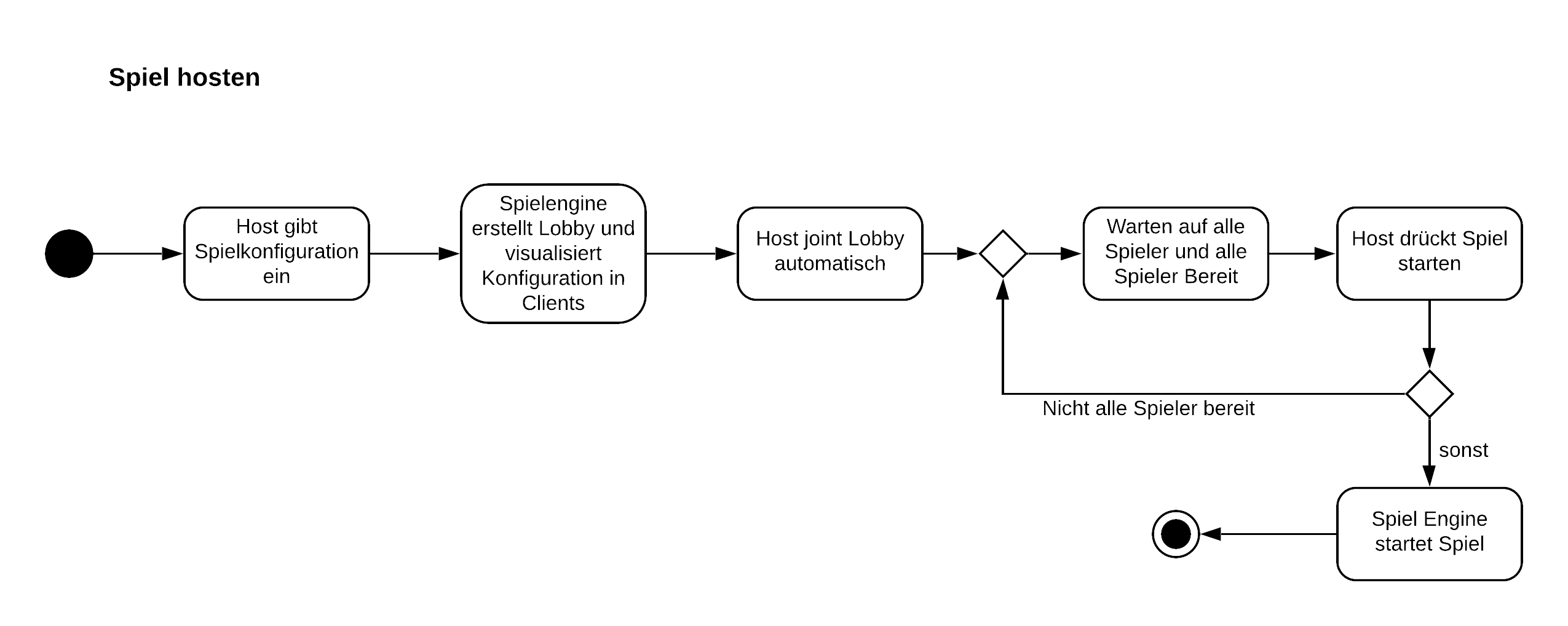


Abbildung : Aktivitätsdiagramm Spiel Hosten

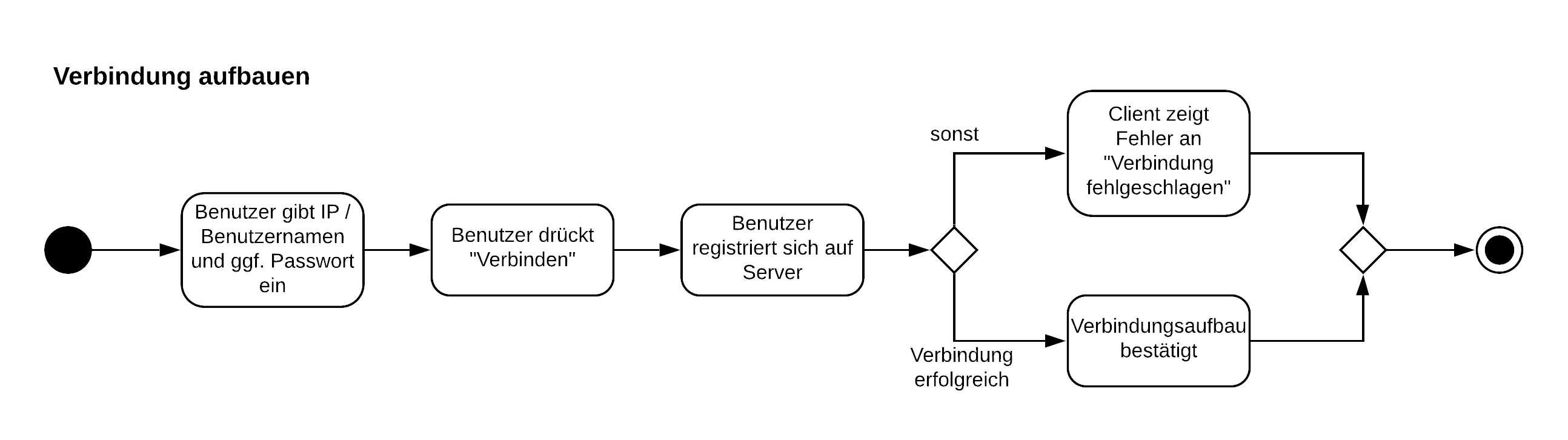


Abbildung : Aktivitätsdiagramm Verbindung Aufbauen

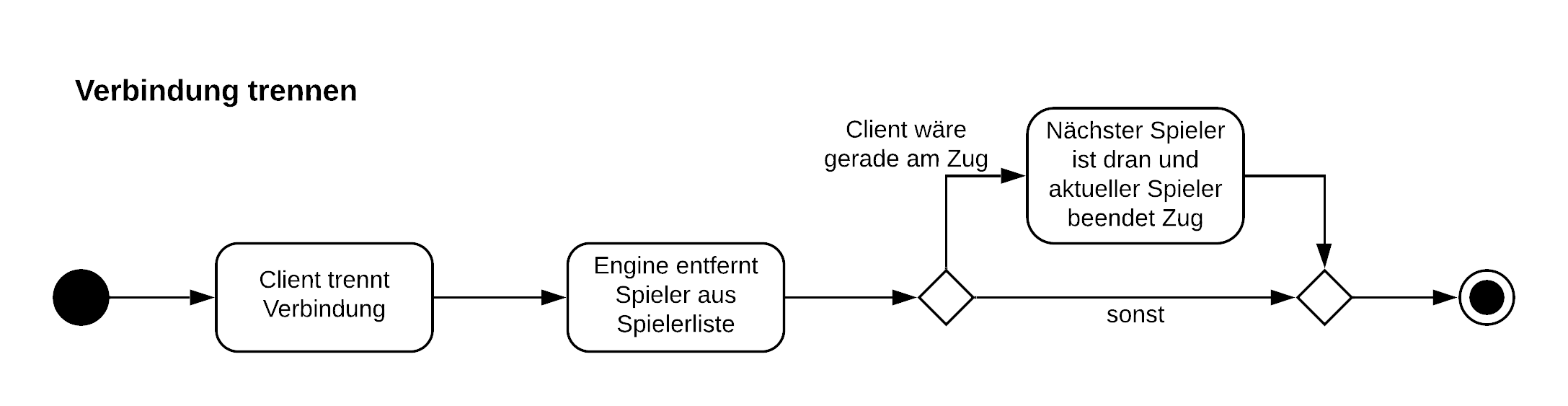


Abbildung : Aktivitätsdiagramm Verbindung trennen

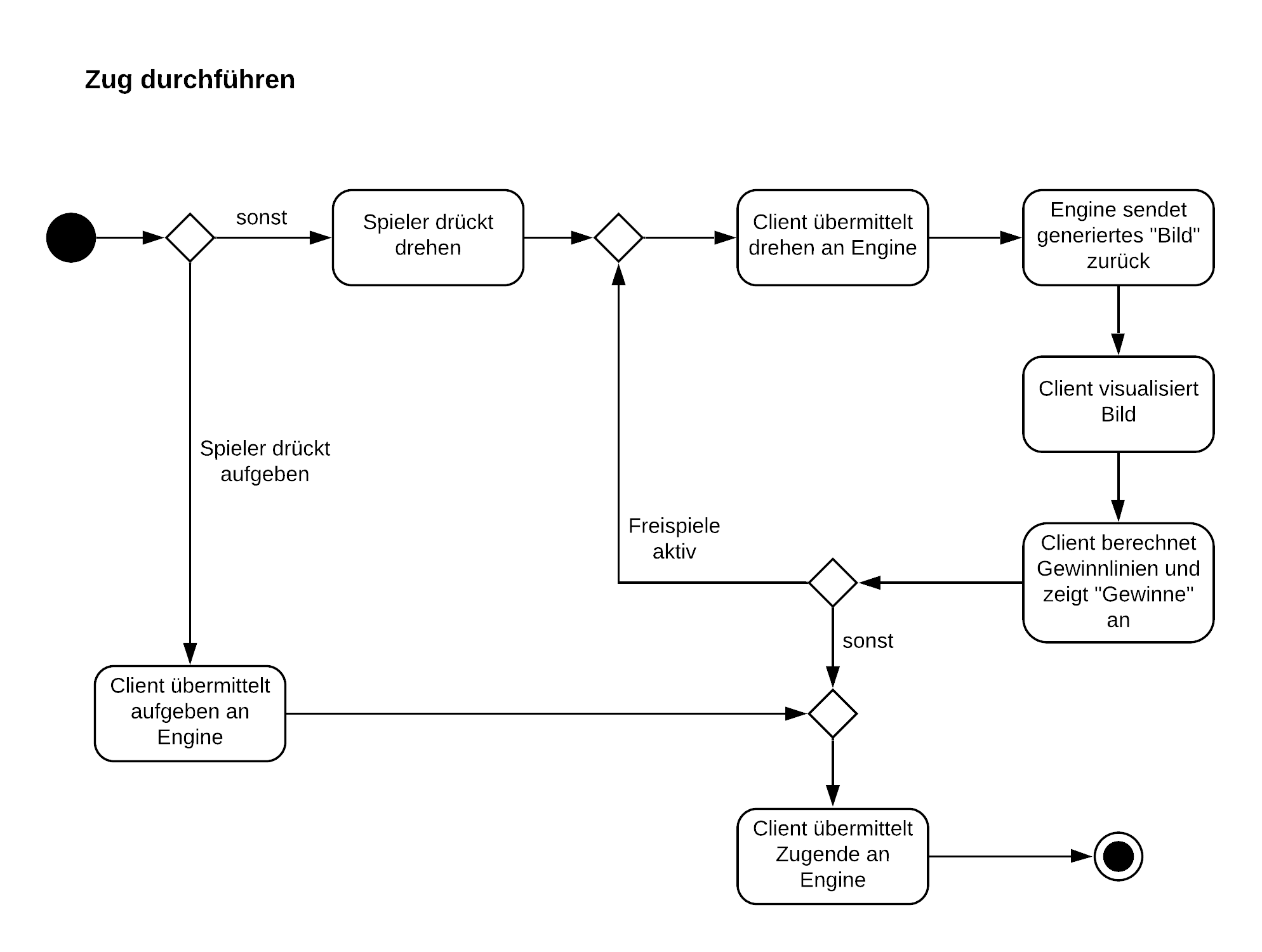


Abbildung : Aktivitätsdiagramm Zug durchführen

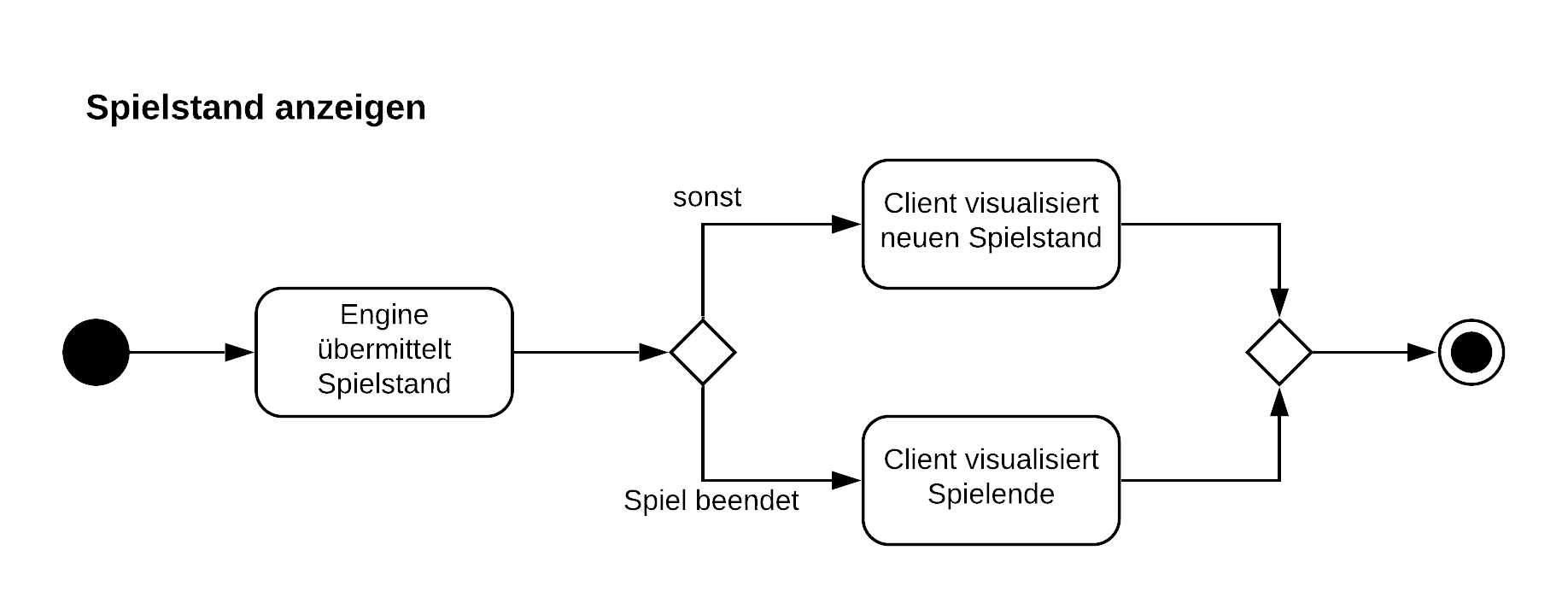


Abbildung : Aktivitätsdiagramm Spielstand anzeigen

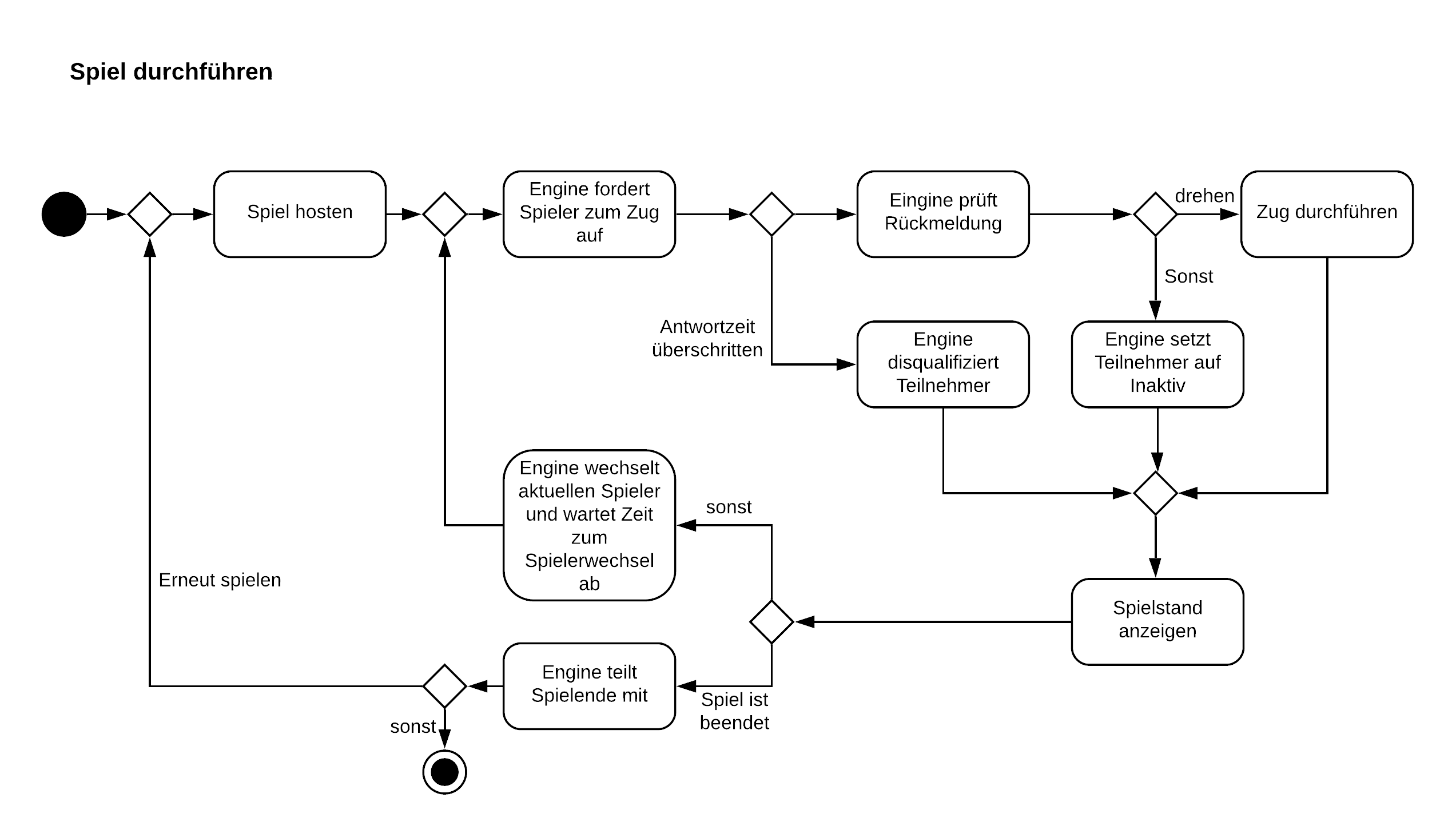


Abbildung : Spiel durchführen

## 3.6 Entwurf

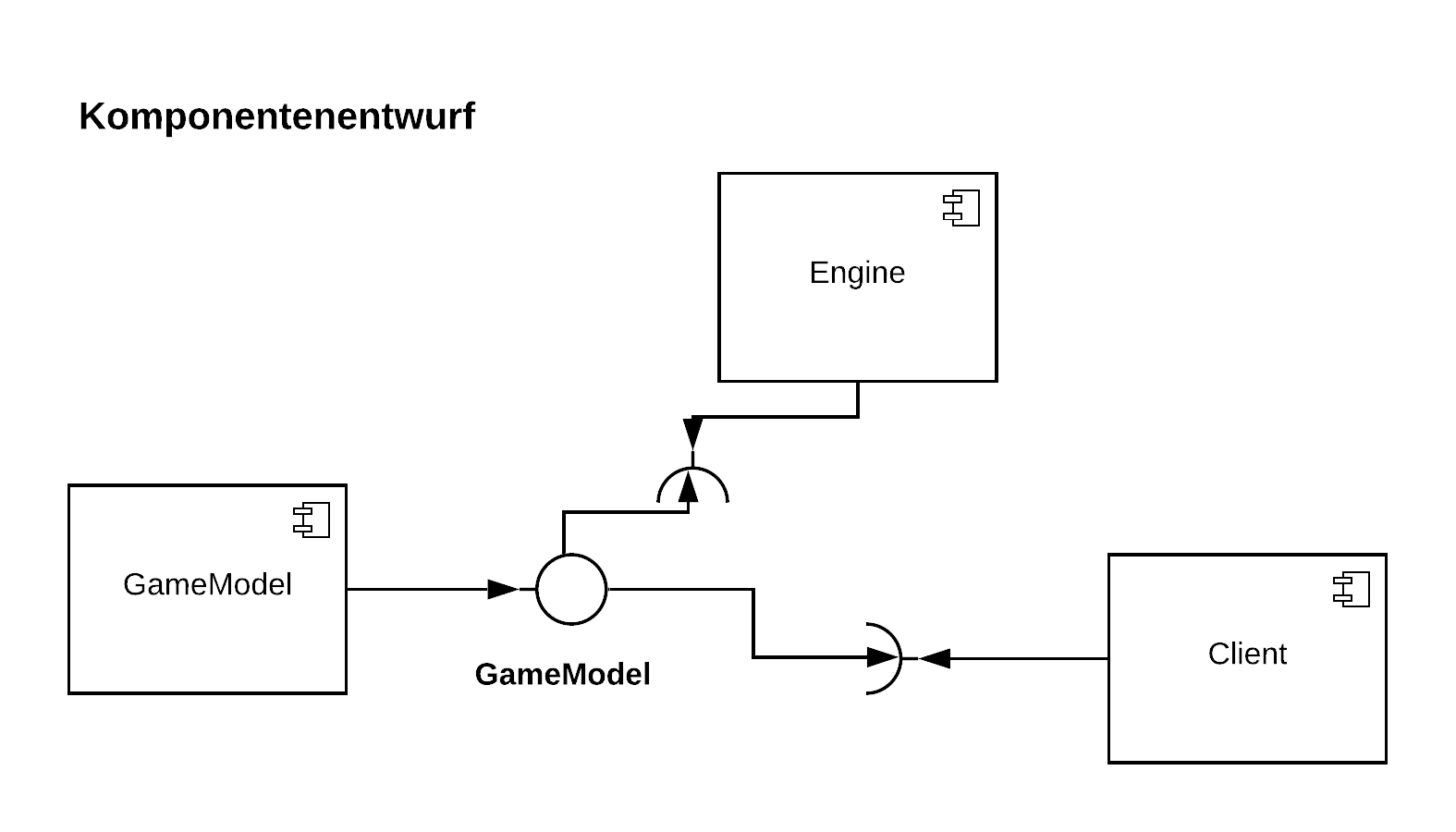


Abbildung : UML-Komponentendiagramm

In Abbildung 17 wird das Zusammenspiel zwischen den Komponenten *Engine*, *Client* und *Gamemodel* beschrieben. Alle diese Komponenten arbeiten auf einem GameModel, welches sie über das Interface IGameModel ansprechen können. Dieses bietet Factory Methoden, um einen neuen Spieler zu erzeugen und ein Spiel zu erstellen. Dieses bietet allen Komponenten die Möglichkeit, eine interne Repräsentation des Spiels „Drunken Slot“ zu erstellen.

## 3.7 Fehlerbehandlung

Keine Kommunikation mit dem Server möglich. (Socket timeout)

Kann IP nicht entschlüsseln.

Fehler in der GUI-Erzeugung.

Fehler in der Input-Erkennung.

Fehler in der internen Kommunikation.

Fehler in der Erzeugung von Zufallswerten.

## 3.8 Validierung

Testfall Spieler verbindet sich mit Server.

Testfall Spieler wartet zu lange mit Rückmeldung an Server.

Testfall Spieler trennt die Verbindung.

Testfall Spieler trennt die Verbindung während seines Zuges.

Testfall alle Spieler außer einem trennen die Verbindung.

Testfall alle Spieler außer einem drücken auf Aufgeben.

Testfall Engine generiert Bild und Client erkennt Gewinnlinien.

Testfall Host drückt erneut spielen und Engine geht zurück in die Lobby mit allen Spielern.

Testfall Engine fordert Spieler zu Zug auf.

Testfall Host will spiel starten und nicht alle Spieler sind bereit.

# 4 Projektorganisation

## 4.1 Annahmen

Wir werden als Softwareumgebungen Eclipse und Android Studio benutzen. Die Hauptprogrammiersprache wird Java sein. Als Kommunikationsprotokoll zwischen Client und Server haben wir JSON ins Auge gefasst. Ebenfalls haben wir ein Git Repository angelegt. Dieses werden wir dann für die verteilte Programmierung benutzen. Die weiteren Betriebsbedingungen sind in 2.3.2 zu finden.

## 4.2 Verantwortlichkeiten

|  |  |
| --- | --- |
| **Softwarebaustein** | **Person** |
| Client / Engine | Dominik Haacke |
| Server / Engine | Jonas vom Braucke |

### 4.2.1 Rollen

|  |  |
| --- | --- |
| **Name** | **Rolle** |
| **Softwarearchitekt / Frontend-Entwickler / Backend Entwickler** | Dominik Haacke |
| **Softwarearchitekt / Frontend-Entwickler / Backend Entwickler** | Jonas vom Braucke |

## 4.3 Grober Projektplan

**Meilensteine**

* KW 20 (11.05) Abgabe Pflichtenheft
* KW 21 (18.05) Aufsetzen und Einarbeitung in genutzte Systeme
* KW 22 (25.05) Basis Programme/Spielelogik
* KW 23 (01.06) Geräte Kommunikation
* KW 24 (08.06) GUI Erzeugung
* KW 25 (21.06) Fertigstellung des Projekts
* KW 26 (22.06-28.06) Puffer zum Nachbessern
* KW 27 (29.06-03.07) Endabgabe

# 5 Anhänge

## 5.1 Glossar

**Slotfeld** Das Slotfeld besteht aus 3 Reihen mit jeweils 5 Feldern und zeigt die gedrehten Symbole an.

**Teilfeld** Ein Teilfeld ist die kleinste Einheit eines Slotfeldes. Es zeigt genau ein Symbol an.

**Spieler** Ein Spieler ist ein Teilnehmer eines Spiels und kann gegen einen anderen Spieler antreten.

**Symbol** Ein Symbol kann aneinander gereiht mit anderen Symbolen zu einem Gewinn führen.

**Gewinnreihe** Es gibt insgesamt neun Gewinnreihen, auf denen je nach Symbolkombination ein Gewinn erzielt werden kann.

**Spielzug** Der Spielzug beginnt mit dem Spielerwechsel. Wenn der Spieler nur drehen wählt, wird eine neue Kombination von Symbolen an der Slotmachine generiert. Er kann aber auch aufgeben und scheidet dann für das ganze weitere Spiel aus.

**Slotmachine** Ursprünglich als einarmiger Bandit bekannt. Ein Glücksspiel Automat, an dem durch Zufall mit einem gewissen Einsatz etwas gewonnen werden kann.

**Wild** Ein Symbol, dass eine Gewinnline erweitert, zu Deutsch könnte man es auch Joker nennen.

**Freispiele** Erscheinen 3 Mal die so genannten Scatter Symbole egal wo auf der Slotmachine werden sie ausgelöst. Der Spieler ist dann eine vorher definierte Anzahl an Drehungen weiter am Zug, und in diesen Drehungen können nur die anderen Spieler verlieren.

**Ausgang des Spiels** Ein Spieler gewinnt das Spiel, wenn alle anderen Spieler entweder aufgegeben haben, oder alle anderen Spieler die Verbindung getrennt haben.

**Client** Ein Client ist entweder ein PC- oder Smartphone Teilnehmer.

**Shot** Ein kleines Glas gefüllt mit einem alkoholischen Getränk

**Einen Trinken** Einen Schluck aus seiner vorher angemischten alkoholischen Mischung nehmen.

## 5.2 Referenzen

## 5.3 Index

[Abbildung 1: Use-Case-Diagramm 4](#_Toc40046394)

[Abbildung 2: Startscreen Mokup 6](#_Toc40046395)

[Abbildung 3: Hostscreen Mokup 7](#_Toc40046396)

[Abbildung 4: Join Mokup 8](#_Toc40046397)

[Abbildung 5: Lobbyscreen Mokup 9](#_Toc40046398)

[Abbildung 6: Gamescreen Mokup 10](#_Toc40046399)

[Abbildung 7: Freispielescreen Mokup 11](#_Toc40046400)

[Abbildung 8: Zwischenstands-Screen Mokup 12](#_Toc40046401)

[Abbildung 9: Endstand Mokup 13](#_Toc40046402)

[Abbildung 10: Navigation zwischen den Screens 13](#_Toc40046403)

[Abbildung 11: Systemarchitektur Diagram 14](#_Toc40046404)

[Abbildung 12: Softwarebausteine 15](#_Toc40046405)

[Abbildung 13: Softwareschnittstellen 15](#_Toc40046406)

[Abbildung 14: Konzeptionelles Analyseklassendiagramm 16](#_Toc40046407)

[Abbildung 15: Aktivitätsdiagramm Spiel Hosten 16](#_Toc40046408)

[Abbildung 16: Aktivitätsdiagramm Verbindung Aufbauen 17](#_Toc40046409)

[Abbildung 17: Aktivitätsdiagramm Verbindung trennen 17](#_Toc40046410)

[Abbildung 18: Aktivitätsdiagramm Zug durchführen 17](#_Toc40046411)

[Abbildung 19: Aktivitätsdiagramm Spielstand anzeigen 18](#_Toc40046412)

[Abbildung 20: Spiel durchführen 18](#_Toc40046413)

[Abbildung 21: UML-Komponentendiagramm 19](#_Toc40046414)