



DESERTS OF DUNE



PFLICHTENHEFT, *Version: 1.0 - TEAM 08*
SOFTWAREGRUNDPROJEKT WISE 2021/2022 UND SOSE 2022



BEARBEITET VON

LEONHARD ALKEWITZ, LOUIS BOCK, ALEXANDER FINK,
JANNIS KIESELBACH, FELIX SCHOPPE, NIKLAS SCHUERRLE

Inhaltsverzeichnis

1 Überblick	3
1.1 Einleitung	3
1.2 Motivation	3
1.3 Vision	3
1.4 Projektkontext	3
2 Anforderungsanalyse	4
2.1 Fachwissen	4
2.2 Anwendungskontext	17
2.2.1 Akteure und Rollen	17
2.2.2 Anwendungsfälle	19
2.2.3 Abläufe	23
2.3 Funktionale Anforderungen	38
3 Softwarespezifikation	83
3.1 Systemschnittstellen	83
3.1.1 Dialogstruktur	84
3.1.2 Graphische Gestaltung und Nutzungskonzept	88
3.2 Domänenmodell	104
4 Randbedingungen	107
4.1 Qualität	107
4.2 Betriebskonzept	114
4.3 Entwicklungsvorgaben	114
4.4 Abnahmekriterien	114
5 Architekturentwurf	115

5.1	Komponentendiagramm	115
5.2	Implementierungsentwurf	119
5.2.1	KI-Client	120
5.3	Geteilte-Logik	122
5.4	Nutzer-Klient	122
5.5	Statistik	123
5.5.1	Netzwerk-Modul	123
5.5.2	Konfiguration	129
5.5.3	Runden-Handler	130

1 Überblick

1.1 Einleitung

In unserem Projekt geht es darum, ein funktionierendes Spiel gemäß Vorgaben aus einem Lastenheft zu erstellen, welches den Namen DESERTS OF DUNE tragen wird. Das Spiel ist für zwei Personen gedacht, wobei mehrere Personen dem Spielgeschehen als Zuschauer folgen können. Hierbei wird alles, also sowohl die gesamte Spiellogik, als auch die graphische Gestaltung von uns selbst entwickelt und implementiert werden.

1.2 Motivation

Das Projekt soll dazu beitragen, uns in dem langfristigen Arbeiten an größeren Projekten mit anderen Personen weiterzubringen und somit unser Arbeiten mit fremdem Code zu verbessern. Außerdem lernen wir dadurch, dass das Spiel auch mit Teilen der Codes anderer Gruppen funktionieren soll, eine universell anwendbare Kompatibilität zu entwickeln. Nicht zu vernachlässigen ist, dass jedes Teammitglied lernt seine Stärken einzusetzen und durch die Stärken anderer mögliche Schwächen verbessern kann. Der Auftraggeber bekommt ein funktionstüchtiges Spiel mit guter Qualität zum Zeitvertreib und kann unsere Fähigkeiten zu programmieren und als Team zu arbeiten besser einschätzen und bewerten.

1.3 Vision

Die fertige Anwendung soll ein rundenbasiertes Multiplayer-Strategiespiel werden, welches wahlweise auch als Einzelspieler gegen eine KI gespielt werden kann. Dafür soll die Anwendung ein Hauptmenü haben, von dem aus der Spieler das Spiel starten kann oder sich eine Übersicht der Häuser mit ihren dazugehörigen Charakteren und deren Statuswerte anzeigen lassen kann. Startet der Spieler das Spiel, so soll ihm eine Auswahl an zwei Spielbaren Häusern zur Verfügung stehen, mit welchen er spielen kann.

Das Spiel selbst soll eine eigene Szenerie haben, die die Dünenlandschaft von Arrakis, mit Flachsand, Dünen, Gesteinsfeldern und Gebirgen darstellen soll. Die Statuswerte der Charaktere auf dem Feld sollen jederzeit für jeden Spieler einsehbar sein und außerdem soll ein Pausenmenü jederzeit im Spiel aufgerufen werden können.

1.4 Projektkontext

An dem Projekt beteiligt sind der Auftraggeber, das arbeitende Team und auch die anderen Teams, da das Spiel auch mit Teilen von Spielen von anderen Teams funktionieren soll. Möglicherweise sind auch außenstehende als Spieldatenbasierte Beteiligung.

Denkbare Weiterentwicklungen für das Spiel sind die Erweiterung auf mehrere Spieler, die gegeneinander spielen können. Daraus ergeben sich dann weitere taktische Möglichkeiten, wie zum Beispiel Bündnisse zwischen Häusern.

Eine weitere denkbare Erweiterung wäre das hinzufügen von neuen spielbaren Charakteren oder Charakter-Typen oder gar neuen Häusern.

2 Anforderungsanalyse

2.1 Fachwissen

Begriff	Aktionspunkte / Action Points (AP)
Beschreibung	Aktionspunkte sind die Eigenschaft, die angibt wie viele Aktionen der Charakter ausführen kann.
Ist-ein	Eine Eigenschaft eines Charakters.
Kann-sein	
Aspekt	ist ein Parameter eines Charakters.
Bemerkung	
Beispiel	Ein Noble hat beispielsweise pro Runde 2 AP und kann damit zwei normale Angriffe durchführen (jeweils 1 AP)

Begriff	Arrakis
Beschreibung	Der Planet auf dem das Spiel stattfindet.
Ist-ein	ein Planet
Kann-sein	
Aspekt	Ist die Szenerie des Spiels DESERTS OF DUNE.
Bemerkung	
Beispiel	

Begriff	Bene Gesserit
Beschreibung	Ist ein schneller und wendiger Charakter-Typ mit hoher Heilungsrate.
Ist-ein	Charakter-Typ
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	HP: 150, Heilungs-HPs: 20, MP: 3, AP: 2, Angriffsschaden: 20, Inventargröße: 5

Begriff	Benutzer-Client
Beschreibung	Ermöglicht es einem menschlichen Benutzer über eine grafische Oberfläche entweder aktiv, als Spieler an einer Partie teilzunehmen oder als passiver Zuschauer im Zuschauermodus eine Partie zwischen zwei anderen Clients zu verfolgen.
Ist-ein	Mensch
Kann-sein	Spielerclient, Zuschauerclient
Aspekt	-
Bemerkung	Kann sich über das Netzwerk bei einem Server für eine Partie registrieren.
Beispiel	Eine Person registriert sich beim Server als Spieler.

Begriff	Charakter
Beschreibung	Eine Figur aus dem Dune-Universum, welche durch eine Einheit auf dem Raster der Spielfelder (Spielbrett) repräsentiert wird.
Ist-ein	bewegliche Einheit
Kann-sein	Player Charakter (PC), Non-Player Character (NPC)
Aspekt	Name, zugehöriger Spieler oder NPC, Charakter-Typ, Health-Points, Inventar, Position auf dem Spielfeld, etc.
Bemerkung	Ein Charakter kann von einem Haus kontrolliert werden, das heißt er ist besiegt, aber noch nicht unbedingt wieder geklont und nicht vom Sandwurm gefressen.
Beispiel	Baron Vladimir Harkonnen, gehört zu House Harkonnen, Character-Typ: Noble HP: 100 Heilungs-HPs: 10 MP: 2 Inventargröße: 5 etc.

Begriff	Charakter-Typ
Beschreibung	Jeder Charakter hat einen Charakter-Typ, der seine Statistiken (HP, MP, AP, ...) bestimmt und ihm ggf. spezielle Aktionen ermöglicht.
Ist-ein	
Kann-sein	Noble, Mentat, Bene Gesserit, Fighter
Aspekt	Die Charaktere haben also unterschiedliche Stärken und Schwächen, in der Art des Schere-Stein-Papier Prinzips.
Bemerkung	
Beispiel	

Begriff	Charakter-Zug
Beschreibung	Züge von Charakter bestehen aus Zugphasen, in denen jeweils entweder ein Bewegungsschritt oder eine Aktion gemacht wird.
Ist-ein	
Kann-sein	Bewegungsschritt, Aktion
Aspekt	Die Abhandlung des Zuges eines Charakters findet in aufeinanderfolgenden Zugphasen statt.
Bemerkung	
Beispiel	Duke Leto ist am Zug. Er hat 2 MP und 2 AP. Zuerst benutzt der Spieler einen MP, um Leto auf ein Feld mit einem Spicekrümel zu bewegen. In der zweiten Zugphase verwendet Leto einen AP, um den Spicekrümel aufzusammeln.

Begriff	Desert of Dune
Beschreibung	Ein rundenbasiertes Taktik-Spiel für zwei Spieler es spielt auf dem Wüstenplaneten Arakis.
Ist-ein	Taktik-Spiel
Kann-sein	
Aspekt	Ist das Grundkonzept des Spiels.
Bemerkung	
Beispiel	

Begriff	Düne
Beschreibung	Hügeliges, höher gelegenes Stück Wüste
Ist-ein	Wüstenfeld
Kann-sein	
Aspekt	Ist betretbar und das Höhenniveau ist hoch.
Bemerkung	
Beispiel	

Begriff	Dünenwanderung
Beschreibung	Dünen wandern ständig umher. Diese Dünenwanderungen werden mittels eines zellulären Automaten berechnet.
Ist-ein	
Kann-sein	
Aspekt	Wüstenfelder mit niedrigem Höhenniveau werden im Automaten als „tot“ betrachtet und Wüstenfelder mit hohem Höhenniveau werden als „lebendig“ betrachtet.
Bemerkung	
Beispiel	

Begriff	Editor
Beschreibung	Der Editor wird zum Erstellen von Spielwelten, Charakteren und Konfigurationen für ein Spiel verwendet.
Ist-ein	-
Kann-sein	-
Aspekt	Szenarios und Partie-Konfigurationen können über eine grafische Oberfläche erzeugt und bearbeitet werden.
Bemerkung	Die mit dem Editor erstellten Ressourcen werden im JSON-Format in Dateien gespeichert.
Beispiel	-

Begriff	Feld
Beschreibung	Ein Feld ist ein Teilbereich des Spielfeldes, der quadratisch ist. Ein Feld wird durch eine Feldart weiter charakterisiert.
Ist-ein	
Kann-sein	Stadt, Flachsand, Düne, Felsplateau, Gebirge
Aspekt	kann frei oder besetzt sein (ein Objekt befindet sich auf dem Feld)
Bemerkung	
Beispiel	

Begriff	Felsplateau
Beschreibung	Flaches, niedrig gelegenes Gelände mit massivem Untergrund.
Ist-ein	Felsenfeld
Kann-sein	
Aspekt	Das Felsenfeld ist betretbar und sein Höhenniveau ist niedrig.
Bemerkung	
Beispiel	

Begriff	Fighter
Beschreibung	Ist ein Charakter-Typ, der im Kampf ordentlich einstecken und austeilen kann.
Ist-ein	Charakter-Typ
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	HP: 200, Heilungs-HPs: 20, MP: 2, AP: 2, Angriffsschaden: 40, Inventargröße: 3

Begriff	Flachsand
Beschreibung	Flachsand ist ein flaches, niedrig gelegenes Wüstenfeld.
Ist-ein	Wüstenfeld
Kann-sein	
Aspekt	Flachsand ist betretbar und sein Höhenniveau ist niedrig.
Bemerkung	
Beispiel	

Begriff	Gebirge
Beschreibung	Das Gebirge ist ein hohes, unpassierbares Felsenfeld.
Ist-ein	Felsenfeld
Kann-sein	
Aspekt	Das Gebirge ist nicht betretbar und sein Höhenniveau ist hoch.
Bemerkung	
Beispiel	

Begriff	Great-Houses
Beschreibung	Great-Houses sind große Adelshäuser, die miteinander konkurrieren. Jedes dieser Häuser besteht aus einer Reihe von Charakteren, die jeweils einen Charakter-Type haben.
Ist-ein	
Kann-sein	
Aspekt	Es gibt im Spiel sechs Great-Houses.
Bemerkung	House Corrino, House Atreides, House Harkonnen, Haus Ordos, House Richese, House Vernius
Beispiel	

Begriff	Great-House-Wahlphase
Beschreibung	Zu Beginn jeder Partie muss der Spieler sich entscheiden, mit welchem Great House er auf Arrakis antreten möchte.
Ist-ein	
Kann-sein	
Aspekt	Der Server nimmt aus der Liste der sechs Great Houses zwei zufällige heraus, die dem Spieler angezeigt werden. Der Spieler wählt eines davon aus. Dies findet nebenläufig für beide Spieler statt, also müssen die Mengen der Great House Wahl disjunkt sein.
Bemerkung	Durch die unterschiedlichen Zusammensetzungen der Charaktere bzw. deren Typen, sollte man das Haus wählen, das dem eigenen Spielstil am nächsten kommt.
Beispiel	

Begriff	Health Points (HP)
Beschreibung	Health Points sind die Eigenschaft wie viele Lebenspunkte ein Charakter hat.
Ist-ein	Eine Eigenschaft eines Charakters.
Kann-sein	
Aspekt	ist ein Parameter eines Charakters.
Bemerkung	
Beispiel	

Begriff	Heilungs-Health Points
Beschreibung	Definiert um wie viele Health Points sich ein Charakter pro Spielrunde Heilen kann, wenn er keine Aktion ausführt.
Ist-ein	Eine Eigenschaft eines Charakters.
Kann-sein	
Aspekt	ist ein Parameter eines Charakters.
Bemerkung	
Beispiel	

Begriff	Höhenniveau
Beschreibung	Es wird zwischen zwei verschiedenen Höhenniveaus unterschieden. Hier geht es darum, ob ein Spielfeld hoch oder niedrig ist.
Ist-ein	Spieldeldeigenschaft.
Kann-sein	niedrig, hoch
Aspekt	
Bemerkung	
Beispiel	Das Gebirge ist hoch. Das Felsplateau ist niedrig.

Begriff	Inventar
Beschreibung	Definiert wie viele Spice krümel ein Charakter maximal tragen kann und wie viele er aktuell trägt.
Ist-ein	Eine Eigenschaft eines Charakters.
Kann-sein	
Aspekt	ist ein Parameter eines Charakters.
Bemerkung	
Beispiel	

Begriff	KI-Client
Beschreibung	Werden autonom von einer Künstlichen Intelligenz gesteuert und könne als Mitspieler an einer Partie teilnehmen.
Ist-ein	Spieler-Client
Kann-sein	-
Aspekt	Der Spieler-Client behandelt Aktionen eines Spielers und nimmt aktiv als Spieler an einer Partie teil.
Bemerkung	-
Beispiel	-

Begriff	Mentat
Beschreibung	Ist jemand, der sehr schlau ist, und deshalb besonders effizient darin, Aktionen auszuführen und Spice zu sammeln. Im Kampf hält dieser Charakter-Typ jedoch nicht sehr viel aus.
Ist-ein	Charakter-Typ
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	HP: 75, Heilungs-HPs: 10, MP: 2, AP: 3, Angriffsschaden: 10, Inventargröße: 10

Begriff	Movement Points (MP)
Beschreibung	Jeder Charakter bekommt zu Beginn seines Zuges Movement Points, mit denen er sich auf ein anderes Feld bewegen kann.
Ist-ein	
Kann-sein	
Aspekt	Für einen Movement Point kann ein Charakter sich auf ein betretbares Nachbarfeld bewegen. Dabei kann er sich im Raster vertikal, horizontal oder diagonal bewegen. Ist das Feld von einem anderen Charakter belegt, so wird dieser weggedrängt und die Plätze werden einfach getauscht.
Bemerkung	Die Movement Points werden für jeden Charakter in der Partie-Konfiguration festgelegt.
Beispiel	Charakter 1 bewegt sich von Feld x auf das Feld y, welches von Charakter 2 besetzt ist. Dann ist Charakter 1 danach auf Feld y und Charakter 2 auf Feld x.

Begriff	Nachbarfeld
Beschreibung	Ein Nachbarfeld ist eines der acht Felder, welche ein spezifisches Feld umgeben.
Ist-ein	Feld auf dem Spielbrett.
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	

Begriff	Noble
Beschreibung	Ist ein hochwohlgeborener Adliger, der durchschnittlich in all seinen Eigenschaften ist.
Ist-ein	Charakter-Typ
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	HP: 100, Heilungs-HPs: 10, MP: 2, AP: 2, Angriffsschaden: 20, Inventargröße: 5

Begriff	Normale Aktion
Beschreibung	Zu Beginn eines Zuges bekommt jeder Charakter so viel Action Points (AP), wie in der Partie-Konfiguration für seinen Charakter-Typ festgelegt ist. Normale Aktionen stehen allen Charakteren zur Verfügung.
Ist-ein	Charakter-Zug
Kann-sein	Angriff, Spice aufsammeln, Spice übergeben
Aspekt	Normale Aktionen kosten 1 AP.
Bemerkung	
Beispiel	

Begriff	Partie
Beschreibung	Ein ganzes Spiel von DESERTS OF DUNE wird als eine Partie bezeichnet und endet, wenn einer der beiden Spieler gewonnen hat.
Ist-ein	
Kann-sein	
Aspekt	Zwei Spieler spielen gegeneinander und dabei kann es Zuschauer geben, die das Spiel anschauen.
Bemerkung	
Beispiel	Spieler A spielt gegen einen KI-Client und Zuschauer B beobachtet das Spiel.

Begriff	Partie-Beginn
Beschreibung	Vor der ersten Runde weist der Server jedem der beiden Häuser eine der beiden Städte zu und platziert dann alle Charaktere eines Hauses auf zufälligen freien betretbaren Felder, die mit der jeweiligen Stadt benachbart sind.
Ist-ein	
Kann-sein	
Aspekt	Zu Beginn stehen alle Charaktere eines Hauses um ihre eigene Stadt herum. Sie stehen also auf den acht Nachbarfeldern der Stadt.
Bemerkung	
Beispiel	

Begriff	Partie-Ende
Beschreibung	Wenn in der Sandwurm-Rundenphase der letzte Charakter, der ein Haus kontrolliert, durch einen Sandwurm-Angriff verschluckt wird, während das andere Haus nach Abhandlung dieser Sandwurm-Rundenphase noch Charaktere hat, gewinnt das andere Haus unmittelbar die Partie. Falls nach einer festgelegten maximalen Rundenanzahl noch immer kein Gewinner feststeht, so werden alle Felsplateaus und Gebirge zu Wüstenfelder. Zusätzlich werden fortan die Sandwurm-Runden durch Shai-Hulud Runden ersetzt.
Ist-ein	
Kann-sein	
Aspekt	Beenden einer Partie.
Bemerkung	
Beispiel	

Begriff	Partie-Konfiguration
Beschreibung	Hier sind fast alle Parameter und Werte gespeichert, welche für das Spiel wichtig sind.
Ist-ein	Datei
Kann-sein	
Aspekt	Die Partie-Konfiguration wird zu Beginn einer Partie vom Server geladen und enthält alle wichtigen Spieleinstellungen.
Bemerkung	
Beispiel	

Begriff	Replay
Beschreibung	Unter Replay versteht man das Abspielen einer vorher aufgezeichneten Partie. Dafür für die Partie mit entsprechenden Informationen in einer speziellen Datei abgespeichert.
Ist-ein	Datei
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	

Begriff	Rundenphase
Beschreibung	DESERTS OF DUNE läuft in Runden ab, in denen Ereignisse passieren und die einzelnen Charaktere ihre Züge nacheinander machen.
Ist-ein	
Kann-sein	Dünenwanderungs-, Sandsturm-, Sandwurm-, Klon-, Charakterzug-Rundenphase
Aspekt	<p>Die Reihenfolge der Rundenphasen ist wie folgt:</p> <ol style="list-style-type: none"> 1. Dünenwanderungs-Rundenphase 2. Sandsturm-Rundenphase 3. Sandwurm-Rundenphase 4. Klon-Rundenphase 5. Charakterzug-Rundenphase
Bemerkung	
Beispiel	

Begriff	Sandsturm
Beschreibung	Zu Beginn jeder Partie wird vom Server ein zufälliges Zentrafeld gewählt. Auf diesem Zentrafeld und allen Nachbarfeldern, also ein 3x3 Quadrat, tobt ein gefährlicher Sandsturm. Charaktere, die sich in diesem Sturm befinden, können keine Aktionen machen und auch selbst nicht Ziel von Aktionen sein. Einzig die Aktion Atomics-Einsatz kann Charaktere beeinflussen.
Ist-ein	
Kann-sein	
Aspekt	Zu Beginn jeder Runde wird in der Sandsturm-Rundenphase das Zentrafeld des Sturms auf ein zufälliges Nachbarfeld bewegt. Das 3x3 Sturmquadrat bewegt sich also Runde für Runde über das Spielfeld. Außerdem wird jedes Wüstenfeld in diesem 3x3 Bereich zufällig auf ein niedriges oder hohes Höhenniveau gesetzt.
Bemerkung	
Beispiel	

Begriff	Sandwurm
Beschreibung	Zu Beginn jeder Runde, in der Sandwurm-Rundenphase, falls es bereits einen Sandwurm gibt, läuft dieser n Felder lang auf seine Zielperson zu. Wenn der Sandwurm auf dem Feld eines Charakters ist, wird dieser Charakter verschluckt und der Sandwurm verschwindet ebenfalls.
Ist-ein	
Kann-sein	
Aspekt	Der verschluckte Charakter muss nicht der Zielcharakter sein, er kann auch unglücklich im Weg stehen. Der Sandwurm kann sich nur über Wüstenfelder bewegen.
Bemerkung	
Beispiel	

Begriff	Server
Beschreibung	Zentrale Komponente, die eine Partie von DESERTS OF DUNE verwalten und abwickeln kann.
Ist-ein	-
Kann-sein	-
Aspekt	Beinhaltet die Anwendungslogik zur Verwaltung von Clients und Partien, sowie der Umsetzung der Spielregeln von DESERTS OF DUNE.
Bemerkung	-
Beispiel	-

Begriff	Siegbedingungen
Beschreibung	Es werden nacheinander Siegmetriken FA82 → p. ⁶⁵ in Reihenfolge betrachtet, wie z.B. welches Haus den größten Spice-Vorrat hat.
Ist-ein	
Kann-sein	
Aspekt	Am Ende einer Partie wird überprüft, welcher der beiden Spieler gewonnen hat.
Bemerkung	Wenn nach einer Metrik Gleichstand herrscht, wird die jeweils nächste Metrik als Tie-Breaker herangezogen.
Beispiel	Alle Charaktere sind Tod. Da Spieler B über mehr Spice als Spieler A verfügt, hat Spieler B gewonnen.

Begriff	Spezielle Aktion
Beschreibung	Zu Beginn eines Zuges bekommt jeder Charakter so viel Action Points (AP), wie in der Partie-Konfiguration für seinen Charakter-Typ festgelegt ist. Spezielle Aktionen sind nur für bestimmte Charakter-Typen verfügbar.
Ist-ein	Charakter-Zug
Kann-sein	Kanly, Family Atomics, Spice Hoarding, Voice, Sword Spin
Aspekt	Spezielle Aktionen kosten so viele APs, wie der Charakter pro Zug zur Verfügung hat.
Bemerkung	Kanly nur für Noble Family Atomics nur für Noble Spice Hoarding nur für Mentat Voice nur für Bene Gesserit Sword Spin nur für Fighter
Beispiel	

Begriff	Spice
Beschreibung	Zu Beginn jeder Runde, falls die Anzahl der auf der Karte liegenden Spicekrümel kleiner als der Spice-Schwellenwert s ist, findet ein Spice-Blow statt, bei dem ein zufälliges Wüstenfeld gewählt wird und alle benachbarten Wüstenfelder werden zufällig auf Flachsand oder Düne gesetzt, und es werden Spicekrümel auf den Felder verteilt.
Ist-ein	
Kann-sein	
Aspekt	Bei einem Spice-Blow werden 3-6 Spicekrümel irgendwo in der Wüste auf Feldern verteilt.
Bemerkung	
Beispiel	

Begriff	Spiceablieferung
Beschreibung	Tritt ein Charakter auf ein Nachbarfeld der eigenen Stadt, so wird direkt nach dieser Zugphase alles Spice aus seinem Inventar in der Stadt abgeladen und zum Spice-Vorrat des Hauses hinzugezählt.
Ist-ein	
Kann-sein	
Aspekt	Alle Charaktere versuchen so viel Spice wie möglich in der Wüste zu sammeln und dann zu ihrer Stadt zurückbringen, da dies die Siegmetrik FA82 \rightarrow p. 65 ist.
Bemerkung	
Beispiel	

Begriff	Spice-Schwellwert
Beschreibung	Der Spice-Schwellwert definiert ist eine feste Anzahl an minimalen Spice Einheiten auf dem Spielfeld. Falls dieser Schwellwert unterschritten wird wird der Spice Blow ausgelöst.
Ist-ein	Ist eine Eigenschaft der Karte.
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	Wenn der spezifische Spiceschwellwert unterschritten wird wird ein Spiceblow ausgelöst.

Begriff	Spielbrett
Beschreibung	Ein Spielbrett besteht aus x mal y schachbrettartigen Feldern.
Ist-ein	
Kann-sein	
Aspekt	Besteht aus verschiedenen Arten von Feldern, die von Charakteren betretbar sein können oder nicht.
Bemerkung	
Beispiel	

Begriff	Spieler
Beschreibung	Der Spieler nimmt aktiv am Spielgeschehen teil und führt Aktionen aus.
Ist-ein	Teilnehmer
Kann-sein	Benutzer-Client, KI-Client
Aspekt	Nimmt aktiv am Spiel teil durch ausführen von Aktionen.
Bemerkung	
Beispiel	Der Spieler A bewegt den Charakter X auf das Wüstenfeld nebenan.

Begriff	Spielstand
Beschreibung	Stellt den aktuellen Spielstand dar mit Spiceguthaben, Position der Charaktere, usw.
Ist-ein	
Kann-sein	
Aspekt	Aktueller Spielstand wird gespeichert, dass es bei einer Pause genau gleich wie davor weitergeht.
Bemerkung	
Beispiel	

Begriff	Stadt
Beschreibung	Auf Arrakis gibt es zwei größere Städte, Arrakeen und Carthag. Beiden Spielern gehört eine davon.
Ist-ein	spezielles Feld
Kann-sein	
Aspekt	Ist nicht betretbar und das Höhenniveau ist hoch.
Bemerkung	
Beispiel	

Begriff	Szenario-Konfiguration
Beschreibung	Eindeutige Beschreibung eines Spielfeldes in JSON-Format.
Ist-ein	Datei
Kann-sein	
Aspekt	So kann man selbst konfigurierte Spielfelder speichern und diese immer wieder verwenden.
Bemerkung	
Beispiel	-

Begriff	Wüstenlautheit
Beschreibung	Damit man die gefährlichen Sandwürmer nicht anlockt, muss man sich in der Wüste sehr langsam fortbewegen, da die Sandwürmer auf Geräusche reagieren. Am Anfang seines Zuges gilt jeder Charakter als „leise“ für diese Runde. Wenn der Charakter während seines Zuges mehr als einmal ein Wüstenfeld betritt oder einen Spicekrümel von einem Wüstenfeld ins Inventar aufnimmt, wird er für diese Runde als „laut“ markiert.
Ist-ein	
Kann-sein	
Aspekt	Um beim Laufen leise zu bleiben, darf also pro Zug höchstens einmal ein Wüstenfeld betreten werden.
Bemerkung	
Beispiel	Charakter A wandert über mehrere Wüstenfelder. Aufgrund des zunehmenden Sandes in seinen Schuhen lockt er feindliche Sandwürmer an.

Begriff	Zufall
Beschreibung	Wenn etwas zufällig gewählt werden soll, ist damit immer eine gleich verteilte zufällige Auswahl unter den entsprechenden Möglichkeiten gemeint.
Ist-ein	
Kann-sein	
Aspekt	
Bemerkung	
Beispiel	

Begriff	Zuschauer
Beschreibung	Der menschliche Benutzer kann sich als Zuschauer im Zuschauermodus und das Spiel anschauen, jedoch nicht ins Spielgeschehen eingreifen.
Ist-ein	Mensch
Kann-sein	
Aspekt	Nimmt passiv am Spielgeschehen teil.
Bemerkung	Man kann von Anfang an zuschauen oder man kann auch während einer Partie schon läuft noch dazukommen.
Beispiel	Dennis möchte aufgrund seiner hohen Verlustrate bei einem reinen KI-Spiel zuschauen, um von diesen zu lernen.

2.2 Anwendungskontext

2.2.1 Akteure und Rollen

AKTEUR:	Server
BESCHREIBUNG:	Zentraler Rechner, der allen Clients das Spiel zur Verfügung stellt
ROLLE:	Der Server ist die zentrale Vermittlungsinstanz zwischen den Clients. Das bedeutet, dass der Server für den Informationsaustausch zwischen den verschiedenen Clients zuständig ist. Dabei ist der Server auch für die Verwaltung der Clients zuständig. Weiterhin verwaltet er auf eine Partie des Spiels DESERTS OF DUNE und beinhaltet die Spielregeln und -logik. Das heißt, der Server simuliert das Spiel, verarbeitet die Befehle von den Clients und sendet ihnen den aktuellen Spielstand. Außerdem kann der Server mithilfe einer Partie- und Szenariokonfiguration eine Partie erstellen.

AKTEUR:	Benutzer-Client
BESCHREIBUNG:	Rechner vom Benutzer, auf dem die Endanwendung läuft
ROLLE:	Der Benutzer-Client dient dem menschlichen Benutzer als grafische Benutzeroberfläche, um mit dem Spiel interagieren oder beobachten zu können (z. B. aktuellen Spielstand ansehen, Zug ausführen, Spiel starten, ...). Der Client muss sich mit einem Server verbinden. Dann kann der Client dem Server durch das Senden von Nachrichten Befehle geben ¹ oder durch den Empfang und die Verarbeitung von Nachrichten vom Server den aktuellen Spielstand anzeigen.

AKTEUR:	menschlicher Benutzer
BESCHREIBUNG:	Mensch, der aktiv oder passiv am Spiel teilnimmt
ROLLE:	Der menschliche Benutzer ist entweder ein passiver Zuschauer oder ein aktiver Spieler. Er kann entweder aktiv oder passiv am Spiel und damit an einer Partie teilnehmen. Dafür startet er den Benutzer-Client und kann über dessen grafische Benutzeroberfläche das Spielgeschehen verfolgen und gegebenenfalls mit der Partie interagieren.

AKTEUR:	Spieler
BESCHREIBUNG:	Menschlicher Benutzer bzw. Teilnehmer mit direktem Einfluss auf das Spielgeschehen
ROLLE:	Ein Spieler ist ein aktiver Teilnehmer an einer Partie. Ein Spieler spielt das Spiel DESERTS OF DUNE gegen eine andere Person und kann dabei von Zuschauern beobachtet werden. Der Spieler kann gemäß der Regeln Aktionen ausführen und Befehle an den Server senden, mit dem Ziel, das Spiel zu gewinnen. Ein Spieler hat auch die Möglichkeit über den Editor das Spiel zu konfigurieren. Außerdem sieht der Spieler den aktuellen Spielstand über den Benutzer-Client.

¹Das gilt nur für Spieler und nicht für Zuschauer

AKTEUR:	Zuschauer
BESCHREIBUNG:	Menschlicher Benutzer ohne Einfluss auf die Partie
ROLLE:	Ein Zuschauer kann passiv eine Partie, das heißt das Spielgeschehen, mitverfolgen. Dafür besitzt der Zuschauer auch einen (eingeschränkten) Benutzer-Client, über den er über die grafische Benutzeroberfläche die Partie angezeigt bekommt. Der Zuschauer hat im Gegensatz zu dem Spieler keine Möglichkeit, aktiv in das Spielgeschehen einzugreifen. Das bedeutet dieser Benutzer kann nicht mit dem Server kommunizieren.

AKTEUR:	KI-Client
BESCHREIBUNG:	Teilnehmer, der von einer künstlichen Intelligenz gesteuert wird
ROLLE:	Ein KI-Client kann als Mitspieler an einer Partie teilnehmen. Der KI-Client kann dabei das Spiel DESERTS OF DUNE aktiv gegen einen anderen Teilnehmer spielen. Dafür muss der KI-Client eine Verbindung beim Server aufbauen und kann dann mithilfe von Nachrichten mit dem Server kommunizieren. Dabei hat er gegenüber dem Spieler nur eingeschränkte Rechte, wie zum Beispiel, dass er keine Partie pausieren kann. Der KI-Client trifft seine Entscheidungen nach vorprogrammierten oder angelernten Regeln, die statisch oder dynamisch an das Spielgeschehen angepasst werden können. Damit versucht der KI-Client das menschliche Verhalten oder eine Spielstrategie zu simulieren.

AKTEUR:	Teilnehmer
BESCHREIBUNG:	Spieler oder KI-Client
ROLLE:	Ein Teilnehmer ist eine aktive Person oder Software, die mit dem Server kommunizieren kann und dadurch aktiv in das Spielgeschehen eingreifen kann. Dafür muss der Teilnehmer eine Verbindung beim Server aufbauen und kann dann mithilfe von Nachrichten mit dem Server kommunizieren. Ein Teilnehmer kann entweder ein Spieler oder ein KI-Client sein.

AKTEUR:	Editor
BESCHREIBUNG:	Grafische Benutzeroberfläche zur Erstellung von Konfigurationen
ROLLE:	Der Editor ist ein Teil des Benutzer-Clients und ist damit eine grafische Benutzeroberfläche. Der Editor soll dem Benutzer die Möglichkeit geben, Inhalt und Konfigurationen, wie zum Beispiel die Szenarios oder die Partiekonfiguration zu erstellen.

AKTEUR:	Tutor
BESCHREIBUNG:	Zugewiesener Tutor: Tim Wibiral
ROLLE:	Der Vertreter des Auftragsgebers, das heißt der für das Projekt zugewiesene Tutor vertritt den Auftraggeber. Das heißt, er überwacht den Entwicklungsprozess, übernimmt während der Implementierungsphase die Rolle eines <i>Product Owners</i> . Zusätzlich veranstaltet der Tutor wöchentliche Treffen, die Tutorien oder <i>Sprint Meetings</i> als <i>Srcum Master</i> und steht für Rückfragen zur Verfügung oder gibt Feedback.

AKTEUR:	Entwickler
BESCHREIBUNG:	Team 008 des Softwaregrundprojekts
ROLLE:	Das Produkt, das heißt das gesamte System, wird von den Entwicklern geplant, modelliert und anschließend implementiert, sowie dokumentiert. Dafür werden basierend auf dem Lastenheft Anforderungen, Definitionen, Analysen und Modelle erstellt, die danach implementiert werden. Am Ende sollen die Entwickler ein funktionierendes Spiel mit allen Komponenten fertigstellen.

AKTEUR:	Standardisierungskomitee
BESCHREIBUNG:	teamübergreifendes Komitee zur Standardisierung von Protokollen
ROLLE:	Das zu entwickelnde Spiel ist eine verteilte Anwendung. Da nicht alle Komponenten von einem Team entwickelt werden und die Komponenten über Protokolle kommunizieren, muss für die Kommunikation ein einheitliches Protokoll existieren. Damit dieses Protokoll teamübergreifend identisch ist und damit von allen Teams verwendet werden kann, muss es von einem Standardisierungskomitee definiert werden. Außerdem muss jedes Team einen Vertreter in dem Komitee haben.

2.2.2 Anwendungsfälle

In diesem Abschnitt werden die Anwendungsfälle des gesamten Systems in zwei UML2-Anwendungsfalldiagramme vorgestellt. In Abbildung 1 werden die Anwendungsfälle vorgestellt, die vor dem Beginn einer Partie auftreten. Diese behandeln die Konfiguration des Systems und die Teilnahme der Clients.

Im zweiten Anwendungsfalldiagramm Abbildung 2 werden die Anwendungsfälle dargestellt, die mit dem Spielablauf verbunden sind, das heißt die Partie und deren Verlauf darstellen.

Die Anwendungsfälle wurden auf diese Weise in zwei Diagramme aufgeteilt, weil die Anwendungsfälle nicht zwangsläufig zusammen hängen und der Übersichtlichkeit halber getrennt werden können.

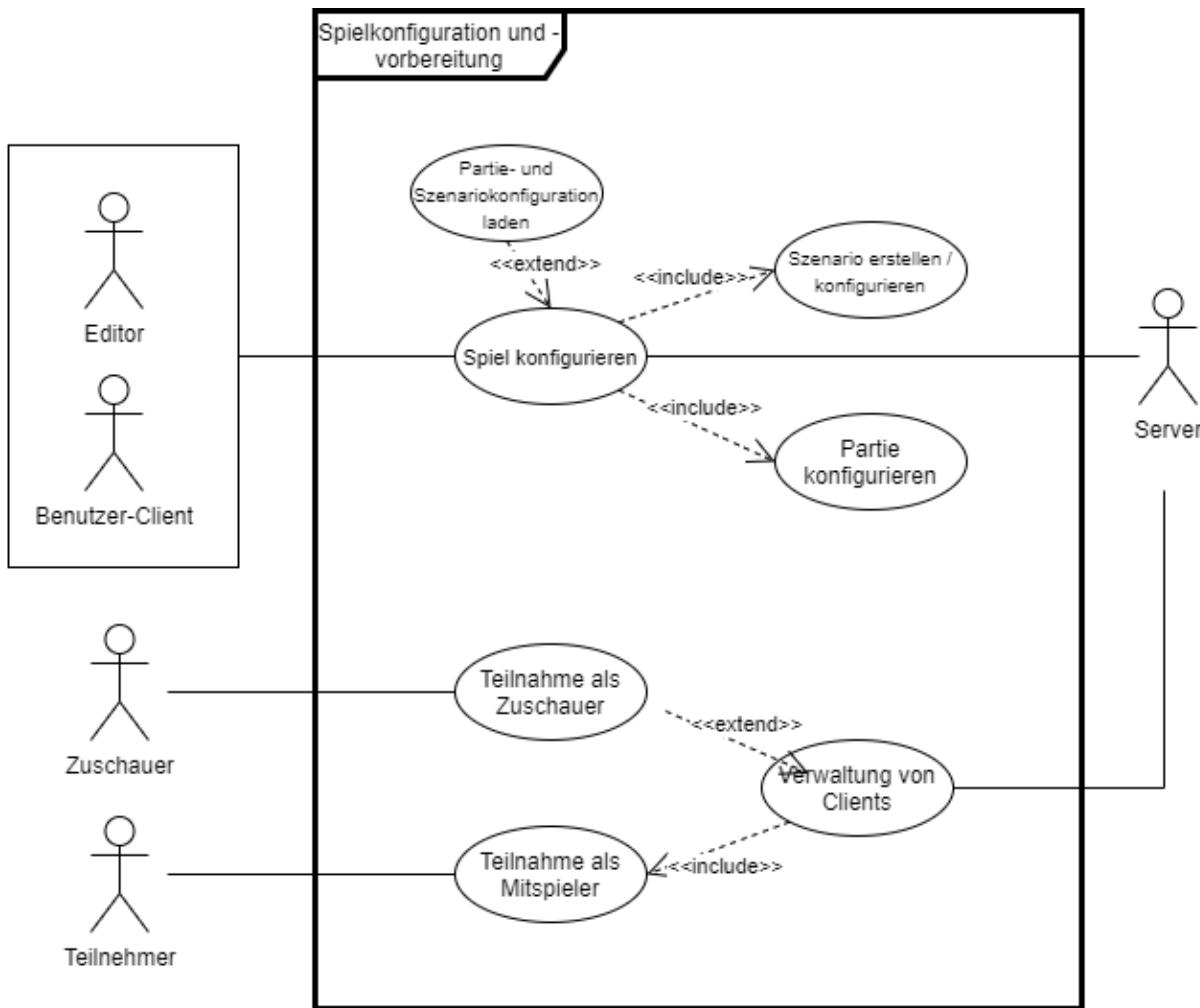


Abbildung 1: Anwendungsfalldiagramm für die Anwendungsfälle bei der Konfiguration und Vorbereitung eines Spieles bzw. einer Partie

Die Akteure Benutzer-Client^{→ p. 17} und Editor^{→ p. 18} wurden gruppiert, um deutlich zu machen, dass der Editor ein Teil des Benutzer-Clients ist. Denn die Anwender des Spiels nutzen den Editor im Benutzer-Client, um das Spiel zu konfigurieren, wodurch beide Akteure gleichermaßen benötigt werden.

Wie man in dem Anwendungsfalldiagramm Abbildung 1 sieht, inkludiert der Anwendungsfall *Spiel konfigurieren* die Anwendungsfälle *Szenario erstellen und konfigurieren* sowie *Partie konfigurieren* und wird durch *Partie- und Szenariokonfiguration laden* erweitert. Diese Gruppenbeziehung besteht so, weil bevor der Server die Konfiguration laden kann, müssen das Szenario und die Partie zunächst konfiguriert werden, weil der Server sonst das Spiel nicht starten kann. Da beide Konfigurationen notwendig sind, müssen sie ablaufen, bevor der Server die Konfiguration laden kann und somit der Anwendungsfall *Spiel konfigurieren* abgeschlossen wird. Die Konfigurationen müssen jedoch nicht geladen werden, da sie auch noch konfiguriert werden können, weswegen dieser Anwendungsfall nur erweitert.

Wie den funktionalen Anforderungen zu entnehmen ist, hat der Server die Aufgabe, die Clients zu verwalten, weswegen der Anwendungsfall *Verwaltung von Clients* mit dem Server verbunden ist.

Dabei sind, um das Spiel anschließend auch spielen zu können, Mitspieler notwendig. Daher inkludiert *Verwaltung von Clients* den Anwendungsfall *Teilnahme als Mitspieler*. Die Tatsache, dass für den Server zwei Mitspieler notwendig sind, lässt sich jedoch nicht darstellen. Außerdem erlaubt der Server, dass Zuschauer der Partie beitreten und über den Spielablauf informiert werden. Deswegen gibt es einen Anwendungsfall *Teilnahme als Zuschauer*, der *Verwaltung von Clients* extended. Das wird damit begründet, dass Zuschauer teilnehmen können, jedoch kein Teilnehmer einer Partie bewohnen muss, das heißt dieser Anwendungsfall optional ausgeführt wird.

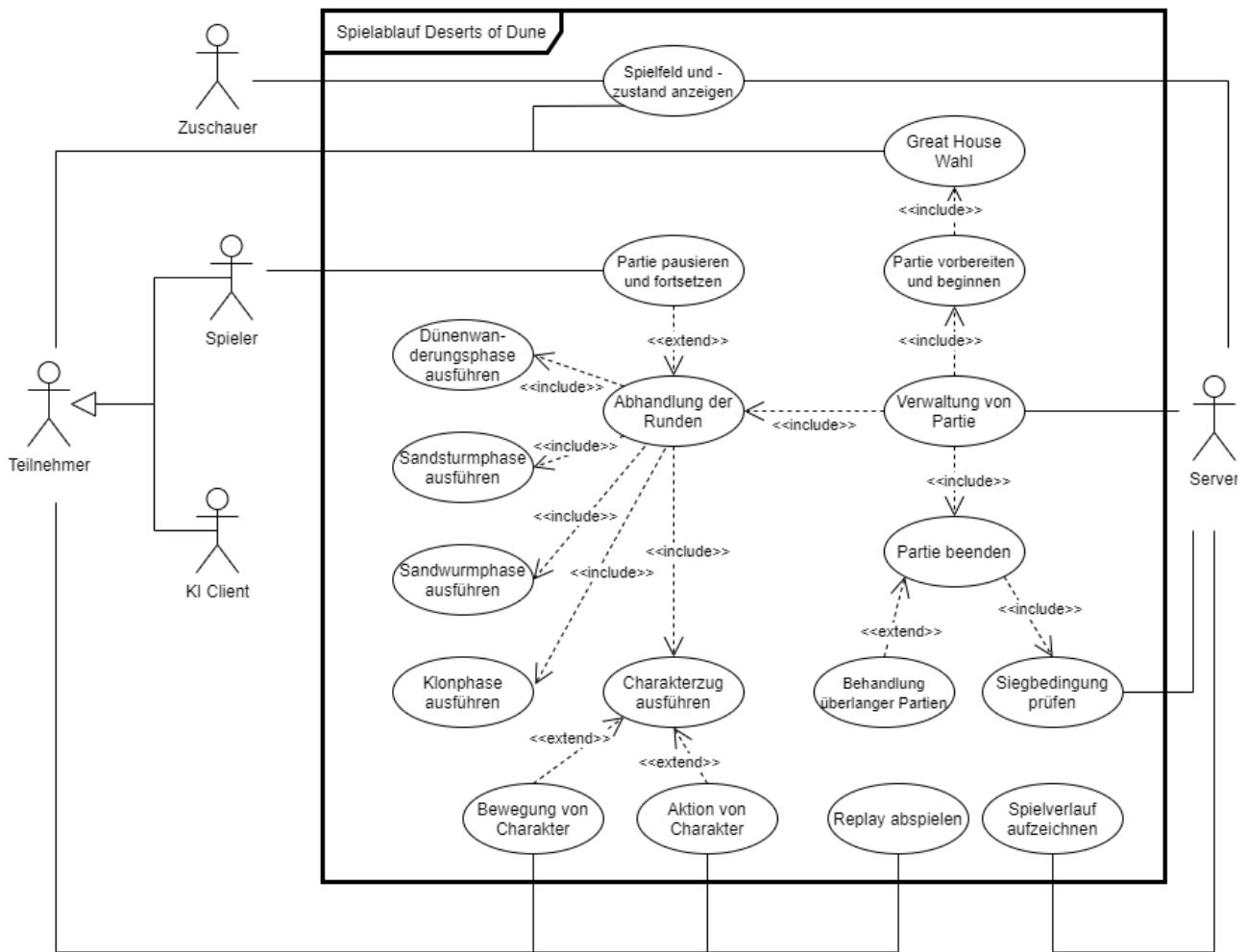


Abbildung 2: Anwendungsfalldiagramm für die Anwendungsfälle Spielablaufs von DUNE

Dieses Anwendungsfalldiagramm Abbildung 2 behandelt alle Anwendungsfälle, die mit der Spieldurchführung zu tun haben. Dabei gibt es Anwendungsfälle, die nur von dem Akteur Spieler^{→ p. 17} ausgeführt werden und manche, die auch vom Akteur KI-Client^{→ p. 18} und damit allgemein dem Mitspieler bzw. Teilnehmer ausgeführt werden. Daher sind Spieler und KI Client Spezialisierung des Teilnehmers.

Wie in den funktionalen Anforderungen definiert wurde, hat der Server ebenso die Aufgabe, die Partie zu verwalten. Das heißt, er hat die Aufgabe, die Partie vorzubereiten, zu beginnen, durchzuführen und anschließend auch zu beenden. Um diese Logik darzustellen, gibt es einen Anwendungsfall *Verwaltung von Partie* der *Partie vorbereiten und beginnen*, *Abhandlung der Runden* (= Spieldurchführung) und *Partie beenden* inkludiert, weil alle diese Anwendungsfälle ausgeführt werden

müssen, um eine Partie abzuhandeln und zu verwalten.

Da die Partievorbereitung die Great-House Wahl beinhaltet, wird *Great House Wahl* von *Partie vorbereiten und beginnen* inkludiert.

Die Abhandlung der Runden, das heißt die Durchführung von einem Rundenzyklus beinhaltet die Durchführung der in den Anforderungen definierten Runden. Das heißt, dass zur Abhandlung einer Runde die Anwendungsfälle *Dünenwanderungsphase*, *Sandsturmphase*, *Sandwurmpphase*, *Klonphase* und *Charakterzugphase* voraussetzt, weswegen diese inkludiert werden.

Der *Charakterzug* wiederum besteht aus Zugphasen, die entweder eine Bewegung oder eine Aktion eines Charakters beinhalten. Da der Charakter sich nicht bewegen muss und auch keine Aktion ausführen muss und damit keiner der beiden Fälle Voraussetzung für einen Charakterzug ist, entenden die Anwendungsfälle *Bewegung von Charakter* und *Aktion von Charakter Charakterzug*.

Ein ähnliches Argument begründet, warum *Behandlung überlanger Partien* den Fall *Partie beenden* extended. Da nicht unbedingt eine überlange Partie auftreten muss, muss dieser Anwendungsfall nicht unbedingt umgesetzt werden. In jedem Fall muss der Server jedoch die Siegbedingung prüfen², das heißt der Anwendungsfall *Siegbedingung prüfen* muss von dem Fall *Partie beenden* inkludiert werden. Bei dem Anwendungsfalldiagramm Abbildung 1 sind den Anwendungsfällen folgende funktionalen Anforderungen aus Unterabschnitt 2.3 zugeordnet:

Anwendungsfall	zugeordnete Anforderungen
Spiel konfigurieren und laden	FA8, FA86, FA87
Szenario erstellen und konfigurieren	FA16, FA121-FA123, FA126-FA128
Partie konfigurieren	FA124, FA125
Teilnahme als Zuschauer	FA4, FA90, FA101
Teilnahme als Mitspieler	FA4, FA7, FA100, FA102, FA114, FA115, FA120,
Verwaltung von Clients	FA3, FA5, FA6, FA4, FA88

Bei dem Anwendungsfalldiagramm Abbildung 2 sind den Anwendungsfällen folgende funktionalen Anforderungen aus Unterabschnitt 2.3 zugeordnet:

²Bei der Behandlung überlanger Partien sieht diese Siegbedingung zwar anders aus, jedoch wird auch auf eine Siegbedingung geprüft.

Anwendungsfall	zugeordnete Anforderungen
Spielfeld und -zustand anzeigen	FA13-FA15, FA17-FA26, FA97, FA103-FA109
Great House Wahl	FA28-FA34, FA73
Partie vorbereiten und beginnen	FA72, FA74, FA89,
Verwaltung von Partie	FA3
Partie beenden	FA78
Siegbedingung prüfen	FA78, FA82, FA98,
Behandlung überlanger Partien	FA79-FA81
Abhandlung der Runden	FA62, FA75
Partie pausieren und fortsetzen	FA94, FA112, FA116,
Dünenwanderungsphase	FA64, FA65, FA66
Sandsturmphase	FA67, FA68
Sandwurmpphase	FA69, FA70
Klonphase	FA71
Charakterzug	FA35-FA44, FA63, FA76, FA77, FA91-FA93, FA95, FA96, FA110, FA111, FA117-FA119, FA27, FA45, FA49, FA83, FA84, FA50-FA61
Bewegung von Charakter	FA99
Aktion von Charakter	FA113
Spielverlauf aufzeichnen	
Replay abspielen	

Die funktionalen Anforderungen FA1-FA18, FA9-FA12 und FA85 finden sich nicht in den Tabellen wieder, da sich nicht eindeutig einem Anwendungsfall zuzuordnen sind. Die Anwendungsfälle FA1-FA18, die sich mit der Architektur befassen finden sich in den gesamten Anwendungsfalldiagrammen Abbildung 2 und Abbildung 1 wieder durch die Aufteilung der Anwendungsfälle auf die Akteure.

Die funktionalen Anforderungen FA9-FA12, die sich mit der Netzwerkkommunikation befassen können allen Anwendungsfällen zugeordnet werden, die etwas mit der Netzwerkkommunikation zu tun haben, wie zum Beispiel *Spielfeld und -zustand anzeigen* oder *Aktion von Charakter*.

Die Anforderung FA85 ist eine Anforderung, die sich ebenfalls nicht eindeutig zuordnen lässt, da die Diagramme voraussetzen, dass Server und Benutzer-Client laufen und bereits gestartet wurden, weswegen diese Anforderungen ebenso nicht dargestellt werden kann.

Im Folgenden Unterabschnitt 2.3 werden die Abläufe und Vorgänge der Anwendungsfälle genauer spezifiziert.

2.2.3 Abläufe

Die folgenden Aktivitäts- und Sequenzdiagramme veranschaulichen die Abläufe der einzelnen Anwendungsfälle aus Unterunterabschnitt 2.2.2 und damit die wichtigen Abläufe vor und während des Spiels bzw. Systems

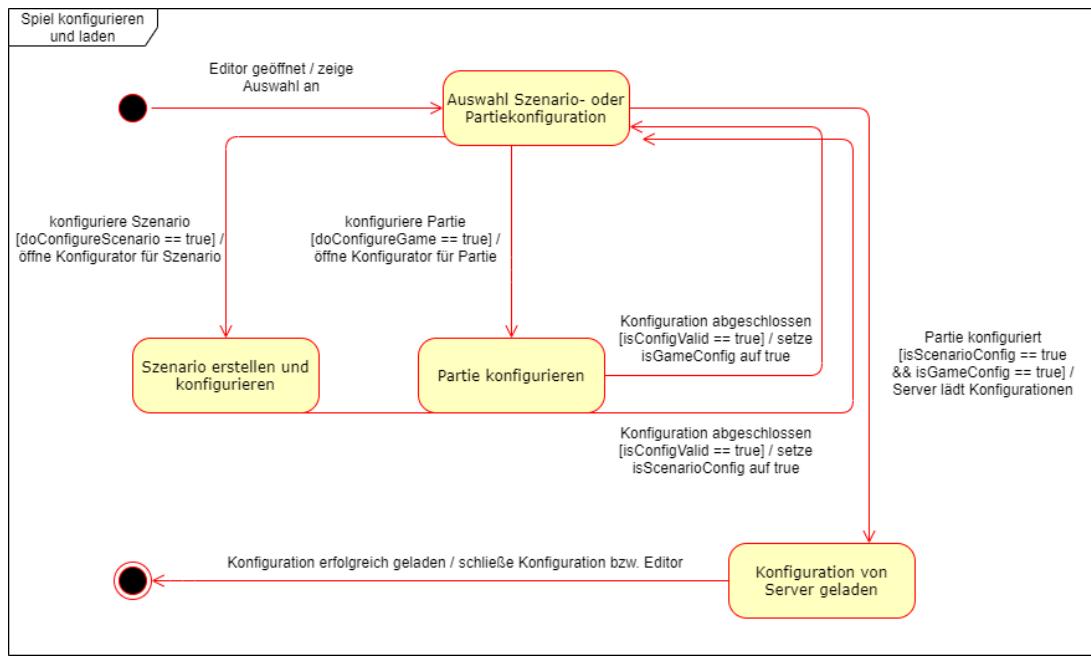


Abbildung 3: Zustandsdiagramm für den Ablauf der Konfiguration eines Spiels und dem anschließenden Laden der Konfiguration durch den Server

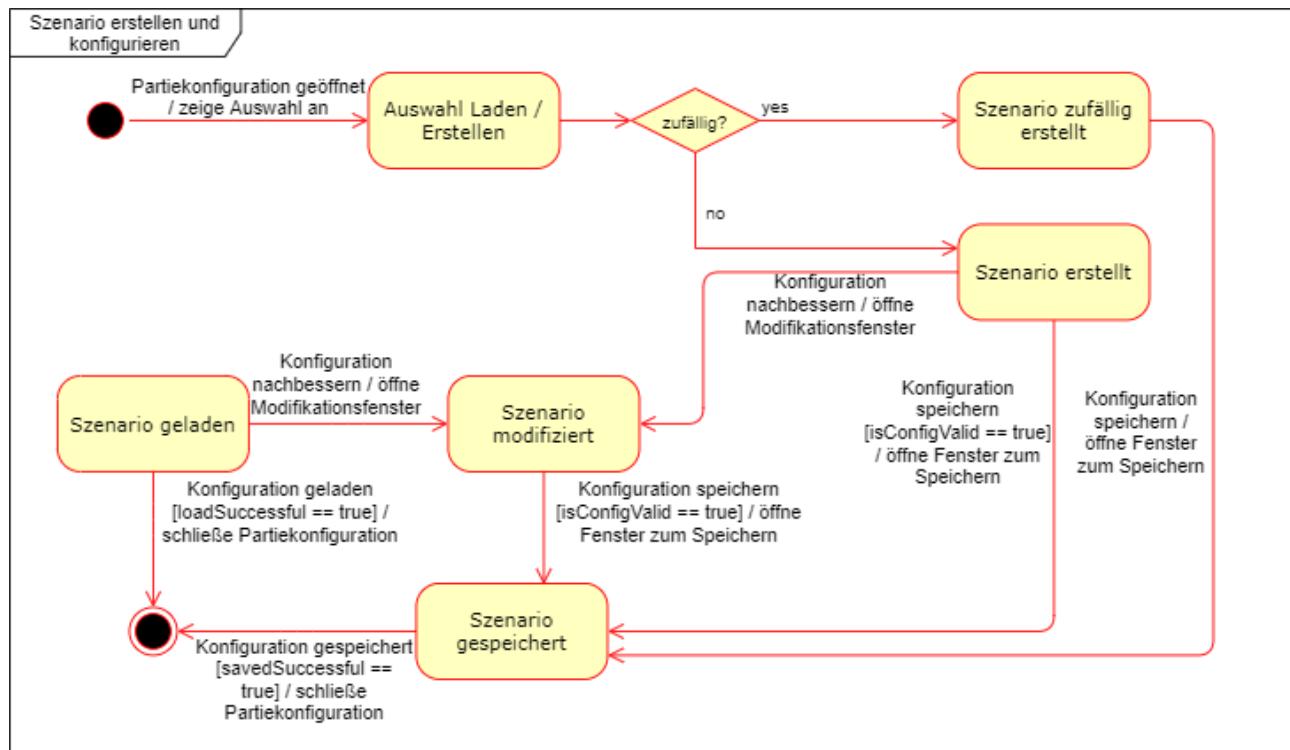


Abbildung 4: Zustandsdiagramm für das erstellen, modifizieren und laden von einer Szenariokonfiguration

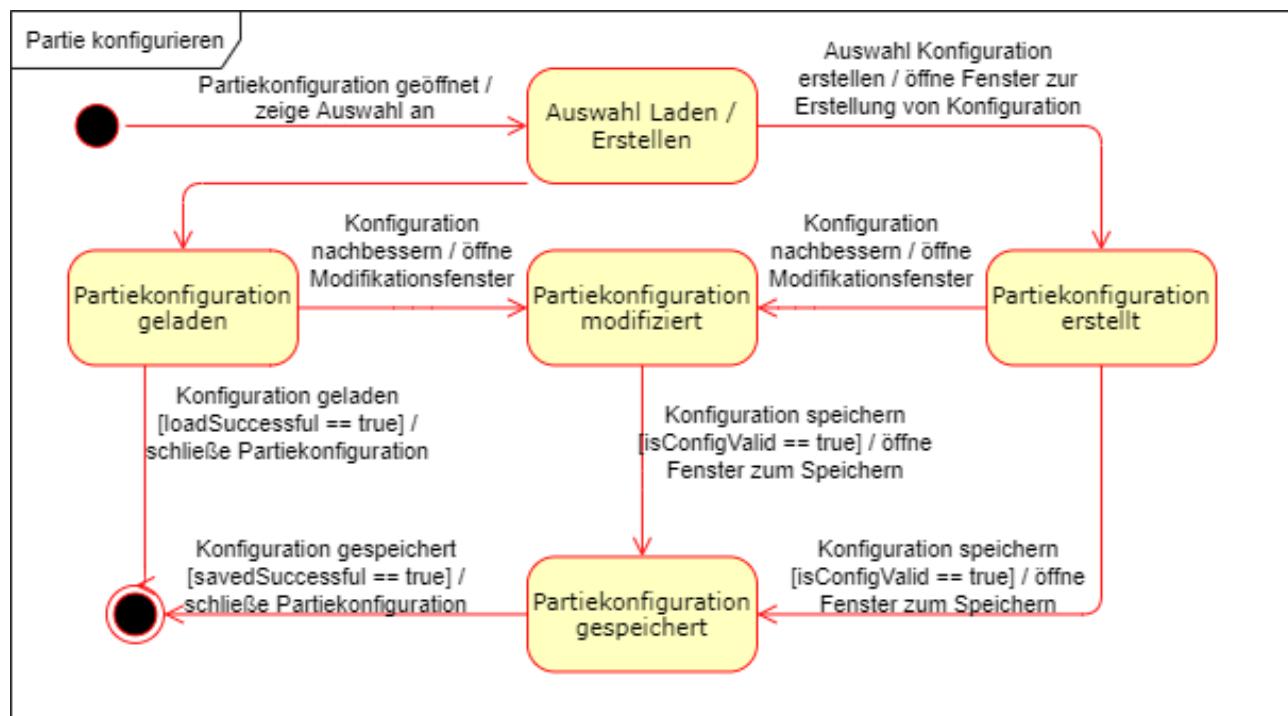


Abbildung 5: Zustandsdiagramm für das erstellen, modifizieren und laden von einer Partiekonfiguration

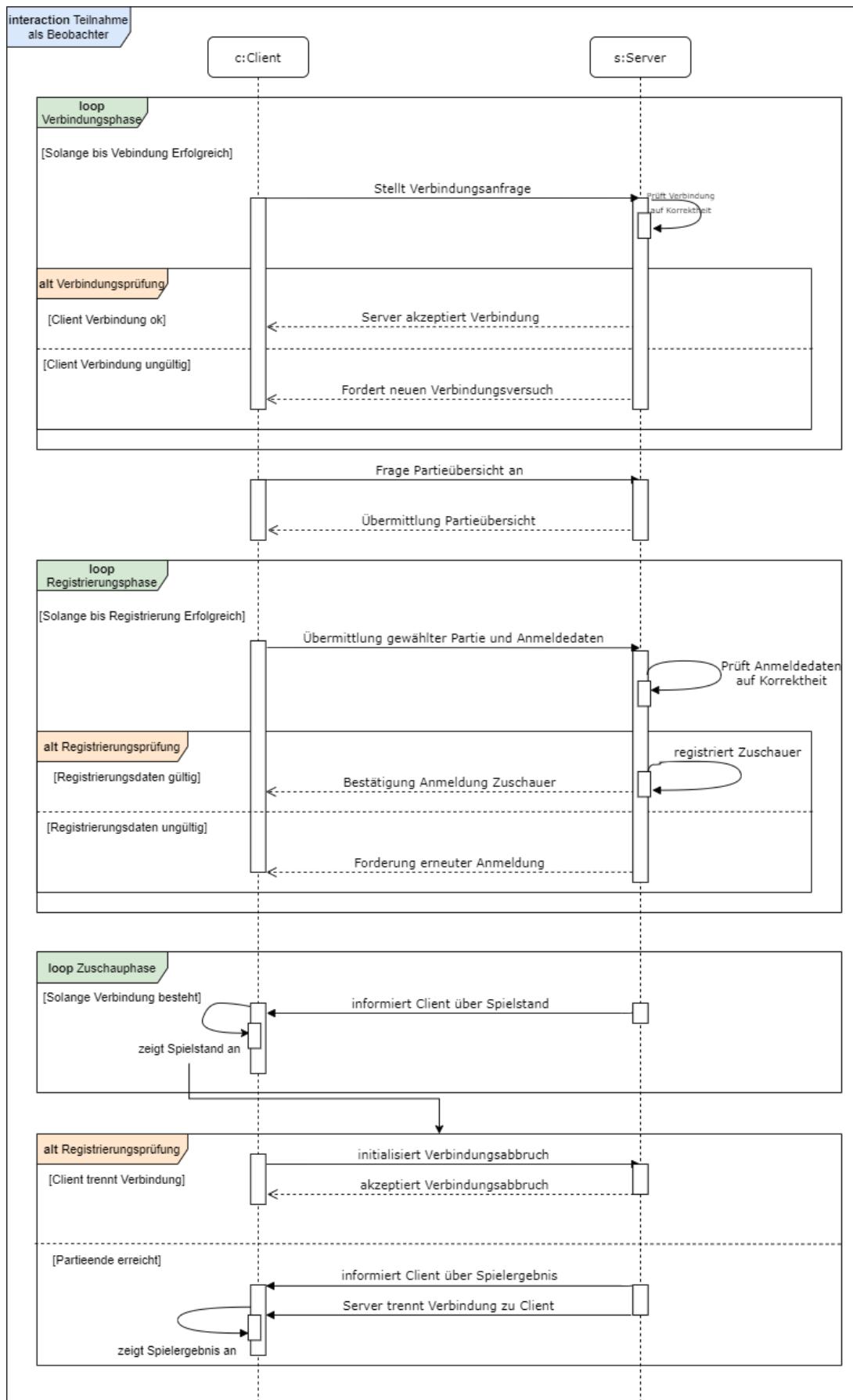
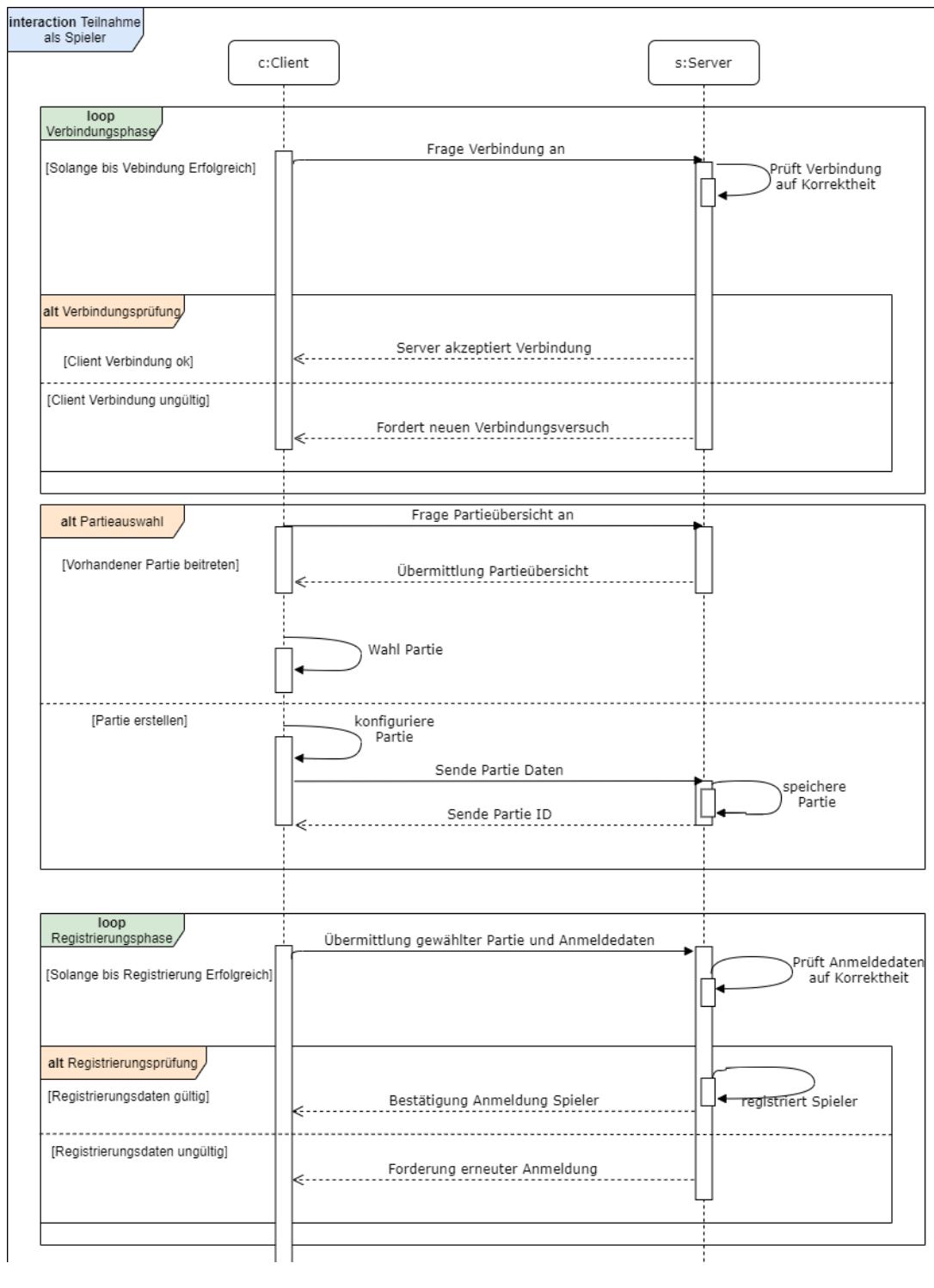


Abbildung 6: Sequenzdiagramm für Ablauf der Teilnahme eines Clients als Zuschauer



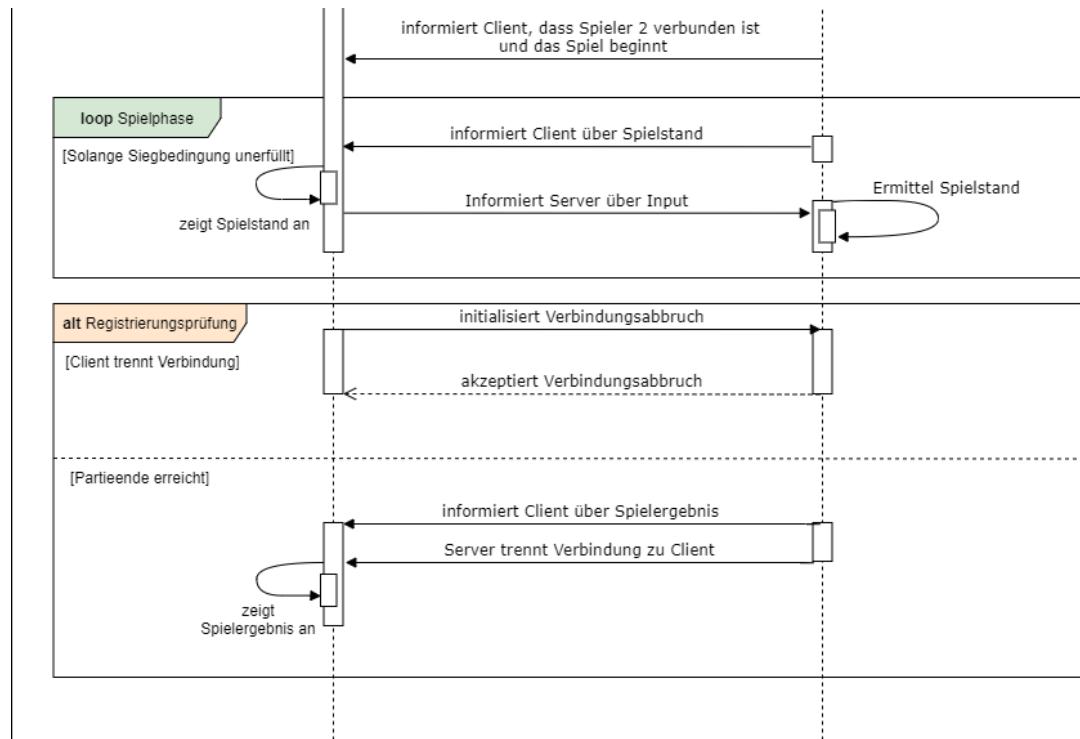


Abbildung 7: Sequenzdiagramm für Ablauf der Teilnahme eines Clients als Mitspieler (Fortsetzung auf nächster Seite)

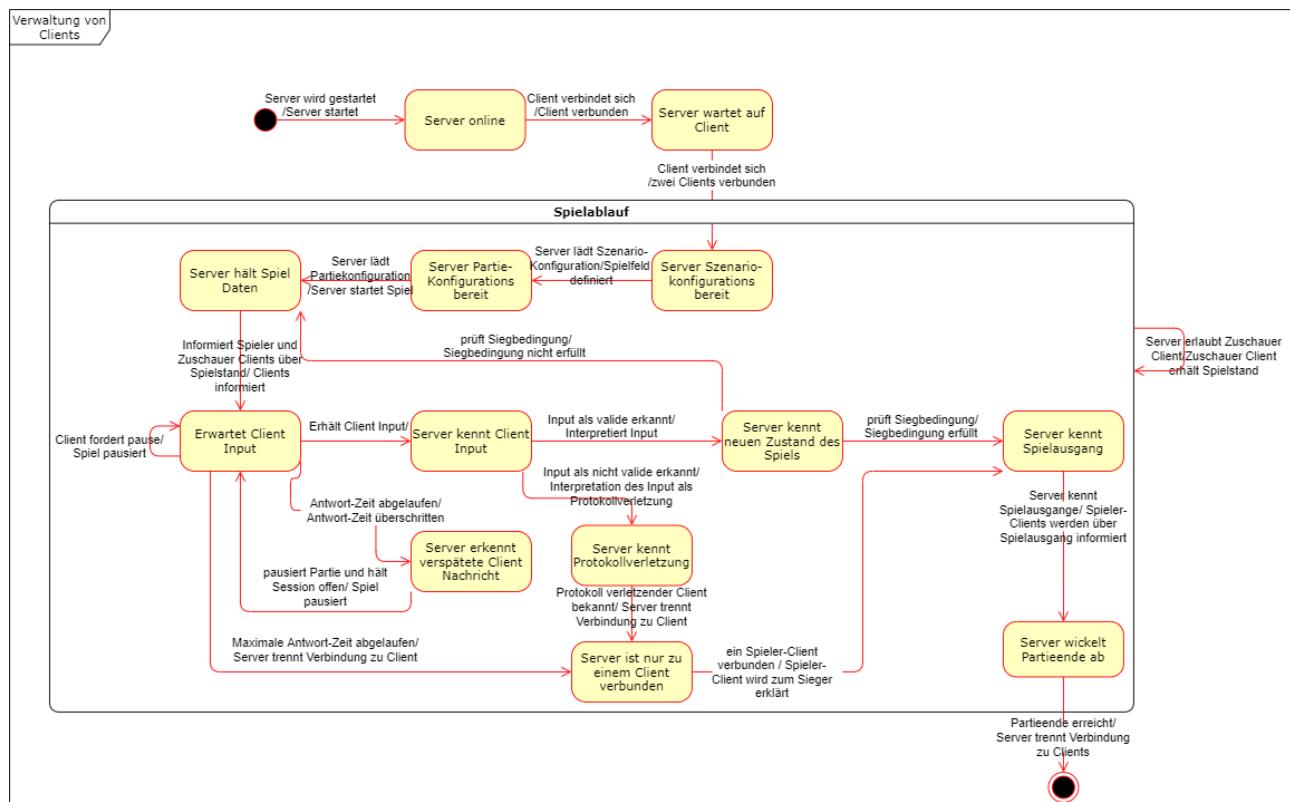


Abbildung 8: Zustandsdiagramm für die Verwaltung von Clients seitens des Servers

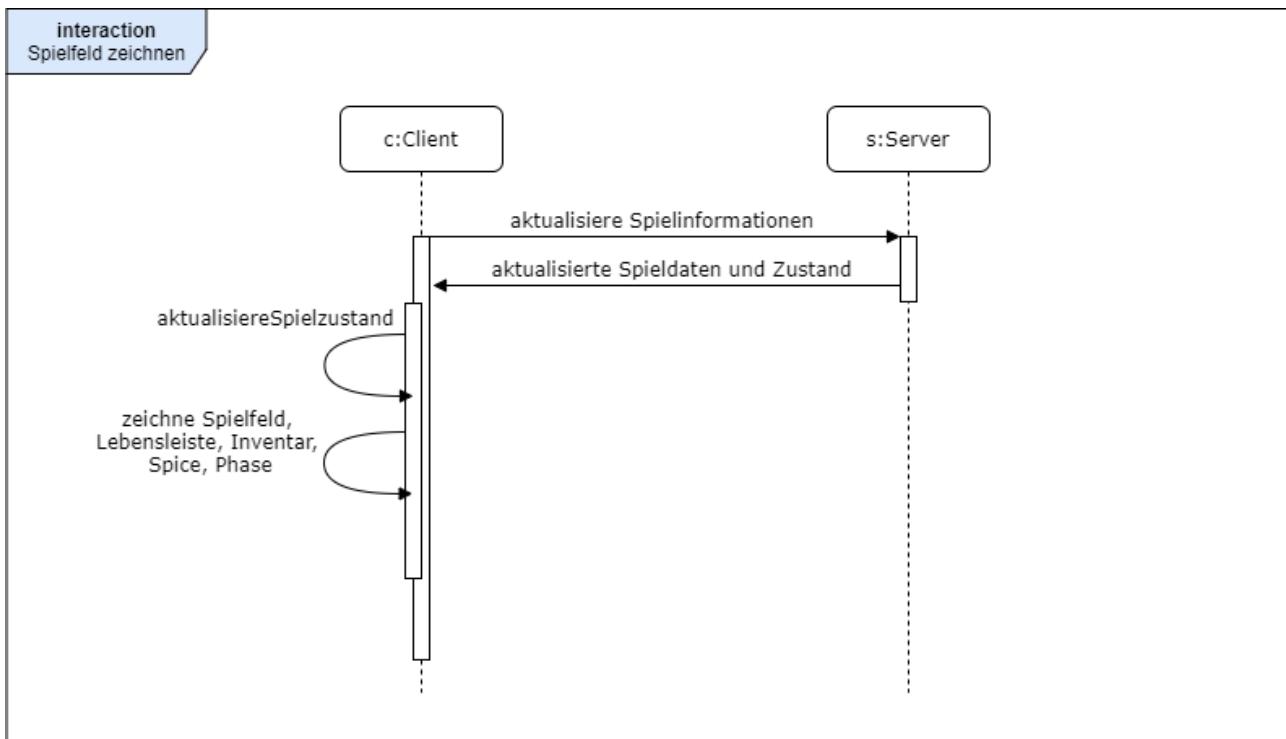


Abbildung 9: Sequenzdiagramm für Aktualisierung des Servers und Anzeige des neuen Spielstands auf dem Benutzer-Client

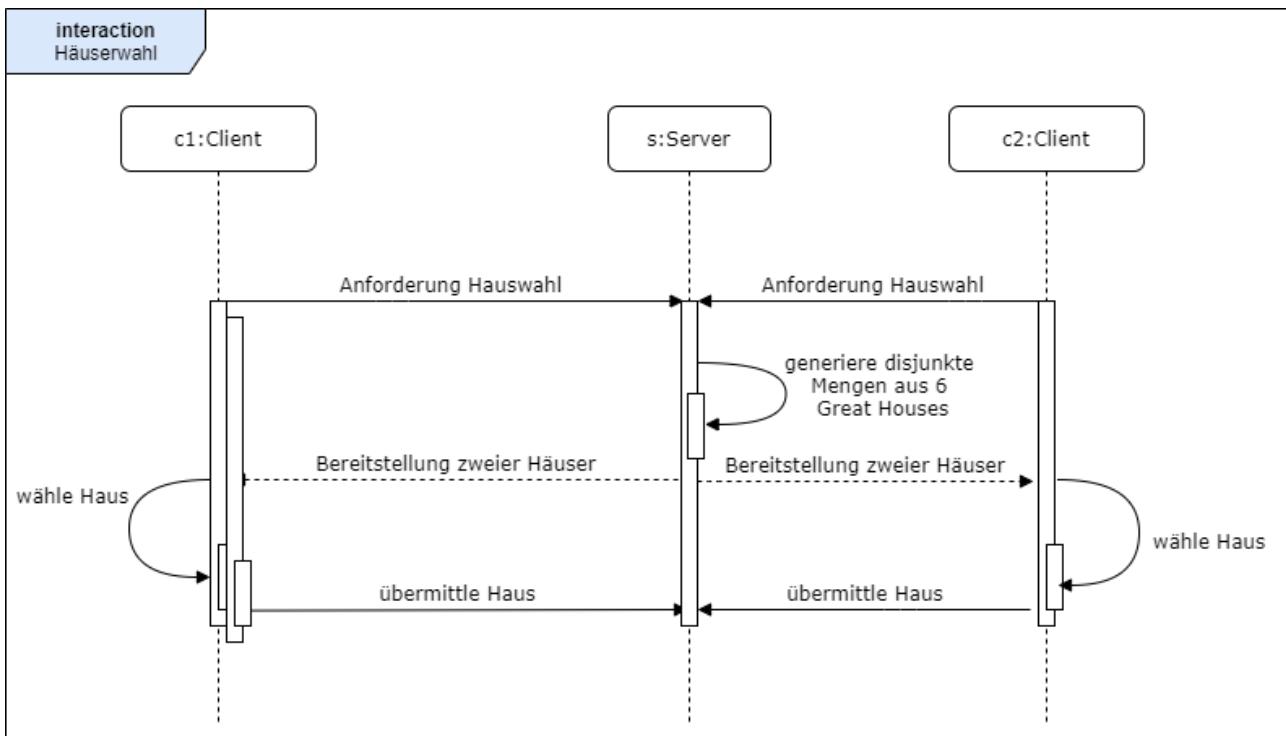


Abbildung 10: Sequenzdiagramm für die Great House Wahl vor Partiebeginn

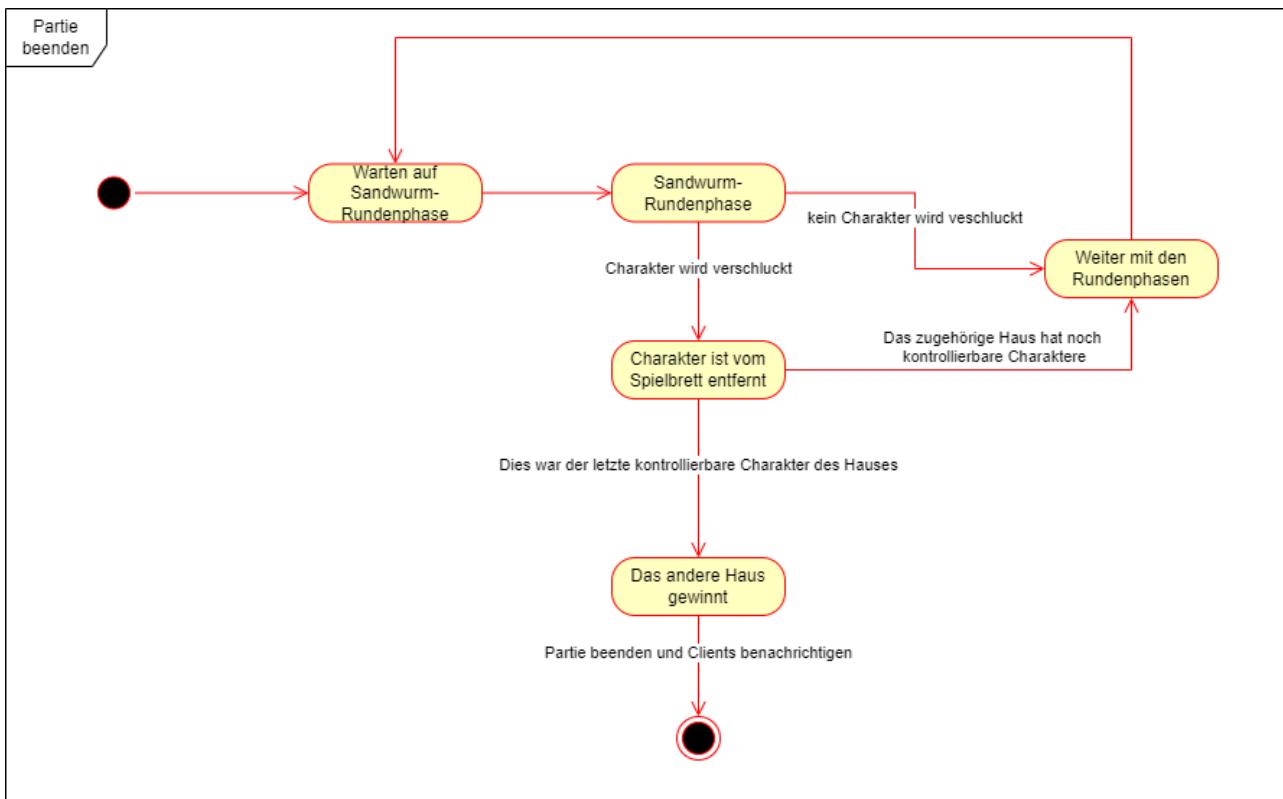


Abbildung 11: Zustandsdiagramm für Beenden der Partie

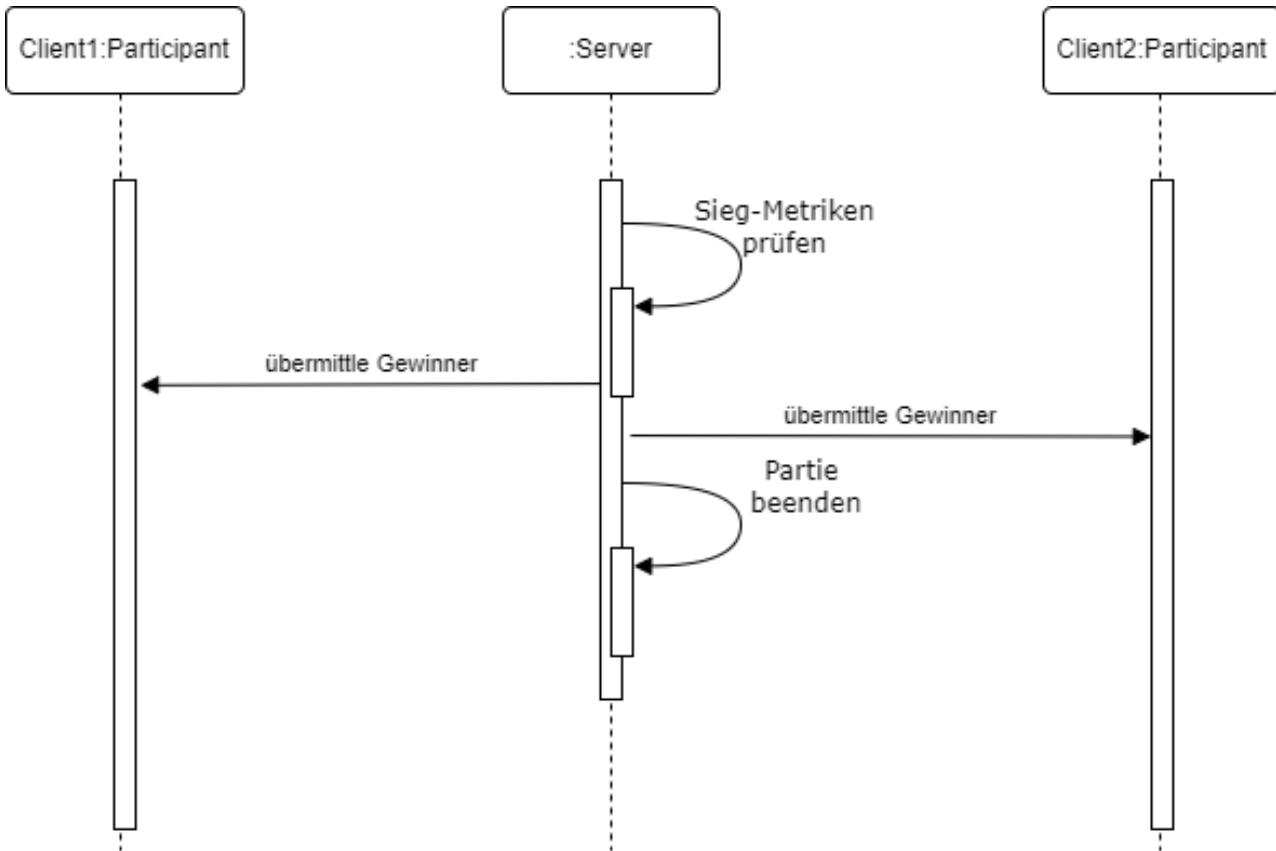


Abbildung 12: Sequenzdiagramm für Beenden der Partie

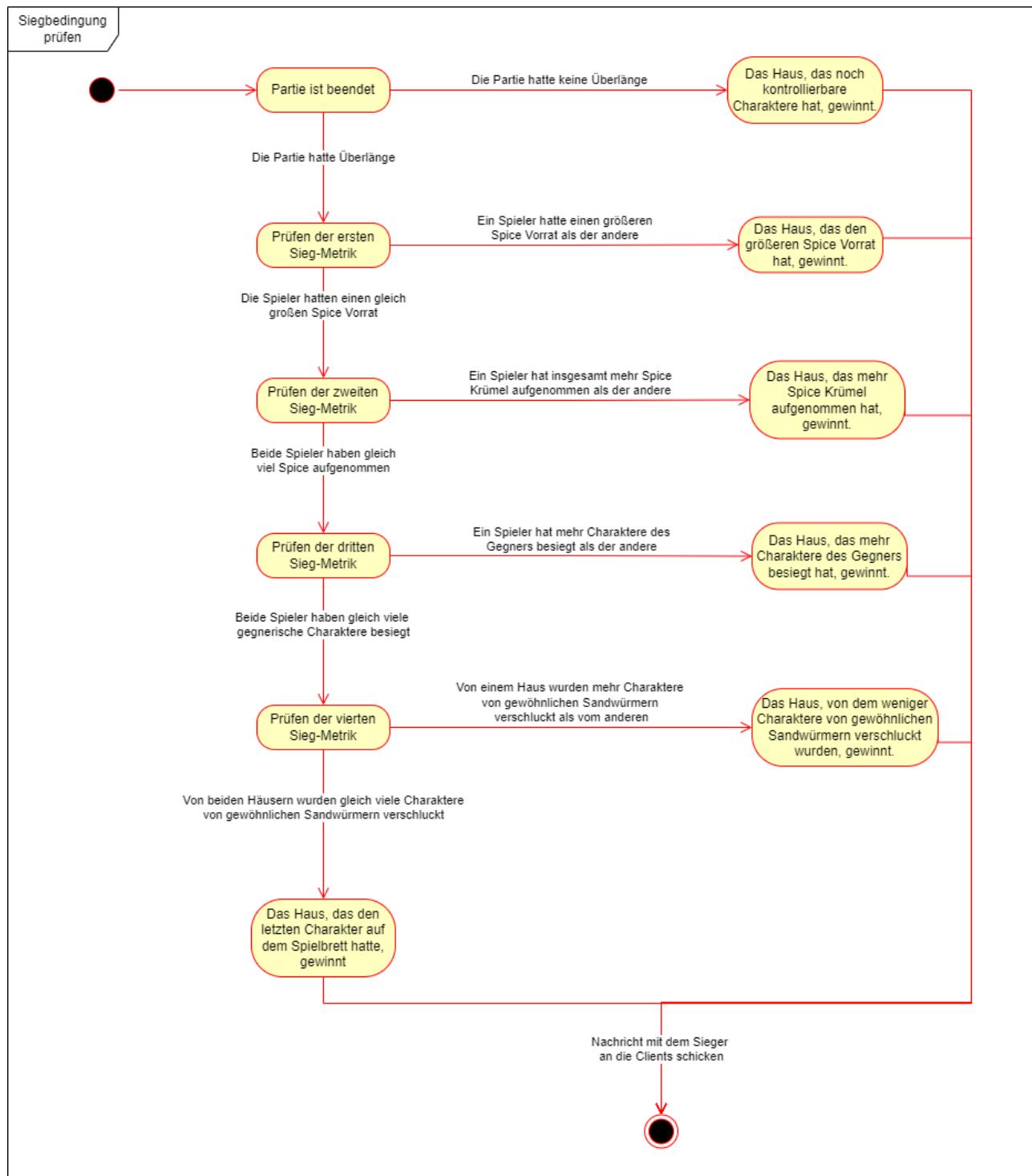


Abbildung 13: Zustandsdiagramm für Prüfen der Siegbedingungen

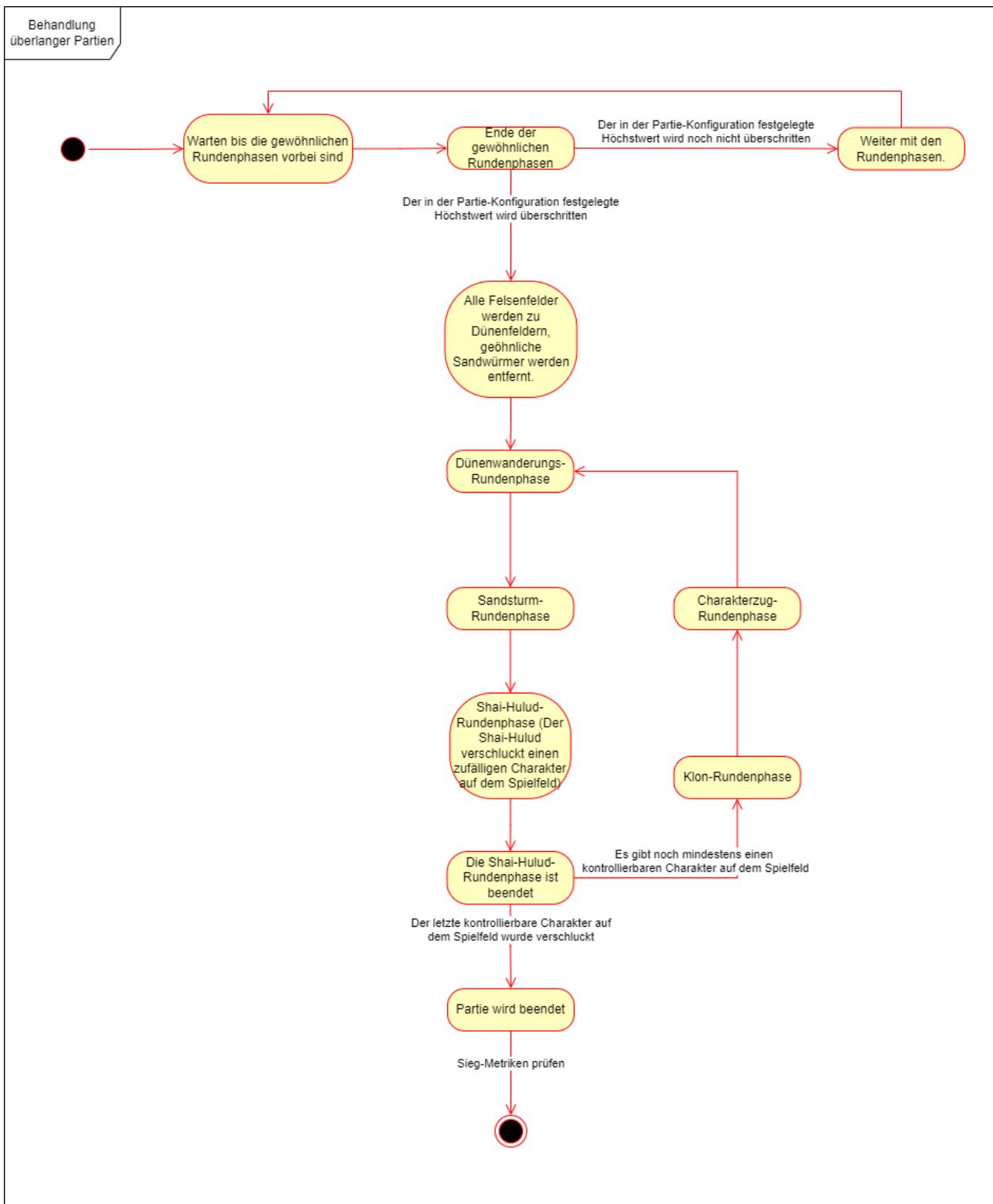


Abbildung 14: Zustandsdiagramm für Behandlung überlanger Partien

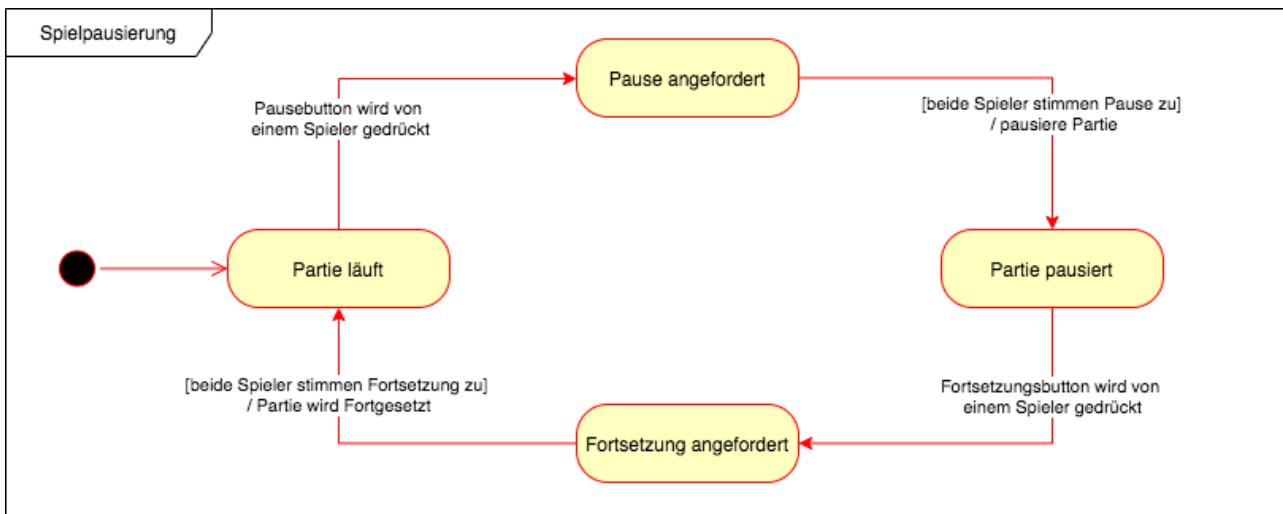


Abbildung 15: Zustandsdiagramm für den Ablauf eines Pausierungs- und Wiederaufnahmeantrags für eine Partie von einem menschlichen Benutzer

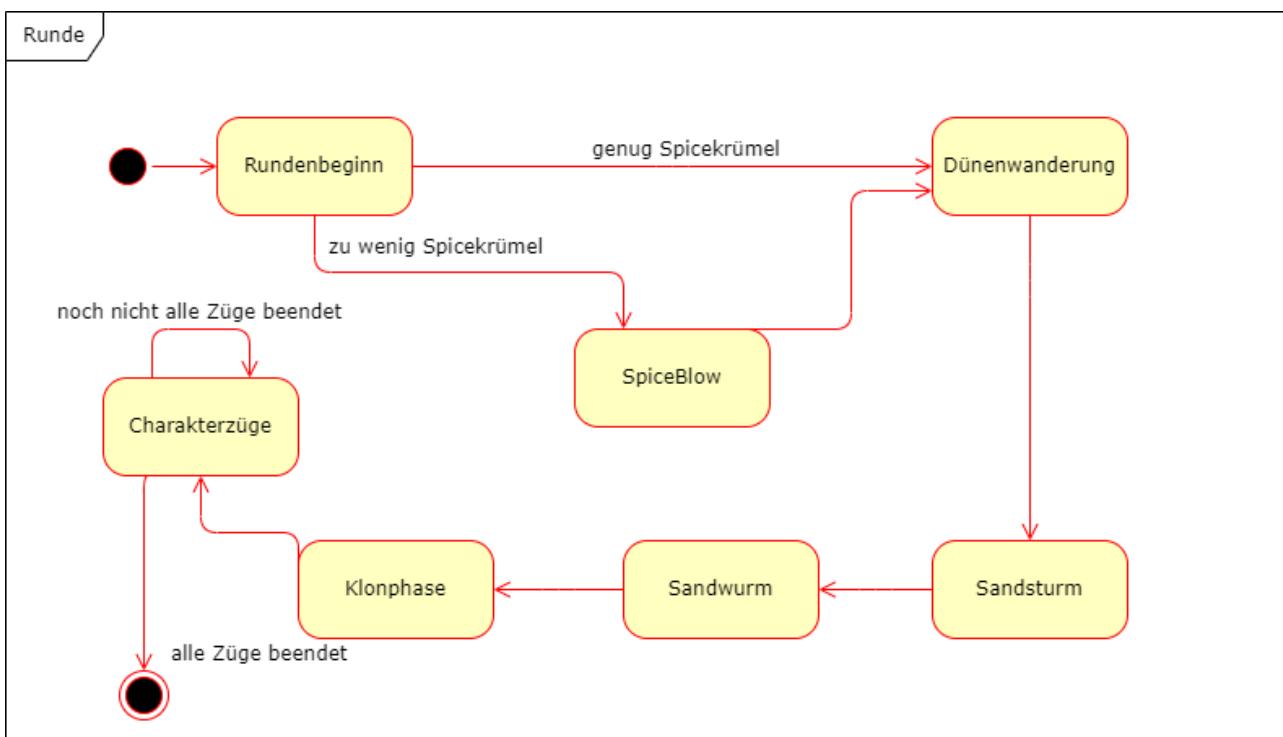


Abbildung 16: Zustandsdiagramm für den Anwendungsfall Abhandlung der Runde

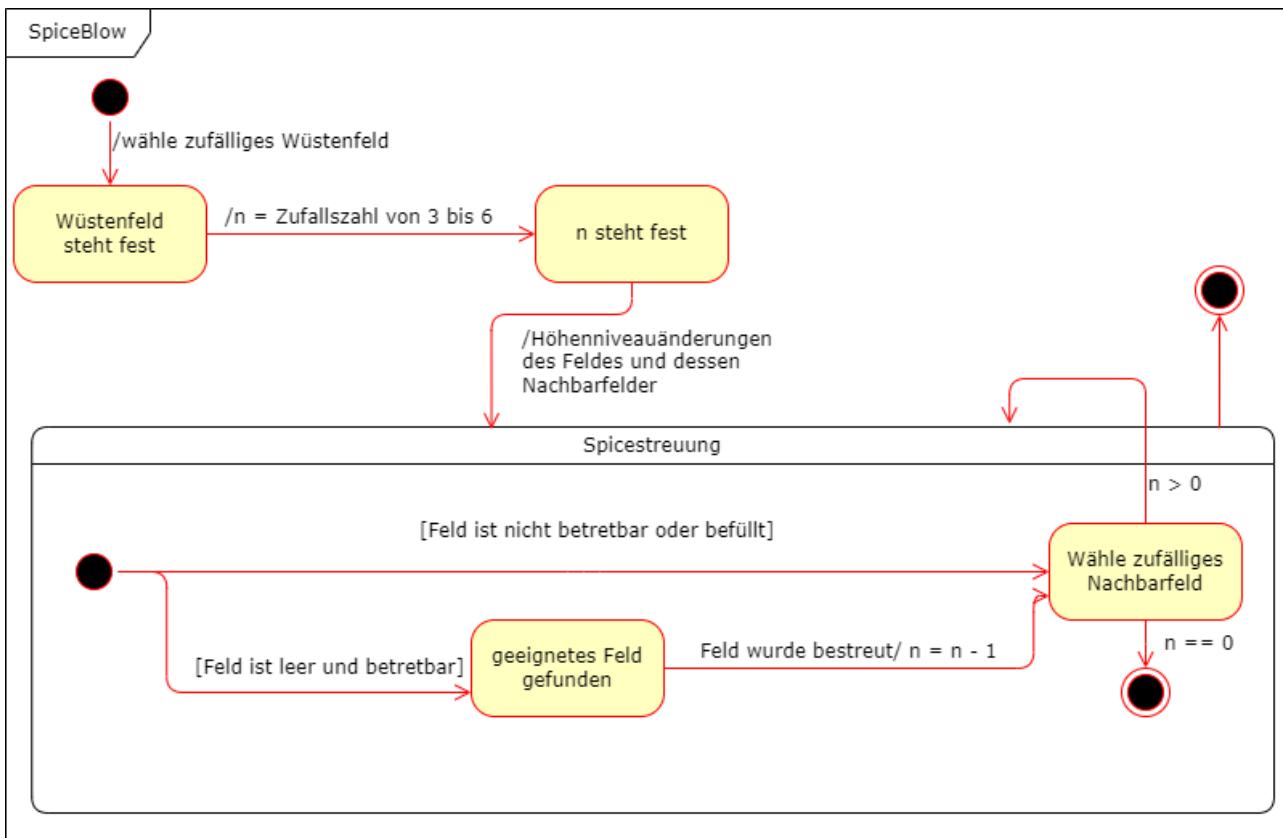


Abbildung 17: Zustandsdiagramm für den Anwendungsfall SpiceBlow

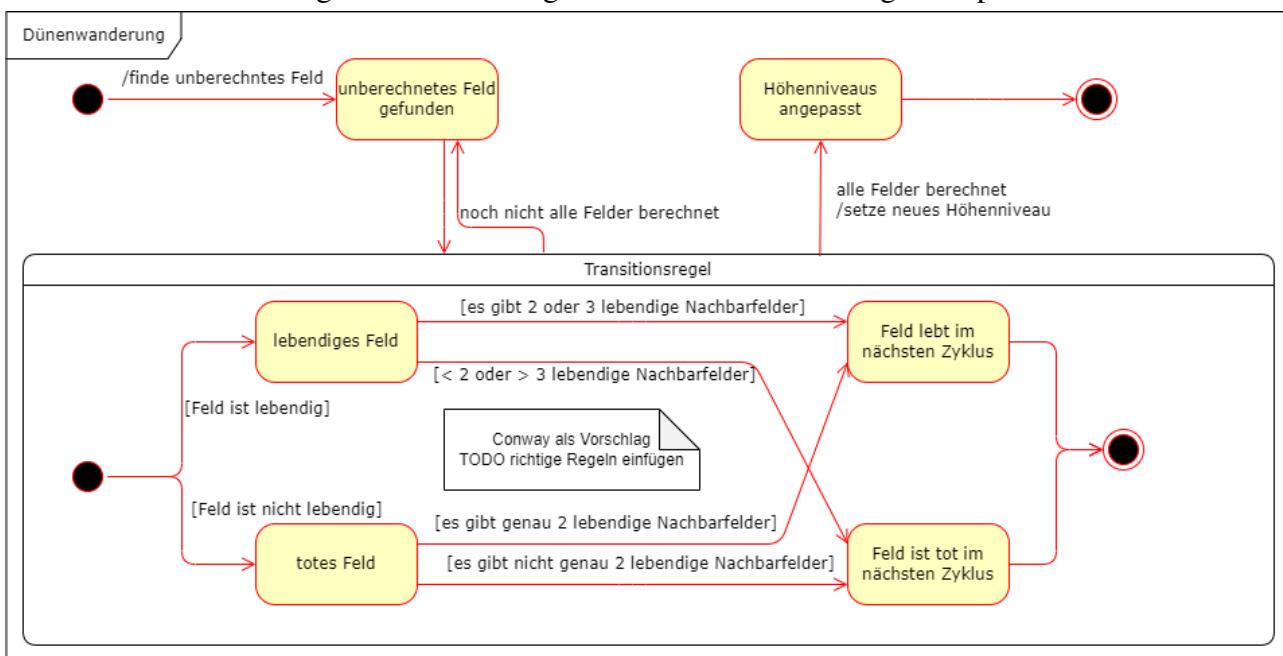


Abbildung 18: Zustandsdiagramm für den Anwendungsfall Dünenwanderung

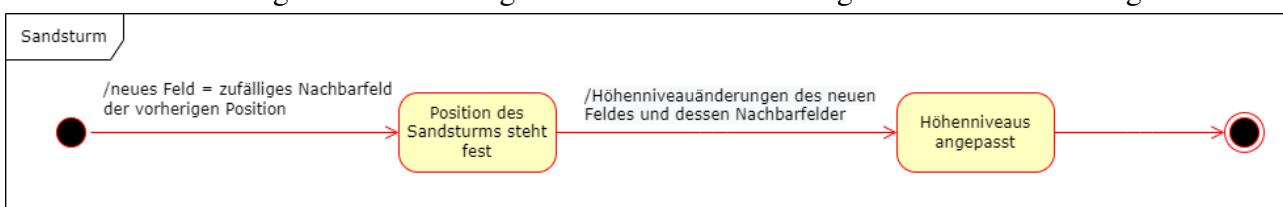


Abbildung 19: Zustandsdiagramm für den Anwendungsfall Sandsturm

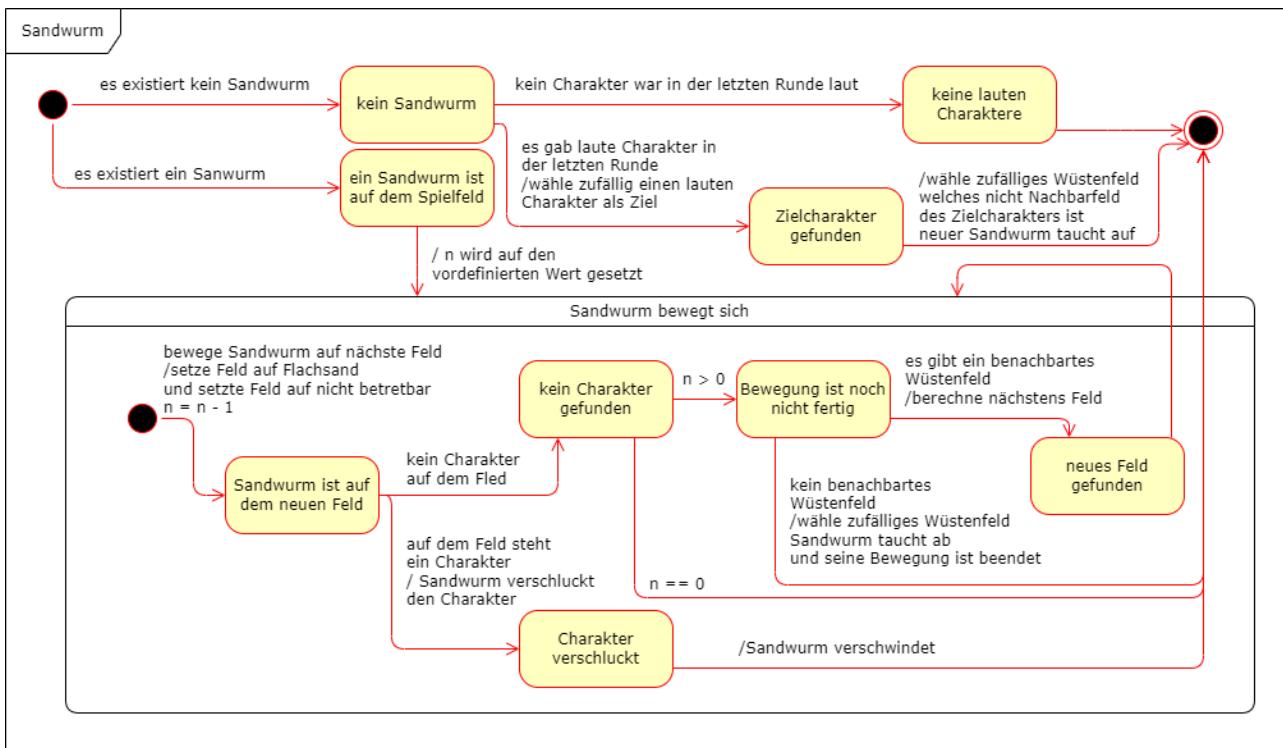


Abbildung 20: Zustandsdiagramm für den Anwendungsfall Sandwurm

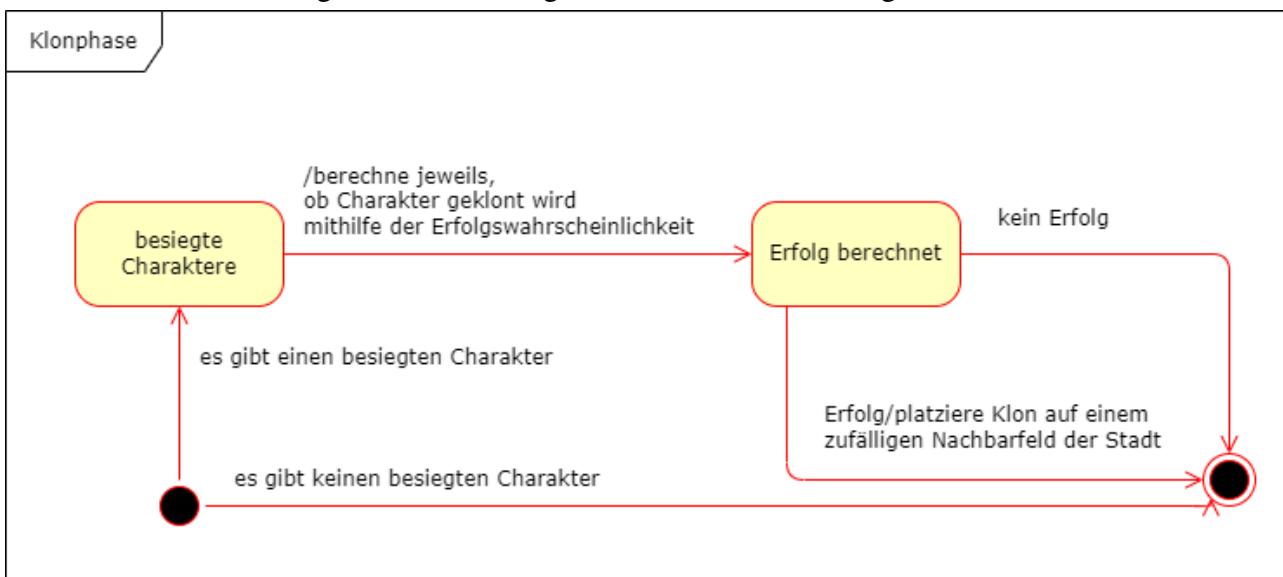


Abbildung 21: Zustandsdiagramm für den Anwendungsfall Klonphase

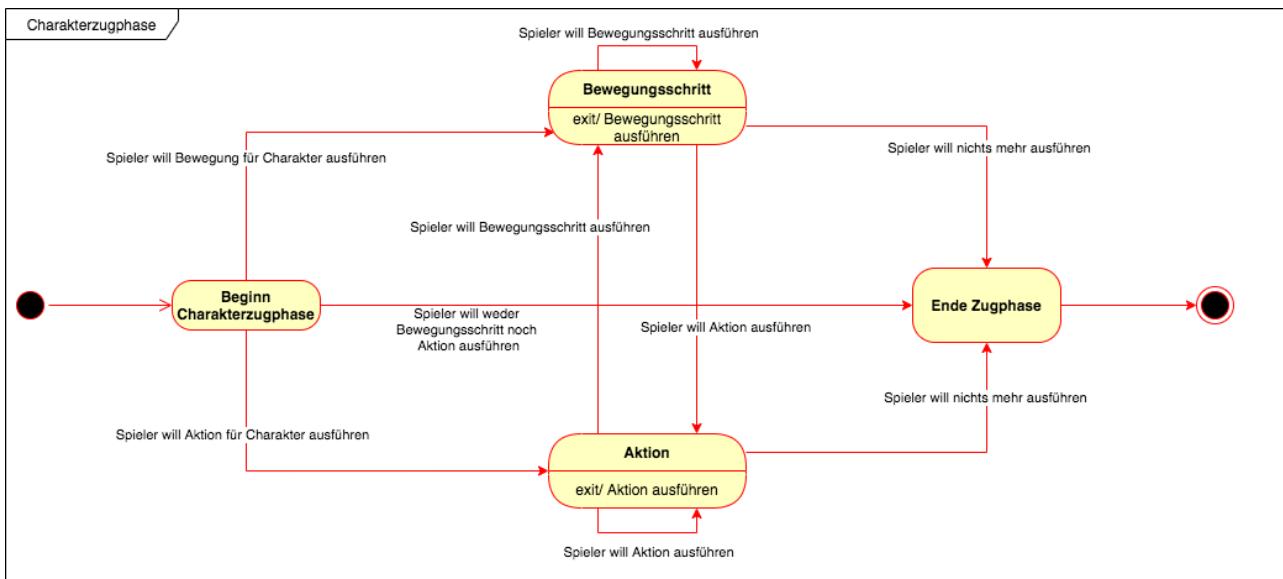


Abbildung 22: Zustandsdiagramm für den Ablauf der Phase des Charakterzugs

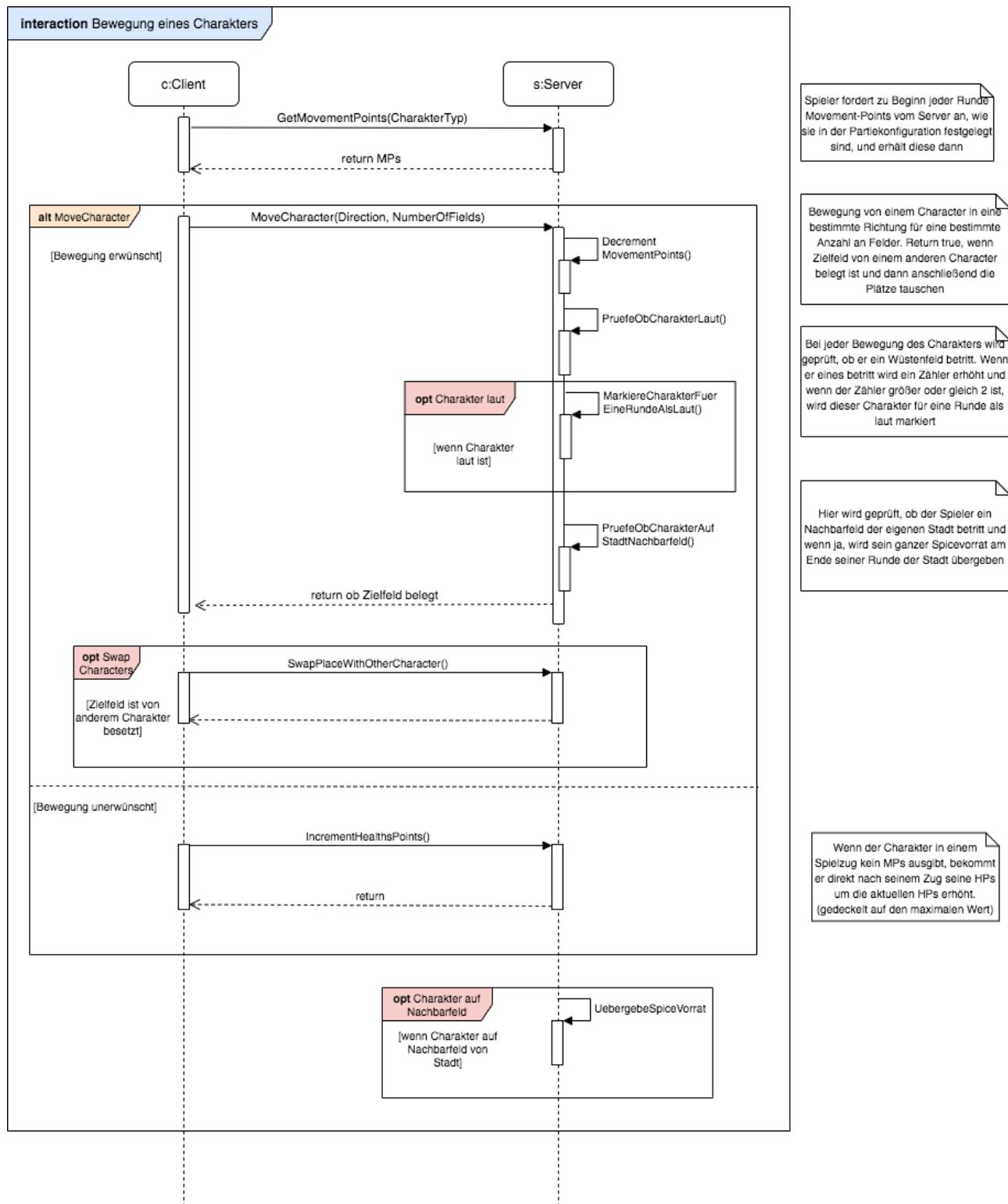


Abbildung 23: Sequenzdiagramm für den Vorgang, wenn ein Charakter bewegt werden soll

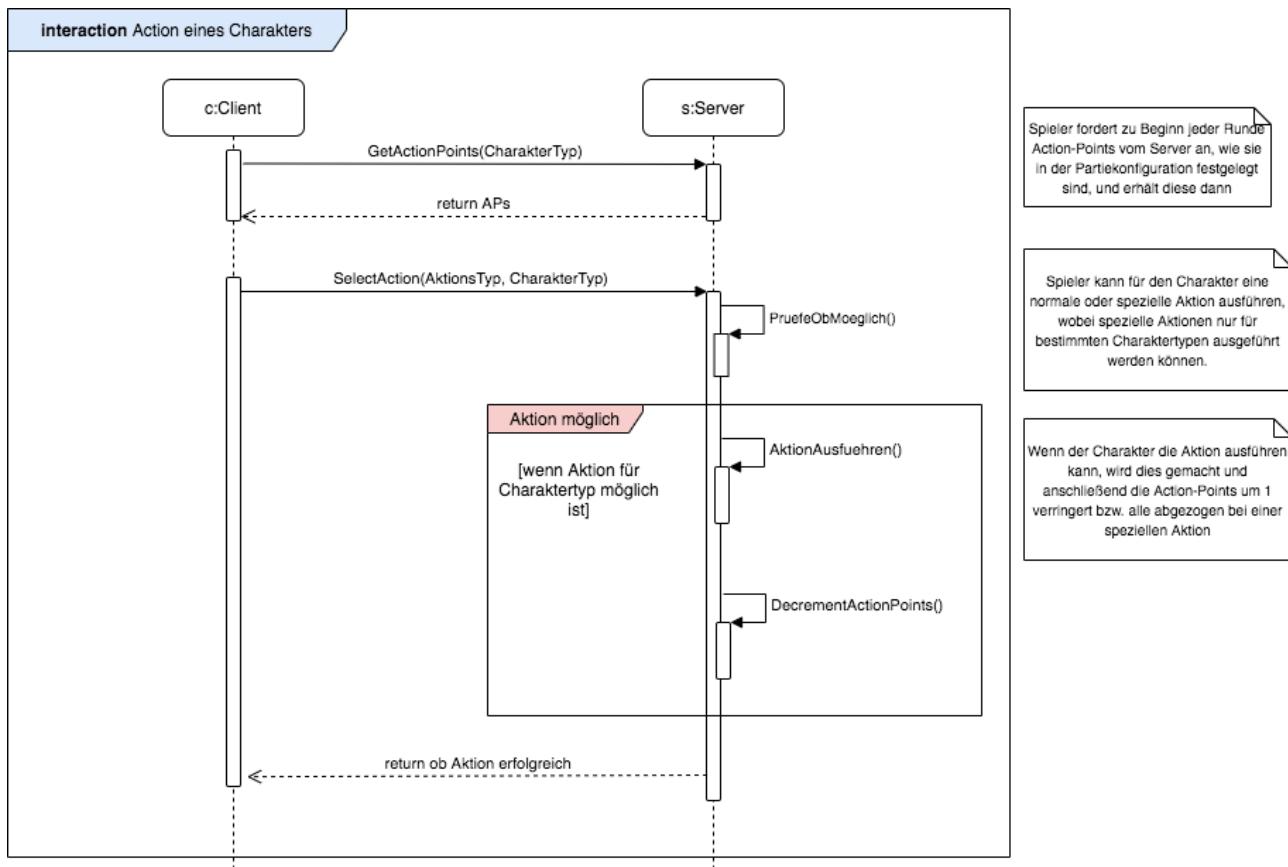


Abbildung 24: Sequenzdiagramm für den Vorgang, wenn ein Charakter eine Aktion ausführen soll

2.3 Funktionale Anforderungen

Dieser Abschnitt enthält alle funktionalen Systemanforderungen, die die grundlegenden Aktionen des Softwaresystems spezifizieren.

Dabei gilt für die Priorisierung, dass eine Anforderung eine Priorität von 1 (sehr gering, kann) bis 5 (essenziell wichtig, muss unbedingt) bekommen kann.

Komponenten und Architektur

ID	FA1
TITEL:	Architektur
BESCHREIBUNG:	Das Spiel DESERTS OF DUNE ist eine verteilte Anwendung mit der Client-Server-Architektur.
BEGRÜNDUNG	Damit hat man das Spiel in Instanzen aufteilen, die das Spiel bereit stellen und organisieren und Instanzen, die es spielen. Außerdem können somit zwei Spieler auf unterschiedlichen Systemen gegeneinander spielen
ABHÄNGIGKEITEN	FA2 → p. 38
PRIORITÄT	5
AKTEUR	Tutor → p. 18

ID	FA2
TITEL:	Komponenten des Spiels
BESCHREIBUNG:	Das gesamte System besteht aus vier Komponenten: Server, Benutzer-Client, KI-Client und Editor
BEGRÜNDUNG	Die Spiellogik soll in diese vier Komponenten aufgeteilt werden als Vorgabe vom Auftraggeber
ABHÄNGIGKEITEN	FA18 → p. 11 ³ , FA3 → p. 39, FA4 → p. 39, FA7 → p. 40, FA8 → p. 40
PRIORITÄT	5
AKTEUR	Tutor → p. 18

ID	FA3
TITEL:	Server
BESCHREIBUNG:	Der Server hält die Anwendungslogik zur Verwaltung von Clients, Partien und zur Umsetzung der Spielregeln.
BEGRÜNDUNG	Der Server soll die gesamte Partie managen und abwickeln, weil diese Aufgabe zentral erledigt werden soll.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17, Teilnehmer → p. 18

ID	FA4
TITEL:	Benutzer-Client
BESCHREIBUNG:	Der Benutzer-Client ermöglicht es einem einzelnen menschlichen Benutzer, als Spieler oder als Zuschauer an einer Partie teilzuhaben.
BEGRÜNDUNG	Der Benutzer braucht eine Anwendung, um eine Partie zu verfolgen oder daran teilzunehmen
ABHÄNGIGKEITEN	FA100 → p. 71, FA101 → p. 71
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17

ID	FA5
TITEL:	Teilnehmeranzahl an Partie
BESCHREIBUNG:	Es nehmen genau zwei gegeneinander spielende Spieler an einer Partie teil.
BEGRÜNDUNG	Das Spiel ist darauf ausgerichtet, dass immer zwei Spieler gegeneinander spielen.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18

ID	FA6
TITEL:	Beobachten von Partien
BESCHREIBUNG:	Das Spiel kann von beliebig vielen Benutzer-Clients beobachtet werden.
BEGRÜNDUNG	Es soll möglich sein ein beliebiges Spiel zu beobachten.
ABHÄNGIGKEITEN	FA101 → p. 71
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17

ID	FA7
TITEL:	KI-Client
BESCHREIBUNG:	Der KI-Client wird autonom durch eine Künstliche Intelligenz gesteuert und kann als Mitspieler an einer Partie teilnehmen.
BEGRÜNDUNG	Um auch einzelnen Benutzer ein spannendes Multiplayer-Erlebnis bieten zu können muss es einen KI-Client geben, welcher gegen diese Benutzer antreten kann.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17

ID	FA8
TITEL:	Editor
BESCHREIBUNG:	Mit dem Editor kann der Spieler das Spiel konfigurieren
BEGRÜNDUNG	Dem Nutzer muss es möglich sein, die Konfiguration für ein Spiel anzupassen.
ABHÄNGIGKEITEN	FA124 → p. 81, FA126 → p. 81, FA121 → p. 80
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17

Netzwerkkommunikation und Nachrichtenprotokoll

ID	FA9
TITEL:	Netzwerkprotokoll
BESCHREIBUNG:	Die Komponenten verwenden für die Kommunikation untereinander ein Netzwerkprotokoll. Das Netzwerkprotokoll ist ein WebSocket-Protokoll.
BEGRÜNDUNG	Die Komponenten müssen nach einem Standard kommunizieren und das WebSocket-Protokoll ist ein etabliertes, gut funktionierendes Protokoll
ABHÄNGIGKEITEN	FA10 → p. 40
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	FA10
TITEL:	Kodierung der Netzwerknachrichten
BESCHREIBUNG:	Die über die WebSocket-Verbindung ausgetauschten Textstrings sind im UTF-8 Format kodiert
BEGRÜNDUNG	Die Textstrings müssen aufgrund des Websocket-Protokoll kodiert werden und UTF-8 ist ein Standardformat
ABHÄNGIGKEITEN	FA9 ^{→ p. 40}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19}

ID	FA11
TITEL:	Nachrichtenformat des Spielprotokolls
BESCHREIBUNG:	Die über die WebSocket-Verbindung ausgetauschten Textstrings enthalten die Nachrichten des Spielprotokolls. Diese Nachrichten sind in dem Format JSON formatiert.
BEGRÜNDUNG	Die Textstrings müssen aufgrund des Websocket-Protokoll kodiert werden und UTF-8 ist ein Standardformat.
ABHÄNGIGKEITEN	FA9 ^{→ p. 40} , FA12 ^{→ p. 41}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19}

ID	FA12
TITEL:	Spielprotokoll
BESCHREIBUNG:	Das Spielprotokoll wird durch das Standardisierungskomitee definiert.
BEGRÜNDUNG	Es muss ein fest definiertes Spielprotokoll verwendet werden, um eine Standardisierung zu erhalten. Dieses Protokoll muss teamübergreifend erstellt werden, da es mehrere Teams verwenden.
ABHÄNGIGKEITEN	FA9 ^{→ p. 40} , FA11 ^{→ p. 41}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19}

Szenarios

ID	FA13
TITEL:	Sichtbarkeit des Spielfeldes
BESCHREIBUNG:	Alle Spieler können zu jedem Zeitpunkt das gesamte Spielfeld sehen.
BEGRÜNDUNG	Die Spieler sollen zu jedem Zeitpunkt des Spiels sämtliche Informationen über den Status des Spiels haben, weil das Spiel ein Spiel mit vollständiger Information sein soll.
ABHÄNGIGKEITEN	FA15 → p. 42
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17, Teilnehmer → p. 18

ID	FA14
TITEL:	Sichtbarkeit Zustand und Charakterinventar
BESCHREIBUNG:	Alle Spieler können zu jedem Zeitpunkt alle Charaktere und deren Inventar sehen.
BEGRÜNDUNG	Die Spieler sollen zu jedem Zeitpunkt des Spiels sämtliche Informationen über den Status des Spiels haben, weil das Spiel ein Spiel mit vollständiger Information sein soll.
ABHÄNGIGKEITEN	FA35 → p. 48
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17, Teilnehmer → p. 18

ID	FA15
TITEL:	Spielbrett
BESCHREIBUNG:	Das Spielbrett ist ein rechteckiges kartesisches Raster aus $x * y$ Feldern. Dieses Spielbrett inklusive aller Felder und Charakter repräsentiert das Szenario.
BEGRÜNDUNG	Alle Spielbretter sollen rechteckig sein, weil das die Berechnung von Zügen vereinfacht, und die einzelnen Felder sollen klar unterscheidbar sein.
ABHÄNGIGKEITEN	FA17 → p. 42, FA16 → p. 42
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17, Teilnehmer → p. 18

ID	FA16
TITEL:	Erstellung Spielbrett
BESCHREIBUNG:	Das Spielbrett wird als Szenario im Editor erstellt und kann dann vom Server geladen werden
BEGRÜNDUNG	Das Spielbrett muss irgendwie erstellt oder definiert werden können.
ABHÄNGIGKEITEN	FA15 → p. 42, FA126 → p. 81, FA121 → p. 80
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17, Teilnehmer → p. 18

ID	FA17
TITEL:	Feld
BESCHREIBUNG:	Ein Feld ist ein Teil auf dem Spielbrett und wird durch eine Koordinate positioniert.
BEGRÜNDUNG	Felder auf dem Spielbrett sollen genau einer Koordinate zugeordnet werden.
ABHÄNGIGKEITEN	FA15 → p. 42
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA18
TITEL:	Eigenschaften von Feldern
BESCHREIBUNG:	Ein Feld hat die Eigenschaft Feldart, Betretbarkeit und ein Höhenniveau
BEGRÜNDUNG	Felder müssen spezifische Eigenschaften haben, um den Spielverlauf durch diese Eigenschaften zu beeinflussen.
ABHÄNGIGKEITEN	FA17 → p. 42, FA21 → p. 43, FA19 → p. 43, FA20 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA19
TITEL:	Feldeigenschaft Betretbarkeit
BESCHREIBUNG:	Die Eigenschaft Betretbarkeit besagt, ob ein Charakter ein Feld betreten kann oder nicht. Wenn dieses betretbare Feld durch ein Objekt blockiert ist, dann heißt es <i>besetzt</i> , ansonsten heißt es <i>frei</i>
BEGRÜNDUNG	Es muss klar definiert sein, ob ein spezifisches Feld durch einen Charakter betreten werden kann oder nicht und ob es frei ist.
ABHÄNGIGKEITEN	FA18 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA20
TITEL:	Feldeigenschaft Höhenniveau
BESCHREIBUNG:	Die Eigenschaft Höhenniveau gibt an, ob ein Feld <i>hoch</i> oder <i>niedrig</i> ist.
BEGRÜNDUNG	Es muss klar definiert sein, ob ein spezifisches Feld hoch oder niedrig ist, was bei der Bewegung und bei den Aktionen von Charakteren eine Rolle spielt
ABHÄNGIGKEITEN	FA18 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA21
TITEL:	Feldarten
BESCHREIBUNG:	Es gibt die verschiedenen Feldarten: <i>Stadt, Flachsand, Düne, Felsplateau, Gebirge</i> . Für jede Feldart sind die Eigenschaften <i>Betretbarkeit</i> und <i>Höhenniveau</i> definiert.
BEGRÜNDUNG	Jedes Feld muss eine definierte Feldart und damit einhergehende Parameter haben.
ABHÄNGIGKEITEN	FA19 ^{→ p. 43} , FA20 ^{→ p. 43}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18}

ID	FA22
TITEL:	Feldart Stadt
BESCHREIBUNG:	Die Stadt ist ein Spezialfeld. Auf dem Spielbrett gibt es zwei große Städte, Arrakeen und Carthag. Beiden Spielern gehört jeweils eine davon. Felder mit der Feldart Stadt sind nicht betretbar und haben ein hohes Höhenniveau.
BEGRÜNDUNG	Jede Feldart hat besondere Eigenschaften und die Parameter müssen wohldefiniert sein.
ABHÄNGIGKEITEN	FA21 ^{→ p. 43} , FA19 ^{→ p. 43} , FA20 ^{→ p. 43}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

ID	FA23
TITEL:	Feldart Flachsand
BESCHREIBUNG:	Flachsand ist ein Wüstenfeld, welches flach und niedrig gelegen ist. Felder der Feldart Flachsand sind betretbar und haben ein niedriges Höhenniveau.
BEGRÜNDUNG	Jede Feldart hat besondere Eigenschaften und die Parameter müssen wohldefiniert sein.
ABHÄNGIGKEITEN	FA21 ^{→ p. 43} , FA19 ^{→ p. 43} , FA20 ^{→ p. 43}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

ID	FA24
TITEL:	Feldart Düne
BESCHREIBUNG:	Die Düne ist ein Wüstenfeld, welches hügelig und höher gelegen ist. Felder der Feldart Düne sind betretbar und haben ein hohes Höhenniveau.
BEGRÜNDUNG	Jede Feldart hat besondere Eigenschaften und die Parameter müssen wohldefiniert sein.
ABHÄNGIGKEITEN	FA21 → p. 43, FA19 → p. 43, FA20 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA25
TITEL:	Feldart Felsplateau
BESCHREIBUNG:	Das Felsplateau ist ein Felsenfeld, welches flach und niedrig gelegen ist. Felder der Feldart Felsplateau sind betretbar und haben ein niedriges Höhenniveau.
BEGRÜNDUNG	Jede Feldart hat besondere Eigenschaften und die Parameter müssen wohldefiniert sein.
ABHÄNGIGKEITEN	FA21 → p. 43, FA19 → p. 43, FA20 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA26
TITEL:	Feldart Gebirge
BESCHREIBUNG:	Das Gebirge ist ein Felsenfeld, welches hoch gelegen und unpassierbar ist. Felder der Feldart Gebirge sind nicht betretbar und haben ein hohes Höhenniveau.
BEGRÜNDUNG	Jede Feldart hat besondere Eigenschaften und die Parameter müssen wohldefiniert sein.
ABHÄNGIGKEITEN	FA21 → p. 43, FA19 → p. 43, FA20 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA27
TITEL:	Entfernung Felder
BESCHREIBUNG:	Die Entfernung zwischen zwei Feldern A und B ist definiert als die minimale Anzahl von aufeinanderfolgenden Schritten auf alle 8 Nachbarfelder (gemäß der Moore Neighborhood-Definition), um von A nach B zu gelangen.
BEGRÜNDUNG	Der Abstand zwischen Feldern muss klar definiert sein.
ABHÄNGIGKEITEN	FA17 → p. 42
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

Great Houses

ID	FA28
TITEL:	Great Houses
BESCHREIBUNG:	Im Spiel gibt es sechs Great Houses, das heißt große Adelshäuser, die miteinander konkurrieren. Jedes der Häuser hat genau einen Namen und genau eine Farbe und besteht aus einer Reihe von Charakteren, die bei FA35 → p. 48 genauer beschrieben werden
BEGRÜNDUNG	Die Great Houses sollen farblich, namentlich und durch die Eigenschaften ihrer Charaktere unterscheidbar sein und dem Spieler Abwechselung bieten und verschiene Strategien in das Spiel bringen.
ABHÄNGIGKEITEN	FA35 → p. 48
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA29
TITEL:	Haus Corrino
BESCHREIBUNG:	Dieses Great House hat den Namen <i>House Corrino</i> , die Farbe <i>Gold</i> und die folgenden Charaktere: Emperor Shaddam IV Corrino (Noble), Princess Irulan Corrino (Bene Gesserit), Count Hasimir Fenring (Mentat), Lady Margot Fenring (Bene Gesserit), Reverend Mother Gaius Helen Mohiam (Bene Gesserit) und Captain Aramsham (Fighter).
BEGRÜNDUNG	Die Great Houses sollen farblich, namentlich und durch die Eigenschaften ihrer Charaktere unterscheidbar sein.
ABHÄNGIGKEITEN	FA28 → p. 46, FA40 → p. 50, FA41 → p. 50, FA42 → p. 50, FA43 → p. 51
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA30
TITEL:	Haus Atreides
BESCHREIBUNG:	Dieses Great House hat den Namen <i>House Atreides</i> , die Farbe <i>Grün</i> und die folgenden Charaktere: Duke Leto Atreides (Noble), Paul Atreides (Noble), Lady Jessica (Bene Gesserit), Thufir Hawat (Mentat), Gurney Halleck (Fighter) und Space Pug, Duke Letos tapferer Mopshund (Fighter).
BEGRÜNDUNG	Die Great Houses sollen farblich, namentlich und durch die Eigenschaften ihrer Charaktere unterscheidbar sein.
ABHÄNGIGKEITEN	FA28 ^{→ p. 46} , FA40 ^{→ p. 50} , FA41 ^{→ p. 50} , FA42 ^{→ p. 50} , FA43 ^{→ p. 51}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

ID	FA31
TITEL:	Haus Harkonnen
BESCHREIBUNG:	Dieses Great House hat den Namen <i>House Harkonnen</i> , die Farbe <i>Rot</i> und die folgenden Charaktere: Baron Vladimir Harkonnen (Noble), Count Glossu Beast Rabban (Fighter), Feyd-Rautha Rabban (Fighter), Piter De Vries (Mentat), Iakin Nefud (Fighter) und Pet Spider (Fighter).
BEGRÜNDUNG	Die Great Houses sollen farblich, namentlich und durch die Eigenschaften ihrer Charaktere unterscheidbar sein.
ABHÄNGIGKEITEN	FA28 ^{→ p. 46} , FA40 ^{→ p. 50} , FA41 ^{→ p. 50} , FA42 ^{→ p. 50} , FA43 ^{→ p. 51}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

ID	FA32
TITEL:	Haus Ordos
BESCHREIBUNG:	Dieses Great House hat den Namen <i>House Ordos</i> , die Farbe <i>Blau</i> und die folgenden Charaktere: Execatrix (Noble), The Speaker (Noble), Ammon (Mentat), Edric (Mentat), Roma Atani (Mentat) und Robot (Fighter).
BEGRÜNDUNG	Die Great Houses sollen farblich, namentlich und durch die Eigenschaften ihrer Charaktere unterscheidbar sein.
ABHÄNGIGKEITEN	FA28 ^{→ p. 46} , FA40 ^{→ p. 50} , FA41 ^{→ p. 50} , FA42 ^{→ p. 50} , FA43 ^{→ p. 51}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

ID	FA33
TITEL:	Haus Richese
BESCHREIBUNG:	Dieses Great House hat den Namen <i>House Richese</i> , die Farbe <i>Silber</i> und die folgenden Charaktere: Count Ilban Richese (Noble), Helena Richese (Noble), Haloa Rund (Mentat), Flinto Kinnis (Mentat), Tenu Chobyn (Mentat) und Yresk (Fighter).
BEGRÜNDUNG	Die Great Houses sollen farblich, namentlich und durch die Eigenschaften ihrer Charaktere unterscheidbar sein.
ABHÄNGIGKEITEN	FA28 → p. 46, FA40 → p. 50, FA41 → p. 50, FA42 → p. 50, FA43 → p. 51
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA34
TITEL:	Haus Vernius
BESCHREIBUNG:	Dieses Great House hat den Namen <i>House Vernius</i> , die Farbe <i>Violett</i> und die folgenden Charaktere: Earl Dominic Vernius (Noble), Lady Shando Vernius (Noble), Kailea Vernius (Noble), Tessia Vernius (Bene Gesserit), Rhombur Vernius (Fighter) und Bronso Vernius (Mentat).
BEGRÜNDUNG	Die Great Houses sollen farblich, namentlich und durch die Eigenschaften ihrer Charaktere unterscheidbar sein.
ABHÄNGIGKEITEN	FA28 → p. 46, FA40 → p. 50, FA41 → p. 50, FA42 → p. 50, FA43 → p. 51
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

Charaktere

ID	FA35
TITEL:	Charaktere
BESCHREIBUNG:	Charaktere haben bestimmte Eigenschaften, deren Werte (abhängig vom Charakter-Typ) in der Partiekonfiguration definiert sind. Sie haben einen <i>Namen</i> , ein <i>Haus</i> , welchem sie angehören, <i>Health Points</i> , <i>Movement Points</i> , <i>Angriffsschaden</i> und ein <i>Inventar</i> . Die Werte und das Inventar aller Charaktere sind für alle Spieler sichtbar.
BEGRÜNDUNG	Charaktere werden strategisch durch ihre Eigenschaften und deren Werte unterschieden.
ABHÄNGIGKEITEN	FA124 → p. 81, FA14 → p. 42
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA36
TITEL:	Eigenschaft Health Points
BESCHREIBUNG:	Charaktere haben sogenannte Health Points, die angeben, wie viel Schaden der Charakter noch bekommen kann, bevor er stirbt.
BEGRÜNDUNG	Charaktere brauchen Lebenspunkte, um zu entscheiden, ob sie noch leben oder nicht.
ABHÄNGIGKEITEN	FA35 → p. 48
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA37
TITEL:	Tod eines Charakters
BESCHREIBUNG:	Wenn der Wert der Health Points eines Charakters auf 0 sinkt, dann ist der Charakter besiegt. Anschließend wird dieser von der Karte entfernt und alle Spicekrümel nach der Spicekrümel-Logik auf dem Spielfeld verteilt.
BEGRÜNDUNG	Es muss definiert sein, was nach dem Tod eines Charakters inklusive seines Spice passiert.
ABHÄNGIGKEITEN	FA36 → p. 48, FA63 → p. 58
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA38
TITEL:	Heilung von Health-Points
BESCHREIBUNG:	Ein Charakter, welcher in seinem Zug keine Movement-Points ausgibt, bekommt am Ende des Zuges seine Health-Points um seine Heilungs-Health Points erhöht.
BEGRÜNDUNG	Es muss klar definiert sein, wann ein Charakter sich heilt.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA39
TITEL:	Charakter-Typen
BESCHREIBUNG:	Der Charakter-Typ bestimmt die <i>Health Points</i> , <i>Heilungs-Health Points</i> , <i>Movement Points</i> , <i>Action Points</i> , <i>Inventargröße</i> und den <i>Angriffsschaden</i> des Charakters. Zudem ermöglicht der Charakter-Typ, dem Charakter eventuell spezielle Aktionen ausführen zu können.
BEGRÜNDUNG	Charaktere werden strategisch durch ihre Eigenschaften und deren Werte unterschieden.
ABHÄNGIGKEITEN	FA35 → p. 48, FA55 → p. 55, FA44 → p. 51
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA40
TITEL:	Charakter-Typ Noble
BESCHREIBUNG:	Noble ist ein hochwohlgeborener Adliger. Dafür sind seine Eigenschaften nur durchschnittlich. Die genauen Werte seiner Eigenschaften sind in der Partiekonfiguration zu konfigurieren.
BEGRÜNDUNG	Die Eigenschaften der Charakter-Typen werden in der Partiekonfiguration konfiguriert und zu einem späteren Zeitpunkt genauer festgelegt, um eine Einfluss-Balance zwischen den Charakter-Typen zu ermöglichen.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA41
TITEL:	Charakter-Typ Mentat
BESCHREIBUNG:	Mentat ist ein Charakter-Typ, der sehr schlau ist, und deshalb besonders effizient darin, Aktionen auszuführen und Spice zu sammeln. Er hält aber im Kampf nicht besonders viel aus. Die genauen Werte der Eigenschaften sind in der Partiekonfiguration zu konfigurieren.
BEGRÜNDUNG	Die Eigenschaften der Charakter-Typen werden in der Partiekonfiguration konfiguriert und zu einem späteren Zeitpunkt genauer festgelegt, um eine Einfluss-Balance zwischen den Charakter-Typen zu ermöglichen.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

ID	FA42
TITEL:	Charakter-Typ Bene Gesserit
BESCHREIBUNG:	Bene Gesserit ist ein Charakter-Typ, der schnell und wendig ist und eine hohe Heilungsrate besitzt. Die genauen Werte der Eigenschaften sind in der Partie-Konfiguration zu konfigurieren.
BEGRÜNDUNG	Die Eigenschaften der Charakter-Typen werden in der Partiekonfiguration konfiguriert und zu einem späteren Zeitpunkt genauer festgelegt, um eine Einfluss-Balance zwischen den Charakter-Typen zu ermöglichen.
ABHÄNGIGKEITEN	FA35 ^{→ p. 48} , FA39 ^{→ p. 49}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

ID	FA43
TITEL:	Charakter-Typ Fighter
BESCHREIBUNG:	Fighter ist ein Charakter-Typ, der im Kampf viel Schaden macht und viele Health-Points hat. Die genauen Werte der Eigenschaften sind in der Partiekonfiguration zu konfigurieren.
BEGRÜNDUNG	Die Eigenschaften der Charakter-Typen werden in der Partiekonfiguration konfiguriert und zu einem späteren Zeitpunkt genauer festgelegt, um eine Einfluss-Balance zwischen den Charakter-Typen zu ermöglichen.
ABHÄNGIGKEITEN	FA35 ^{→ p. 48} , FA39 ^{→ p. 49}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

ID	FA44
TITEL:	Stärken und Schwächen von Charakter-Typen
BESCHREIBUNG:	Charakter-Typen haben unterschiedliche Stärken und Schwächen ähnlich dem Schere-Stein-Papier Prinzip.
BEGRÜNDUNG	Kein Charakter-Typ soll alleine alles gut können, sie sollen sich gegenseitig ergänzen und effektiv gegeneinander sein.
ABHÄNGIGKEITEN	FA35 ^{→ p. 48}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18} , menschlicher Benutzer ^{→ p. 17}

Bewegung

ID	FA45
TITEL:	Bewegung von Charakteren
BESCHREIBUNG:	Zu Beginn seines Zuges bekommt jeder Charakter so viele Movement Points (MP), wie in der Partiekonfiguration für seinen Charakter-Typ festgelegt ist. Für einen Movement Point kann sich der Charakter auf ein betretbares Nachbarfeld bewegen.
BEGRÜNDUNG	Es muss klar definiert sein, wie weit und wohin sich Charaktere jede Runde bewegen dürfen.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA17 → p. 42, FA19 → p. 43, FA46 → p. 52
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA46
TITEL:	Bewegungsrichtung von Charakteren
BESCHREIBUNG:	Ein Charakter kann sich in horizontaler, vertikaler oder diagonaler Richtung bewegen.
BEGRÜNDUNG	Es muss klar definiert sein, in welche Richtungen sich ein Charakter-Typ bewegen darf.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA17 → p. 42, FA19 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA47
TITEL:	Drängeln von Charakteren
BESCHREIBUNG:	Bewegt sich ein Charakter auf ein Feld, auf dem bereits ein Charakter steht, tauschen die beiden Charaktere die Felder.
BEGRÜNDUNG	Die Interaktion zwischen einem Charakter, der bereits auf einem Feld steht und einem zweiten Charakter, welcher das Feld betritt, muss eindeutig definiert sein.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA17 → p. 42, FA21 → p. 43, FA19 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA48
TITEL:	Lautheit in der Wüste
BESCHREIBUNG:	Am Anfang seines Zuges gilt ein Charakter als leise. Charaktere werden als laut markiert, wenn sie mindestens zweimal aktiv ein Wüstenfeld in einem Zug betreten.
BEGRÜNDUNG	Es muss klar definiert werden, wann Charaktere als laut markiert werden.
ABHÄNGIGKEITEN	FA35 → p. 48
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA49
TITEL:	Abliefern von Spice
BESCHREIBUNG:	Ist ein Charakter auf einem Nachbarfeld seiner Stadt, so wird das Spice aus seinem Inventar direkt der Stadt übergeben und dem Spice-Vorrat des Hauses hinzugezählt. Das heißt der Charakter hat genau 0 Spice im Inventar und die Stadt genau die Menge mehr, die der Charakter vorher im Inventar hatte.
BEGRÜNDUNG	Es muss klar geregelt sein, wie Charaktere Spice bei ihrer Stadt abliefern.
ABHÄNGIGKEITEN	FA35 → p. 48
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

Aktionen

ID	FA50
TITEL:	Aktionen
BESCHREIBUNG:	Aktionen sind aktive Handlungen, welche Charaktere ausführen können. Das Einsetzen von Aktionen kostet den Charakter immer Action Points, die der Charakter zu Beginn seines Zuges erhält.
BEGRÜNDUNG	Aktionen sind notwendig, um zu definieren, welche Handlungen welche Charaktere ausführen können.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA51
TITEL:	normale Aktion
BESCHREIBUNG:	Normale Aktionen können von allen Charakteren ausgeführt werden. Dies ist unabhängig davon, welchen Charakter-Typ ein Charakter hat. Sie kosten den Charakter immer genau einen Action Point. Zu den normalen Aktionen gehören Angriff, Spice aufsammeln und Spice übergeben.
BEGRÜNDUNG	Normale Aktionen kategorisieren alle Aktionen, welche durch jeden Charakter ausgeführt werden können.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA50 → p. 53
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA52
TITEL:	Aktion: Angriff
BESCHREIBUNG:	Mit dieser normalen Aktion kann ein beliebiger Charakter einen gegnerischen Charakter auf einem benachbarten Feld verletzen. Der gegnerische Charakter bekommt, wenn beide Charaktere auf dem gleichen Höhenniveau sind, den Angriffsschaden des Angreifers von seinen Health Points abgezogen. Ist der Angreifer auf einem hohen Feld und der gegnerische Charakter auf einem niedrigen Feld, so bekommt der gegnerische Charakter $\frac{4}{3} \times$ Angriffsschaden des Angreifers von seinen Health Points abgezogen. Bei gegenteiliger Situation erhält der gegnerische Charakter $\frac{2}{3} \times$ Angriffsschaden des Angreifers von seinen Health Points abgezogen.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion Angriff abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA51 → p. 53, FA20 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA53
TITEL:	Aktion: Spice aufsammeln
BESCHREIBUNG:	Ein Charakter kann mit einer Aktion einen Spicekrümel von dem Feld, auf dem er steht, in sein Inventar aufnehmen, sofern dieses nicht voll ist.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion Spice aufsammeln abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA51 → p. 53, FA20 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA54
TITEL:	Aktion: Spice übergeben
BESCHREIBUNG:	Ein Charakter kann mit einer Aktion eine beliebige Teilmenge der Spicekrümel in seinem Inventar auf das Inventar eines benachbarten verbündeten Charakters übertragen, sofern dieser genügend Platz im Inventar hat.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion Spice übergeben abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA51 → p. 53
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA55
TITEL:	spezielle Aktion
BESCHREIBUNG:	Über diese Kategorie von Aktionen verfügen nur spezielle Charakter-Typen. Sie kosten immer die gesamten Action Points, die der Charakter in einer Runde zur Verfügung hat.
BEGRÜNDUNG	Spezielle Aktionen kategorisieren alle Aktionen, welche nur durch bestimmte Charakter-Typen ausgeführt werden können.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA50 → p. 53
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA56
TITEL:	Aktion: Kanly
BESCHREIBUNG:	Kanly ist eine spezielle Aktion, welche nur Charaktere vom Charakter-Type Noble ausführen können. Und die nur auf Charaktere vom Charakter-Type Nobel angewendet werden kann. Dies ist ein ritualisierter Angriff zwischen Nobel, welcher das Ziel des Angriffs mit einer Kanly-Erfolgswahrscheinlichkeit, welche in der Partie-Konfiguration festgehalten ist, besiegt. Bei Misserfolg passiert nichts.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion Kanly abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA55 → p. 55, FA53 → p. 54
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA57
TITEL:	Aktion: Family Atomics
BESCHREIBUNG:	Die spezielle Aktion Family Atomics kann nur durch Nobel eingesetzt werden. Dies ist für jedes Great House bis zu drei mal möglich, da der Atomic vorrat jedes Hauses drei entspricht. Hierbei wird ein beliebiges Zielfeld gewählt Auf diesem Zielfeld und auf dem gesamten 3x3 Quadrat ums Zielfeld werden Gebirge zu Felsplateaus und Dünen zu Flachsand Städten passiert nichts, alle Charaktere werden besiegt, Sandwürmer und Spicekrümel verschwinden. Wird hierbei ein gegnerischer Charakter besiegt wird und der Gegner zuvor nicht die Greatkonvention verletzt hat wird die Greatkonvention verletzt.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 ^{→ p. 48} , FA39 ^{→ p. 49} , FA55 ^{→ p. 55} , FA53 ^{→ p. 54} , FA58 ^{→ p. 56}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18}

ID	FA58
TITEL:	Great Convention
BESCHREIBUNG:	Die Great Convention besagt, dass wenn ein Great House Atomics gegen einen Charakter eines anderen Great House einsetzt und zuvor keiner gegen die Great Convention verstoßen hat, dass dieses Great House geächtet wird. Das gegnerische, welches nicht geächtet wird, Great House erhält dann von den anderen vier Great Houses zu Anfang der nächsten Runde jeweils einen zufälligen Charakter. Dieser wird auf ein zufälliges Feld auf der Karte platziert. Zudem darf das Great House, dass durch den Atomic Einsatz angegriffen wurde unbefreit auch Atomics auf das andere Great House einsetzen.
BEGRÜNDUNG	Die Great Convention muss eindeutig definiert sein.
ABHÄNGIGKEITEN	FA57 ^{→ p. 55}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18}

ID	FA59
TITEL:	Aktion: Spice Hoarding
BESCHREIBUNG:	Die spezielle Aktion Spice Hoarding kann nur von einem Charakter mit dem Charakter-Typ Mentat eingesetzt werden. Hierbei werden alle Spicekrümel auf dem aktuellen Feld des Charakters und auf den Nachbarfeldern eingesammelt und in das Inventar des Charakters aufgenommen.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion Spice Hoarding abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA55 → p. 55, FA53 → p. 54
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA60
TITEL:	Aktion: Voice
BESCHREIBUNG:	Die spezielle Aktion Voice kann nur von einem Charakter mit dem Charakter-Typ Bene Gesserit eingesetzt werden. Hierbei kann die Aktion auf einen benachbarten Charakter verbündet oder feindlich eingesetzt werden. Dieser Charakter übergibt dann hypnotisiert sofort das gesamte Spice in seinem Inventar an den Charakter, der die Aktion eingesetzt hat, bis dessen Inventar voll ist.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion Voice abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA55 → p. 55, FA54 → p. 54
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA61
TITEL:	Aktion: Sword Spin
BESCHREIBUNG:	Die spezielle Aktion Sword Spin kann nur von einem Charakter mit dem Charakter-Typ Fighter eingesetzt werden. Die Aktion Sword Spin macht allen benachbarten Charakteren schaden wie bei einem normalen Angriff.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Aktion Sword Spin abläuft, welche Vorbedingungen gelten müssen und welche Auswirkungen sie hat.
ABHÄNGIGKEITEN	FA35 → p. 48, FA39 → p. 49, FA55 → p. 55,
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

Spice

ID	FA62
TITEL:	Spice Blow
BESCHREIBUNG:	Spice Blow findet statt, wenn zu Beginn eine Runde die Spicekrümel Anzahl auf der Karte kleiner als der Spiceschwellwert ist. Es wird ein zufälliges Wüstenfeld gewählt und eine Zufallszahl im Intervall [3,6] ermittelt. Das gewählte Wüstenfeld und die benachbarten Wüstenfelder werden zufällig auf Flachland oder Düne gesetzt. Entsprechend der ermittelten Zufallszahl werden Spicekrümel über die Felder mittels Spice Streuung verstreut.
BEGRÜNDUNG	Es muss klar definiert sein, wie der Spice Blow funktioniert.
ABHÄNGIGKEITEN	FA17 → p. 42, FA21 → p. 43, FA63 → p. 58,
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA63
TITEL:	Spice Streuung
BESCHREIBUNG:	Spicekrümel werden bei Tod eines Charakters oder einem Spice Blow verteilt. Es wird das aktuelle Feld mit Spice bestreut, falls kein Spice auf dem aktuellen Feld liegt. Wenn das aktuelle Feld bestreut ist, werden n-1 zufällige Nachbarfeldern bestreut, falls dieses betretbar sind.
BEGRÜNDUNG	Es muss klar definiert sein wie die Spice Streuung funktioniert.
ABHÄNGIGKEITEN	FA17 → p. 42, FA21 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18, menschlicher Benutzer → p. 17

Dünenwanderung

ID	FA64
TITEL:	Dünenwanderung
BESCHREIBUNG:	Die Dünenwanderung muss mittels zellulären Automaten implementiert werden. Jedes Wüstenfeld hat ein niedriges oder hohes Höhenniveau, ist also Flachsand oder eine Düne. Wir betrachten niedrige Felder als "tot" und hohe als "lebendig" im Sinne des zellulären Automaten.
BEGRÜNDUNG	Es muss klar definiert sein wie die Dünenwanderung funktioniert.
ABHÄNGIGKEITEN	FA17 → p. 42, FA21 → p. 43, FA65 → p. 58
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA65
TITEL:	Dünenwanderung-Rundenphase
BESCHREIBUNG:	In der Dünenwanderungs-Rundenphase wird eine Iteration für den zellulären Automaten gemacht. Dabei wird für jedes Wüstenfeld aus dem eigenen Höhenniveau-Zustand und dem der Nachbar-Felder der Folgezustand berechnet, also ob das Feld Flachsand oder eine Dune bleiben bzw. werden soll. Städte und Gebirge gelten dabei als konstant hohe, also “lebendige”, Felder, niedrige Felsplateaus entsprechend als konstant “tote” Felder.
BEGRÜNDUNG	Es muss klar definiert sein wie die Dünenwanderung-Rundenphase funktioniert.
ABHÄNGIGKEITEN	FA17 → p. 42, FA21 → p. 43, FA64 → p. 58, FA66 → p. 59
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA66
TITEL:	Transitionsregel des zellulären Automaten
BESCHREIBUNG:	Die Transitionsregel des zellulären Automaten beschreibt, für den zellulären Automaten, bei welcher Anzahlen an lebendigen Nachbarfeldern eine tote Zelle lebendig wird oder eine lebendige Zelle überlebt. Die Transitionen orientieren sich an Conway’s Original Game of Life. Eine spezifische zu verwendete Regel wird in der Partie-Konfiguration festgehalten. Das Format ist hierbei ein String mit dem Inhalt B<Ziffernfolge>/S<Ziffernfolge> (B für Born und S für survive).
BEGRÜNDUNG	Es muss klar definiert sein, was mit den Transitionsregeln gemeint ist, und wo diese spezifiziert werden.
ABHÄNGIGKEITEN	FA65 → p. 58, FA64 → p. 58
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

Sandsturm

ID	FA67
TITEL:	Sandsturm
BESCHREIBUNG:	Zu Beginn der Partie wird vom Server ein zufälliges Feld gewählt zentrales Feld des Sandsturms gewählt. Auf diesem Feld und den Nachbarfeldern, also einem 3x3-Quadrat aus Feldern tobt ein gefährlicher Sandsturm. Zu Beginn jeder Runde wird die Sandsturm-Rundenphase ausgelöst. Charaktere, die sich im Sturm befinden, können keine Aktionen machen, und auch selbst nicht das Ziel von Aktionen sein. Lediglich durch einen Atomics-Einsatz können sie Ziel einer Aktion sein.
BEGRÜNDUNG	Es bedarf einer klaren Definition des Sandsturms.
ABHÄNGIGKEITEN	FA68 ^{→ p. 60}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18}

ID	FA68
TITEL:	Sandsturm-Rundenphase
BESCHREIBUNG:	In der Sandsturm-Rundenphase wird das Zentrafeld des Sturms auf ein zufälliges Nachbarfeld gesetzt. Danach wird jedes Feld im 3x3 Bereich des Sturms zufällig auf ein niedriges oder hohes Höheniveau gesetzt. Der Sandsturm bringt also kleine lokale Störungen in den zellulären Automaten ein, und verhindert damit statische Zustände des Automaten.
BEGRÜNDUNG	Es bedarf einer klaren Definition des Ablaufes der Sandsturm-Rundenphase.
ABHÄNGIGKEITEN	FA67 ^{→ p. 60} , FA66 ^{→ p. 59} , FA20 ^{→ p. 43}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18}

Sandwurm

ID	FA69
TITEL:	Sandwurm
BESCHREIBUNG:	Auf Arakis leben riesige Sandwürmer, sie können sich über Wüstenfelder bewegen, aber nicht über Felsenfelder. Der Sandwurm wird in der Sandwurm-Rundenphase aktiv. Wüstenfelder, auf denen sich ein Sandwurm befindet, gelten als nicht betretbar. Alle Wüstenfelder, von denen ein Sandwurm sich weg bewegt, werden auf Flachsand gesetzt.
BEGRÜNDUNG	Es muss klar definiert sein, was ein Sandwurm ist und welche Eigenschaften er hat.
ABHÄNGIGKEITEN	FA70 → p. 61, FA19 → p. 43, FA21 → p. 43
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA70
TITEL:	Sandwurm-Rundenphase
BESCHREIBUNG:	Zu Beginn jeder Runde in der Sandwurm-Rundenphase bewegt sich der Sandwurm, falls vorhanden n Felder auf eine Zielperson zu (n ist in der Partie-Konfiguration definiert). Wenn der Sandwurm sich auf das Feld eines Charakters bewegt, so wird dieser verschluckt und der Sandwurm verschwindet. Falls der Sandwurm keine nur über Wüstenfelder führende Strecke hat, um zu seiner Zielperson zu gelangen, verschwindet er und taucht direkt danach auf einem zufällig gewählten Wüstenfeld wieder auf. Seine Rundenphase ist damit beendet. Falls es keinen Sandwurm gibt, wird geprüft, ob in der vorangegangenen Rundenphase ein Charakter als Lauf markiert wurde. Falls keiner als laut markiert wurde passiert nichts. Wenn es mindestens einen lauten Charakter gibt, wird dieser oder ein zufälliger lauter Charakter vom Sandwurm ausgewählt. Der Sandwurm taucht dann auf einem zufälligen Wüsten Feld, welches kein Nachbarfeld des ausgewählten Charakters ist auf. Seine Rundenphase endet und er verfolgt später weiter den gleichen Charakter.
BEGRÜNDUNG	Die Sandwurm-Rundenphase muss wohl definiert sein.
ABHÄNGIGKEITEN	FA69 → p. 61, FA35 → p. 48, FA48 → p. 52
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

Klonen von besieгten Charakteren

ID	FA71
TITEL:	Klon-Rundenphase
BESCHREIBUNG:	Jeder besieгte Charakter eines der beiden Great Houses wird mit einer Klon-Erfolgswahrscheinlichkeit aus der Partiekonfiguration geklont. Der geklonte Charakter wird nun auf ein zufälliges freies Nachbarfeld der Stadt platziert. Charaktere die vom Sandwurm verschluckt wurden, können nicht geklont werden.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Klon-Rundenphase abläuft und von welchen Parameter sie beeinflusst wird.
ABHÄNGIGKEITEN	FA69 → p. 61, FA35 → p. 48
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

Partievorbereitung

ID	FA72
TITEL:	Partievorbereitung
BESCHREIBUNG:	Wenn zwei Spieler in einer Partie sind und der Server alle Daten geladen hat, wird eine Partie vorbereitet. Das heißt, die Spieler wählen eines der Great Houses für sich aus. Dies geschieht in der Great House Wahl.
BEGRÜNDUNG	Es muss klar definiert sein, was in der Partie-Vorbereitung passiert.
ABHÄNGIGKEITEN	FA73 → p. 62, FA28 → p. 46
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA73
TITEL:	Great House Wahl
BESCHREIBUNG:	Der Server nimmt aus der Liste der sechs Great Houses zwei zufällige heraus, die dem Spieler angezeigt werden. Der Spieler wählt eines davon aus. Dies findet Nebenläufig für beide Spieler statt. Der Server muss den beiden Spielern jeweils eine Menge aus zwei Häusern zur Auswahl anbieten, diese Mengen müssen disjunkt sein.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Great House Wahl abläuft.
ABHÄNGIGKEITEN	FA28 → p. 46
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

Beginn einer Partie

ID	FA74
TITEL:	Beginn einer Partie
BESCHREIBUNG:	Vor der ersten Runde weist der Server jedem der Häuser eine der beiden Städte zu, und platziert dann alle Charaktere eines Hauses auf zufälligen freien betreibbaren Felder benachbart der jeweiligen Stadt.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Partie beginnt.
ABHÄNGIGKEITEN	FA28 → p. 46, FA35 → p. 48, FA17 → p. 42
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

Runden und Züge

ID	FA75
TITEL:	Runden und Züge
BESCHREIBUNG:	DESERTS OF DUNE läuft in Runden ab, in denen Ereignisse passieren, und die einzelnen Charaktere nacheinander ihre Züge machen. Zu Beginn jeder Runde handelt der Server einige Ereignisse ab. Die Reihenfolge der Rundenphasen ist: 1. Dünenwanderungs-Rundenphase 2. Sandsturm-Rundenphase 3. Sandwurm-Rundenphase 4. Klon-Rundenphase 5. Charakterzug-Rundenphase
BEGRÜNDUNG	Es muss klar definiert sein, welche Rundenphasen in welcher Reihenfolge stattfinden.
ABHÄNGIGKEITEN	FA65 → p. 58, FA68 → p. 60, FA71 → p. 62, FA76 → p. 63
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA76
TITEL:	Charakterzug-Rundenphase
BESCHREIBUNG:	In der Charakterzug-Rundenphase werden zunächst alle Charaktere vom Server für diese Runde in eine zufällige Reihenfolge gebracht. Die Charaktere kommen nun der Reihe nach dran, und können, wenn sie zu diesem Zeitpunkt leben, ihren Zug machen, der aus mehreren Zugphasen besteht.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Charakterzug-Rundenphase abläuft.
ABHÄNGIGKEITEN	FA35 → p. 48
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	FA77
TITEL:	Züge von Charakteren
BESCHREIBUNG:	Züge von Charakteren bestehen aus Zugphasen, in denen jeweils entweder ein Bewegungsschritt oder eine Aktion gemacht wird. Wenn ein Charakter mit seinem Zug an der Reihe ist, werden zuerst seine Movement Points und Action Points auf die seinem Charakter-Typ entsprechenden Werte aus der Partie-Konfiguration gesetzt. Der Spieler kann dann für den aktiven Charakter diese Punkte in beliebiger Reihenfolge, Zugphase für Zugphase, einen nach dem anderen ausgeben.
BEGRÜNDUNG	Es muss klar definiert sein, wie Züge von Charakteren ablaufen.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Teilnehmer ^{→ p. 18}

Ende der Partie und Bestimmung des Siegers

ID	FA78
TITEL:	Ende der Partie
BESCHREIBUNG:	Wenn in der Sandwurm-Rundenphase der letzte Charakter, den ein Haus kontrolliert, durch einen gewöhnlichen Sandwurm-Angriff verschluckt wird, während das andere Haus nach Abhandlung dieser Sandwurm-Rundenphase noch Charaktere hat, gewinnt das andere Haus unmittelbar die Partie.
BEGRÜNDUNG	Es muss klar definiert sein, wann die Partie endet und wie der Sieger ermittelt wird.
ABHÄNGIGKEITEN	FA35 ^{→ p. 48} , FA69 ^{→ p. 61} , FA70 ^{→ p. 61}
PRIORITÄT	5
AKTEUR	Server ^{→ p. 17} , Tutor ^{→ p. 18}

ID	FA79
TITEL:	Behandlung überlanger Partien
BESCHREIBUNG:	Wenn die Partie über mehr Runden läuft, als der in der Partie-Konfiguration festgelegter Höchstwert. Wird die Partie durch einen speziellen Überlängenmechanismus einem beschleunigten Ende zugeführt.
BEGRÜNDUNG	Es muss klar definiert sein, wann der Überlängenmechanismus Anwendung findet.
ABHÄNGIGKEITEN	FA80 ^{→ p. 64}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18}

ID	FA80
TITEL:	Überlängenmechanismus
BESCHREIBUNG:	Durch ein großes Erdbeben werden alle Felsfelder zu Dünenfeldern. Die Sandwurm-Rundenphase wird durch die Shai-Hulud-Rundenphase ersetzt. Falls gerade ein gewöhnlicher Sandwurm unterwegs ist, verschwindet dieser, und taucht nicht mehr auf. Auch die Regel, dass ein Haus unmittelbar verliert, wenn alle von ihm kontrollierten Charaktere verschluckt wurden, wird außer Kraft gesetzt. Sobald in einer Shai-Hulud-Rundenphase der letzte Charakter, der noch auf der Karte stand, verschluckt wird, ist die Partie beendet.
BEGRÜNDUNG	Es muss klar definiert sein, wie der Überlängenmechanismus funktioniert.
ABHÄNGIGKEITEN	FA79 → p. 64
PRIORITÄT	5
AKTEUR	Tutor → p. 18

ID	FA81
TITEL:	Shai-Hulud-Rundenphase
BESCHREIBUNG:	In der Shai-Hulud-Rundenphase wählt sich Shai-Hulud einen zufälligen Charakter, und macht unvermittelt einen Sandwurmangriff auf dessen Feld. Der Charakter wird also verschluckt, und ist damit unwiederbringlich aus dem Spiel entfernt.
BEGRÜNDUNG	Es muss klar definiert sein, was in der Shai-Hulud-Rundenphase passiert.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18

ID	FA82
TITEL:	Sieg-Metrik überlanger Runden
BESCHREIBUNG:	Nachdem die Partie durch Shai-Hulud beendet wurde, wird der Sieger anhand der Sieg-Metriken ermittelt. Erste Metrik: Welches Haus hat den größten Spice-Vorrat? Zweite Metrik: Welches Haus hat mehr Spicekrümel aufgenommen? Dritte Metrik: Welches Haus hat mehr Charaktere des Gegners besiegt? Vierter Metrik: Bei welchem Haus wurden weniger Charaktere durch gewöhnliche Sandwürmer verschluckt? Fünfter Metrik: Welches Haus hatte den letzten Charakter auf dem Spielbrett?
BEGRÜNDUNG	Es muss klar definiert sein, wie der Sieger ermittelt wird, wenn das Spiel durch den Überlängenmechanismus beendet wird.
ABHÄNGIGKEITEN	FA80 → p. 64, FA81 → p. 65
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Server → p. 17

ID	FA83
TITEL:	Behandlung kein freies Nachbarfeld
BESCHREIBUNG:	Falls kein Nachbarfeld frei ist, wird rekursiv zufällig ein besetztes Feld gesucht und ein zufällig freies Nachbarfeld gesucht.
BEGRÜNDUNG	Es muss klar definiert sein, wie die Behandlung von besetzten Nachbarfeldern bei der Suche nach einem freien Nachbarfelde funktioniert.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18}

ID	FA84
TITEL:	Behandlung von gleichwertigen Alternativen
BESCHREIBUNG:	Liegen zwei gleichwertige Alternativen vor, dann wird zufällig eine der beiden gewählt.
BEGRÜNDUNG	Es muss klar definiert werden, wie zwei gleichwertige Alternativen behandelt werden.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Server ^{→ p. 17} , Tutor ^{→ p. 18}

Server

ID	FA85
TITEL:	Starten des Servers über Docker Container
BESCHREIBUNG:	Ein Server muss nicht-interaktiv über die Kommandozeile in Form eines Docker-Containers gestartet werden können. Dabei können in einer vom Standardisierungskomitee definierten Form die nötigen Argumente übergeben werden.
BEGRÜNDUNG	Es muss möglich sein den Server, ohne kompliziertes lokales Build verfahren, mit allen notwendigen Parametern zu starten.
ABHÄNGIGKEITEN	FA3 ^{→ p. 39}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19} , Teilnehmer ^{→ p. 18} , Server ^{→ p. 17}

ID	FA86
TITEL:	Server lädt Partie-Konfiguration
BESCHREIBUNG:	Der Server lädt beim Start eine Partie-Konfiguration. Alle Einstellungen für eine Partie werden in der Partie-Konfiguration gespeichert. Enthaltene Konfigurationsdaten sind bspw. erlaubte Zeitspannen für Aktionen in den Rundenphasen, Rundenanzahl bis zum Eintritt der Überlängenbedingung für eine Partie, etc.
BEGRÜNDUNG	Alle Konfigurationsdaten müssen aus der Partie-Konfiguration gelesen werden.
ABHÄNGIGKEITEN	FA3 → p. 39, FA124 → p. 81
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA87
TITEL:	Server lädt Szenario-Konfiguration
BESCHREIBUNG:	Der Server lädt beim Start eine Szenario-Konfiguration, die das Spielfeld definiert, auf dem die Partie stattfindet.
BEGRÜNDUNG	Die Spielfeldkonfiguration soll aus der Szenario-Konfiguration entnommen werden.
ABHÄNGIGKEITEN	FA3 → p. 39, FA126 → p. 81
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17 126 → p. 81

ID	FA88
TITEL:	Server erlaubt Clientverbindung
BESCHREIBUNG:	Der Server erlaubt genau zwei Clients, sich über das Netzwerk bei ihm für eine Partie als Mitspieler anzumelden.
BEGRÜNDUNG	Der Server muss es genau zwei Clients ermöglichen, sich, als Mitspieler anzumelden.
ABHÄNGIGKEITEN	FA3 → p. 39, FA4 → p. 39, FA5 → p. 39
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA89
TITEL:	Server startet Partie
BESCHREIBUNG:	Sobald sich zwei mitspielende Clients beim Server registriert haben, startet der Server eine Partie und wickelt sie gemäß der Spielregeln ab.
BEGRÜNDUNG	Es muss klar definiert sein, wann der Server die Partie startet.
ABHÄNGIGKEITEN	FA3 → p. 39, FA74 → p. 63
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA90
TITEL:	Server erlaubt Zuschauer
BESCHREIBUNG:	Der Server erlaubt Benutzer-Clients sich als Zuschauer für eine Partie zu registrieren. Sie bekommen dann den aktuellen Spielzustand und zukünftige Updates geschickt.
BEGRÜNDUNG	Es muss klar definiert sein, ob und wie viele Zuschauer Clients der Server zulässt.
ABHÄNGIGKEITEN	FA3 → p. 39, FA101 → p. 71
PRIORITÄT	3
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA91
TITEL:	Server trennt Verbindung
BESCHREIBUNG:	Falls von einem Client erwartet wird, innerhalb einer in der Partie-Konfiguration festgelegten Zeitspanne eine Nachricht an den Server zu schicken, ein Fortgang der Partie ohne diese Nachricht nicht sinnvoll möglich ist, und der Server keine Nachricht rechtzeitig empfängt, soll der Server die Verbindung zum Client abbrechen. Im Falle eines mitspielenden Clients wird dieser disqualifiziert.
BEGRÜNDUNG	Es muss klar definiert sein, unter welchen Bedingungen der Server die Verbindung zum Client trennt.
ABHÄNGIGKEITEN	FA3 → p. 39, QA3 → p. 107
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA92
TITEL:	Server übernimmt Client Input
BESCHREIBUNG:	In Situationen, in denen der Server eine nicht rechtzeitig empfangene Nachricht als Wunsch des Clients interpretieren kann, nichts zu tun, sollte er tolerant sein, und für den Clients ein Default-Verhalten wählen. Damit kann eine Partie manchmal fortgesetzt werden, wenn ein Client kurzzeitig die Verbindung verliert, oder für eine Nachricht zu lange braucht. Der Client verliert lediglich einen Zug, wird aber nicht sofort aus dem Spiel geworfen.
BEGRÜNDUNG	Der Server soll im Falle eines kurzzeitigen Verbindungsabbruchs den Client vertreten, um den Spielfluss in Gang zu halten, und dafür zu sorgen, dass der möglichst weiter spielen kann.
ABHÄNGIGKEITEN	FA3 → p. 39, QA3 → p. 107
PRIORITÄT	3
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA93
TITEL:	Server Erkennung verspäteter Nachrichten
BESCHREIBUNG:	Verspätet eintreffende Nachrichten für eine Rundenphase, die in einer späteren Rundenphase beim Server eingehen, sollten entsprechend vom Server nicht als Protokollverletzung betrachtet, sondern als verspätet erkannt und einfach verworfen werden.
BEGRÜNDUNG	Der Server soll klar definiert wissen, ob eine Client-Nachricht eine Protokollverletzung darstellt oder verspätet ist.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA94
TITEL:	Server pausiert Partie
BESCHREIBUNG:	Falls ein mitspielender Benutzer-Client eine Pausierung der Partie wünscht, unterbricht der Server diese, bis irgendeiner der Mitspieler anzeigt, dass er weiterspielen möchte. KI-Clients dürfen keine Pausen verlangen, und auch keine Pausen beenden.
BEGRÜNDUNG	Es muss klar definiert sein, welche Clients eine Pausierung wünschen können und wie diese Pausierung vom Server gehandhabt wird.
ABHÄNGIGKEITEN	
PRIORITÄT	4
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA95
TITEL:	Server hält Session offen
BESCHREIBUNG:	Falls ein mitspielender Client seine Connection zum Server verliert, so bleibt seine Session zunächst bestehen. Der Client kann sich, innerhalb einer gewissen Zeitspanne, erneut mit dem Server verbinden, und seine Session fortsetzen. Der Client bekommt dazu vom Server den vollständigen aktuellen Spielzustand geschickt, für den Fall, dass der Client bspw. neu gestartet werden musste.
BEGRÜNDUNG	Es muss definiert sein, wie der Server mit dem Verbindungsverlust eines Clients umgeht.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19} , Teilnehmer ^{→ p. 18} , Server ^{→ p. 17}

ID	FA96
TITEL:	Server Handhabung von Protokollverletzungen
BESCHREIBUNG:	Falls ein Client sich nicht an das Kommunikationsprotokoll hält, oder eine nach den Spielregeln unzulässige Aktion durchführen will, sendet der Server dem Client eine Nachricht mit einer aussagekräftigen Fehlermeldung, beendet die Verbindung zu ihm und schließt ihn damit vom weiteren Verlauf der Partie aus. Im Fall eines mitspielenden Clients gewinnt dadurch der gegnerische Mitspieler.
BEGRÜNDUNG	Es muss definiert sein, wie der Server darauf reagiert, wenn ein Client sich nicht an das Kommunikationsprotokoll hält.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19} , Teilnehmer ^{→ p. 18} , Server ^{→ p. 17}

ID	FA97
TITEL:	Server informiert Clients über Spielzustand
BESCHREIBUNG:	Der Server informiert alle Clients über die Aktionen der Spieler und die Ereignisse, die sich daraus ergeben haben, und den daraus resultierenden Spielzustand.
BEGRÜNDUNG	Es muss definiert sein, welche Clients wie über den Spielzustand informiert werden.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19} , Teilnehmer ^{→ p. 18} , Server ^{→ p. 17}

ID	FA98
TITEL:	Server überprüft Siegbedingungen
BESCHREIBUNG:	Der Server überprüft zu den relevanten Zeitpunkten, ob ein Spieler gemäß der Siegbedingungen gewonnen hat. Wenn dies der Fall ist, beendet er die Partie und benachrichtigt alle Clients entsprechend.
BEGRÜNDUNG	Der Server muss dazu in der lange sein zu den relevanten Zeitpunkten die Spielbedingung zu prüfen.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19} , Teilnehmer ^{→ p. 18} , Server ^{→ p. 17}

ID	FA99
TITEL:	Server loggt Partie
BESCHREIBUNG:	Der Server schreibt alle Informationen in eine Logdatei, die zum Nachvollziehen des Partieverlaufs nötig sind. Dadurch können Clients ein Replay der Partie abspielen. Das Standardisierungskomitee legt das Format für Logs bzw. Replay-Dateien fest. Am Ende einer Partie kann das Replay auch direkt den teilnehmenden Clients zugeschickt werden.
BEGRÜNDUNG	Der Server solle alle Informationen in eine Logdatei schreiben, um den Informationsgewinn über den Verlauf der Partie zu optimieren.
ABHÄNGIGKEITEN	
PRIORITÄT	3
AKTEUR	Tutor ^{→ p. 18} , Entwickler ^{→ p. 19} , Teilnehmer ^{→ p. 18} , Server ^{→ p. 17}

Benutzer-Client

ID	FA100
TITEL:	Registrierung als Spieler
BESCHREIBUNG:	Ein menschlicher Teilnehmer muss sich über den Benutzer-Client beim Server als Spieler registrieren können. Diese Registrierung gilt für genau eine Partie.
BEGRÜNDUNG	Der Server muss wissen, dass ein Spieler eine Partie spielen möchte
ABHÄNGIGKEITEN	FA102 ^{→ p. 72} , FA103 ^{→ p. 72}
PRIORITÄT	5
AKTEUR	Spieler ^{→ p. 17} , Tutor ^{→ p. 18}

ID	FA101
TITEL:	Registrierung als Zuschauer
BESCHREIBUNG:	Ein menschlicher Teilnehmer muss sich über den Benutzer-Client beim Server als Zuschauer registrieren können. Diese Registrierung gilt für genau eine Partie, die entweder noch ausstehend ist oder schon läuft.
BEGRÜNDUNG	Der Server muss wissen, dass ein Benutzer bei einer Partie zuschauen will (auch wenn erst später einsteigen will) und ihm dann auch den Spielstand aktualisiert immer zusenden.
ABHÄNGIGKEITEN	FA102 → p. 72, FA103 → p. 72
PRIORITÄT	5
AKTEUR	Zuschauer → p. 17, Tutor → p. 18

ID	FA102
TITEL:	Verbindungsaufbau zum Server
BESCHREIBUNG:	Der Benutzer-Client muss sich mit einem Server verbinden können. Dabei müssen sie dem Server die Rolle mitteilen, das heißt ob sie ein Spieler sind oder nicht, sowie ihren Namen.
BEGRÜNDUNG	Der Server verwaltet die Partien und Teilnehmer, deswegen muss ein Benutzer-Client sich damit verbinden, um an einer Partie teilnehmen oder sie beobachten zu können. Außerdem muss der Server wissen, ob es sich bei dem Benutzer um einen Spieler handelt, um die entsprechenden Rechte zu vergeben. Der Name ist für den Server wichtig, damit man die Clients auf der Benutzeroberfläche identifizieren kann
ABHÄNGIGKEITEN	FA100 → p. 71, FA101 → p. 71, FA103 → p. 72
PRIORITÄT	5
AKTEUR	Spieler → p. 17, Tutor → p. 18, Server → p. 17

ID	FA103
TITEL:	Graphische Benutzeroberfläche
BESCHREIBUNG:	Der Benutzer-Client hat eine graphische Benutzeroberfläche, über die dem menschlichen Benutzer alle Daten zum Spiel angezeigt werden können und über die der Spieler mit dem Spiel interagieren, das heißt seine Züge ausführen kann.
BEGRÜNDUNG	Der menschliche Benutzer muss das Spiel sehen können, da er im Gegensatz zu Computern nicht gut mit rohen Daten umgehen kann, sondern lieber die Daten visualisiert verwertet. Außerdem macht ein Spiel mit einer graphischen Oberfläche und einem schönen Design mehr Spaß. Zudem braucht der Spieler eine Möglichkeit zur Eingabe seiner Züge.
ABHÄNGIGKEITEN	FA104 → p. 73, FA105 → p. 73, FA106 → p. 74, FA107 → p. 74, FA109 → p. 75, FA110 → p. 75, FA112 → p. 76, FA108 → p. 75
PRIORITÄT	5
AKTEUR	Tutor → p. 18, menschlicher Benutzer → p. 17

ID	FA104
TITEL:	Visualisierung Spielgeschehen
BESCHREIBUNG:	Der Benutzer-Client muss das Spielgeschehen auf der graphischen Benutzeroberfläche visualisieren können. Das bedeutet, dass der Benutzer-Client die Karte beziehungsweise das Spielfeld anzeigen muss, sowie alle sich darauf befindenden Charaktere. Die Charaktere müssen so angezeigt werden, dass sie auf dem Spielfeld unterscheiden werden können, durch zum Beispiel unterschiedliche Avatare oder Namen.
BEGRÜNDUNG	Der Benutzer muss den aktuellen Spielstand sehen können und die Charaktere auf dem Spielfeld unterscheiden können. Ohne Unterscheidung oder Anzeige des Spielstandes, kann der Benutzer den aktuellen Spielstand nicht (korrekt) wahrnehmen und damit dem Spiel nicht folgen oder Züge basierend auf dem aktuellen Stand ausführen.
ABHÄNGIGKEITEN	FA103 → p. 72, FA109 → p. 75
PRIORITÄT	5
AKTEUR	menschlicher Benutzer → p. 17, Tutor → p. 18

ID	FA105
TITEL:	Visualisierung Spielstatus
BESCHREIBUNG:	<p>Der Benutzer-Client muss alle spiel-relevanten Informationen, das heißt den Status des Spiels, visualisieren. Das heißt auf der graphischen Benutzeroberfläche müssen die Werte und die Zustände der Charaktere angezeigt werden. Zu den spielrelevanten Informationen zählen mindestens:</p> <ul style="list-style-type: none"> • Health-Bar der Charakter • das Inventar und gesammelte Spice • aktuelle Phase • Spielerinformationen, wie MP, AP oder Angriffsschaden
BEGRÜNDUNG	Der Benutzer müssen den aktuellen Status des Spiels sehen, um zu wissen welche Aktionen möglich oder sinnvoll sind und dem Spielverlauf als Zuschauer zu folgen oder als Spieler fundierte Entscheidungen für eine Aktion treffen.
ABHÄNGIGKEITEN	FA103 → p. 72, FA109 → p. 75
PRIORITÄT	5
AKTEUR	menschlicher Benutzer → p. 17, Tutor → p. 18

ID	FA106
TITEL:	Visualisierung Aktionen
BESCHREIBUNG:	Der Benutzer-Client muss die Aktionen des Spielers visualisieren. Das bedeutet, dass Angriffe animiert werden müssen und zwar so, dass man sieht, welcher Charakter welchen anderen angegriffen hat und wie viel Schaden er ihm zugefügt hat. Weiterhin sollen alle Veränderungen auf dem Spielfeld, wie zum Beispiel eine Family Atomic oder die Bewegung des Sandwurms animiert werden.
BEGRÜNDUNG	Der Benutzer muss sehen können, welche Aktion gerade ausgeführt wird und welche Veränderungen dadurch auf dem Spielfeld entstehen. Außerdem ist eine Animation von Zügen ein gutes Benutzererlebnis.
ABHÄNGIGKEITEN	FA103 → p. 72, FA109 → p. 75
PRIORITÄT	5
AKTEUR	menschlicher Benutzer → p. 17, Tutor → p. 18

ID	FA107
TITEL:	Visualsierung Phasen
BESCHREIBUNG:	Der Benutzer-Client muss die Abwicklung der einzelnen Runden- und Zugphasen animieren. Dabei muss die Dauer der Animation an die vorgegebene Dauer für die Phase in der Partiekonfiguration angepasst werden.
BEGRÜNDUNG	Der Benutzer möchte sehen, in welcher Phase er sich gerade befindet. Außerdem zeigt eine Animation der Phasen den Spielverlauf besser an und stellt ein gutes Benutzererlebnis dar.
ABHÄNGIGKEITEN	FA103 → p. 72, FA109 → p. 75, FA124 → p. 81
PRIORITÄT	5
AKTEUR	menschlicher Benutzer → p. 17, Tutor → p. 18

ID	FA108
TITEL:	Visualisierung Spielausgang
BESCHREIBUNG:	Am Ende der Partie soll der Benutzer-Client dem Benutzer den Gewinner anzeigen. Optional sollen noch interessante Statistiken angezeigt werden können, wie zum Beispiel in welcher Phase man dem Gegener wie viel Schaden zugefügt hat.
BEGRÜNDUNG	Der Benutzer möchte sehen, wer gewonnen hat und kann sich mit Hilfe der Statistiken eventuell verbessern.
ABHÄNGIGKEITEN	FA103 → p. 72, FA109 → p. 75
PRIORITÄT	4
AKTEUR	menschlicher Benutzer → p. 17, Tutor → p. 18

ID	FA109
TITEL:	Visualisierungsdauer
BESCHREIBUNG:	Die Animation von bestimmten Aktionen oder Phasen darf nur eine in der Partiekonfiguration festgesetzte Zeit dauern.
BEGRÜNDUNG	Wenn die Animationen zu lange dauern, dann kommt das Spiel dem Benutzer ruckelig vor und stört den Spielfluss. Außerdem sehen lange Animationen nicht schön aus und nerven den Benutzer.
ABHÄNGIGKEITEN	FA103 → p. 72, FA124 → p. 81
PRIORITÄT	5
AKTEUR	menschlicher Benutzer → p. 17, Tutor → p. 18

ID	FA110
TITEL:	Ermöglichung Spielinteraktion
BESCHREIBUNG:	Der Spieler kann über die Oberfläche Aktionen vornehmen, die der Client an den Server sendet. Dabei muss der Benutzer-Client gewährleisten, dass nur regelkonforme Eingaben getätigt werden können.
BEGRÜNDUNG	Der Spieler muss irgendwie mit dem Spiel interagieren können und seine Züge dem Server mitteilen. Damit der Server nicht abstürzt dürfen nur valide Züge an ihn gesendet werden.
ABHÄNGIGKEITEN	FA103 → p. 72
PRIORITÄT	5
AKTEUR	Spieler → p. 17, Tutor → p. 18

ID	FA111
TITEL:	Vereinfachung Spielinteraktion
BESCHREIBUNG:	Der Benutzer-Client soll die Möglichkeit haben, dem Benutzer mit Hilfe von Hotkeys die Interaktion zu erleichtern. Damit kann der Spieler bestimmte Aktionen auch über Tastenkombinationen ausführen.
BEGRÜNDUNG	Hotkeys können dem Spieler die Bedienung des Spiels komfortabler gestalten.
ABHÄNGIGKEITEN	FA103 → p. 72, FA110 → p. 75
PRIORITÄT	2
AKTEUR	Spieler → p. 17, Tutor → p. 18

ID	FA112
TITEL:	Antrag auf Pausierung und Spielfortsetzung
BESCHREIBUNG:	Der Spieler muss über den Benutzer-Client dem Server einen Antrag auf Pause oder Wiederaufnahme der Partie senden können. Der Antrag auf Wiederaufnahme kann nur geschehen, wenn ein Spiel pausiert ist und ein Spiel kann nur pausiert werden, wenn nicht schon pausiert ist.
BEGRÜNDUNG	Der Spieler soll die Möglichkeit haben, ein Spiel anzuhalten und fortzusetzen, weil er zwischenzeitlich etwas anderes erledigen muss, das Spiel aber nicht beenden will.
ABHÄNGIGKEITEN	FA103 → p. 72
PRIORITÄT	5
AKTEUR	Spieler → p. 17, Tutor → p. 18

ID	FA113
TITEL:	Abspielen eines Replay
BESCHREIBUNG:	Der Benutzer kann mit Hilfe einer Log-Datei vom Server ein Replay laden und somit eine gespielte Partie nochmal anschauen
BEGRÜNDUNG	Der Spieler möchte vielleicht alte Partien analysieren und sich so verbessern oder sehen, wie andere Spieler spielen, auch wenn er nicht live die Partie folgen konnte. Für die Entwicklern kann ein Replay beim Debuggen helfen, um so sehen, wie gut der KI-Client spielt.
ABHÄNGIGKEITEN	FA103 → p. 72, FA99 → p. 71
PRIORITÄT	2
AKTEUR	menschlicher Benutzer → p. 17, Tutor → p. 18, Entwickler → p. 19

KI-Client

ID	FA114
TITEL:	Start des KI-Client
BESCHREIBUNG:	Der Start des KI-Client erfolgt über die Kommandozeile mit Hilfe eines Docker-Containers durch eine am Projekt beteiligten Person. Dafür werden ihm die von dem Standardisierungskomitee nötigen Argumente übergeben.
BEGRÜNDUNG	Der KI-Client braucht keine graphische Oberfläche und der Start über die Kommandozeile als Docker-Container erleichtert den Build- und Startprozess
ABHÄNGIGKEITEN	FA115 → p. 77
PRIORITÄT	5
AKTEUR	Tutor → p. 18

ID	FA115
TITEL:	Verbindungsauftbau zum Server
BESCHREIBUNG:	Mit den übergebenen Argumenten beim Start des KI-Clients, wie zum Beispiel der IP-Adresse und dem Port des Servers, baut der KI-Client eine Verbindung zum Server auf und teilt diesem mit, dass dieser Client eine KI ist und welchen Namen der KI-Client hat. Danach kann er einer Partie beitreten und nicht mehr mit dem Benutzer kommunizieren
BEGRÜNDUNG	Um an einer Partie teilnehmen, muss sich auch ein KI-Client mit dem Server verbinden. Außerdem muss der Server wissen, dass es sich um eine KI handelt, da diese andere Regeln hat, als ein Spieler (zum Beispiel keine Möglichkeit der Pausierung)
ABHÄNGIGKEITEN	FA116 → p. 77, FA114 → p. 77
PRIORITÄT	5
AKTEUR	Server → p. 17, Tutor → p. 18

ID	FA116
TITEL:	Pausierung KI-Client
BESCHREIBUNG:	KI-Clients dürfen keine Pausierung der Partie bei dem Server anfragen. Jedoch muss dieser Client mit einer Pausierung, die beliebig lang sein kann, von einem Benutzer-Client umgehen können. Das heißt sie müssen den Fall behandeln, dass die Partie keinen Fortschritt machen und einen Wartezustand einnehmen
BEGRÜNDUNG	KI-Client müssen keine Pause einlegen, jedoch sollte der Spieler die Möglichkeit haben, ein Spiel anzuhalten. Damit muss dann der KI-Client umgehen können, damit das Spiel nicht abbricht.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Spieler ^{→ p. 17} , Tutor ^{→ p. 18}

ID	FA117
TITEL:	Zugwahl KI-Client
BESCHREIBUNG:	In jeder Zug- und Rundenphase muss der KI-Client regelkonforme und nach Möglichkeit sinnvolle Züge beziehungsweise Aktionen auswählen. Diese Wahl wird von einem Algorithmus getroffen und darf nur eine gewisse Zeit andauern. Danach muss der KI-Client die Aktion dem Server über das definierte Protokoll mitteilen.
BEGRÜNDUNG	Der KI-Client muss auch Züge machen, die jedoch nicht zu lange dauern sollten, damit der Spieler nicht ewig warten muss. Außerdem sollten die Züge auch regelkonform sein, damit der Server die Züge auch ausführen kann und nicht abstürzt, sowie sinnvoll, damit der andere Spieler auch gegen einen Gegner spielt, bei dem Spiel Spaß macht.
ABHÄNGIGKEITEN	FA118 ^{→ p. 78} , FA119 ^{→ p. 78}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Server ^{→ p. 17} , Spieler ^{→ p. 17}

ID	FA118
TITEL:	Zugdauer KI-Client
BESCHREIBUNG:	Der KI-Client hat nur eine in der Konfiguration festgesetzte Zeitspanne, um einen Zug auszuführen.
BEGRÜNDUNG	Das Spiel soll nicht ewig dauern und der andere Spieler will ebenso Züge machen. Über eine variable Länge kann man jedoch die Schwierigkeit der KI anpassen
ABHÄNGIGKEITEN	FA117 ^{→ p. 78} , FA124 ^{→ p. 81}
PRIORITÄT	5
AKTEUR	Tutor ^{→ p. 18} , Server ^{→ p. 17} , Spieler ^{→ p. 17}

ID	FA119
TITEL:	Anpassung Intelligenzstufen
BESCHREIBUNG:	<p>Der KI-Client soll verschiedene Intelligenzstufen haben, die in der Partiekonfiguration eingestellt werden. Dabei entspricht eine Intelligenzstufe dem Schwierigkeit der KI, der vorher von den Entwicklern individualisiert wurden. Es wird die folgenden Schwierigkeitsgrade geben:</p> <ul style="list-style-type: none"> • EASY • MEDIUM • HARD
BEGRÜNDUNG	Durch die unterschiedlichen Schwierigkeitsgrade kann der Spieler das Spiel individualisieren. Dadurch kann man leicht in das Spiel einsteigen, kann sich steigern und hat lange eine Herausforderung, aber keine Überforderung.
ABHÄNGIGKEITEN	FA117 → p. 78
PRIORITÄT	5
AKTEUR	Spieler → p. 17, Tutor → p. 18

ID	FA120
TITEL:	Einbindung in Benutzer-Client
BESCHREIBUNG:	Der KI-Client soll auch in den Benutzer-Client einbindbar sein, das heißt aus dem Benutzer-Client heraus gestartet werden.
BEGRÜNDUNG	Der Benutzer möchte vielleicht nicht selber spielen, sondern ein Spiel nur beobachten. Eine Einbindung in den Benutzer-Client ermöglicht den Start eines KI-Clients auch für menschliche Benutzer, die nicht wissen, wie man den KI-Client über die Kommandozeile startet
ABHÄNGIGKEITEN	FA114 → p. 77
PRIORITÄT	5
AKTEUR	menschlicher Benutzer → p. 17

Editor

ID	FA121
TITEL:	Konfiguration über graphische Benutzeroberfläche
BESCHREIBUNG:	<p>Der Benutzer kann über die graphische Benutzeroberfläche des Editors Konfigurationen vornehmen. Dabei gibt es die folgenden Konfigurationen:</p> <ul style="list-style-type: none"> • Partiekonfiguration (siehe FA124 → p. 81) • Szenariokonfiguration (siehe FA126 → p. 81) <p>Diese müssen in dem Editor erzeugt und bearbeitet werden können.</p>
BEGRÜNDUNG	Die Teilnehmer sollen das Spiel individualisieren können, um zum Beispiel verschiedene Schwierigkeitslevel zu erstellen. Außerdem ermöglicht eine ausgelagerte Konfiguration, dass der Entwickler spiel-relevante Daten nicht festlegt, sondern diese später leicht auch von Teilnehmern angepasst werden können.
ABHÄNGIGKEITEN	FA124 → p. 81, FA126 → p. 81
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Teilnehmer → p. 18, Server → p. 17

ID	FA122
TITEL:	Format der Konfigurationen
BESCHREIBUNG:	Die Szenario- und Partiekonfigurationen müssen in einem von dem Standardisierungskomitee definierten JSON-Schema erstellt werden. Die Konfigurationen werden in diesem Format in Dateien gespeichert und können aus diesen Daten zur weiteren Bearbeitung geladen werden.
BEGRÜNDUNG	Zur späteren Validierung und Verarbeitung durch den Server (Server wird unabhängig von Editor und Client entwickelt) müssen die Konfigurationen einem Schema folgen, wobei sich JSON als weitverbreitetes Format anbietet.
ABHÄNGIGKEITEN	FA121 → p. 80, FA123 → p. 80
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Spieler → p. 17, Server → p. 17

ID	FA123
TITEL:	Laden und Modifikation von Konfigurationen
BESCHREIBUNG:	Der Spieler soll im Editor erstellte Konfigurationen laden oder abspeichern können. Die geladenen Konfigurationen müssen validiert werden.
BEGRÜNDUNG	Der Spieler soll die Möglichkeit haben beliebte Konfigurationen wiederzuverwenden zu können, ohne sie immer neu erstellen zu müssen. Trotzdem soll der Server keine ungültigen Daten bekommen, weswegen die Konfigurationen validiert werden müssen
ABHÄNGIGKEITEN	FA121 → p. 80, FA122 → p. 80
PRIORITÄT	3
AKTEUR	Entwickler → p. 19, Spieler → p. 17
ID	FA124
TITEL:	Partiekonfiguration
BESCHREIBUNG:	Der Benutzer kann in dem Editor Konfigurationen zur Partie vornehmen. Damit kann er Einstellungen zu einer bestimmten Partie vornehmen.
BEGRÜNDUNG	Der Spieler soll Einstellungen für die Partie vornehmen können, um die Partien zu individualisieren und die Länge des der Partie beeinflussen zu können
ABHÄNGIGKEITEN	FA121 → p. 80, FA122 → p. 80
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Spieler → p. 17, Server → p. 17
ID	FA125
TITEL:	Parameter der Partiekonfiguration
BESCHREIBUNG:	Der Spieler kann mindestens folgende Punkte bezüglich einer Partie einstellen: <ul style="list-style-type: none"> • Zeitdauer der verschiedene Phasen • maximale Rundenanzahl, bis die Bedingung für überlange Runden eintritt
BEGRÜNDUNG	Diese Parameter müssen definiert werden.
ABHÄNGIGKEITEN	FA124 → p. 81, FA125 → p. 81
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Spieler → p. 17, Server → p. 17

ID	FA126
TITEL:	Szenariokonfiguration
BESCHREIBUNG:	Der Benutzer kann in dem Editor Konfigurationen zu den Szenarios vornehmen. Damit kann er Einstellungen zu den Szenarios vornehmen, das heißt das Spielfeld konfigurieren und weitere Kenngrößen des Spiels anpassen.
BEGRÜNDUNG	Der Spieler soll das Spielfeld und andere Kenngrößen des Spiels anpassen sollen, um die Partien zu individualisieren und die Schwierigkeit oder das Design des Spielfeldes selbst zu bestimmten.
ABHÄNGIGKEITEN	FA121 → p. 80, FA127 → p. 82 FA128 → p. 82, FA122 → p. 80
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Spieler → p. 17, Server → p. 17

ID	FA127
TITEL:	Parameter der Szenariokonfiguration
BESCHREIBUNG:	<p>Der Spieler kann mindestens folgende Punkte bezüglich eines Szenarios einstellen:</p> <ul style="list-style-type: none"> • soll ein Spielfeld automatisch generiert werden (siehe FA128 → p. 82) • Dimension des Spielfelds • Position und Größe der Städte, sowie die Landschaft, das heißt wo Gebirge oder Wüste ist. • Darstellung, das heißt bestimmte Farben oder Texturen der Charaktere
BEGRÜNDUNG	Diese Parameter müssen definiert werden.
ABHÄNGIGKEITEN	FA126 → p. 81
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19, Spieler → p. 17, Server → p. 17

ID	FA128
TITEL:	zufällige Szenariokonfiguration
BESCHREIBUNG:	Der Spieler soll im Editor die Möglichkeit haben, ein Szenario zufällig generieren zu lassen. Dabei soll der Spieler ebenso die Möglichkeit haben, in den Generierungsprozess einzugreifen und zum Beispiel die minimale und maximale Dimension des Spielfeldes einzustellen oder im Nachhinein das Spielfeld anpassen können.
BEGRÜNDUNG	Die Konfiguration eines Spielfeldes ist aufwendig und erfordert Erfahrung. Eine automatische Generierung oder Konfiguration erleichtert den Spieleinstieg und spart Zeit. Jedoch soll trotzdem noch nachjustiert werden können.
ABHÄNGIGKEITEN	FA126 ^{→ p. 81}
PRIORITÄT	3
AKTEUR	Entwickler ^{→ p. 19} , Spieler ^{→ p. 17}

3 Softwarespezifikation

In den folgenden Kapiteln wird für jede Komponente eine sinnvolle Schnittstellenart vorgestellt und im Falle einer graphischen Oberfläche die definierten Dialoge vorgestellt. Dafür werden den Dialogen die entsprechenden Anwendungsfälle und Komponenten zugeordnet und die Struktur der Dialoge, das heißt, wie die Dialoge miteinander zusammenhängen, in einem Dialogstrukturdigramm dargestellt. Zudem ist für jeden Dialog eine graphische Skizze in Form eines „Mockups“ zu finden, sowie eine textuelle Beschreibung, wie dieser Dialog benutzt werden kann.

3.1 Systemschnittstellen

Das System bzw. Spiel DESERTS OF DUNE besteht hauptsächlich aus vier großen Komponenten (siehe FA2^{→ p. 38}). Im Folgenden wird für jede Komponente eine Schnittstellenart definiert:

Server Der Server soll über eine Konsole bedient werden, das heißt als Schnittstelle eine Kommandozeile besitzen.

Dies ist einerseits eine Anforderungen aus dem Lastenheft (siehe FA85^{→ p. 66}), andererseits bietet die Kommandozeile den Vorteil, dass sie für die Entwickler weniger Arbeitsaufwand bedeutet und die Benutzer den Server so einfach und schnell über die Konsole starten können. Da für die Aufgabe eine Konsole also besser geeignet ist und keine weitere Interaktion besteht, ist eine Kommandozeile ausreichend.

Benutzer-Client Der Benutzer-Client soll über eine graphische Benutzeroberfläche bedient werden, das heißt als Schnittstelle eine GUI besitzen.

Der Benutzer-Client dient hauptsächlich dem Spielen des Spiels und dementsprechend auch der Anzeige des aktuellen Spielzustands und Spielfeldes. Daher ist eine graphische Oberfläche ein schöneres Spielerlebnis, als die Daten textuell zu sehen.

Da die Benutzer ebenfalls mit dem Umgang einer Kommandozeile nicht vertraut sein werden, eignet sich eine graphische Benutzeroberfläche bezüglich der Eingabe besser.

KI-Client Der KI-Client soll über eine Konsole bedient werden, das heißt als Schnittstelle eine Kommandozeile besitzen.

Dies ist einerseits Anforderungen aus dem Lastenheft (siehe FA114^{→ p. 77}), andererseits bietet die Kommandozeile den Vorteil, dass sie für die Entwickler weniger Arbeitsaufwand bedeutet und die Benutzer den KI-Client so einfach und schnell über die Konsole starten können. Da für die Aufgabe eine Konsole also besser geeignet ist und keine weitere Interaktion besteht, ist eine Kommandozeile ausreichend.

Jedoch besteht auch die Möglichkeit, den KI-Client aus dem Benutzer-Client heraus zu starten, was bedeuten würde, dass der KI-Client eine graphische Schnittstelle hätte. Da diese jedoch in den Benutzer-Client eingebettet wäre, würde es keine extra Schnittstelle sein.

Editor Der Editor soll über eine graphische Benutzeroberfläche bedient werden, das heißt als Schnittstelle eine GUI besitzen.

In dem Editor sollen die Benutzer das Spiel und damit insbesondere das Spielfeld konfigurieren. Das bedeutet, dass die Benutzer viele Interaktionen mit dem System vornehmen müssen und vermutlich keine besonderen Kenntnisse haben, da sie das Spiel spielen wollen. Dementsprechend ist eine graphische Benutzeroberfläche komfortabler und intuitiver zu bedienen, da viele Benutzer den Umgang mit einer Kommandozeile vermutlich nicht gewohnt wären. Außerdem kann der Benutzer so einfacher ein Szenario konfigurieren und sieht das Ergebnis sofort.

3.1.1 Dialogstruktur

Da sowohl der Editor, als auch der Benutzer-Client eine graphische Benutzeroberfläche haben sollen, müssen für diese Oberflächen die entsprechenden Dialoge definiert werden und wie diese voneinander abhängen. Dies wird in den folgenden Dialogstrukturdigrammen zusammengefasst³

³Dabei sind nicht alle Dialoge in einem Diagramm zusammengefasst, sondern die Dialoge in mehrere Diagramme aufgeteilt und dann in den entsprechenden Diagramme verwiesen mit «Subdiagramm».

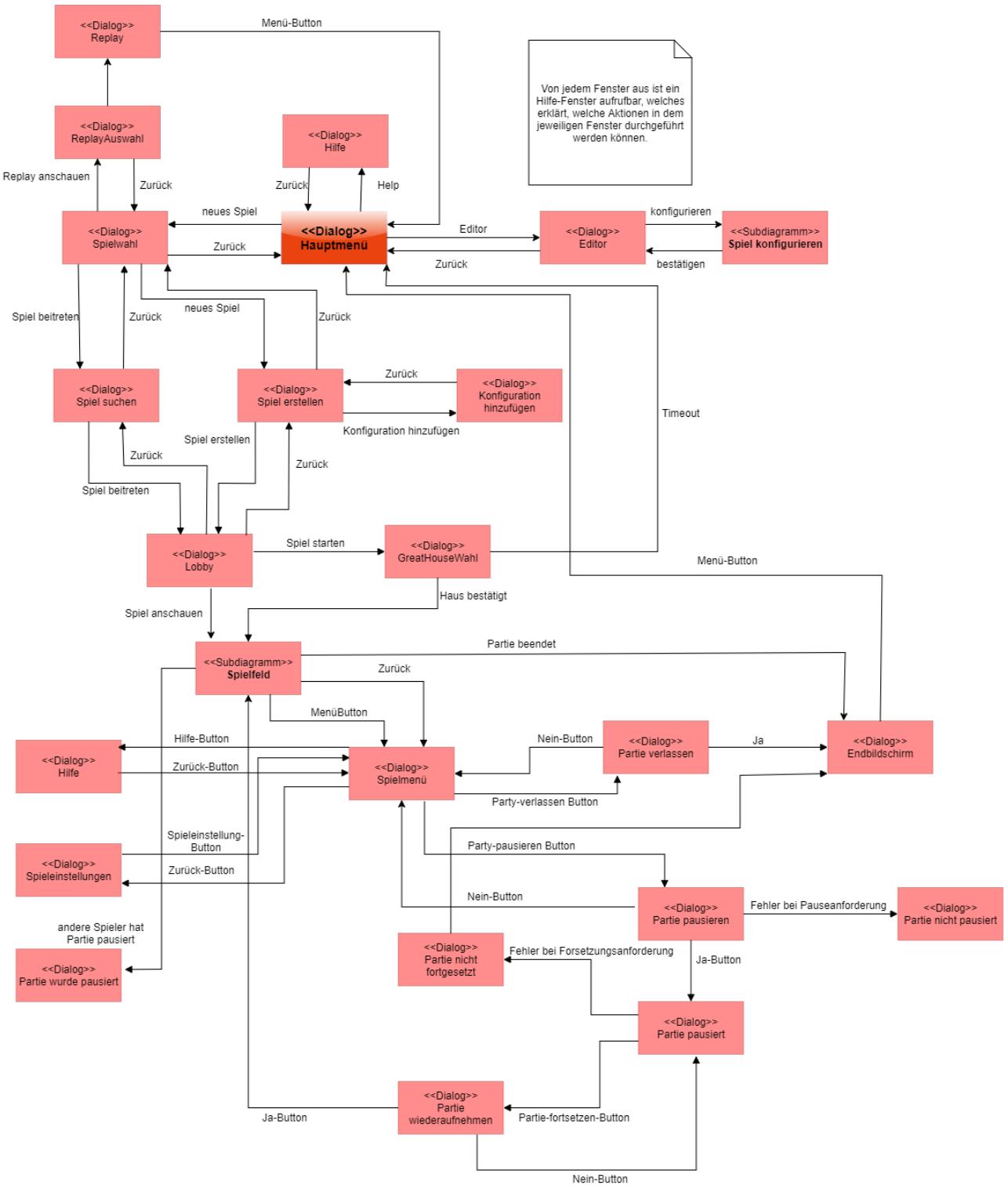


Abbildung 25: Dialogstruktur für den grundlegenden Spielstart und -ablauf

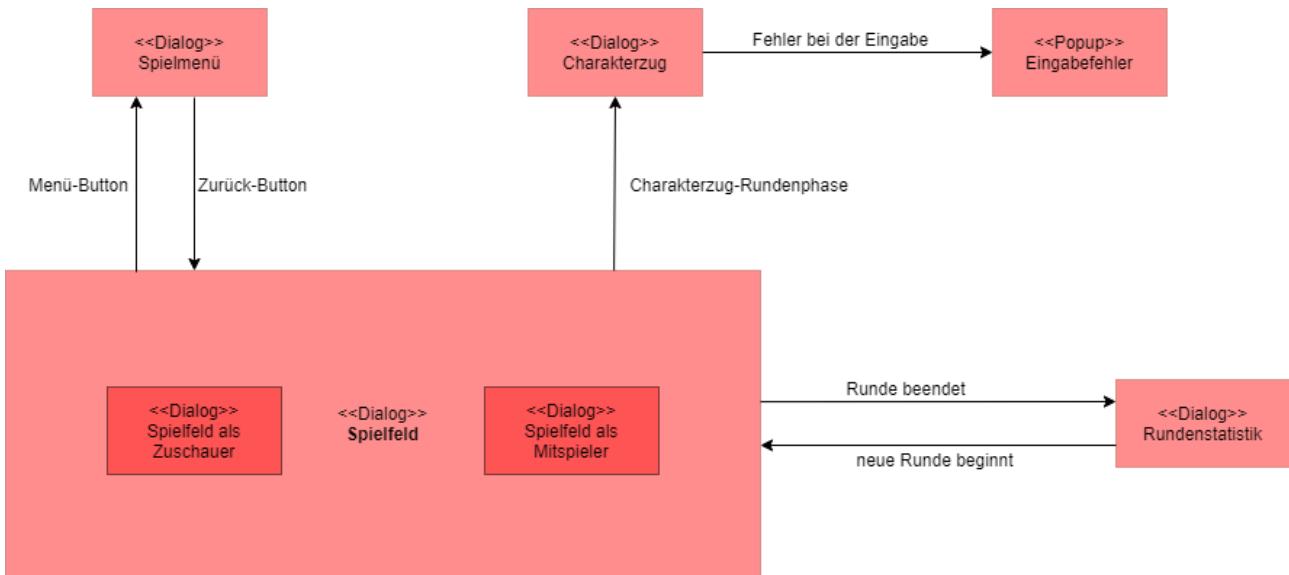


Abbildung 26: Dialogstruktur für das Spielfeld und die Optionen während des Spiels. Generell gibt es einen Dialog für das Spielfeld, welcher jedoch je nach Benutzer anders aussieht. Wenn das Spielfeld für einen Zuschauer aufgerufen wird, dann sieht das Spielfeld anders aus, als bei einem Spieler. Deswegen gibt es spezialisierte Dialoge für das „Spielfeld“

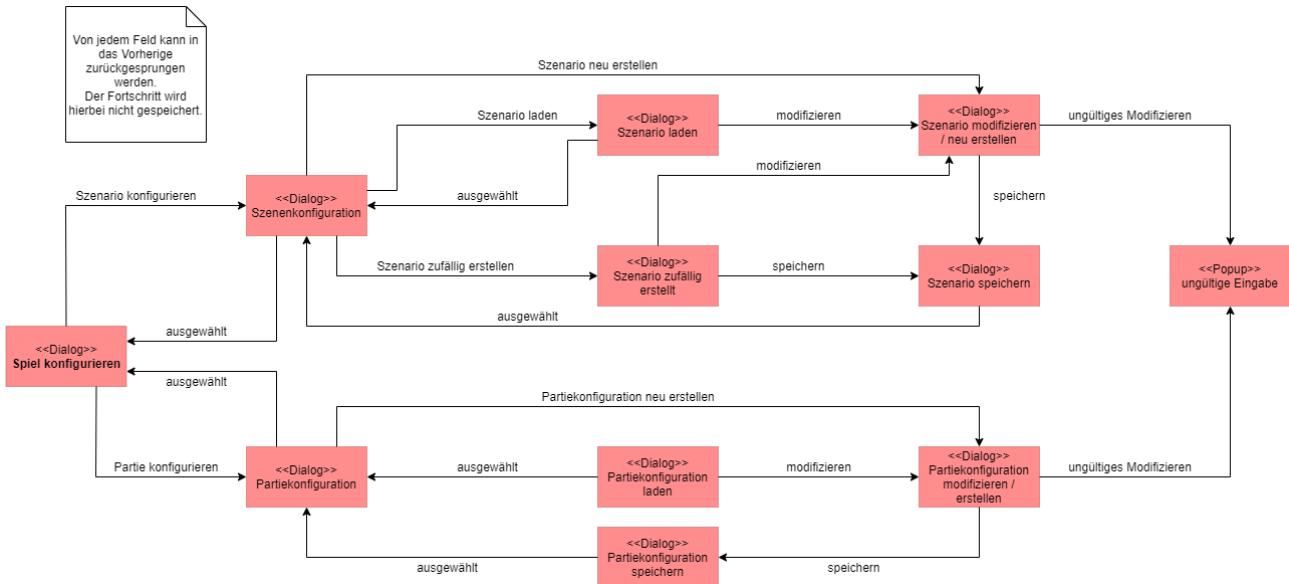


Abbildung 27: Dialogstruktur für die Spielkonfiguration

Das für das Spiel zu viele Dialoge existieren wurden die Dialoge bzw. das Dialogstrukturdigramm der Übersichtlichkeit halber aufgeteilt. In das jeweilige Subdiagramm kommt man, wenn man einen Dialog erreicht, der mit «Subdiagramm» gekennzeichnet ist.

Dabei gibt es folgende Zuordnung der Dialoge und Anwendungsfällen aus Unterunterabschnitt 2.2.2, wobei in dem Kapitel auch die entsprechenden Anforderungen zugeordnet sind:

Dialog	zugeordneter Anwendungsfall	Typ
Replay	Replay abspielen	Dialog
Replay/Auswahl	Replay abspielen	Dialog
Hilfe	kein konkreter Anwendungsfall; siehe QA8 → p. 109	
Spielwahl	Teilnahme als Zuschauer / Mitspieler	Dialog
Hauptmenü	Teilnahme als Zuschauer / Mitspieler	Dialog
Editor	Spiel konfigurieren und laden	Dialog
Spiel suchen	Teilnahme als Zuschauer / Mitspieler	Dialog
Spiel erstellen	Teilnahme als Zuschauer / Mitspieler	Dialog
Konfiguration hinzufügen	Spiel konfigurieren und laden	Dialog
Lobby	Teilnahme als Zuschauer / Mitspieler	Dialog
Great House Wahl	Great House Wahl	Dialog
Spielmenü	Partie pausieren und fortsetzen	Dialog
Partie wurde pausiert	Partie pausieren und fortsetzen	Hinweisfenster
Partie verlassen	Partie beenden	Bestätigungsfenster
Endbildschirm	Partie beenden	Hinweisfenster
Partie pausieren	Partie pausieren und fortsetzen	Bestätigungsfenster
Partie nicht pausiert	Partie pausieren und fortsetzen	Fehlerbenachrichtigung
Partie pausiert	Partie pausieren und fortsetzen	Hinweisfenster
Partie nicht fortgesetzt	Partie pausieren und fortsetzen	Fehlerbenachrichtigung
Partie wiederaufnehmen	Partie pausieren und fortsetzen	Bestätigungsfenster
Spielfeld	Spieldfeld und -zustand anzeigen	Dialog
Charakterzug	Charakterzug (Bewegung und Aktion von Charakter)	Dialog
Eingabefehler	Charakterzug (Bewegung und Aktion von Charakter)	Fehlerbenachrichtigung
Rundenstatistik	Spieldfeld und -zustand anzeigen	Hinweisfenster
Spiel konfigurieren	Spiel konfigurieren und laden	Dialog
Szenariokonfiguration	Szenario erstellen und konfigurieren	Dialog
Szenario laden	Szenario erstellen und konfigurieren	Dialog
Szenario modifizieren / neu erstellen	Szenario erstellen und konfigurieren	Dialog
Szenario zufällig erstellt	Szenario erstellen und konfigurieren	Dialog
Szenario speichern	Szenario erstellen und konfigurieren	Bestätigungsfenster
ungültige Eingabe	Spieldfeld konfigurieren und laden	Fehlerbenachrichtigung
Partiekonfiguration laden	Partie konfigurieren	Dialog
Partiekonfiguration modifizieren / neu erstellen	Partie konfigurieren	Dialog
Partiekonfiguration speichern	Partie konfigurieren	Bestätigungsfenster
Partiekonfiguration	Partie konfigurieren	Dialog

3.1.2 Graphische Gestaltung und Nutzungskonzept

Im Folgenden sind die verschiedenen graphischen Prototypen in Form von „Mockup“-Zeichnung aufgelistet. Für jeden graphischen Prototypen wird in textuellen Form zusätzlich beschrieben, was die Elemente auf der Oberfläche bedeuten und wie dieser Dialog bedient werden kann.



Abbildung 28: **Hauptmenü:**

Der „Hauptmenü“-Dialog wird beim Starten der Anwendung zu Beginn angezeigt. In diesem Dialog hat der Benutzer die Möglichkeit den Editor zu starten, ein neues Spiel zu starten oder die Anwendung wieder zu schließen, indem er die entsprechenden Buttons drückt.



Abbildung 29: **Hilfe:**

Der „Hilfe“-Dialog kann von jedem Dialog, bei dem das Fragezeichen rechts oben im Eck, aufgerufen werden und hier wird zu dem Dialog, wovon es aufgerufen wurde, eine kurze Beschreibung zu dem jeweiligen Dialog stehen

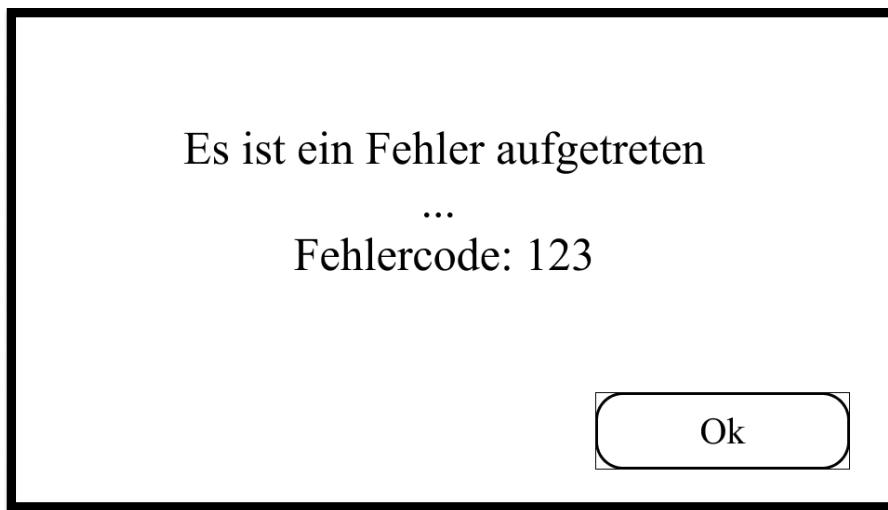


Abbildung 30: **allgemeine Fehlerbenachrichtigung (analog Hinweisfenster):**

Der „Fehlerbenachrichtung“-Dialog taucht immer dann auf, wenn es zu einem Fehler kam. Dieser Dialog zeigt dem Benutzer einen Text an und der Dialog lässt sich schließen, wenn der Benutzer den *OK*-Button betätigt. Damit bestätigt er, dass er den Text gelesen hat oder das Fenster nicht mehr sehen will.

Der Dialog für ein Hinweisfenster sieht identisch aus, nur das ein anderer Text angezeigt wird.

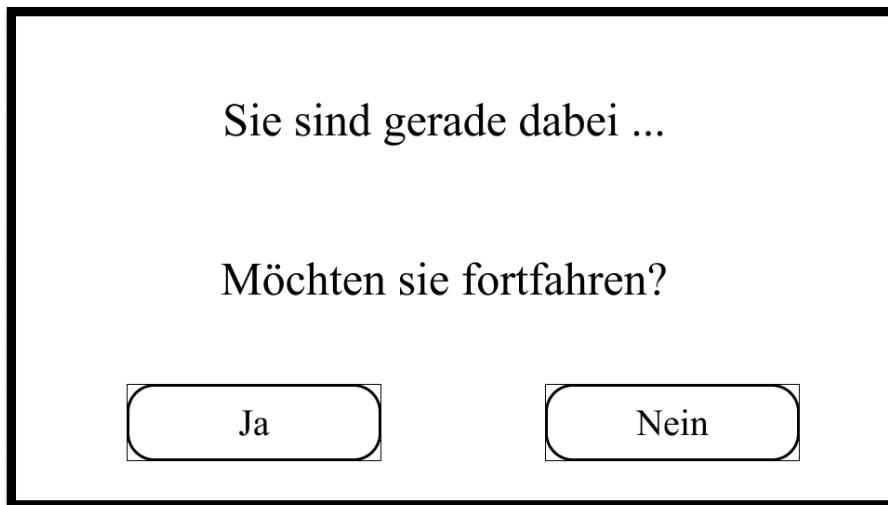


Abbildung 31: **Bestätigungsfenster:**

Der Dialog „Bestätigungsfenster“ wird immer dann dem Benutzer angezeigt, wenn eine wichtige Aktion des Benutzers bestätigt werden soll, zum Beispiel wenn der Benutzer das Spiel verlassen will. Dann hat der Nutzer, die Aktion mit *Ok* zu bestätigen oder mit *Abbrechen* zu dem Dialog zurückkehren, von dem er gekommen ist.



Abbildung 32: **Spielwahl:** Hier kann der Nutzer auswählen, ob er einem bereits erstellten Spiel beitreten will, selbst ein Spiel erstellen möchte oder ein vergangenes Spiel nochmals anschauen will.

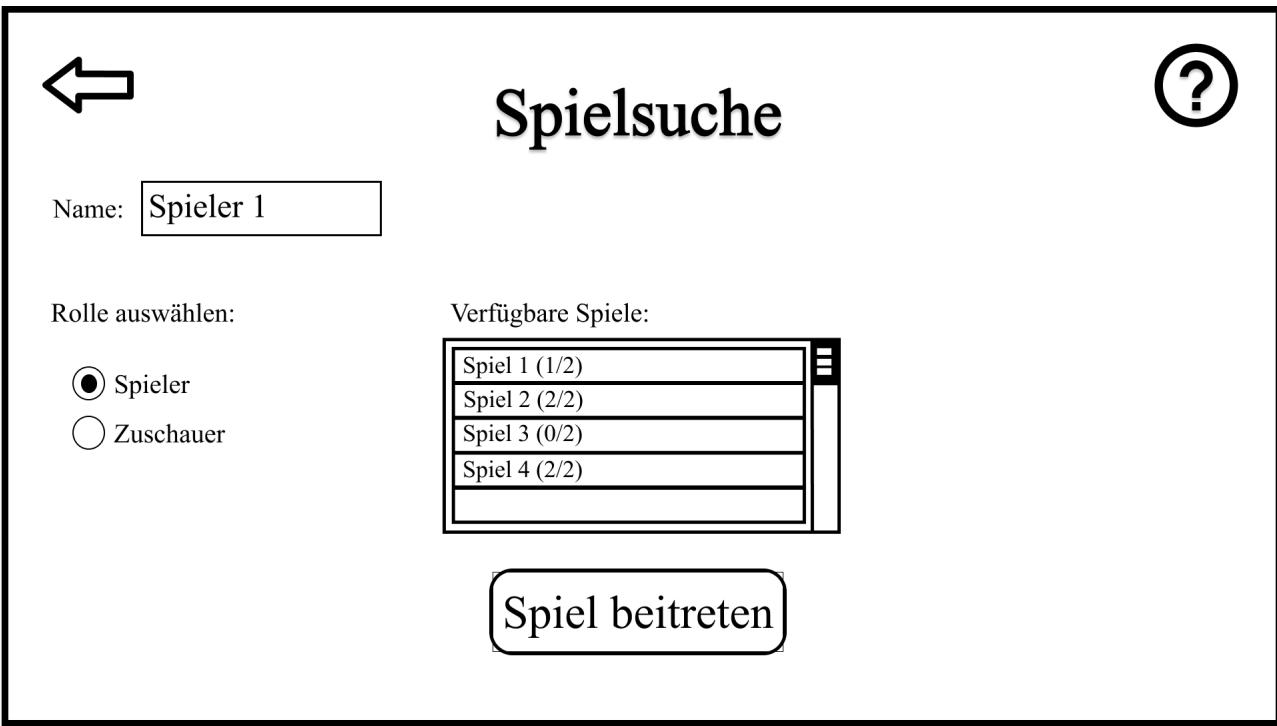


Abbildung 33: **Spiel suchen:**

Wurde im Spielwahl-Dialog der „Spiel beitreten“-Button gedrückt, kann der Nutzer jetzt einem bereits erstellten Spiel beitreten.

Hier kann der Nutzer seinen Namen eingeben und seine Rolle wählen, also ob er als Spieler oder als Zuschauer beitreten möchte. Dem Nutzer werden alle verfügbaren Spiele angezeigt und wenn in einem Spiel schon zwei Spieler sind, kann man dort nur noch als Zuschauer beitreten. Durch drücken auf den „Spiel beitreten“-Button kommt der Nutzer in den „Lobby“-Dialog.

The dialog box has a black border and rounded corners. In the top-left corner is a black arrow pointing left, and in the top-right corner is a circle containing a question mark. The title 'Spiel erstellen' is centered at the top in a large, bold, serif font. Below the title are several input fields and buttons:

- A text input field labeled 'Name:' containing the value 'Spieler 1'.
- A label 'Rolle auswählen:' followed by two radio buttons. The first radio button, labeled 'Spieler', is selected (indicated by a black dot). The second radio button, labeled 'Zuschauer', is not selected.
- A text input field labeled 'Spielname:' containing the value 'Spiel 1'.
- A dropdown menu labeled 'Konfiguration auswählen:' with the option 'Konfiguration 1' selected. A small downward arrow icon is at the bottom right of the dropdown.
- A button labeled 'Konfiguration hinzufügen' with a black border and rounded corners.
- A large, rounded rectangular button at the bottom labeled 'Spiel erstellen'.

Abbildung 34: **Spiel erstellen:**

Wurde im Spielwahl-Dialog der „Neues Spiel“-Button gedrückt, kann der Nutzer jetzt ein neues Spiel erstellen.

Er kann dabei seinen Namen eingeben und den Namen des Spiels. Außerdem kann er auch hier seine Rolle auswählen, ob er Spieler oder Zuschauer sein möchte. Er hat zusätzlich ein Dropdown-Menü, bei dem er eine bereits hinzugefügte Konfiguration auswählen kann oder er kann auch selbst eine neue Spielkonfiguration hinzufügen mit dem „Konfiguration hinzufügen“-Button.

The dialog box has a black border and rounded corners. In the top-left corner is a black arrow pointing left, and in the top-right corner is a circle containing a question mark. The title 'Konfiguration hinzufügen' is centered at the top in a large, bold, serif font. Below the title is an input field:

- A text input field labeled 'Pfad für Konfiguration eingeben:' containing the value 'home/dod/konfiguration/konfiguration1.dod'.
- A large, rounded rectangular button at the bottom labeled 'Konfiguration hinzufügen'.

Abbildung 35: **Konfiguration hinzufügen:**

Wurde im „Spiel erstellen“-Dialog der „Konfiguration hinzufügen“-Button gedrückt, kommt man in diesen Dialog, bei dem man in einer Textbox den Pfad und den Dateinamen für eine erstellte Konfiguration eingeben kann und dieses dann hinzugefügt wird.

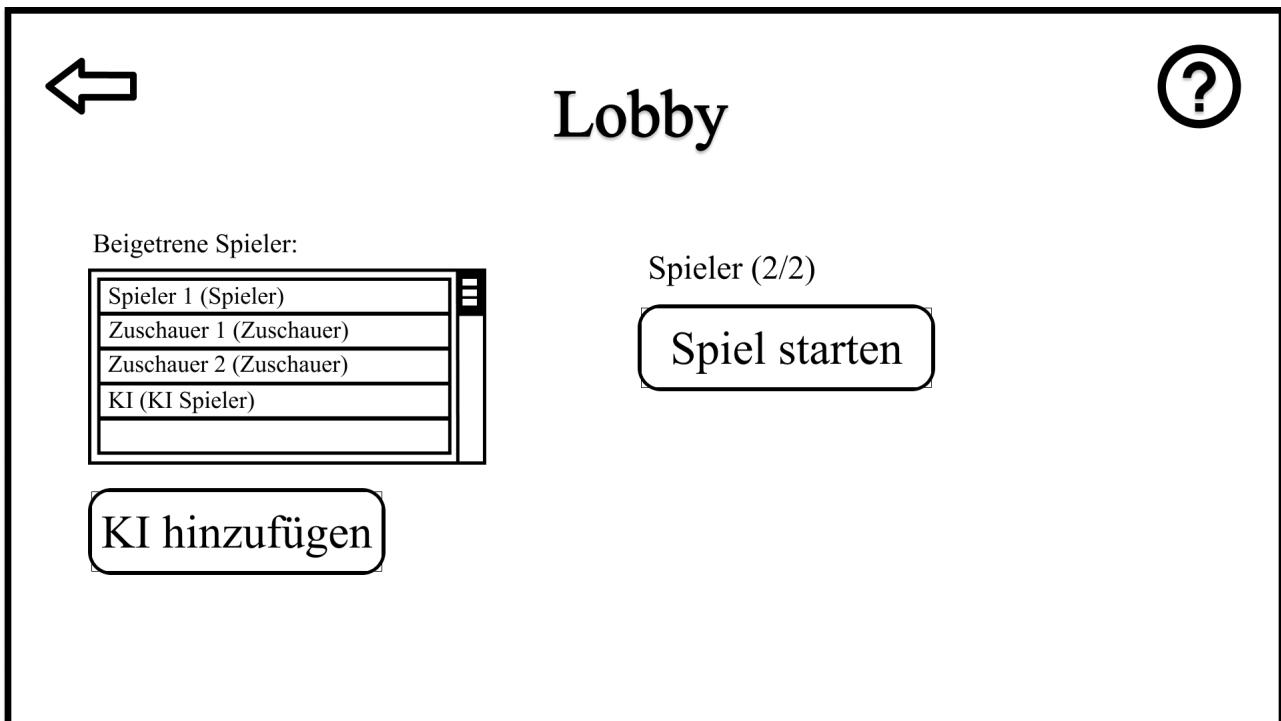


Abbildung 36: **Lobby:**

In dem „Lobby“-Dialog werden in einer Liste alle bereits beigetretenen Spieler und Zuschauer angezeigt. Hier besteht auch die Option einen KI-Spieler über einen Button hinzuzufügen. Außerdem ist hier noch ein „Spiel starten“-Button, womit man das Spiel starten kann unter der Bedingung, dass zwei Spieler in der Lobby sind.

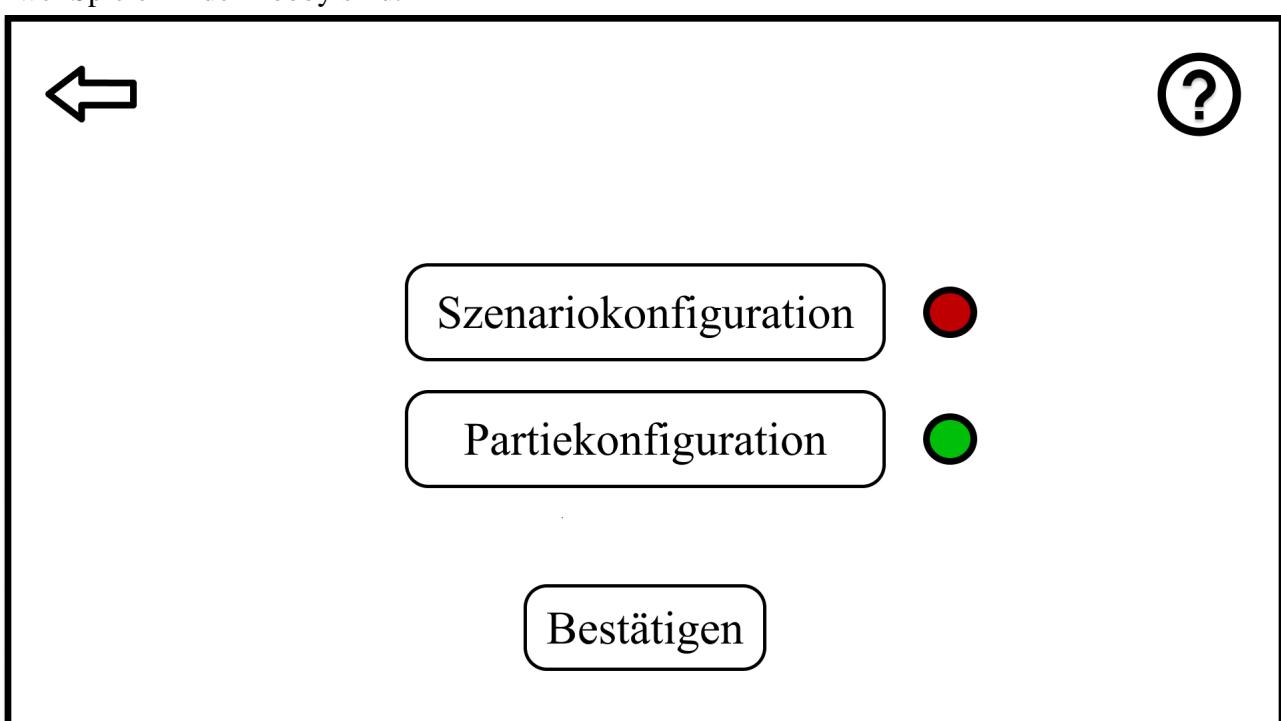
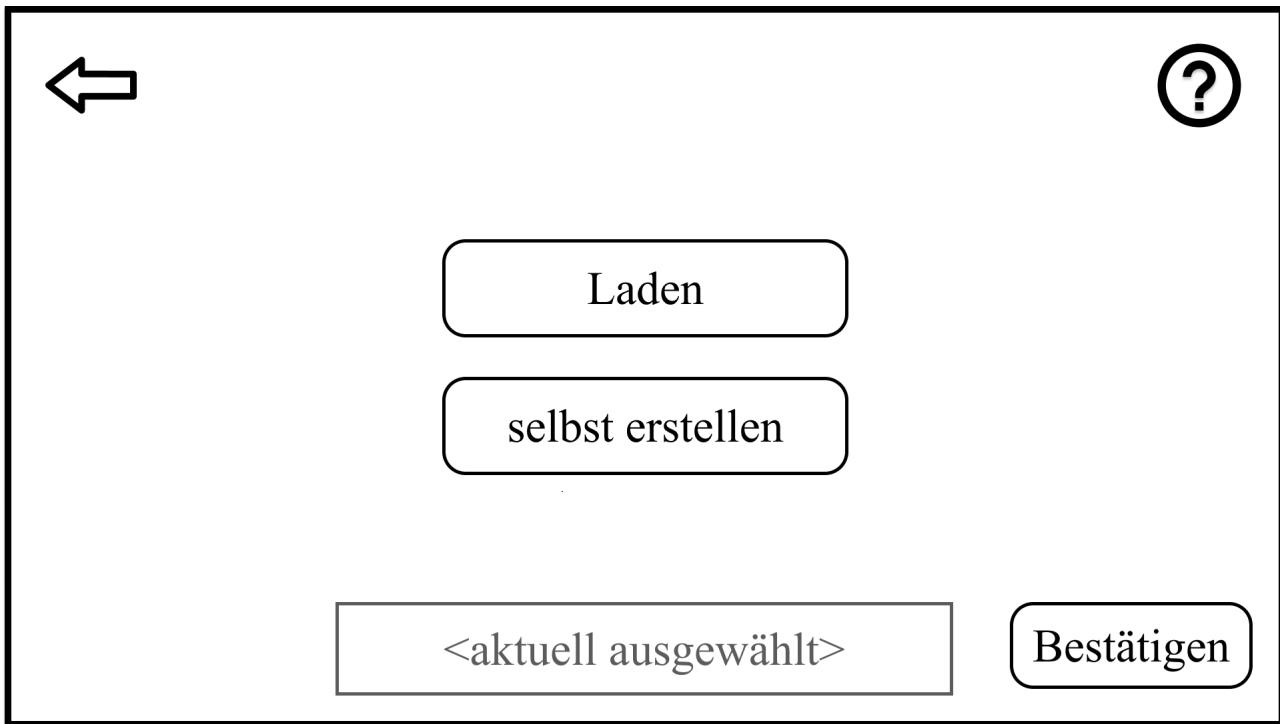
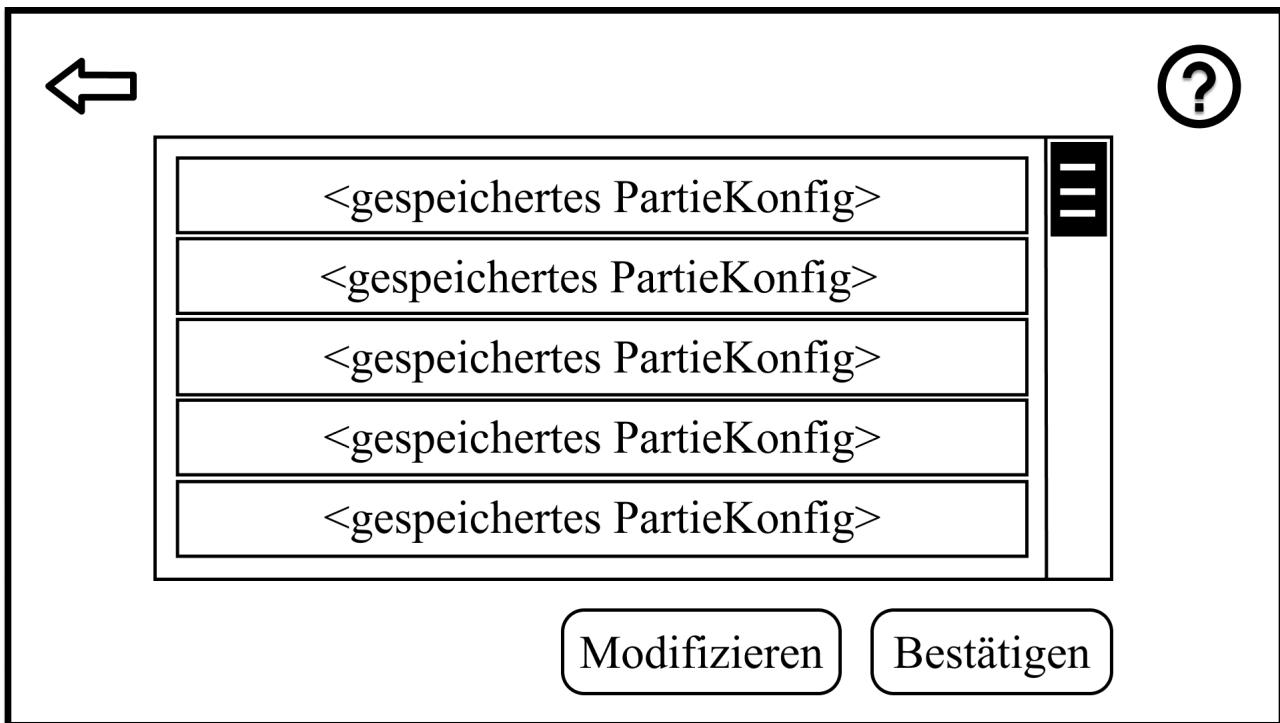


Abbildung 37: **Spiel-Konfiguration:**

Um eine gültige Spielkonfiguration zu erstellen, muss der Benutzer sowohl eine Scenario-Konfiguration, als auch eine Partie-Konfiguration auswählen.

Abbildung 38: **Partie-Konfiguration:**

Der Benutzer kann entweder eine gespeicherte Partie-Konfiguration laden, oder eine neue Partie-Konfiguration erstellen. Der Benutzer kann seine Auswahl bestätigen, aber auch nochmals ändern.

Abbildung 39: **Partie-Konfiguration laden:**

Der Benutzer sieht eine Liste mit gespeicherten Partie-Konfigurationen und kann eine beliebige auswählen oder modifizieren.

The screenshot shows a user interface for creating or modifying a party configuration. At the top left is a back arrow icon. At the top right is a help icon (a question mark inside a circle). On the right side of the screen is a vertical toolbar with a list icon. The main area contains a table with five rows, each consisting of a parameter name and its corresponding value. The table has two columns: 'parametername' and 'wert'. The rows are as follows:

<parametername>	<wert>

At the bottom right of the main area is a rounded rectangular button labeled 'Speichern'.

Abbildung 40: **Partie-Konfiguration erstellen / modifizieren:**

Der Benutzer legt für jeden Parameter aus der angezeigten Liste einen gewünschten Wert fest. Der Benutzer kann die erstellte/ modifizierte Partie-Konfiguration speichern.

The screenshot shows a confirmation dialog box. The title is 'Partiekonfiguration speichern'. Inside the dialog, there is a label 'Konfig Name' followed by a text input field containing '< Name der Konfig>'. At the bottom are two buttons: 'Ja' (Yes) on the left and 'Abbrechen' (Cancel) on the right.

Abbildung 41: **Partie-Konfiguration speichern:**

Der Benutzer kann eine soeben erstellte/ modifizierte Partie-Konfiguration speichern, indem er sie benennt und bestätigt.

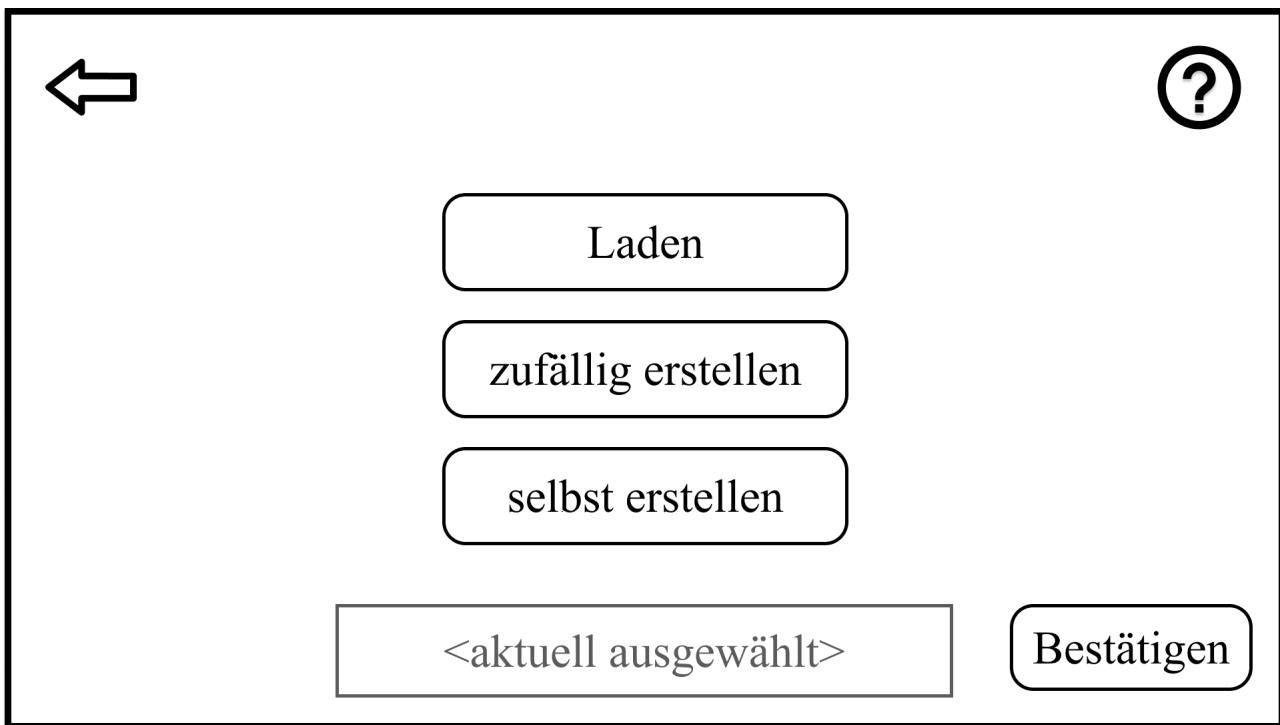


Abbildung 42: **Szenario-Konfiguration:**

Der Benutzer kann entweder eine gespeicherte Szenario-Konfiguration laden, selbst eine Neue erstellen, oder eine Szenario-Konfiguration zufällig erstellen lassen. Der Benutzer kann seine Auswahl bestätigen, aber auch nochmals ändern.

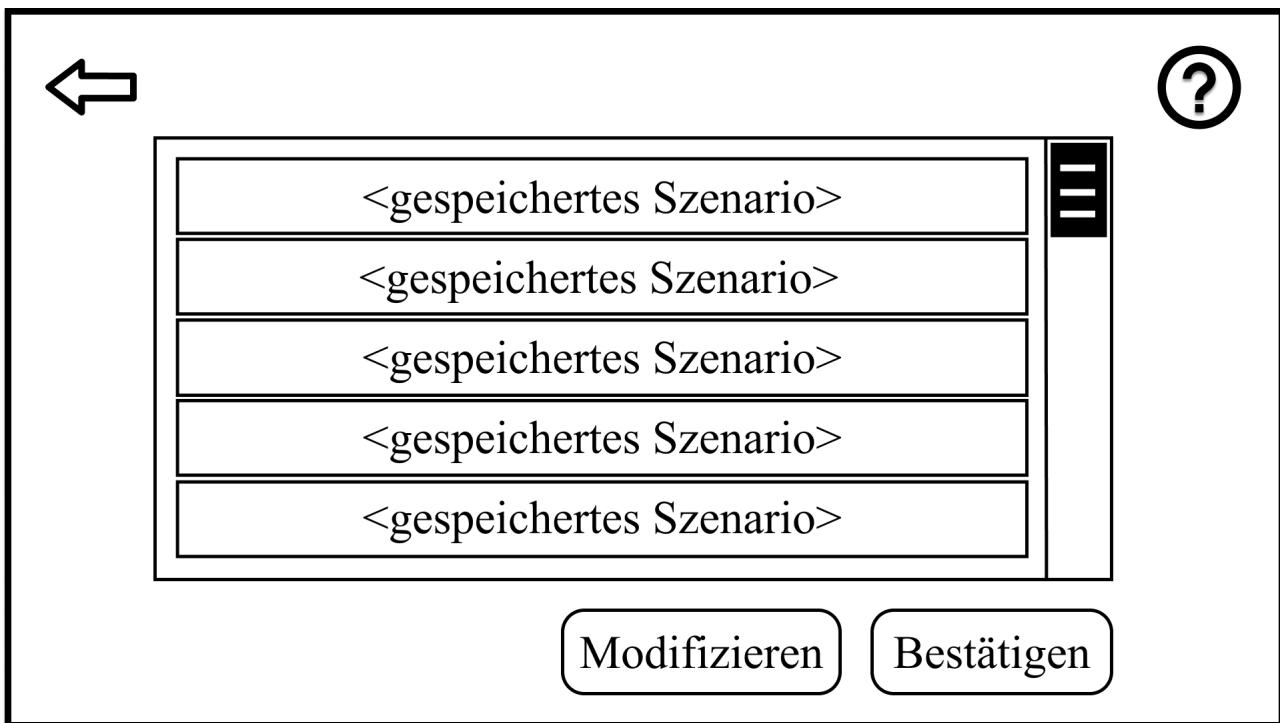


Abbildung 43: **Szenario laden:**

Der Benutzer sieht eine Liste mit gespeicherten Szenario-Konfigurationen und kann eine Beliebige auswählen oder modifizieren.

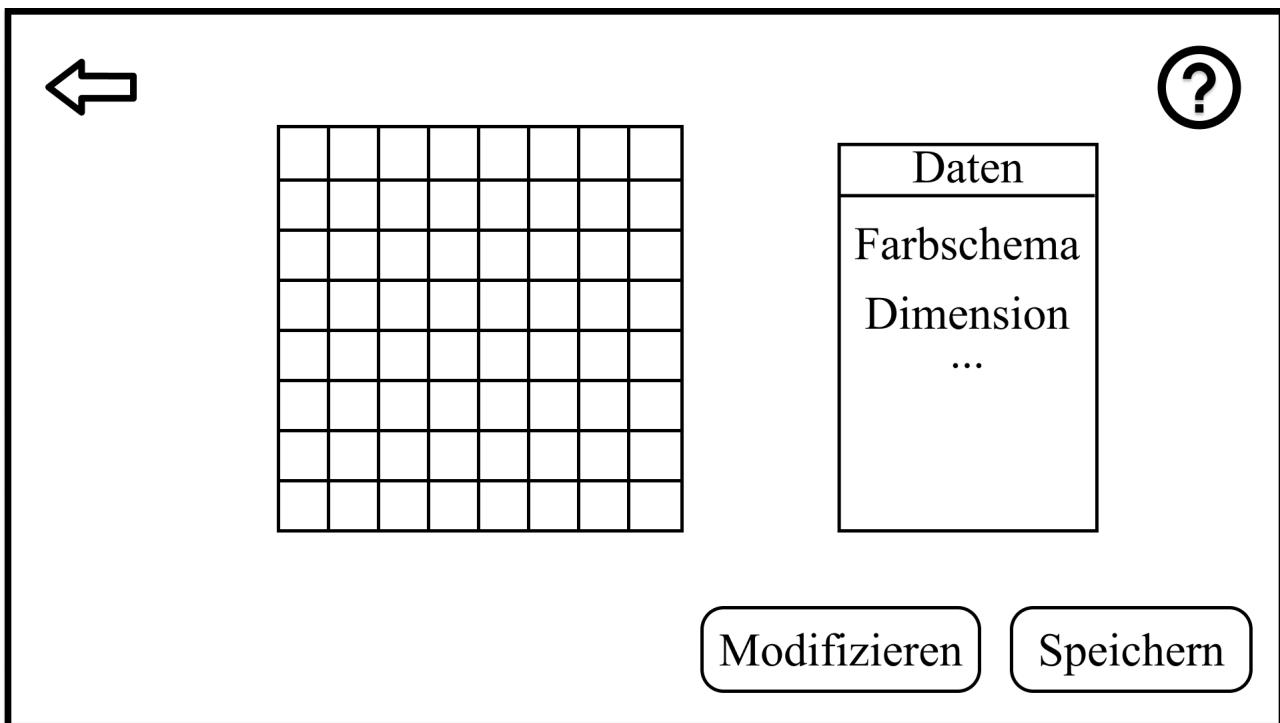


Abbildung 44: **Szenario zufällig erstellen:**

Der Benutzer kann eine Szenario-Konfiguration zufällig erstellen lassen und bekommt diese dann mit ihren zugehörigen Daten angezeigt. Der Benutzer kann die Szenario-Konfiguration direkt speichern oder nochmal selbst modifizieren.

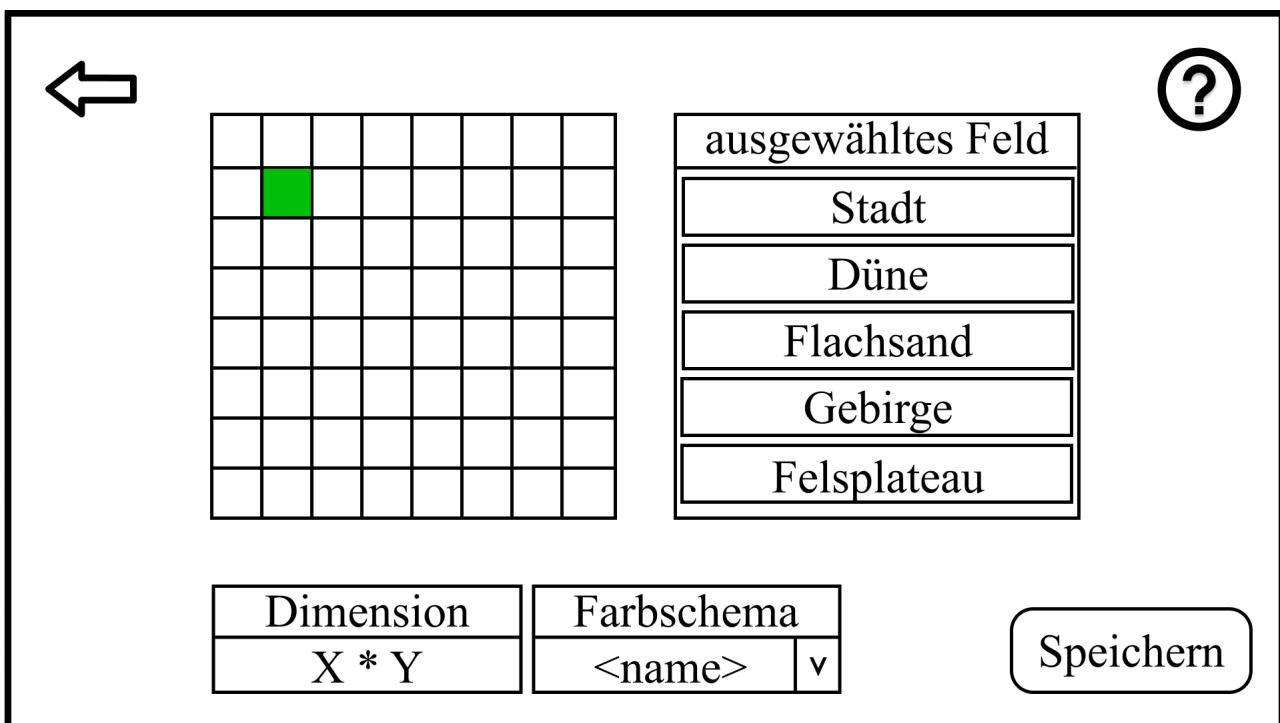


Abbildung 45: **Szenario erstellen / modifizieren:**

Der Benutzer kann verschiedene Werte, wie Dimension oder Farbschema festlegen. Der Benutzer kann für die einzelnen Felder die jeweilige Feldart festlegen.

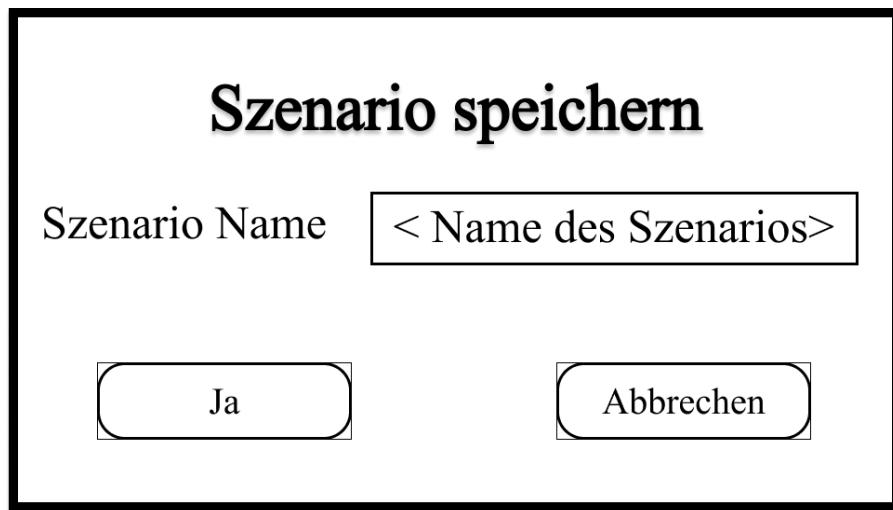


Abbildung 46: **Szenario speichern:**

Der Benutzer kann eine soeben erstellte/ modifizierte Szenario-Konfiguration speichern, indem er sie benennt und bestätigt.

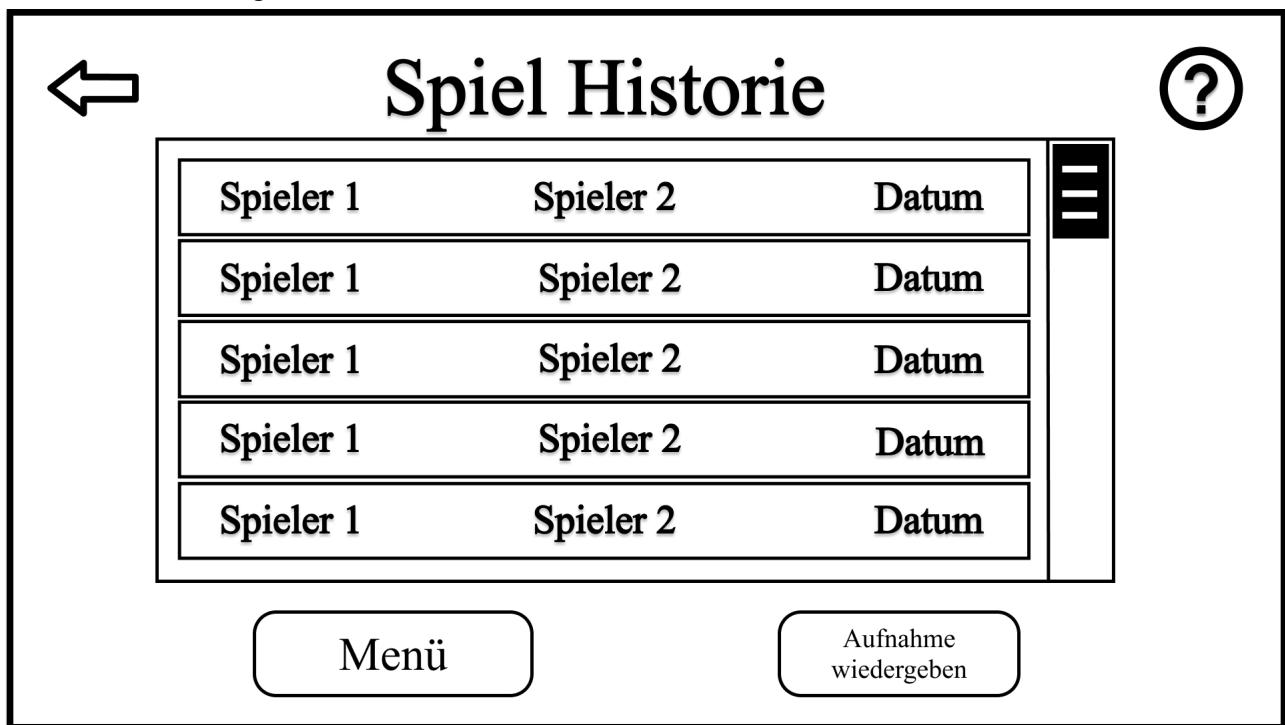


Abbildung 47: **ReplayAbspielen:**

Der Benutzer kann eine Spielaufzeichnung durch Anklicken auswählen und mit Klicken des Knopfes Aufnahmen wiedergeben die Wiedergabe starten. Durch Klicken des Knopfes Menü kann der Spieler in das Menü zurückkehren.

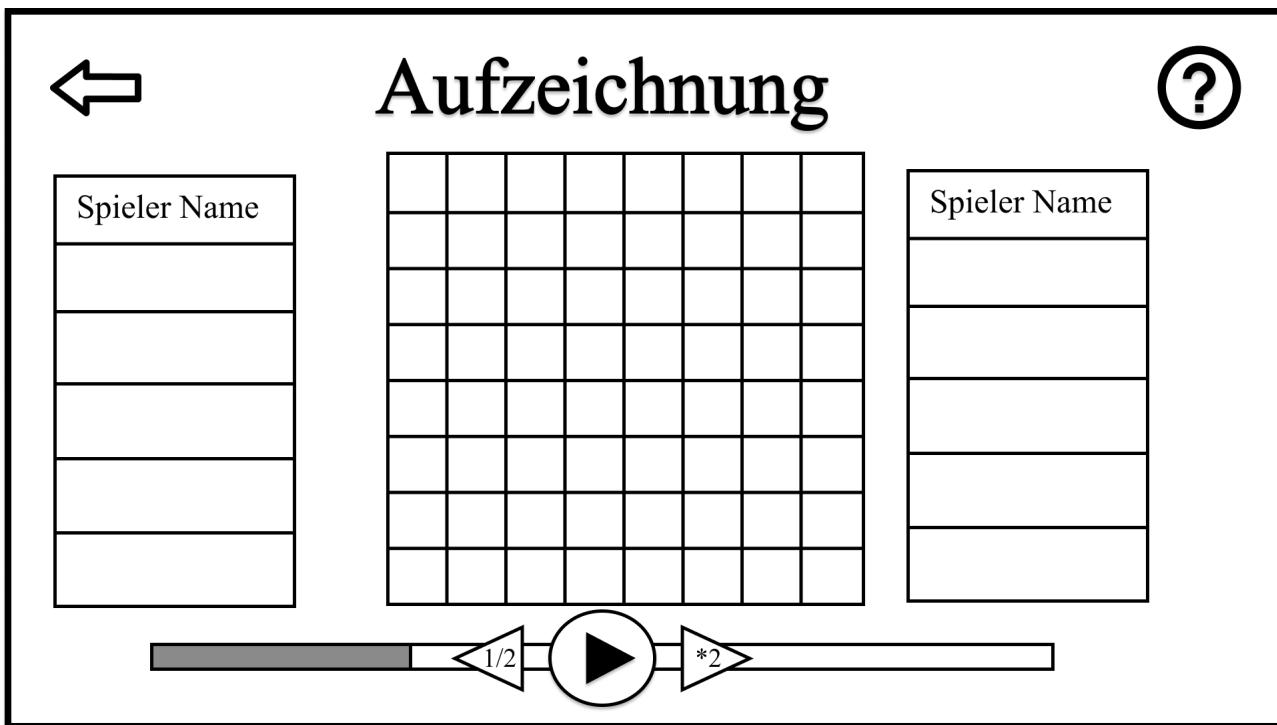


Abbildung 48: **Aufzeichnung:**

Der Spieler kann die Aufzeichnung durch Klicken der Taste zurück in das Menü zurückkehren. Zudem kann der Spieler mit dem runden Knopf die Aufzeichnung pausieren, und mit den Knöpfen rechts und links davon die Aufzeichnung doppelt so schnell oder halb so schnell abspielen.

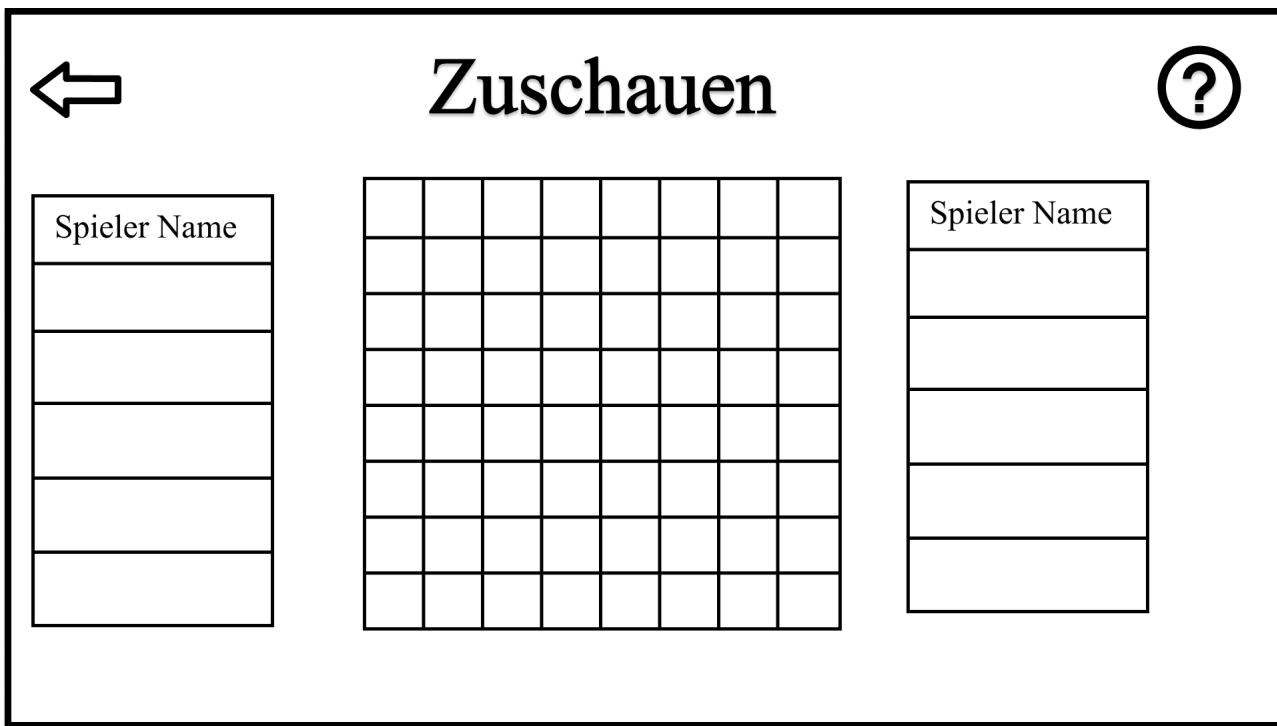


Abbildung 49: **Zuschauen:**

Der Nutzer hat keine Möglichkeit das Spiel zu beeinflussen und schneller oder langsamer abzuspielen. Der Nutzer kann lediglich alles sehen, was auch im Spiel sichtbar ist. Der Nutzer kann vom zuschauen Bildschirm in das Menü wechseln, indem er den Knopf Menü klickt.



Abbildung 50: **GreatHouseWahl**:

Der Nutzer kann durch die Radiobuttons House 1 und House 2 eines der beiden Häuser markieren. Durch Selektieren des Buttons bestätigt der Spieler seine Auswahl. Wenn der Nutzer über ein definiertes Zeitintervall keine Auswahl bestätigt, wird die Partie abgebrochen. Der Nutzer kehrt dann in die Lobby zurück.

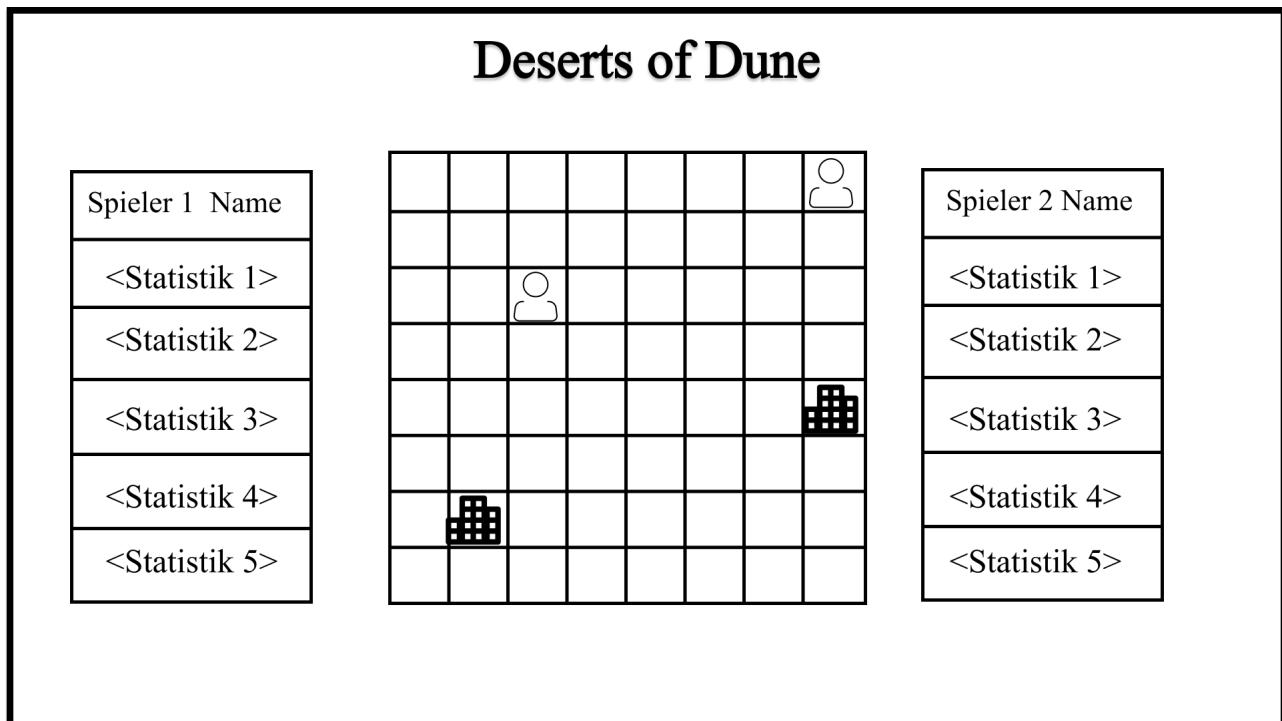
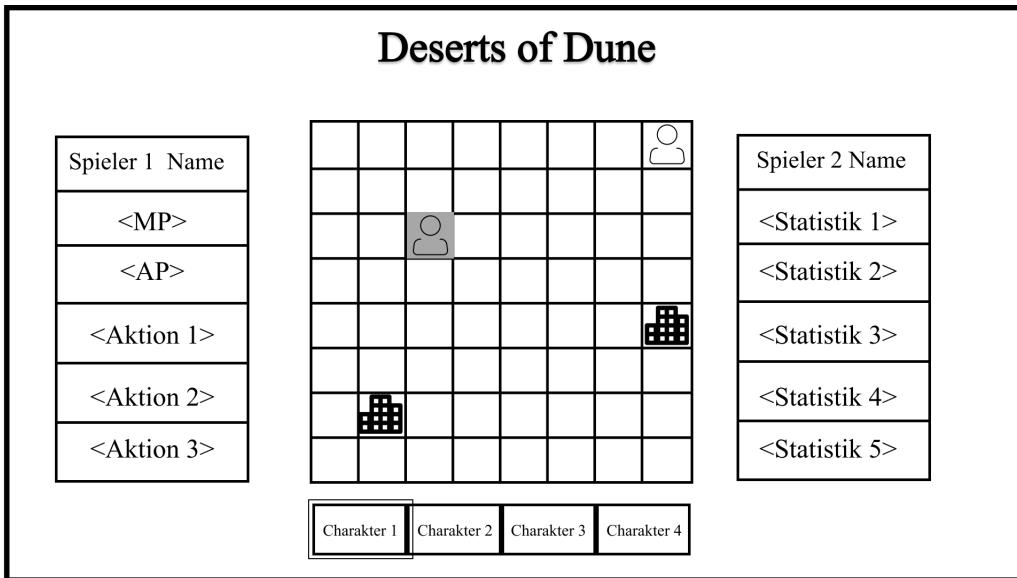


Abbildung 51: **Spielfeld:**

Der Dialog „Spielfeld“ zeigt das aktuelle Spielfeld und den aktuellen Spielzustand an. Dafür sieht man in der Mitte das Spielfeld und auf der linken und rechten Seite jeweils die Statistiken der Spieler, das heißt zum Beispiel wie viel MP der Spieler hat oder in welcher Runde sie sich befinden. Das ist auch die Oberfläche, die ein Zuschauer sehen kann. Wenn der Server die Charakterzugphase abhandelt, dann ändert sich der Dialog für den Spieler und der „Spielfeld“-Dialog wird zum „Charakterzug“-Dialog.

Abbildung 52: **Charakterzug:**

Im „Charakterzug“-Dialog hat der Benutzer die Möglichkeit seine Charakterzüge durchzuführen. Dafür wird quasi ein Overlay erzeugt, das heißt der Dialog des Spielfeldes wird erweitert für den Spieler, der am Zug ist.

Dann sieht der Spieler unten in einer Leiste alle auf der Karte platzierten Charakter und kann diese mit einem Klick (Kasten zeigt aktuelle Auswahl) auswählen. Die Leiste ist wie eine horizontale Scrollbar. Wenn der Spieler einen Charakter ausgewählt hat, dann sieht er an der linken Seite Informationen zu dem Charakter und zu dieser Phase. Das heißt wie viele MP und AP der Spieler noch hat und dieser Charakter verbrauchen würde. Außerdem existieren in diesem Abschnitt Buttons für die verschiedenen Aktionen, die man mit dem Charakter machen kann.

Wenn der Spieler den Charakter bewegen will, dann muss er mit der Maus auf das Feld des Charakters klicken und den Charakter an die gewünschte Stelle ziehen. Damit wird die Bewegung ausgeführt.

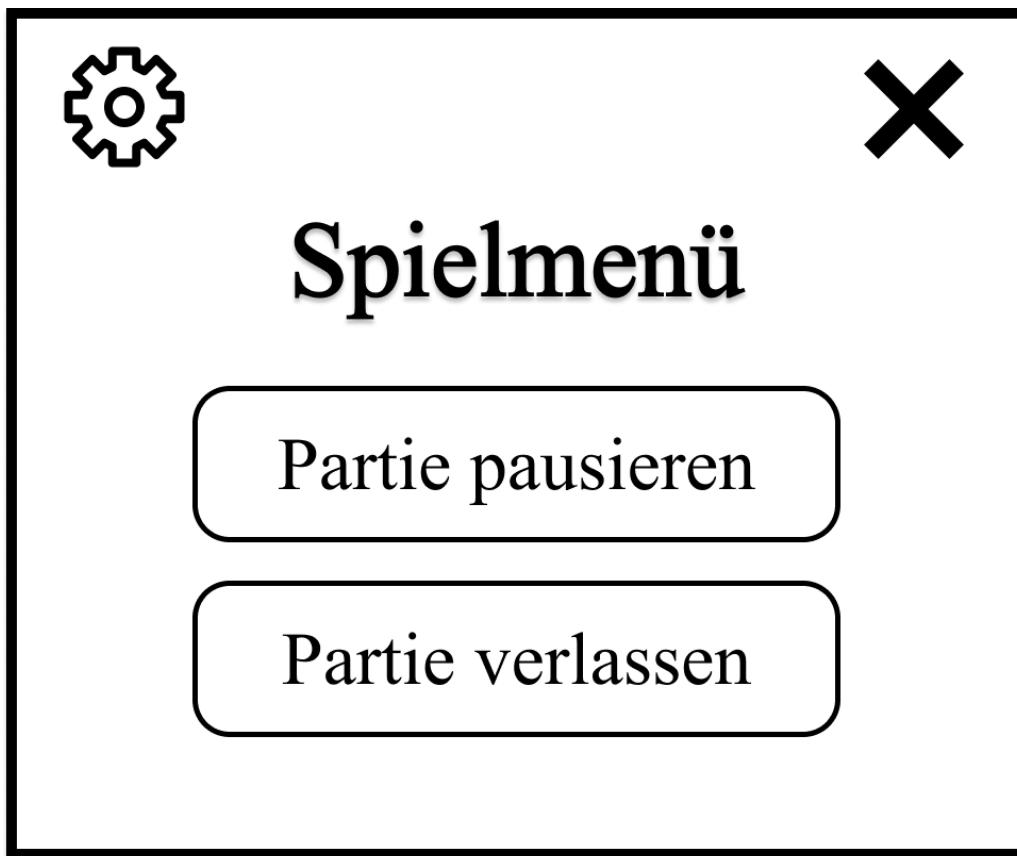


Abbildung 53: **Spielmenü**:

Den „Spielmenü“-Dialog erreicht der Benutzer über das drücken der „ESC“-Taste. Hier hat der Benutzer die Möglichkeit die Partie zu pausieren oder zu verlassen. Über das Zahnrad kommt der Benutzer zu den Einstellungen. Der Dialog kann über den „X“-Knopf oder erneutes drücken der „ESC“-Taste geschlossen werden.

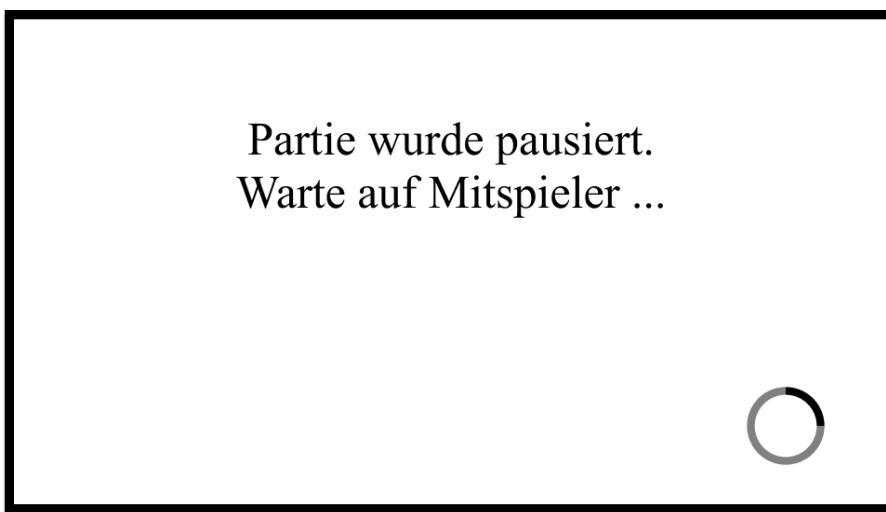


Abbildung 54: **Partie wurde pausiert**:

Zudem kann es passieren, dass man die Partie spielt und der gegnerische Mitspieler die Partie pausiert. Dann wird ebenfalls ein „Pause“-Dialog angezeigt, in dem der Spieler jedoch keine Interaktion vornehmen kann, sondern nur auf die Partiefortsetzung warten kann.

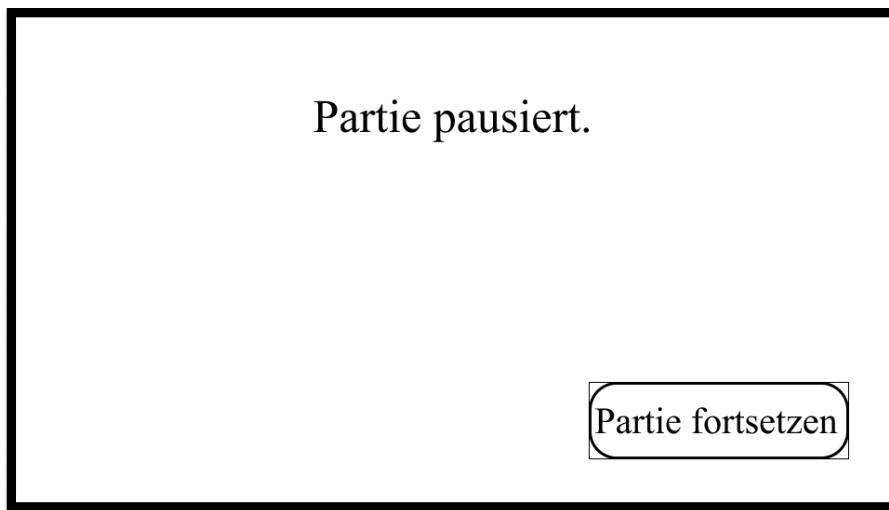


Abbildung 55: **Partie pausiert:**

Im „Pause“-Dialog hat der Spieler dann wieder die Möglichkeit, die Partie fortzusetzen, wofür es ebenfalls einen Button gibt. Dieser öffnet einen Bestätigungsdialog. Drückt der Benutzer in diesem Dialog „Ja“, dann wird das Spiel fortgesetzt und der Spieler kehrt zu dem „Spielfeld“-Dialog zurück.

Runde #<Runden-NR>		
Spieler
...
...
...
...
...

Abbildung 56: **Rundenstatistik:**

Der Dialog „Rundenstatistik“ stellt eine Art Hinweisfenster dar. In diesem Dialog wird dem Benutzer die aktuellen Ergebnisse der letztem Runde anzeigt, solange der Server die Runde beendet, die nächste vorbereitet und startet. Das heißt der Spieler sieht aktuelle Statistiken, wie in dem „Sieg/Niederlage“-Dialog.

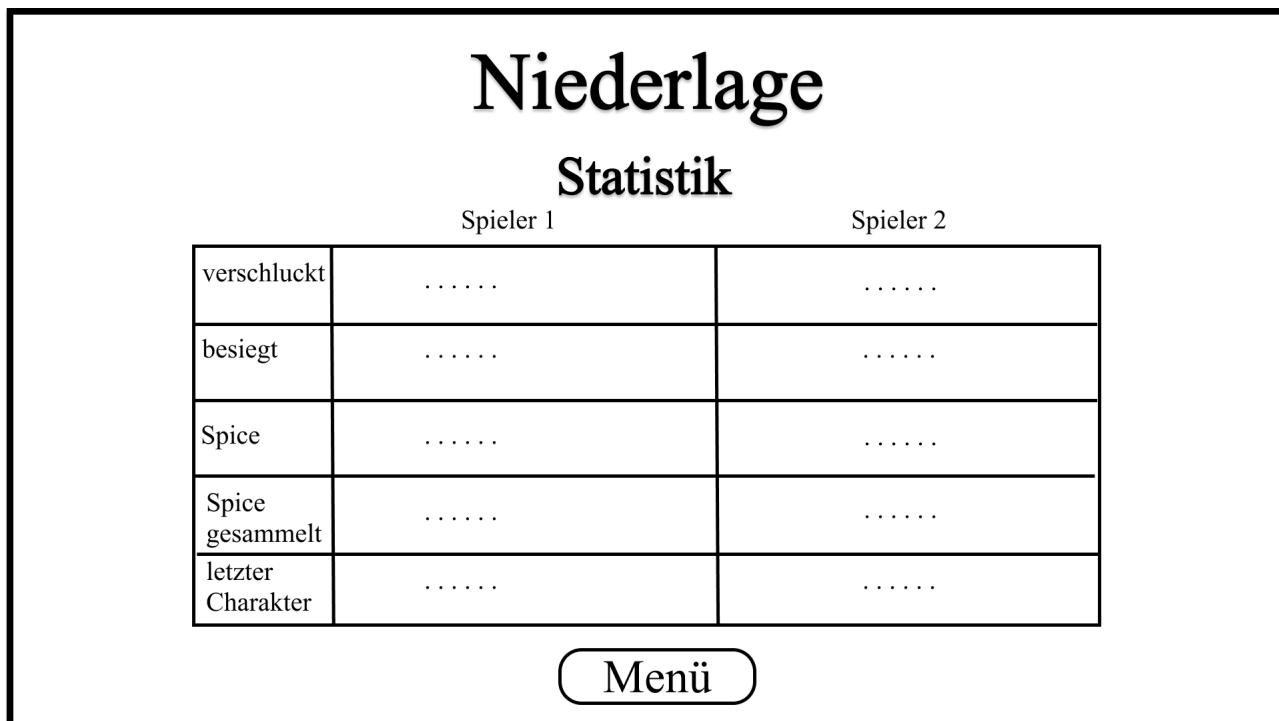


Abbildung 57: **end(lose)Screen:**

Im Endbildschirm wird dem Spieler bei einer Niederlage das Label Niederlage und bei einem Sieg das Label Sieg angezeigt. Zudem werden dem Spieler hier die Statistiken beider Spieler angezeigt. Durch Klicken der Taste Menü kommt der Spieler zurück in das Menü.

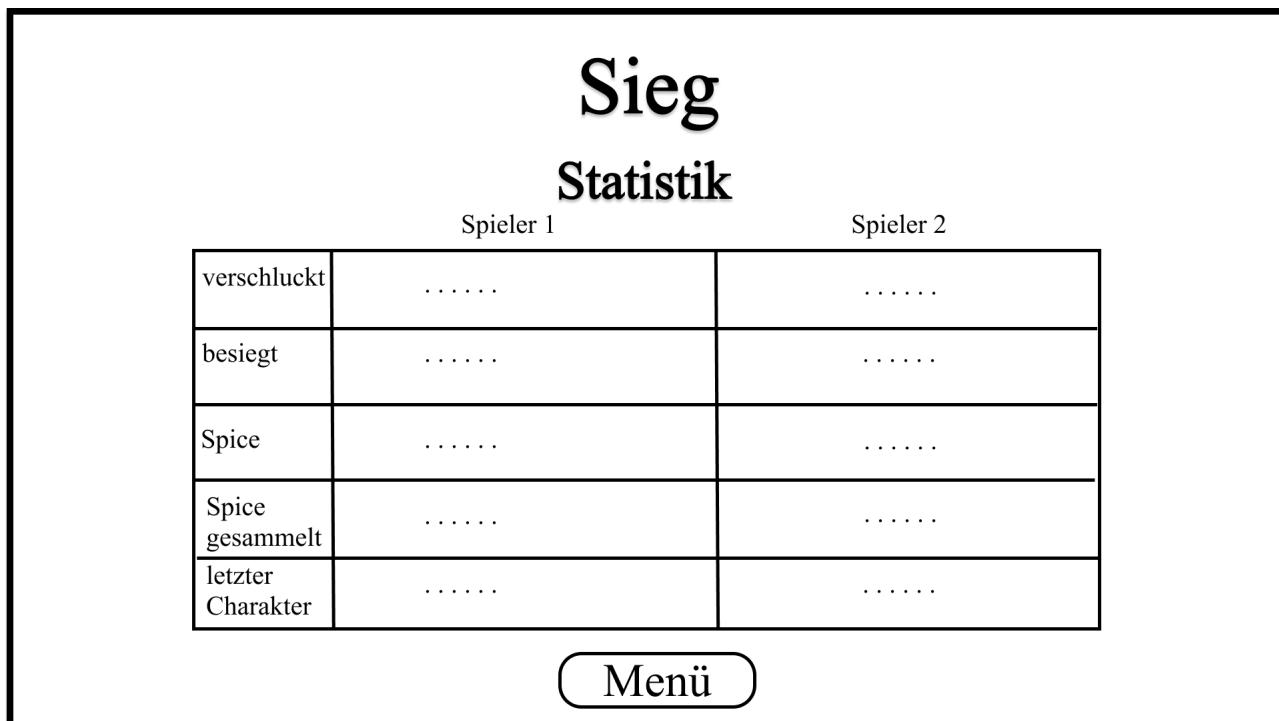


Abbildung 58: **end(win)Screen:**

Im Endbildschirm wird dem Spieler bei einer Niederlage das Label Niederlage und bei einem Sieg das Label Sieg angezeigt. Zudem werden dem Spieler hier die Statistiken beider Spieler angezeigt. Durch Klicken der Taste Menü kommt der Spieler zurück in das Menü.

3.2 Domänenmodell

Dieser Abschnitt enthält das Domänenmodel, welches in fünf kleinere Modelle unterteilt wurde. Zu Beginn die Gesamtstruktur und darauf folgend die zugehörigen Teilmodelle.

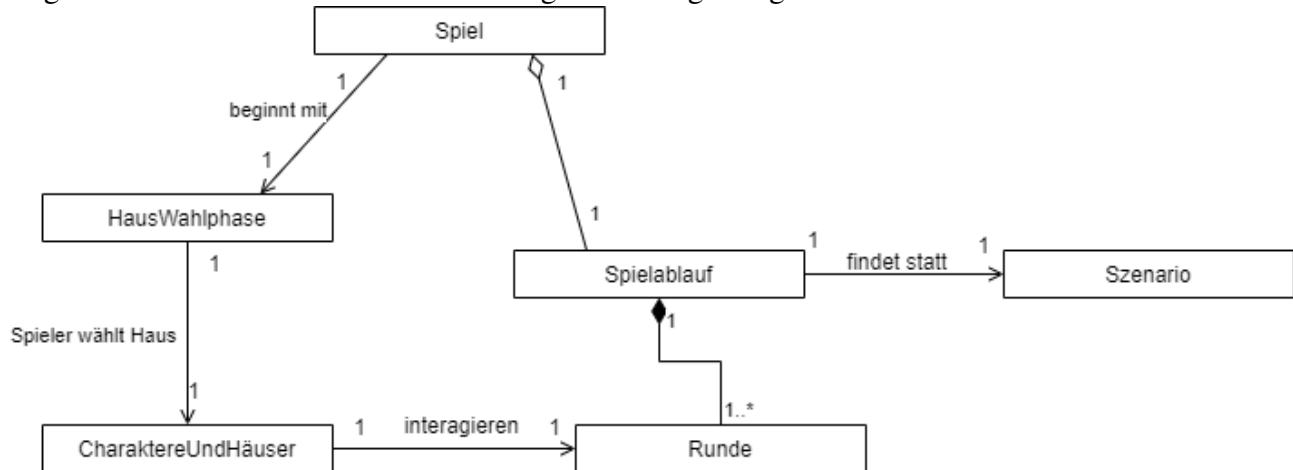


Abbildung 59: Domänenemodel: Gesamtstruktur (enthält Teilmodel: 60, 61, 62)

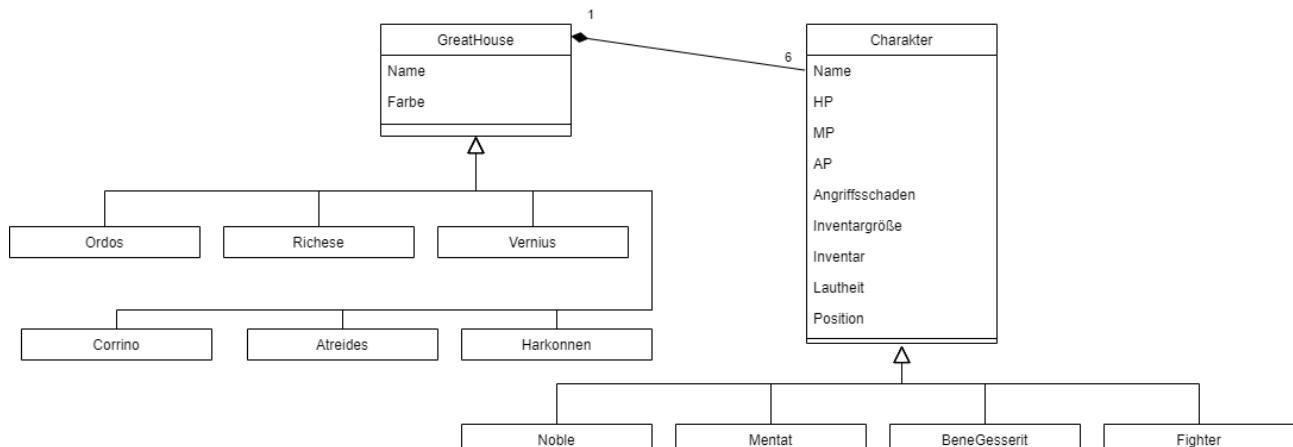


Abbildung 60: Teilmodel: Häuser und Charaktere

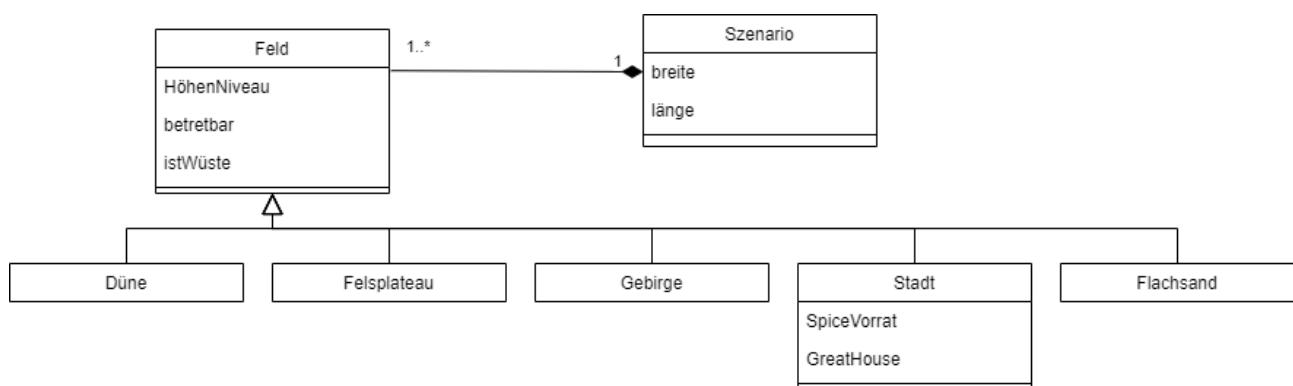


Abbildung 61: Teilmodel: Szenario

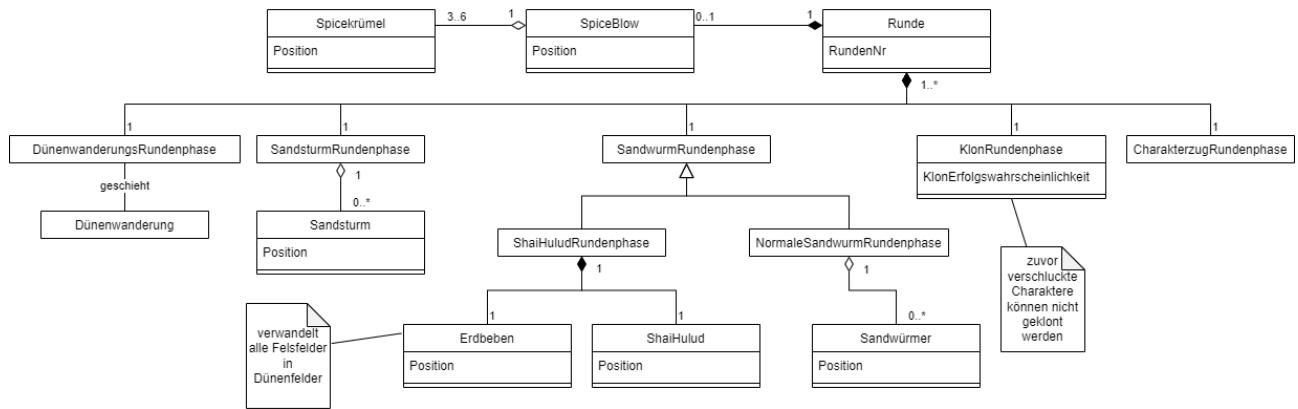


Abbildung 62: Teilmodel: Runde (enthält Teilmodel: 64)

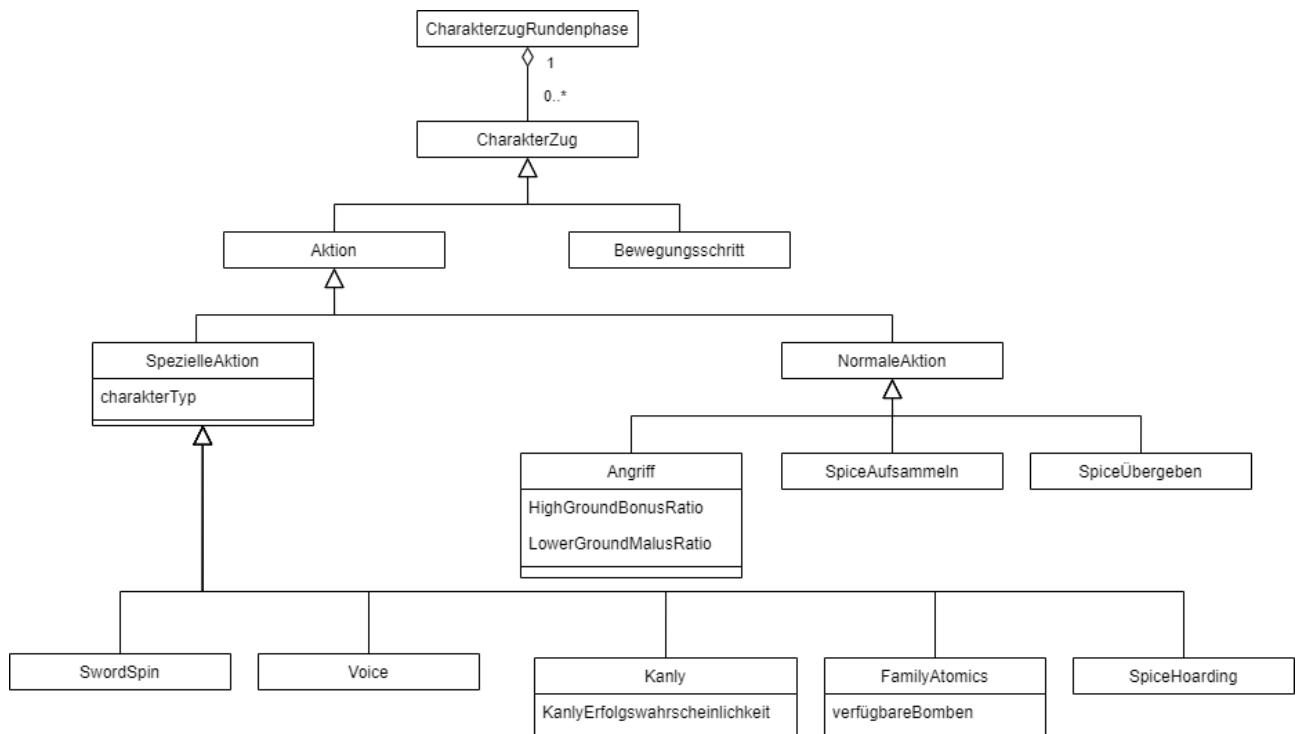


Abbildung 63: Teilmodel: Charakterzug-Rundphase

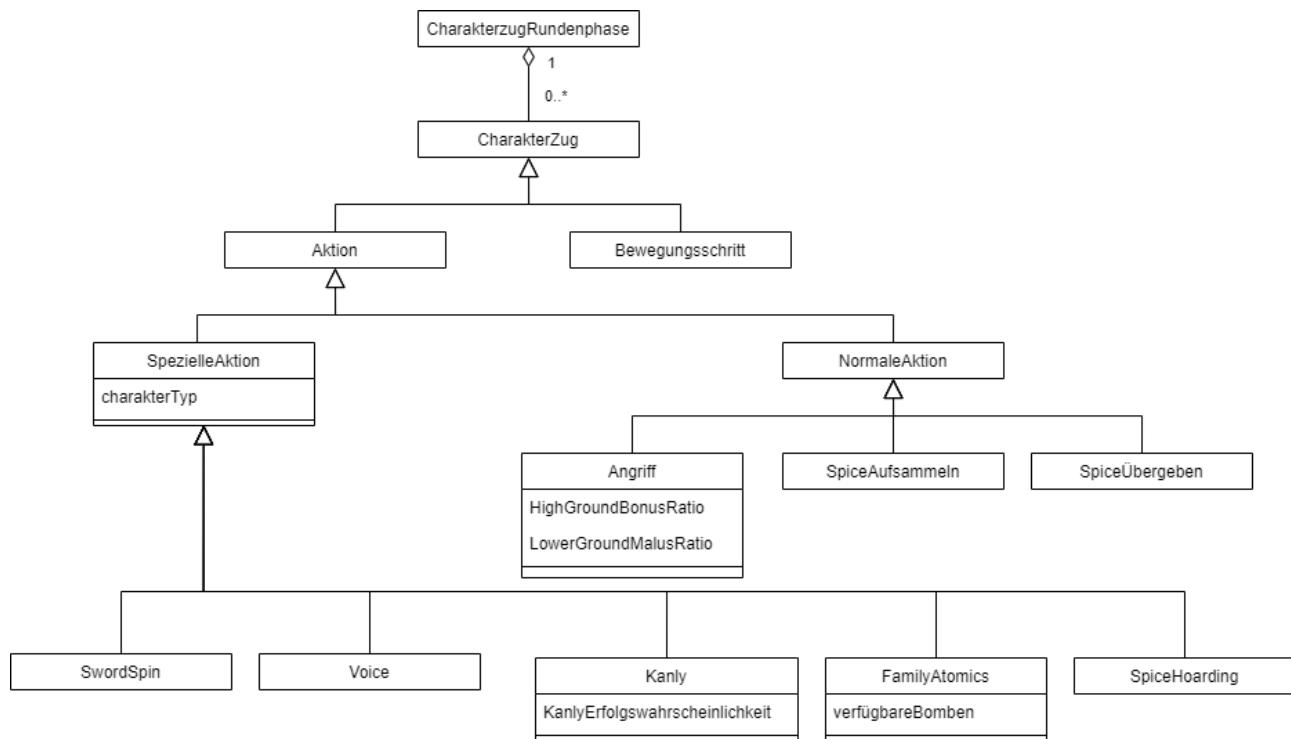


Abbildung 64: Teilmödel: Charakterzug-Rundphase

4 Randbedingungen

4.1 Qualität

Es müssen folgende nicht-funktionalen Anforderungen an das gesamte System und die Entwickler erfüllt werden:

ID	QA1
TITEL:	Robustheit
BESCHREIBUNG:	Das gesamte System darf bei 100 Partien maximal einmal abstürzen. Das gesamte System stürzt genau dann ab, wenn eine Komponente des Systems abstürzt, das heißt aufgrund eines nicht behandelten Fehlers beendet wird.
BEGRÜNDUNG	Das gesamte System soll robust sein, das heißt stabil laufen, weil der Benutzer spielen möchte und nicht ständig das Spiel neu starten oder warten will. Zudem nervt ein nicht stabiles System den Benutzer und vertreibt ihn damit möglicherweise.
ABHÄNGIGKEITEN	
PRIORITÄT	2
AKTEUR	Teilnehmer ^{→ p. 18}

ID	QA2
TITEL:	Zuverlässigkeit des Servers
BESCHREIBUNG:	Der Server soll zuverlässig laufen. Das heißt, er soll eine Uptime von mindestens 95% haben und innerhalb von 1 Minute neu gestartet werden können.
BEGRÜNDUNG	Der Server ist für die Verwaltung der Spieler und der Partie zuständig und ist deshalb ein integraler Bestandteil der Anwendung. Daher sollte er zuverlässig und verfügbar sein.
ABHÄNGIGKEITEN	
PRIORITÄT	2
AKTEUR	Teilnehmer ^{→ p. 18}

ID	QA3
TITEL:	Toleranz des Servers
BESCHREIBUNG:	Der Server soll bei Verbindungsabbrüchen oder zu spät gesendeten oder nicht interpretierbaren Nachrichten des Clients tolerant sein. Das bedeutet, der Server schließt die Session bei einem Verbindungsabbruch nicht sofort. Außerdem wählt der Server bei zu spät gesendeten Nachrichten oder nicht interpretierbaren Nachrichten ein Standardverhalten. Der Client soll jedoch ebenfalls versuchen, das Problem zu beheben und zum Beispiel einen erneuten Verbindungsauftakt zu initiieren.
BEGRÜNDUNG	Durch das tolerante Verhalten des Servers kann die Partie auch bei Problemen des Clients fortgesetzt werden und wird nicht jedes Mal abgebrochen, wenn zum Beispiel bei dem Client das WLAN ausfällt. Dies trägt zur Robustheit der Anwendung bei.
ABHÄNGIGKEITEN	QA1 → p. 107
PRIORITÄT	4
AKTEUR	Benutzer-Client → p. 17, Teilnehmer → p. 18

ID	QA4
TITEL:	Plattformen
BESCHREIBUNG:	Die Komponenten <i>Benutzer-Client</i> und <i>Editor</i> müssen auf einer aktuellen Version von Microsoft Windows oder einer aktuellen Debian Linux-Distribution (zum Beispiel Ubuntu) lauffähig sein. Die Komponenten <i>KI-Client</i> und <i>Server</i> müssen inklusive ihrer Abhängigkeit als Docker-Container lauffähig sein.
BEGRÜNDUNG	Der <i>Benutzer-Client</i> und der <i>Editor</i> soll auf mehreren Betriebssystemen lauffähig sein, um eine möglichst große Zielgruppe zu erreichen und keine Anforderung an den Benutzer zu stellen. <i>KI-Client</i> und <i>Server</i> sollen als Docker-Container lauffähig sein, weil der Benutzer nicht direkt damit interagiert und die Docker-Images ohne komplizierte lokale Build-Prozesse direkt gestartet werden können. Damit erreicht man eine Unabhängigkeit von der Plattform und kann somit die Komponenten einfacher zum Laufen bekommen.
ABHÄNGIGKEITEN	QA5 → p. 108
PRIORITÄT	5
AKTEUR	Tutor → p. 18

ID	QA5
TITEL:	Portierbarkeit
BESCHREIBUNG:	Es muss möglich sein, die Anwendung auf andere Betriebssysteme zu portieren. Das bedeutet, dass innerhalb von 4 Arbeitswochen eines Teammitglieds, die Anwendung auf einem anderen Betriebssystem lauffähig sein sollte.
BEGRÜNDUNG	Die Anwendung soll portierbar sein, weil sich nach der Entwicklung herausstellen könnte, dass die Benutzer ein anderes Betriebssystem bevorzugen und daher das Spiel auf dieser Plattform nutzen wollen.
ABHÄNGIGKEITEN	QA4 → p. 108
PRIORITÄT	1
AKTEUR	Entwickler → p. 19, Teilnehmer → p. 18

ID	QA6
TITEL:	einfache Installation
BESCHREIBUNG:	Die Anwendung sollte mithilfe eines Skripts oder einem Programm installiert werden können. Dieses Skript oder Programm nimmt dem Benutzer die typischen Aufgaben beim Installationsprozess, wie zum Beispiel das Herunterladen und Installieren von Abhängigkeiten, ab.
BEGRÜNDUNG	Der Benutzer möchte das Spiel spielen und nicht die Zeit damit verbringen, das Spiel zunächst zum Laufen zu bekommen. Ein aufwendiger Installationsprozess könnte den Benutzer frustrieren.
ABHÄNGIGKEITEN	QA4 → p. 108
PRIORITÄT	1
AKTEUR	Teilnehmer → p. 18

ID	QA7
TITEL:	Testbarkeit
BESCHREIBUNG:	Die Anwendung soll mit entsprechenden Tests, insbesondere mit Unit-Tests, geprüft werden. Diese Tests sollen zu 100% erfüllt werden und mindestens 80% des nicht automatisch generierten Source Codes abdecken. Diese Tests beziehen sich nicht auf die Benutzeroberfläche. Zudem sollen die Tests automatisiert bei jedem Commit auf dem <i>release</i> -Branch ausgeführt werden.
BEGRÜNDUNG	Die Verwendung von Tests kann die Entwicklung des Codes vereinfachen, weil dann bei der Implementierung durch die Tests ein genaues Verständnis vorliegen muss, was der Code machen soll. Außerdem lassen sich so Fehler oder Bugs frühzeitig finden und die Qualität des Codes steigt.
ABHÄNGIGKEITEN	QA1 → p. 107, QA2 → p. 107
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA8
TITEL:	Bedienbarkeit
BESCHREIBUNG:	Die Anwendung soll einfach und intuitiv sein. Das bedeutet, dass die Anwendung mithilfe des Benutzerhandbuchs innerhalb von 2 Stunden vom Benutzer bedient werden kann.
BEGRÜNDUNG	Durch eine einfache Bedienbarkeit kann sich der Teilnehmer ganzheitlich auf das Spiel konzentrieren und wird nicht durch ein unübersichtliches Design vom Spiel abgehalten oder frustriert
ABHÄNGIGKEITEN	QA13 → p. 111
PRIORITÄT	3
AKTEUR	Teilnehmer → p. 18

ID	QA9
TITEL:	Programmiersprache
BESCHREIBUNG:	Für die Entwicklung der Anwendung wird hauptsächlich die Programmiersprache JAVA 11 verwendet. Die Bibliotheken sind frei wählbar.
BEGRÜNDUNG	JAVA ist die Lehrsprache an der Universität Ulm und damit den Entwicklern und dem Auftraggeber bekannt. Außerdem wird SonarQube zur Analyse des Source Codes verwendet und die Anwendung benötigt JAVA 11.
ABHÄNGIGKEITEN	QA10 → p. 110
PRIORITÄT	2
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA10
TITEL:	Dokumentation des Source Code
BESCHREIBUNG:	Der Source Code soll verständlich dokumentiert werden. Dabei wird der Javadoc-Stil verwendet. Automatisch generierter Code, wie zum Beispiel getter-Methoden sollen nicht dokumentiert werden. 90% des nicht automatisch erzeugten Codes sollen ausführlich dokumentiert werden.
BEGRÜNDUNG	Das ist eine Vorgabe des Auftraggebers. Außerdem erleichtert die Dokumentation von Code, dass andere Personen (unabhängig, ob sie Entwickler sind oder nicht) dokumentierten Code leichter verstehen und weiterentwickeln können. Ebenso kann die Dokumentation des Codes bei der Entwicklung helfen, sich bewusst zu werden, was der Code machen soll oder bei späterer Weiterentwicklung, was der Code macht.
ABHÄNGIGKEITEN	QA9 → p. 110
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA11
TITEL:	Implementierungs- und Dokumentationssprache
BESCHREIBUNG:	Der Source Code und die Dokumentation im Code, das heißt die Kommentare, müssen in Englisch verfasst werden.
BEGRÜNDUNG	Das ist eine Vorgabe des Auftraggebers. Außerdem ist englisch die etablierte Sprache bei der Implementierung und der Dokumentation, weil dadurch der Code international verständlich ist.
ABHÄNGIGKEITEN	QA10 → p. 110
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA12
TITEL:	Sprache der Benutzerschnittstelle
BESCHREIBUNG:	Die Texte der Benutzerschnittstelle werden auf Deutsch verfasst.
BEGRÜNDUNG	Die Anwendung wird vor allem von Personen genutzt, deren Muttersprache deutsch ist und damit würde die englische Sprache wenig Vorteile bieten, jedoch eventuell Schwierigkeiten bei der Verständlichkeit hervorrufen.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	QA13
TITEL:	Benutzerhandbuch
BESCHREIBUNG:	Für die Anwendung soll ein Benutzerhandbuch bereitgestellt werden. Dieses Dokument wird auf Deutsch verfasst und soll dem Benutzer eine einfache Bedienung der Anwendung ermöglichen. In diesem Dokument sollen alle spiel-relevanten Informationen und Komponenten, sowie die Benutzerschnittstelle anhand von Beispielen erklärt werden.
BEGRÜNDUNG	Ein Benutzerhandbuch kann die Bedienung einer Anwendung und den Einstieg in das Spiel erleichtern. Dies vermeidet Frustration bei den Benutzern, weil sie die Anwendung nicht verstehen oder Probleme bei der Bedienung haben.
ABHÄNGIGKEITEN	QA8 → p. 109, QA12 → p. 111
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Teilnehmer → p. 18

ID	QA14
TITEL:	Dokumentation
BESCHREIBUNG:	Der gesamte Entwicklungsprozess und auch das entstandene Produkt muss dokumentiert werden. Das bedeutet, dass Tests, Reviews und weitere Maßnahmen zur Qualitätssicherung dokumentiert werden müssen. Außerdem soll ein Projekttagebuch und ein Entwicklerhandbuch angelegt werden.
BEGRÜNDUNG	Die Dokumentation ermöglicht nach und während der Entwicklung der Anwendung anderen Kunden, den Prozess zu beurteilen oder sich mit der Architektur und Umsetzung der Komponenten auseinanderzusetzen.
ABHÄNGIGKEITEN	QA7 → p. 109, QA17 → p. 112, QA15 → p. 112, QA16 → p. 112
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA15
TITEL:	Projekttagebuch
BESCHREIBUNG:	Es soll ein Projekttagebuch zum Erfassen der Tätigkeiten (zum Beispiel Implementierung oder Treffen) geführt werden
BEGRÜNDUNG	Mit dem Projekttagebuch kann man später die Entwicklungsprozess nachvollziehen und ermitteln, wie viel jeder gearbeitet hat.
ABHÄNGIGKEITEN	QA14 → p. 111
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA16
TITEL:	Entwicklerhandbuch
BESCHREIBUNG:	Es soll ein Entwicklerhandbuch angelegt werden. Dieses Entwicklerhandbuch soll die Architektur und die implementierten Funktionen beschreiben, sowie die verwendeten Technologien auflisten und begründen.
BEGRÜNDUNG	Das Entwicklerhandbuch anderen Entwicklern das die Qualität des System zu beurteilen. Besonders auf der Messe, wenn man andere Komponenten einkauft, kann das die Entscheidung vereinfachen.
ABHÄNGIGKEITEN	QA14 → p. 111
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA17
TITEL:	Dokumentations-Sprache
BESCHREIBUNG:	Alle im Rahmen des Projekts, das heißt der Entwicklung der Anwendung, anfallenden Dokumente, wie zum Beispiel das Benutzerhandbuch werden auf Deutsch verfasst.
BEGRÜNDUNG	Die Anwendung wird vor allem von Personen genutzt, deren Muttersprache deutsch ist und damit würde die englische Sprache wenig Vorteile bieten, jedoch eventuell Schwierigkeiten bei der Verständlichkeit hervorrufen.
ABHÄNGIGKEITEN	QA14 → p. 111
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA18
TITEL:	Kombinierbarkeit von Komponenten
BESCHREIBUNG:	Die Komponenten: Server, Benutzer-Client, KI-Client und Editor sollen beliebig über Teamgrenzen hinweg kombinierbar sein.
BEGRÜNDUNG	Dies ist nötig, um in einer späteren Phase des Projektes Komponenten auszutauschen.
ABHÄNGIGKEITEN	FA3 → p. 39, FA4 → p. 39, FA7 → p. 40, FA2 → p. 38
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

ID	QA19
TITEL:	Versionskontrolle GitLab
BESCHREIBUNG:	Der Entwicklungsprozess und der zu entwickelnde Code soll auf der Plattform GitLab versioniert werden und damit die Versionskontrolle Git zum Einsatz kommen. Dafür sollen Issues, Milestones und Boards verwendet werden, um den SCRUM-Prozess umsetzen sowie Commits und Branches, um den Code zu versionieren und zu strukturieren. Außerdem soll eine CI/CD-Pipeline verwendet werden, um automatisiert zu testen usw.
BEGRÜNDUNG	Dies ist hilfreich bei der Zusammenarbeit von mehreren Personen an einem Projekt und der Verwaltung von Code. So kann man beispielsweise bei fehlgeschlagenen Features einfach wieder zurückspringen oder durch das automatische Testen Zeit sparen und trotzdem die Qualität sichern.
ABHÄNGIGKEITEN	
PRIORITÄT	5
AKTEUR	Tutor → p. 18, Entwickler → p. 19

4.2 Betriebskonzept

Das Programm ist für Systeme mit Windows als Betriebssystem gedacht. Die Anwendung lässt den Benutzer das Spiel 'Dune' spielen. Hierbei kann entweder gegen einen echten Gegner oder eine KI gespielt werden, wobei auch laufenden Partien zugesehen werden kann. Es wird im Hauptmenü eine Hilfsoption mit Erklärungen zum Spielablauf geben.

4.3 Entwicklungsvorgaben

Das Softwaregrundprojekt wird von sechs Studenten geplant und implementiert. Der Entwicklungsprozess ist in Meilensteine unterteilt, welche die Projektphasen Analyse, Entwurf und Implementierung abdecken. Während des gesamten Prozesses steht das Team in Kontakt mit dem Vertreter des Auftraggebers (Tutor). Für die Implementierung wird die Entwicklungsumgebung IntelliJ und UnityHub benutzt und zur Versionsverwaltung ein GitLab-Repository verwendet. Der Programmcode wird in englischer Sprache kommentiert und mit Javadoc dokumentiert.

4.4 Abnahmekriterien

Kriterien für eine erfolgreiche Abnahme sind eine fristgerechte und korrekte Abgabe aller Team-Meilensteine mit einer ausreichenden Bewertung. Zudem wird vorausgesetzt, dass jedes Team-Mitglied $n-1$ der insgesamt n Einzelübungsblätter, fristgerecht abgegeben und bestanden hat. Des Weiteren ist eine regelmäßige Anwesenheit und Mitarbeit im Tutorium notwendig und auch eine aktive Mitarbeit im Team, die durch das Projekttagebuch dokumentiert wird. Dazu gehört eine eigene Programmertätigkeit in ausreichendem Umfang. Voraussetzungen für eine erfolgreiche Abnahmesitzung zum Ende des Projekts sind die Vorlage einer lauffähigen Version der Software und die Umsetzung der Kernfunktionalität, also aller verbindlichen Anforderungen. Zudem ist eine gute Qualität der entwickelten Software mit einer sinnvollen Testabdeckung und nachgewiesen durch umfangreiche und dokumentierte Maßnahmen zur Qualitätssicherung erforderlich. Die Einhaltung des Scrum-Entwicklungsprozesses und die sinnvolle Verwendung des Team-Repository und Scrum-Board im GitLab sind ebenfalls erforderlich. Dazu gehört auch eine ordentliche Dokumentation der Software und des Projektverlaufs. Die Abnahme ist dann erfolgreich, wenn der Kunde mit dem endgültigen Produkt zufrieden ist.

5 Architekturentwurf

5.1 Komponentendiagramm

Das zu entwickelnde Spiel DESERTS OF DUNE ist eine verteilte Anwendung und besteht aus vier großen Subsystemen (siehe FA2 → p. 38). Diese Subsysteme sind die vier großen Komponenten (die auch ausgetauscht werden können), die wiederum aus einzelnen Teilsystemen bzw. Unterkomponenten bestehen. Die strukturelle Gliederung des gesamten Systems von dem Spiel ist in den folgenden Komponentendiagrammen visualisiert und begründet.

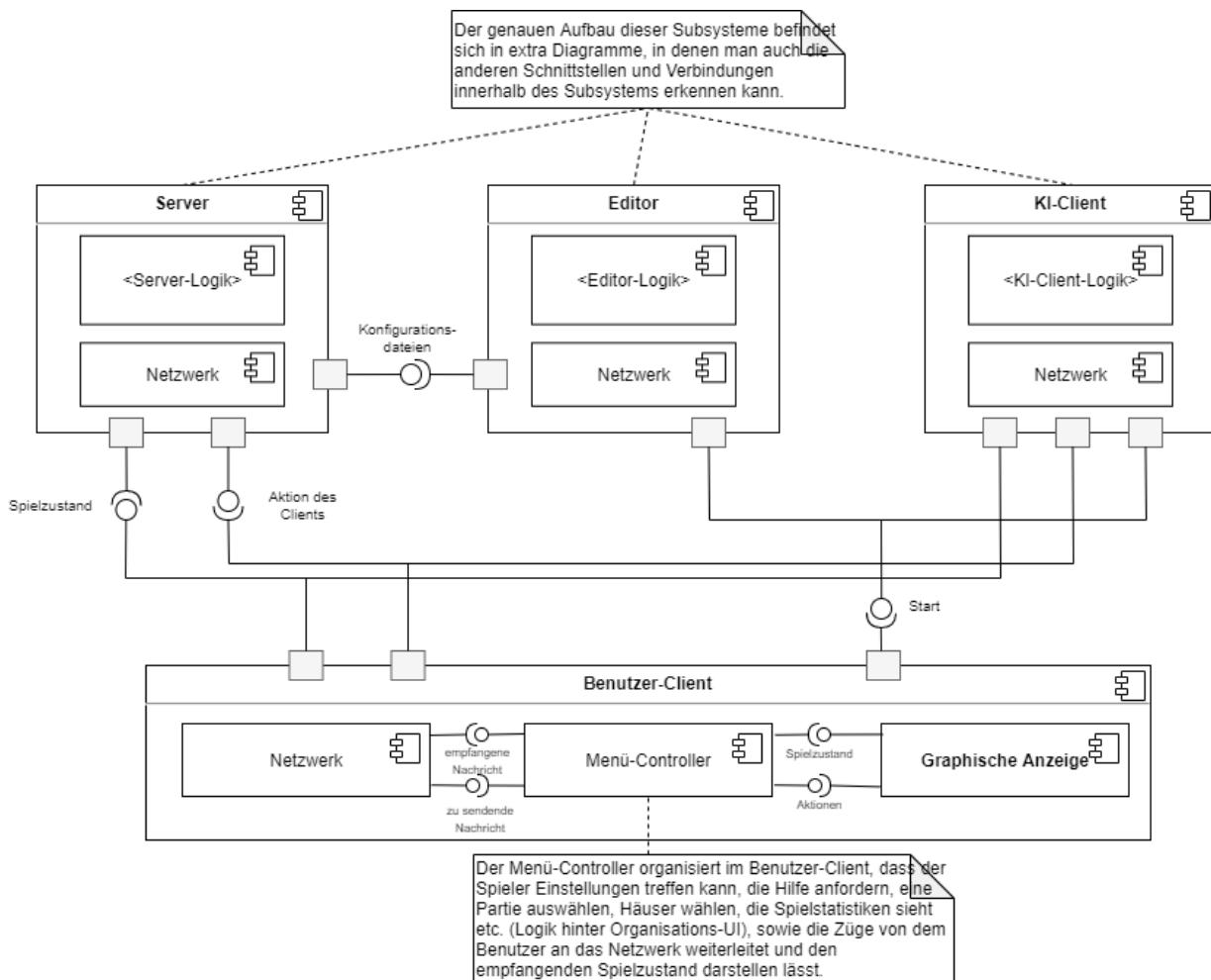


Abbildung 65: Komponentendiagramm des Gesamtsystems. Dabei kann man die Gesamtstruktur erkennen, sowie die Gliederung des Benutzer-Clients in die graphische Anzeige und den Controller. Diese grobe Architektur wurde von den Auftraggebern festgelegt und im Benutzer-Client bietet sich eine Aufteilung in graphische Anzeige und Controller an, damit man das MVC-Pattern umsetzt, was bei der Testbarkeit und Entwicklung Vorteile bietet.

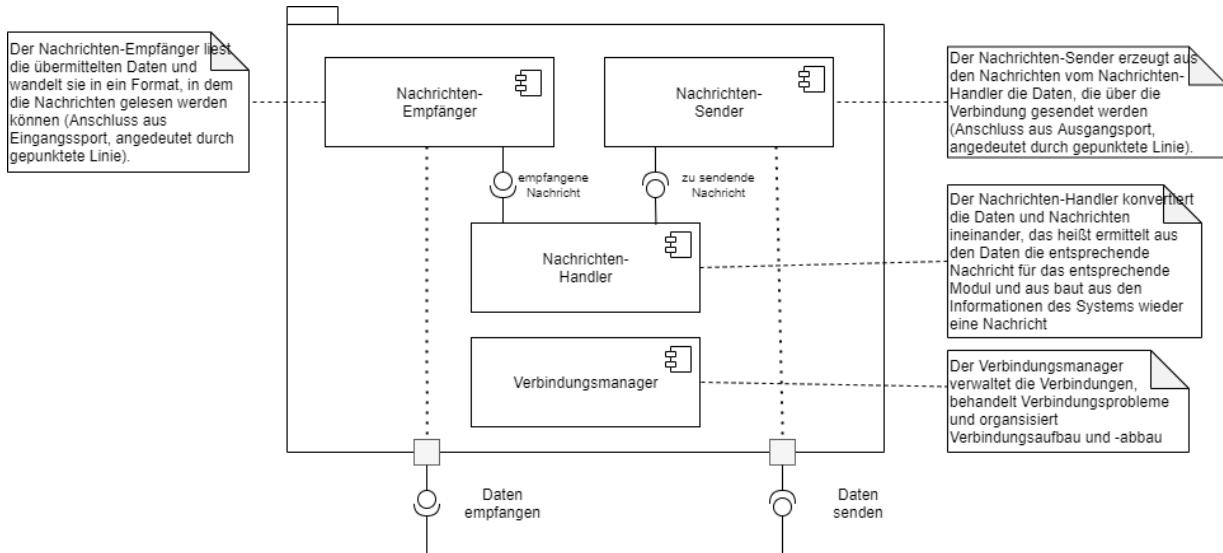


Abbildung 66: Komponentendiagramm des Moduls *Netzwerk*. Da das Spiel eine verteilte Anwendung ist, müssen die einzelnen Systeme über ein Netzwerk kommunizieren und benötigen daher jeweils ein Modul, das die Schnittstelle zur Außenwelt darstellt und die ausgetauschten Daten in für die Komponenten entsprechend lesbare Nachrichten umwandelt.

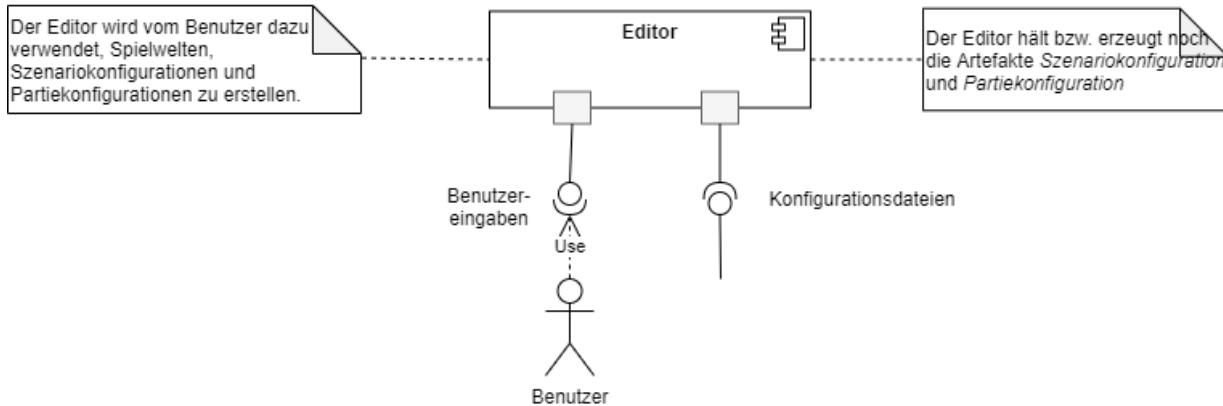


Abbildung 67: Komponentendiagramm des Editors. Dieses wurde jedoch nur angedeutet, da der Editor nicht von Team 08 entwickelt wird, sondern später eingekauft wird und somit nicht modelliert werden muss.

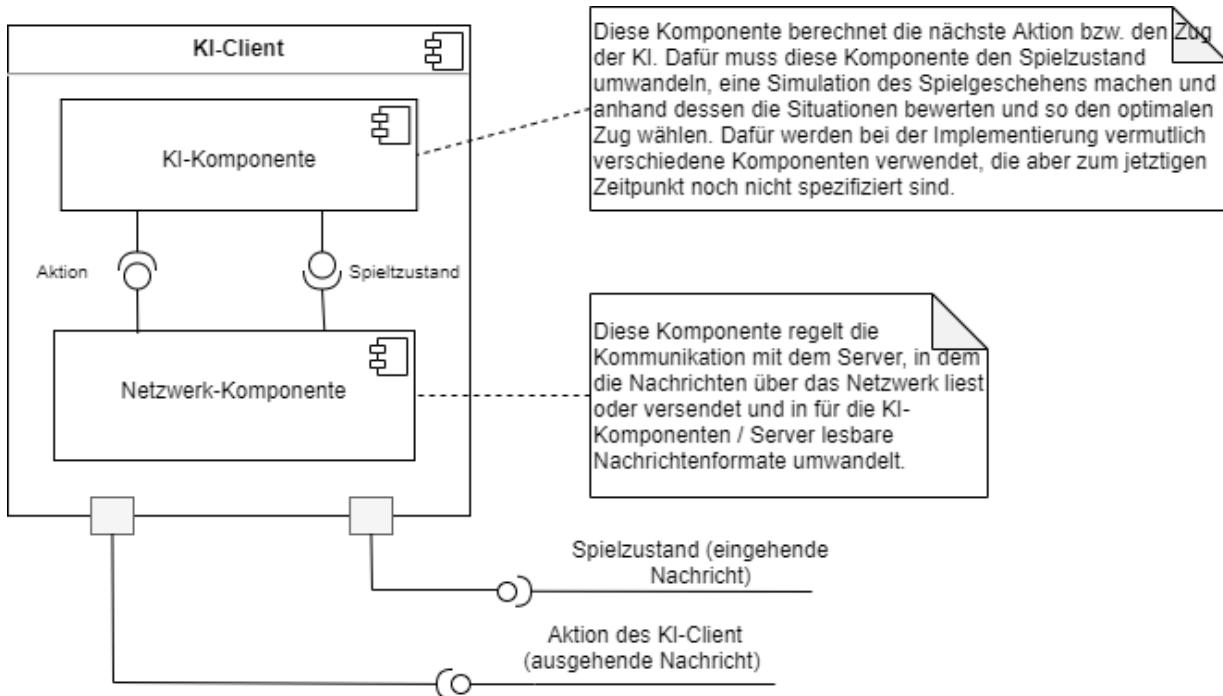


Abbildung 68: Komponentendiagramm des KI-Clients. Der KI-Client beinhaltet vor allem eine Komponente, die die Züge vorschlägt, da dieser keine Visualisierung oder Eingaben benötigt, sondern nur das Spiel spielen muss.

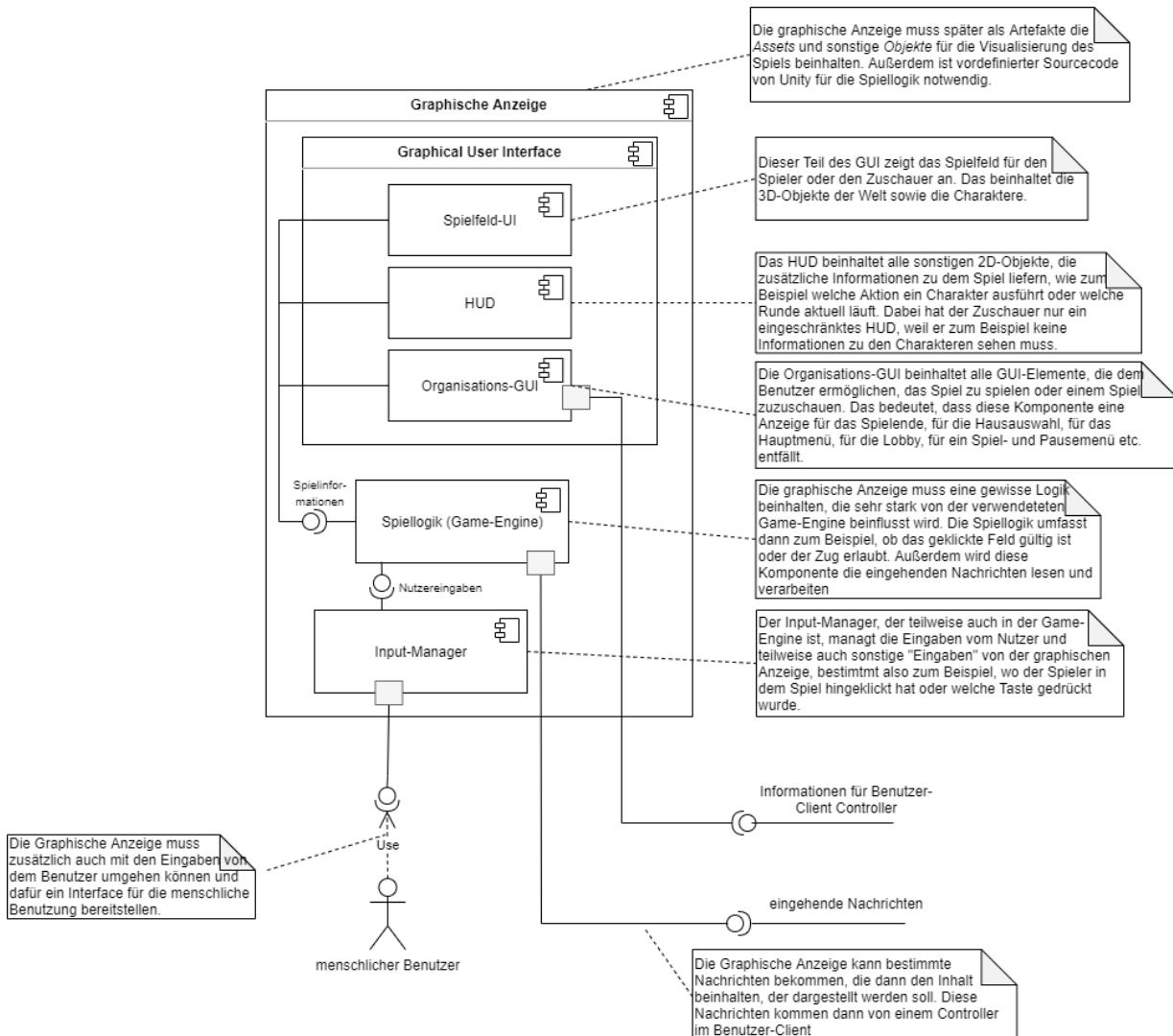


Abbildung 69: Komponentendiagramm der graphischen Anzeige bzw. der GUI. Dabei muss die GUI nicht nur das Spielfeld und alle sonstigen Informationen darstellen, sondern auch mit Nutzereingaben umgehen können, weswegen eine Komponente für die Darstellung (Aufteilung in 2D und 3D, da dies verschiedene Implementierungen benötigt) und eine für die Eingaben zuständig ist. Außerdem müssen die Eingaben verarbeitet werden das Spiel korrekt dargestellt, worum sich eine Spiellogik kümmert. Diese Komponente ist dem Subsystem *Benutzer-Client* zugeordnet

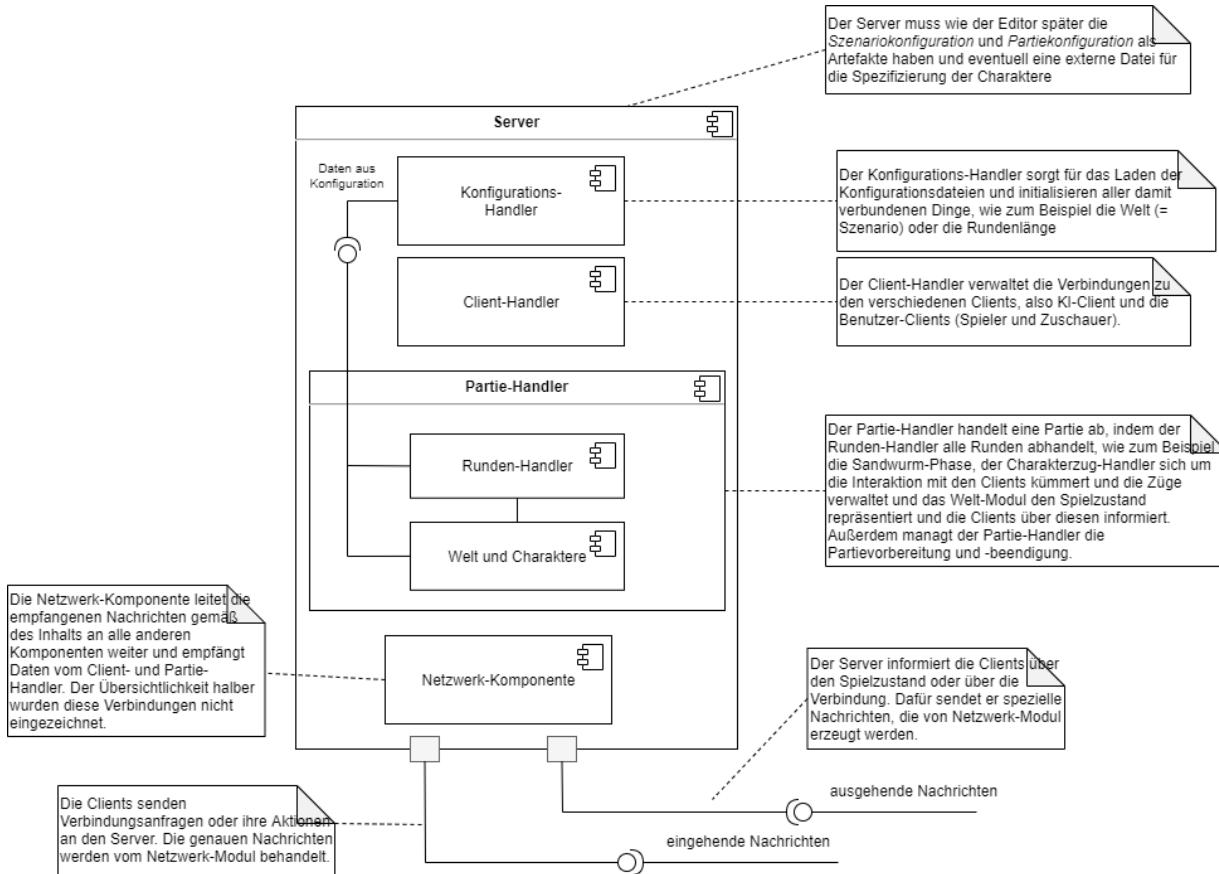


Abbildung 70: Komponentendiagramm des Servers. Der Server muss wie in den funktionalen Anforderungen (siehe Unterabschnitt 2.3) die Konfigurationen laden, die Clients verwalten und die Partie abwickeln. Da diese Aufgaben weitestgehend voneinander unabhängig ausgeführt werden können, sollte man diese in eigene gekapselte Komponenten auslagern. Bei der Partieabwicklung kann man sogar die Welt (= Model), und die Abhandlung der Runden (= Controller) voneinander trennen, was die Testbarkeit und Implementierung vereinfacht.

Im Folgenden ist die Zuordnung der funktionalen Anforderungen aus Unterabschnitt 2.3 und den Komponenten zu erkennen. Dabei wird für jede Komponente des Systems (siehe Komponentendiagramme oben) aufgelistet, welche Anforderung sie abgedeckt:

Komponente	zugeordnete Anforderungen
Gesamtarchitektur	FA1, FA2
Editor	FA8, FA15 - FA26, FA121, FA123- FA128
KI-Client	FA7, FA114
KI-Komponente	FA116 - FA119
Netzwerk-Komponente	FA9 - FA12, FA88, FA90 - FA93, FA95, FA102, FA115
Server	FA3, FA85, FA96, FA122
Konfigurations-Handler	FA87, FA86
Client-Handler	FA5, FA6, FA97, FA100, FA101, FA120
Partie-Handler	FA27 - FA84, FA89, FA94, FA98, FA99, FA112
Benutzer-Client	FA4, FA111
Graphische Anzeige	FA113
GUI	FA103, FA110
Spielfeld	FA13, FA104, FA106, FA107, FA108
HUD	FA14, FA105, FA109

5.2 Implementierungsentwurf

Im Folgenden ist der Implementierungsentwurf ausgeführt. Das heißt für jede in Unterabschnitt 5.1 angegebene Komponente wird nun erläutert, wie diese Komponente implementiert werden soll. Damit wird die grobe Sicht auf das System, gegebenen durch die Komponentendiagramme und die Domänenmodelle verfeinert.

Dabei sind die Klassendiagramme auch entsprechend der Komponenten aufgeteilt. Zu den Klassendiagramme folgen gegebenenfalls noch Erklärungen zu den Klassen und den Methoden, insofern diese nicht selbsterklärend sind⁴. Am Ende jeden Abschnitts findet sich die Zuordnung der Methoden zu den funktionalen Anforderungen aus Unterabschnitt 2.3, die sie implementieren und umsetzen.

⁴Als selbsterklärend werden beispielsweise Aufzählungsklassen (gekennzeichnet mit «enum»), Konstrukturen oder aussagekräftige Methoden angesehen

5.2.1 KI-Client

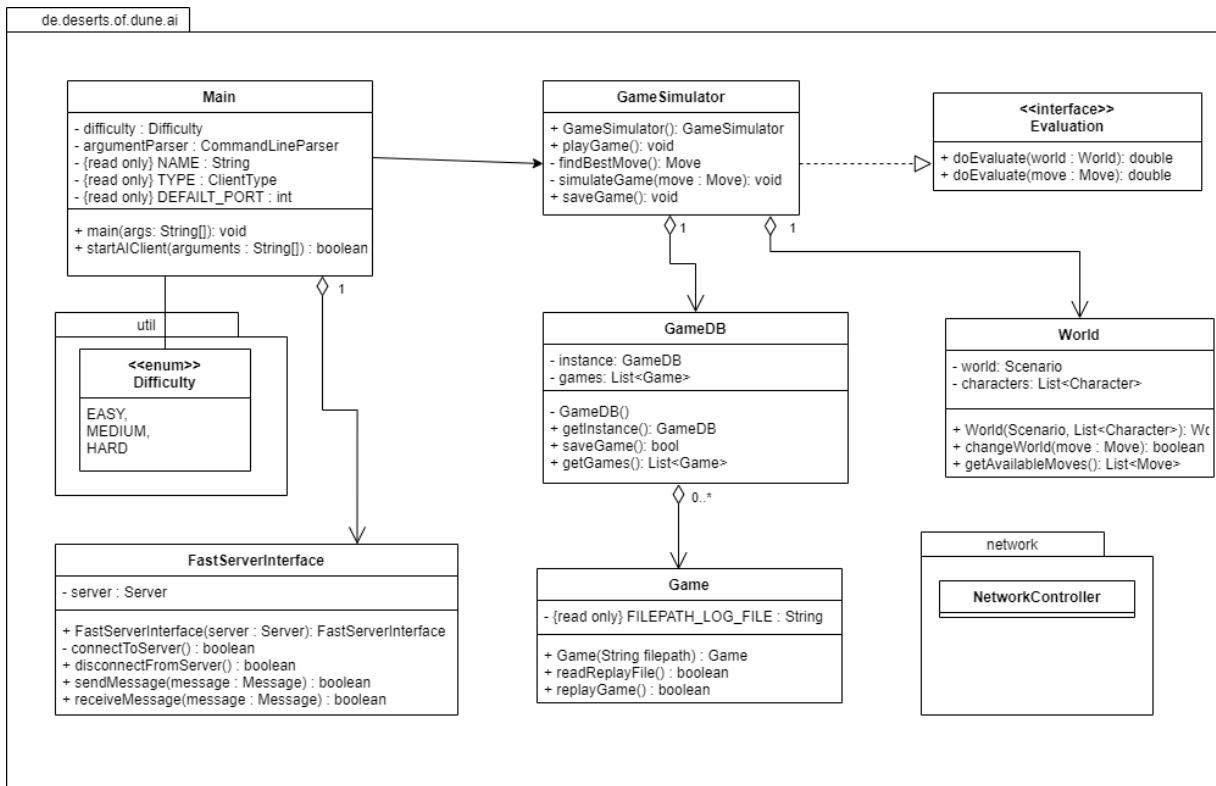


Abbildung 71: Übersicht über die Klassen, die der KI-Client voraussichtlich beinhaltet wird. Da zu dem aktuellen Zeitpunkt die genaue Logik bzw. Strategie, die der KI-Client verwenden wird um zu spielen, noch nicht endgültig festgelegt wurde, sind diese Klassen möglichst generisch gehalten. Zudem verwendet der KI-Client für die Simulation des Spiels und Validierung der Züge die *shared logic*

Main Da der KI-Client eine eigenständige Komponente von DESERTS OF DUNE sein wird, muss diese Komponente auch gestartet werden und dafür einen Einstiegspunkt haben. Dafür bietet sich eine *Main*-Klasse an. Mithilfe der *main*-Methode kann der KI-Client dann gestartet werden, wobei der Start in der Konsole stattfindet und entsprechende Parameter als Argumente übergeben werden. Um den Client dann zu starten, wird die *startAiClient*-Methode aufgerufen, die entweder mit dem entsprechend implementieren Netzwerkcontroller (*NetworkController*-Klasse aus Netzwerk-Paket) oder einem schnellen Interface (*FastServerInterface*-Klasse) eine Verbindung zu dem Server aufbaut.

FastServerInterface Diese Klasse realisiert eine schnelle Verbindung zum Server. Normalerweise kommuniziert der KI-Client über das Netzwerk mit dem Server. Da diese Kommunikation über die WebSockets aber langsam ist und man beim Training des KI-Clients viele Spiele in kurzer Zeit spielen will, muss der KI-Client auch eine direkte Verbindung zum Server haben. Mithilfe der *connectToServer*-Methode und *disconnectFromServer*-Methode kann der KI-Client direkt eine Instanz vom Server erstellen bzw. zerstören, sich mit dem Server verbinden oder trennen und dann schneller mit dem Server kommunizieren. Diese findet dann mithilfe der *sendMessage*-Methode und *receiveMessage*-Methode statt, welche die Funktionalität von den Methoden aus dem Netzwerkmodul nachahmen.

GameSimulator Diese Klasse spielt oder simuliert eine Partie, in der der KI-Client gegen einen anderen Client spielt. Dafür kann das Spiel mit der *playGame*-Methode gespielt werden, wie es ein menschlicher Spieler in dem Benutzer-Client ebenso macht. Dafür wird mithilfe der *findBestMove*-Methode der beste Zug ermittelt. Die Implementierung und das Verhalten der Methode ist noch nicht festgelegt. Generell wird jedoch in dieser Methode mit der *simulateGame*-Methode das Spiel vor simuliert (das heißt alle erlaubten Züge gespielt und am Ende wieder rückgängig gemacht) und die möglichen Stellungen bewertet. Dafür muss die *doEvaluate*-Methode aus dem Interface *Evaluation* implementiert werden. Dann wird anhand der Bewertung der beste Zug ausgewählt und dem Server mitgeteilt. Am Ende der Partie kann diese noch gespeichert werden, was ein Replay ermöglicht (durch *saveGame*-Methode realisiert). Dafür wird jede Nachricht in einem String gespeichert und dieser in eine Textdatei gespeichert.

GameDB Die *GameDB*-Klasse repräsentiert eine Art Datenbank für die gespielten und gespeicherten Spiele. Da diese Datenbank nur einmal benötigt wird, implementiert diese Klasse das Singleton-Pattern durch die *getInstance*-Methode. Mithilfe der *getGames*-Methode kann man dann später die Partie wieder anfordern und gegebenenfalls nochmal wiedergeben oder für das Training einsetzen.

Game Diese Klasse repräsentiert eine gespielte Partie, indem dort der Verweis auf die entsprechende Textdatei mit dem Nachrichtenverlauf der Partie gespeichert ist. Diese Datei kann mithilfe der *readReplayFile*-Methode gelesen werden. Diese Methode liest dann diese Datei wieder aus und speichert die Nachrichten wieder in einem String. Mit der *replayGame*-Methode kann dieser String dann verwendet werden, um die Nachrichten darin Stück für Stück zu analysieren und an den Server zu senden. Damit kann das Spiel nochmal abgespielt werden

World Diese Hilfs-Klasse repräsentiert eine Welt, in der Szenario und alle Charaktere gespeichert sind. Diese dient der Simulation der nächsten Züge aus der *GameSimulator*-Klasse. Denn mit der *changeWorld*-Methode kann ein Zug simuliert werden, indem diese fiktive Welt entsprechend den Spielregeln verändert wird, ohne dabei das richtige Spielfeld vom Server zu verändern. Mit der *getAvailableMoves*-Methode können alle nach den Spielregeln erlaubten Züge angefragt werden, um einen Spielbaum zu erstellen. Dafür wird jeder mögliche Zug generiert und überprüft, ob dieser erlaubt ist.

funktionale Anforderung	Methode(n), die diese umsetzt
FA7	<i>playGame, simulateGame, findBestMove, doEvaluate</i>
FA113	<i>saveGame, readReplayFile, replayGame</i>
FA114	<i>main, startAIClient (connectToServer)</i>
FA115	<i>startAIClient (connectToServer)</i>
FA116	<i>playGame</i>
FA117	<i>playGame, simulateGame, findBestMove, doEvaluate</i>
FA118	<i>findBestMove</i>
FA119	<i>Difficulty-enum, findBestMove, doEvaluate</i>
FA120	<i>main</i>

5.3 Geteilte-Logik

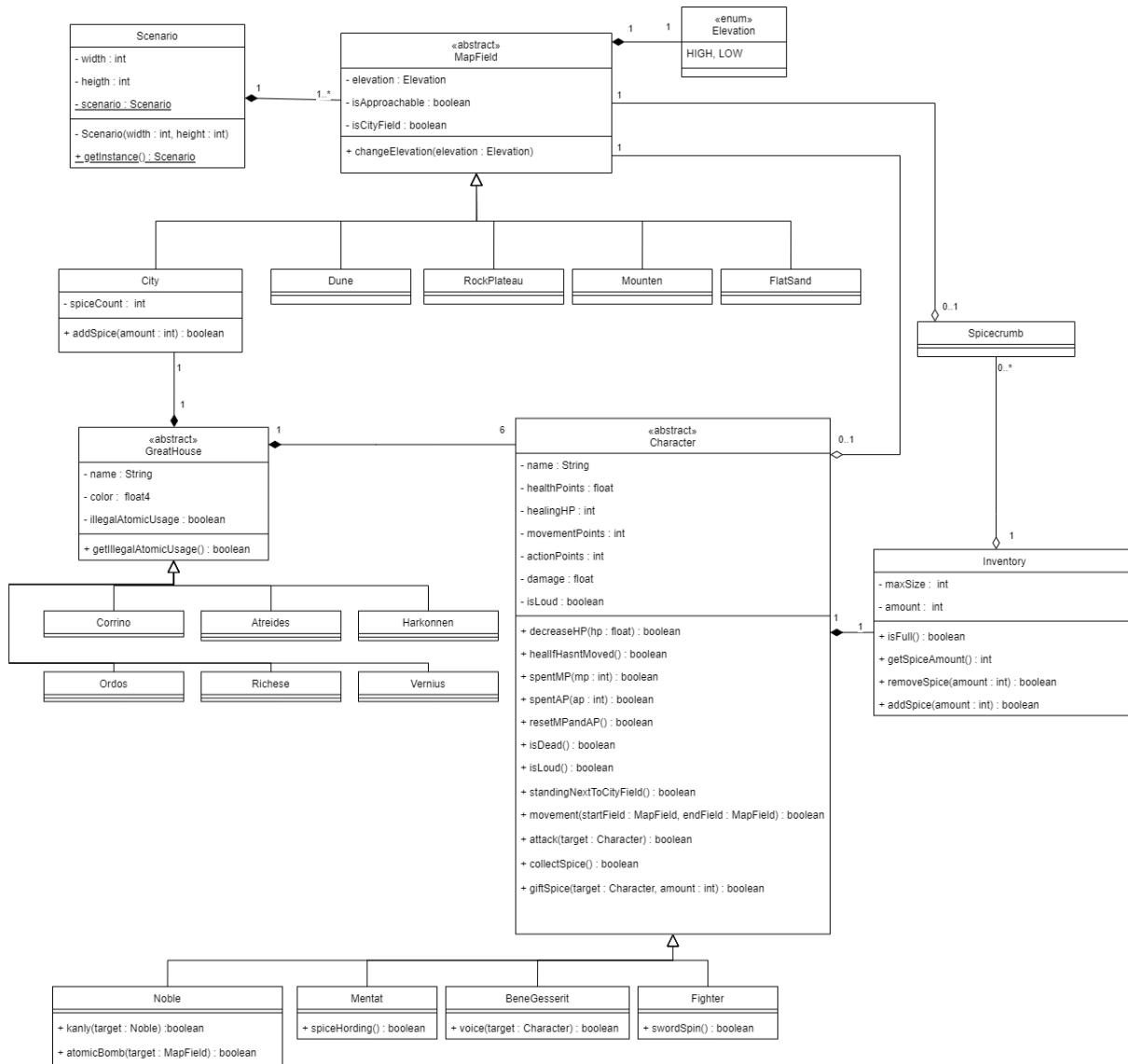


Abbildung 72: Klassendiagramm über Welt und Charaktere.

5.4 Nutzer-Klient

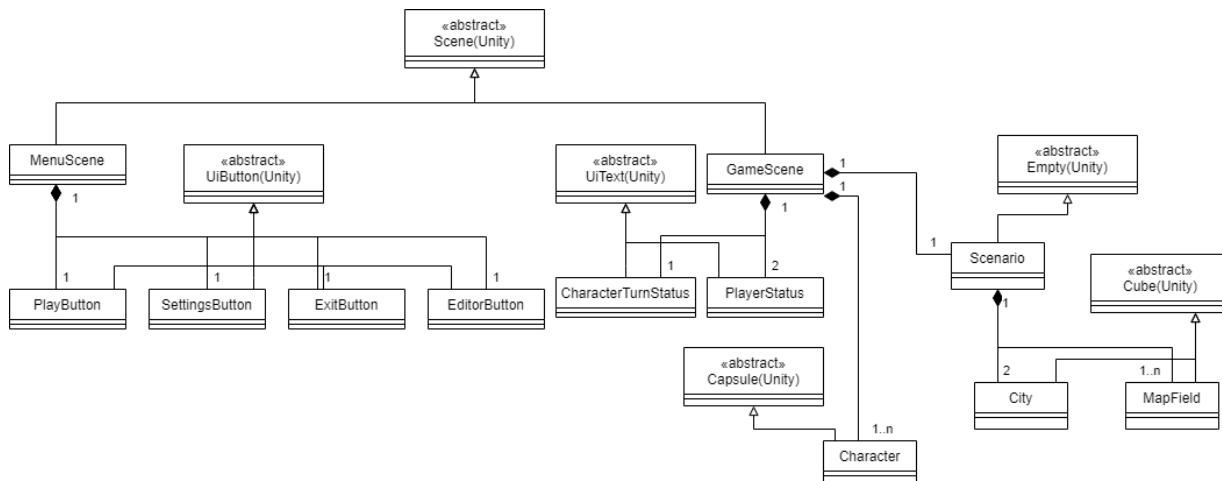


Abbildung 73: Klassendiagramm über Nutzer-Klient.

5.5 Statistik

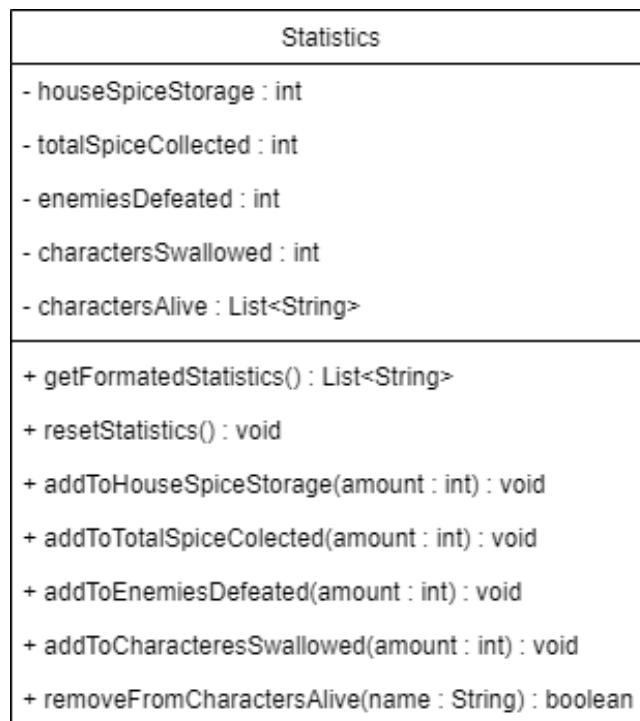


Abbildung 74: Klassendiagramm über Statistik.

5.5.1 Netzwerk-Modul

Im Folgenden sind die Klassendiagramme zu sehen, die das Netzwerk-Modul umfasst. Diese Klassen müssen allen Komponenten zugänglich sein, jedoch werden manche Klassen spezifisch von jeder Komponente erweitert oder genauer implementiert.

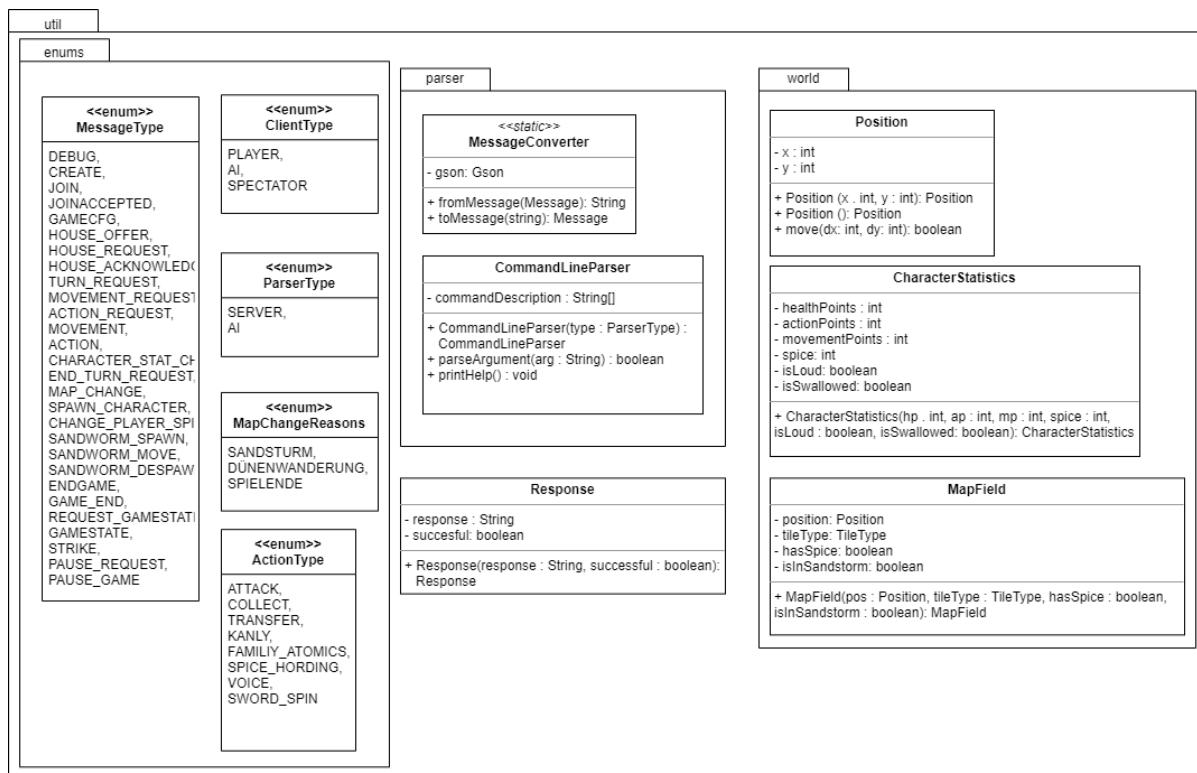
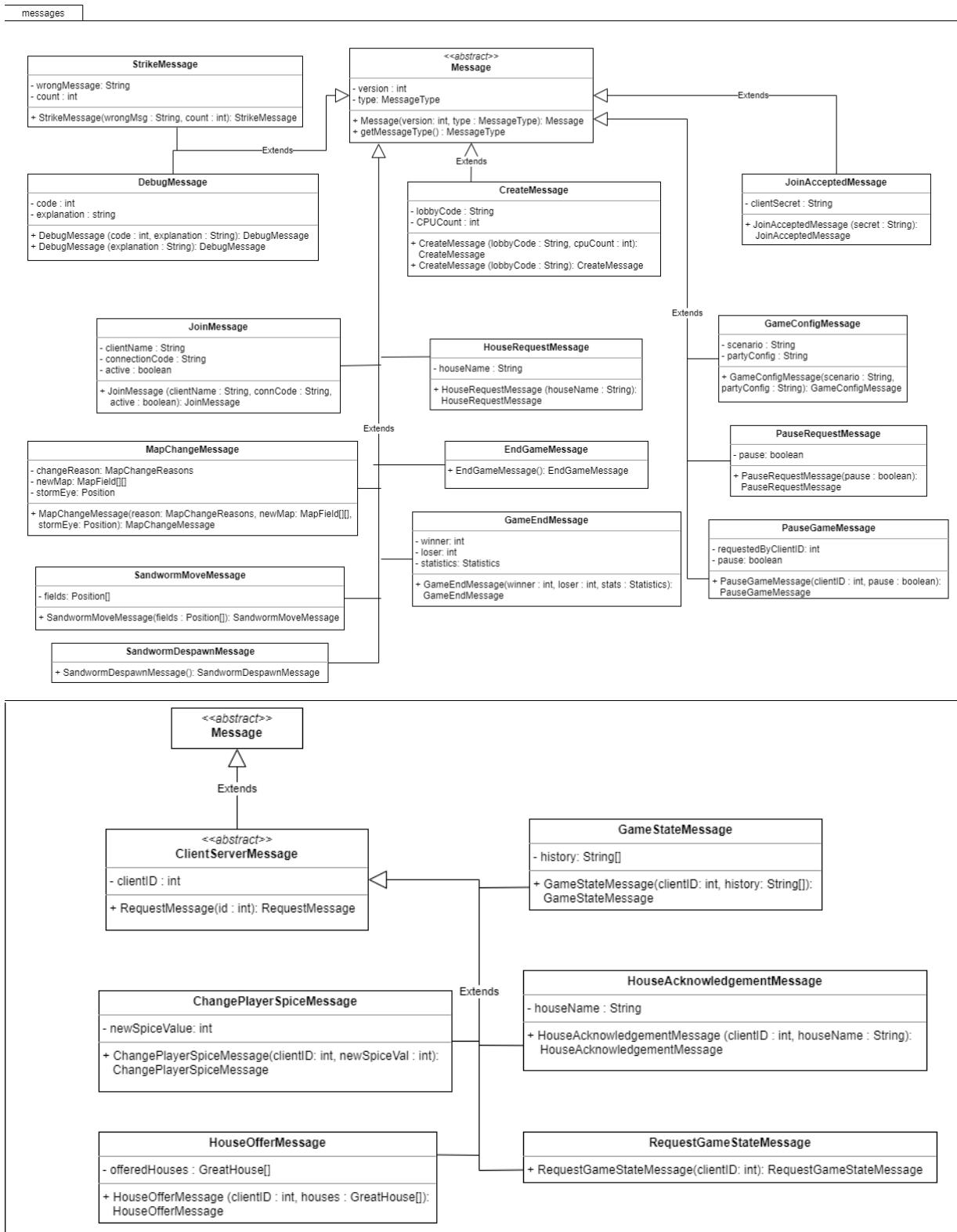


Abbildung 75: Übersicht über die Hilfs-Klassen aus dem Netzwerk-Modul. Diese Paket beinhaltet vor allem Aufzählungsklassen, Parser sowie Klassen, die die Informationen von Netzwerk-Nachrichten kapseln. Diese Klassen müssen von allen Komponenten des Systems verwendet werden können.



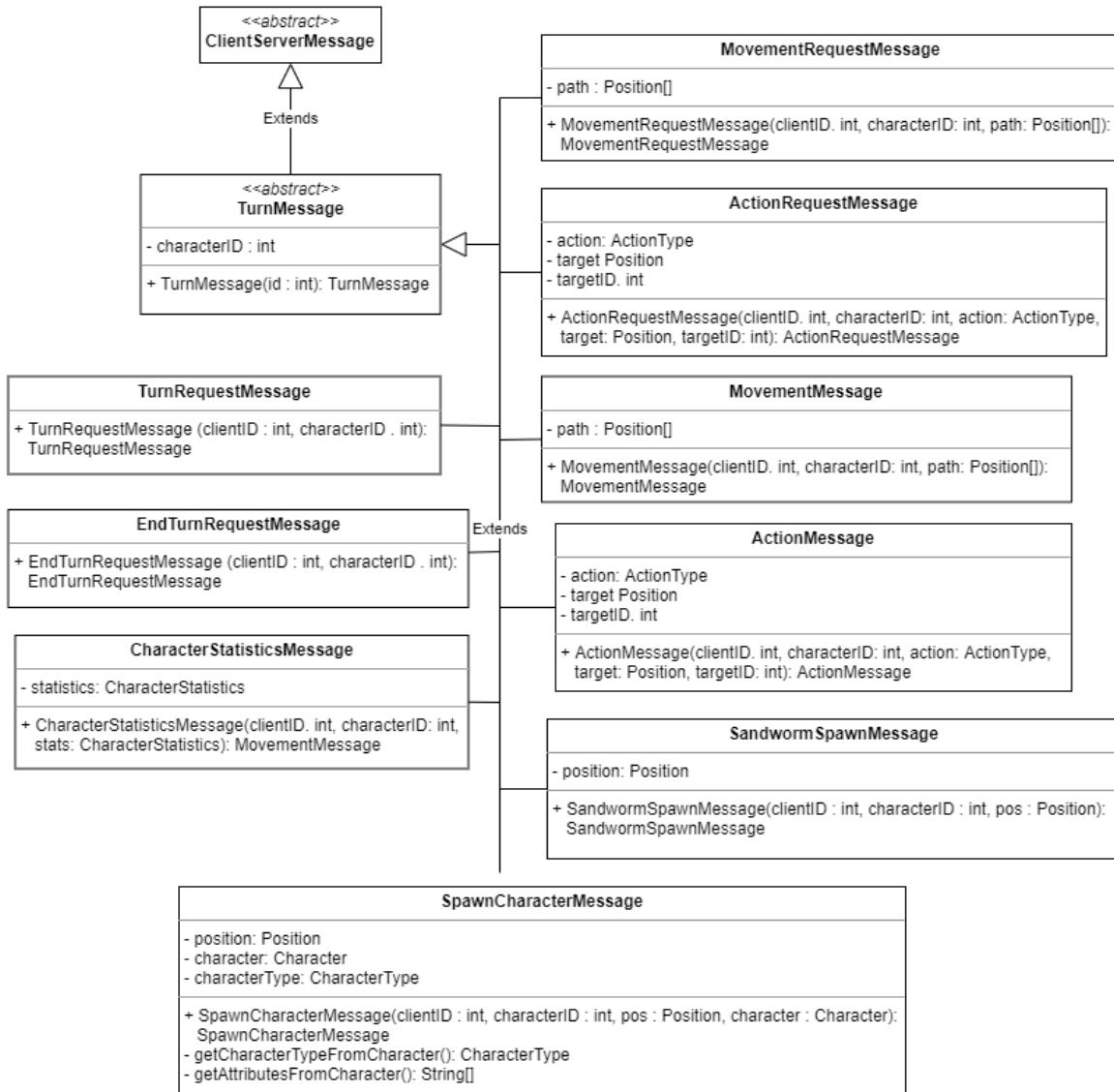


Abbildung 76: Übersicht über die Klassen, die die Nachrichten gemäß des Standardisierungsdokuments darstellen. Diese können gegebenenfalls nochmal angepasst werden, müssen aber auch von allen Komponenten verwendet werden.

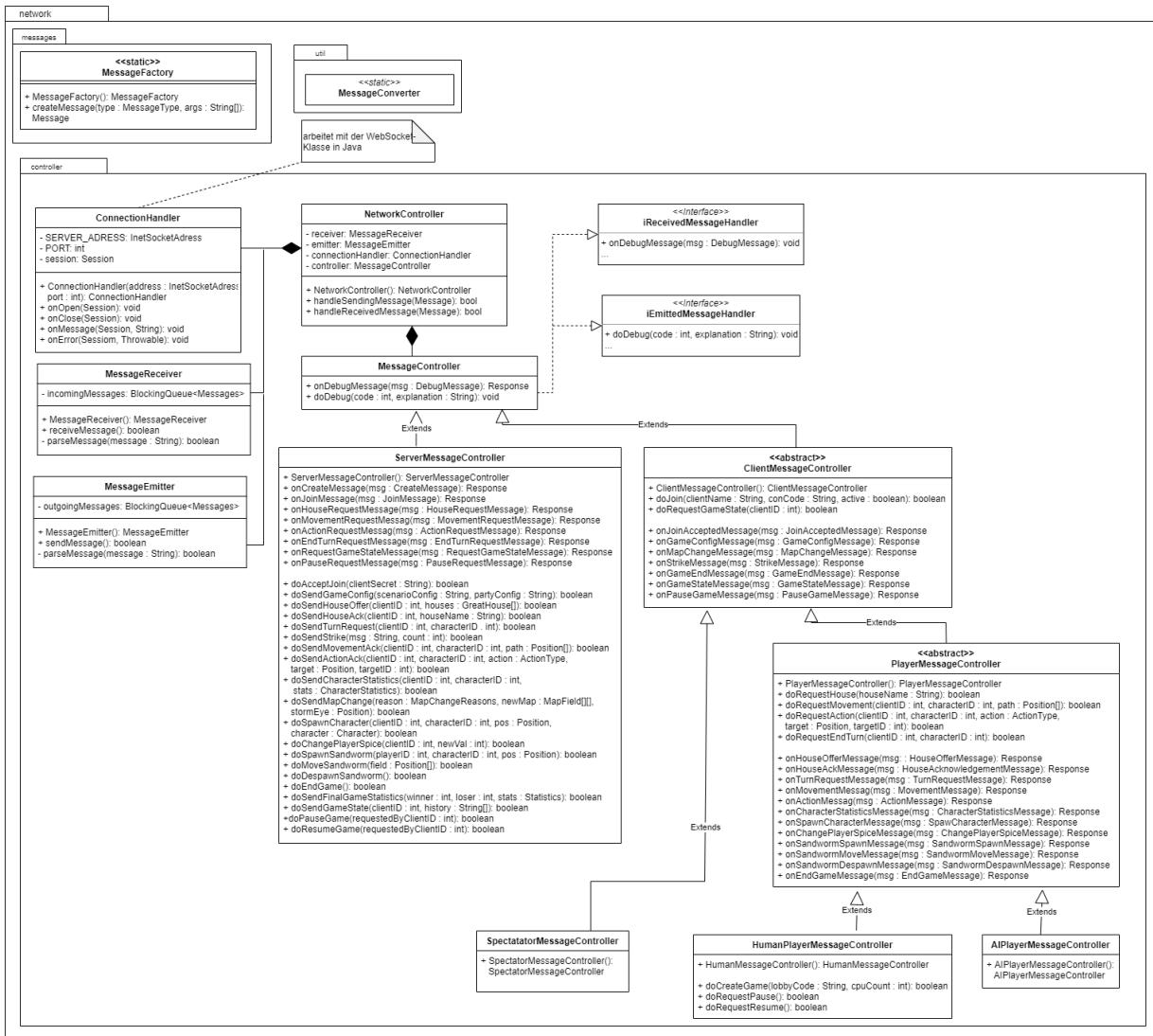


Abbildung 77: Übersicht über die Klassen, die die Kommunikation der Subsysteme über das Netzwerk ermöglichen. Dabei sind vor allem die Controller entscheidend, die die Nachrichten verarbeiten. Diese Klassen müssen ebenso in allen Komponenten verwendet werden können, jedoch muss die `NetworkController`-Klasse je nach Komponente unterschiedlich implementiert werden, da dies der übergeordnete Controller ist, der die Kommunikation leitet und die Komponenten zum Beispiel verschiedene Netzwerkendpunkte sind.

MessageConverter Der `MessageConverter` ist eine statische Klasse, die dem umwandeln von Nachrichten und Strings dient. Dafür wird ein JSON-Parser verwendet, der die Nachrichten, die als Objekte vorliegen, in JSON-Strings umwandelt und umgekehrt. Dafür sind die `fromMessage`-Methode (wandelt ein Nachrichten-Objekt in den entsprechenden String um) und die `toMessage`-Methode (wandelt den JSON-String in das passende Nachrichten-Objekt um) verfügbar.

CommandLineParser Diese Klasse dient dem auswerten der Parameter, die in der Kommandozeile eingegeben werden können. Diese Klasse dient also dem Zweck, die Parameter die beim Start des Servers oder des KI-Client über die Kommandozeile übergeben werden, zu verarbeiten und die Befehle umzusetzen. Dafür gibt es einerseits die `printHelp`-Methode, die alle Parameter und ihre Bedeutung in der Konsole ausgibt und die `parseArgument`-Methode, die ein Parameter bekommt und die entsprechenden Befehle umsetzt. Das bedeutet, dass zum Beispiel bei dem Parameter, der dafür

steht die Hilfe anzufordern, alle Befehle mit deren Erklärungen (durch die *printHelp*-Methode) auszugeben oder durch den entsprechenden Parameter der Port des Servers in *ConnectionsHandler* gesetzt wird.

Response Diese Hilfs-Klasse dient als Antwort für gesendete Nachrichten über das Netzwerk. Das heißt, wenn eine Nachricht empfangen wird und der entsprechende Controller diese verarbeitet, wird ein *Response* zurückgegeben, der angibt ob die Verarbeitung erfolgreich war oder sonst den Fehler als String.

Position Die *Position*-Klasse ist eine Hilfs-Klasse, die eine x- und eine y-Koordinate speichert und somit eine Position auf dem Spielfeld repräsentieren kann. Diese Kapselung ist sinnvoll, weil in manchen Nachrichten Position versendet werden. Eine Position kann bewegt werden, wenn sich ein Charakter zum Beispiel bewegt, was durch die *move*-Methode umgesetzt wird.

CharacterStatistics Diese Hilfs-Klasse speichert die Statistiken eines Characters, da diese ebenfalls über das Netzwerk versendet werden können. Dafür werden in dem Objekt die Statistiken in den entsprechenden Attributen gespeichert. Diese Klasse dient alleine dem Zweck, Informationen zu speichern.

MapField Die Klasse *MapField* repräsentiert ein einziges Teil aus dem Spielfeld. Diese Klasse beinhaltet keine Funktionalität, da sie nur die Informationen über ein Teil des Spielfeldes speichern muss (Position, Feldtyp, Spice und ob es in einem Sandsturm ist), da diese Klasse nur verwendet wird um Änderungen des Spielfeldes über das Netzwerk mitzuteilen.

Message Die *Message*-Klasse ist eine abstrakte Message, die sozusagen das Grundgerüst jeder Nachricht darstellt. Das heißt, dass jede Nachrichtem-Klasse von dieser Klasse erbt und somit eine Version und einen Typ hat (siehe Standardisierungsdokument). Der Typ von jeder Nachricht kann mit der *getMessageType*-Methode abgefragt werden, um die Nachricht zu klassifizieren. Die in Abbildung 75 aufgezeigten Nachrichten-Klassen erben direkt von dieser Klasse und haben somit keine weitere Gemeinsamkeiten. Alle aufgelisteten Nachrichten-Klassen implementieren eine Nachricht aus dem Standardisierungsdokument.

ClientServerMessage Diese Klasse ist eine Erweiterung der *Message*-Klasse und ist die Vorlage für weitere Klassen, weshalb sie auch abstrakt ist. Die von dieser Klasse erbenden Klassen halten weiter eine Client-ID, um bei der Kommunikation über das Netzwerk den Adressaten oder Sender eindeutig identifizieren zu können. Wieder implementieren alle erbenden Nachrichten-Klassen (siehe Abbildung 75) die entsprechenden Nachrichten aus dem Standardisierungsdokument.

TurnMessage Diese Klasse ist nochmal eine Erweiterung der *ClientServerMessage*-Klasse und ist eine abstrakte Vorlage für alle Nachrichten-Klassen, die auch noch eine Charakter-ID speichern müssen, weil sie eine Nachricht repräsentieren, die in Zusammenhang mit einem Charakter oder einer Spielentität stehen, die eindeutig identifiziert werden muss. Wieder implementieren alle erbenden Nachrichten-Klassen (siehe Abbildung 75) die entsprechenden Nachrichten aus dem Standardisierungsdokument.

MessageFactory Die statische Klasse *MessageFactory* setzt das Factory-Pattern um und ermöglicht es, Nachrichten-Objekte zu erstellen. Dafür wird mithilfe der *createMessage*-Methode eine Nachricht erstellt, indem der gewünschte Typ angegeben wird und die Argumenten für die Nachricht in einem Array. Die Methode erstellt dann Objekt der entsprechenden Nachricht.

ConnectionHandler Der *ConnectionHandler* ist für die Umsetzung des Netzwerkendpunkts zuständig. Das heißt diese Klasse implementiert den WebSocket und sorgt für den Verbindungsauftbau und die Kommunikation über das Netzwerk. Dafür gibt es die Methoden *onOpen*, *onClose*, *onMessage* und *onError*, die implementieren, was beim Verbindungsauftbau und -abbau passiert sowie bei Fehlern oder dem Erhalt einer Nachricht.

MessageReceiver Diese Klasse sorgt für die Verwaltung der eingehenden Nachrichten. Dafür gibt es eine Schlange, die alle eingehenden Nachrichten speichert, damit diese später weiterverarbeitet werden können. Mit der *receiveMessage*-Methode können dann Nachrichten „empfangen“ werden und der Warteschlange hinzugefügt werden, die dann mit der *parseMessage*-Methode nacheinander ausgewertet werden.

MessageEmitter Diese Klasse sorgt für die Verwaltung der ausgehenden Nachrichten. Dafür gibt es eine Schlange, die alle ausgehenden Nachrichten speichert, damit diese nacheinander geparsed und abgesendet werden können. Mit der *sendMessage*-Methode können die Nachrichten dann „versendet“ werden, wobei diese vorher mit der *parseMessage*-Methode in das entsprechende UTF-8 JSON-Format gebracht werden müssen.

NetworkController Der *NetworkController* ist übergeordnete Verwaltungseinheit der Netzwerkkomponente. Er hält Implementierungen der Klassen *ConnectionHandler*, *MessageReceiver* und *MessageEmitter*, sowie einen *MessageController*. Damit kann der *NetworkController* die eingehenden und ausgehenden Nachrichten, die Verbindung zu den anderen Netzwerk-Clients und die Nachrichten an sich verwalten und auswerten. Mit der *handleSendingMessage*-Methode werden dann zu sendende Nachrichten verarbeitet und gesendet und mit der *handleReceivedMessage*-Methode eingehende Nachrichten verarbeitet. Diese Methoden sind die, die von der entsprechenden Komponenten zugänglich sind und verwendet werden, um über das Netzwerk zu kommunizieren. Die anderen Klassen und Methoden werden kann von diesem *NetworkController* verwaltet.

MessageController Die *MessageController*-Klasse ist der Controller, der die Logik enthält, die beim Erhalt oder beim Senden einer Nachricht ausgeführt werden soll. Durch die Aufteilung in diesen Controller und die Nachrichten-Klassen (Model) wird das MVC-Pattern implementiert. Der *MessageController* implementiert dann die Interfaces *iReceivedMessageHandler* und *iEmittedMessageHandler*, die alle Methoden enthalten, die die Controller für eingehende und zu sendende Nachrichten repräsentieren.

Der *MessageController* wird von weiteren Controllern erweitert, um je nach Komponenten und Implementierung des Netzwerk-Moduls verschiedene Controller-Funktionalitäten umzusetzen. Jeder Controller muss jedoch Debug-Nachrichten versenden und handeln können, was durch die Methoden *onDebugMessage* und *doDebug* umgesetzt wird.

Generell sind die Methoden mit dem Namen *on<Nachrichtenname>* dafür da, eingehende Nachrichten zu verarbeiten und die Logik zu implementieren, die bei Erhalt der entsprechenden

Nachricht auszuführen ist. Die Methoden mit dem Namen `do<Nachrichtenname>` beinhalten die Logik, die ausgeführt werden soll, wenn eine bestimmte Nachricht versendet werden soll.

Dabei gibt es weitere speziellere Controller, die den `MessageController` erweitern. So verwendet der Server die `ServerMessageController`-Klasse, welche alle Methoden beinhaltet, die notwendig sind um alle Nachrichten zu empfangen und zu versenden, die der Server laut dem Standardisierungsdokument verarbeiten muss. Die `ClientMessageController`-Klasse umfasst alle Methoden, die jeder Client braucht, wobei diese einmal von der `SpectatorMessageController`-Klasse erweitert wird (Klasse, die der Benutzer-Client des Zuschauers verwendet) und von der `PlayerMessageController`-Klasse. Diese Klasse beinhaltet alle Methoden, die die Nachrichten verarbeiten, die die Mitspielenden Clients versenden und empfangen müssen. Dabei gibt es weiterhin eine Unterteilung in die Klassen `HumanPlayerMessageController` und `AIPlayerMessageController`, die der Benutzer-Client für den Spieler und der KI-Client beinhalten. Diese Aufteilung besteht, da der menschliche Spieler mehr Nachrichten versenden kann (zum Beispiel Antrag auf Pausierung) als der KI-Client und somit auch einen erweiterten Controller benötigt, der diese zusätzliche Logik umsetzen kann.

5.5.2 Konfiguration

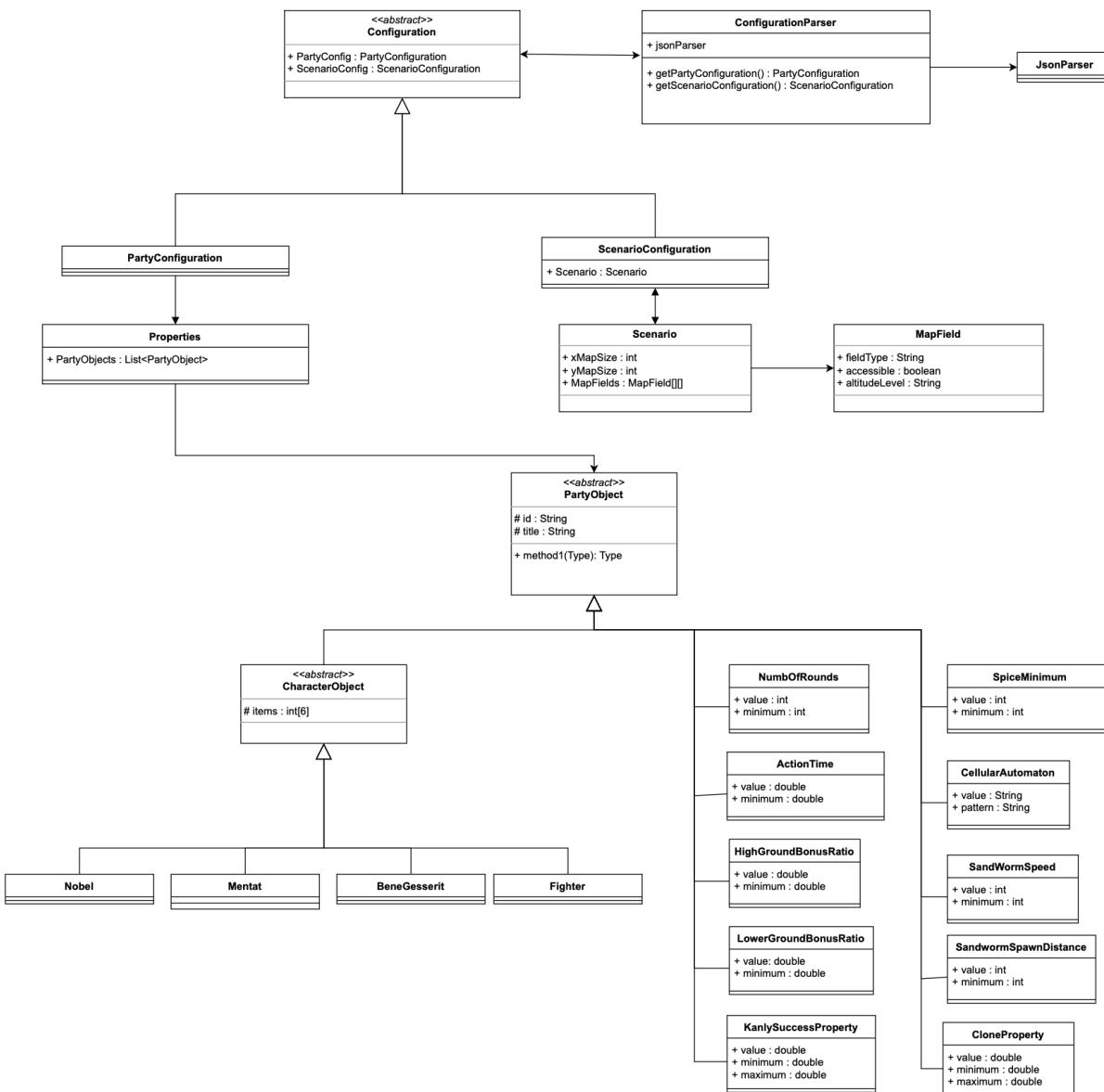


Abbildung 78: Übersicht über die Klassen, die für die Konfiguration von Partie und Szenario nötig sind.

5.5.3 Runden-Handler

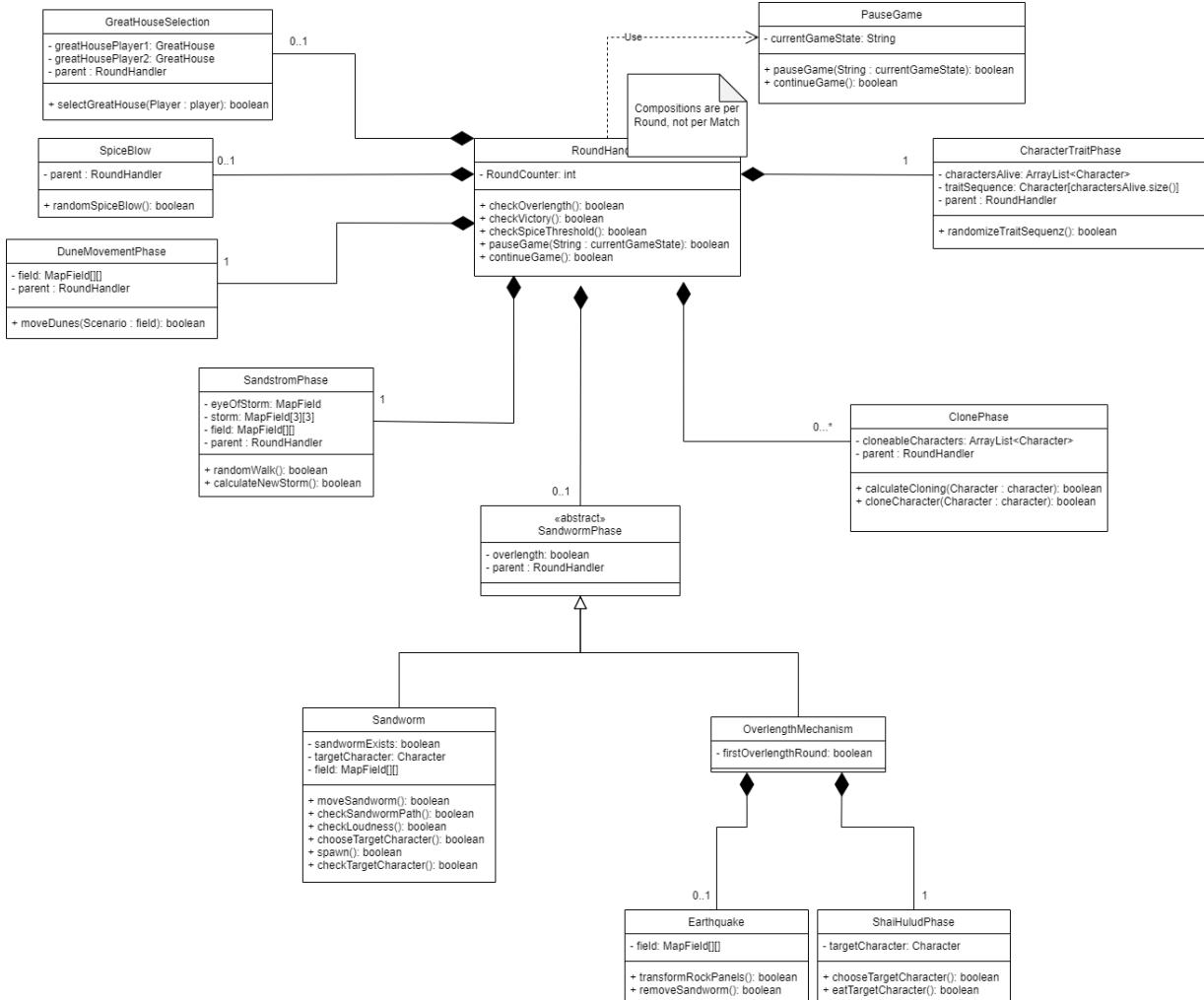


Abbildung 79: Übersicht über die Klassen, die für den korrekten Ablauf der einzelnen Runden zuständig sind. Während das Spiel läuft, also die Runden abgespielt werden, soll das Spiel pausiert werden können.

funktionale Anforderung	Methode(n), die diese umsetzt
FA9	<i>onOpen, onClose, onMessage, onError</i>
FA10	<i>fromMessage, toMessage</i>
FA11	<i>fromMessage, toMessage</i>
FA12	<i>alle Methoden der MessageController bzw. die der Interfaces</i>
FA85	<i>parseArgument</i>
FA114	<i>parseArgument</i>

Im Folgenden ist die Zuordnung der funktionalen Anforderungen aus Unterabschnitt 2.3 und den Klassendiagrammen zu erkennen. Dabei wird für die wichtigen Klassen jedes Diagramms aufgelistet, welche Anforderungen es abdeckt:

Klassendiagramm Charaktere und Häuser		zugeordnete Anforderungen
Charaktere		FA13, FA24, FA38, FA48, FA52, FA53, FA59, FA60, FA55, FA56, FA41, FA51, FA35 ^{→ p. 48} , FA52, FA53
Häuser		FA28, FA29, FA30, FA31, FA32, FA33, FA34

Klassendiagramm Szenario und Felder		zugeordnete Anforderungen
Feld		FA20, FA17, FA21, FA19, FA22, FA23, FA24, FA25, FA26
Szenario		FA15, FA27, FA14, FA13

Klassendiagramm Charakterzug-Rundphase		zugeordnete Anforderungen
Charakterzug-Rundphase		FA35, FA75, FA76, FA77, FA77
CharakterZug		FA77, FA50, Bewegungsschritt
Bewegungsschritt		FA77, FA46, FA47, FA45
Aktion		FA50, FA55, FA51
SpezielleAktion		FA55, FA56, FA57, FA59, FA60, FA61, FA???, FA???, FA???, FA???, FA???
NormaleAktion		FA51, FA52, FA53, FA54

Klassendiagramm Runde		zugeordnete Anforderungen
Runde (ohne FA's von Charakterzug)		FA62, FA64, FA65, FA66, FA67, FA68, FA69, FA70, FA71

Klassendiagramm network		zugeordnete Anforderungen
MessageConverter		FA9, FA10, FA12
ConnectionHandler		FA88, FA91, FA95, FA96, FA102, FA3
MessageEmitter		FA90, FA99, FA97
MessageReceiver		FA92
ClientMessageController		FA92
HumanMessageController		FA100,
SpectatorMessageController		FA101, FA6
AIMessageController		FA115, FA118

Klassendiagramm AI		zugeordnete Anforderungen
Main		FA115, FA119, FA114
Difficulty		FA119
FastServerInterface		FA92, FA97 FA-Verbindungsauftbau-KI
GameSimulator		FA117, FA122
World		FA15, FA35
Game		FA99, FA113

Klassendiagramm UserClient		zugeordnete Anforderungen
MenuScene		FA100, FA101, FA102
GameScene		FA105, FA104, FA103, FA107, FA106, FA107, FA108, FA109, FA110, FA111, FA112, FA113

Klassendiagramm Configuration	zugeordnete Anforderungen
ConfigurationParser	FA86, FA87, FA123
Configuration	FA128, FA126, FA124
PartyConfiguration	FA122, FA125, FA124
ScenarioConfiguration	FA122, FA126, FA127