Pflichtenheft

Bayram, Burak Kirsch, Julian Lang, Linda
Schott, Erik Weggel, Tom

2. Januar 2022

Inhaltsverzeichnis

1 Einleitung

1.1 Zweck und Ziel dieses Dokuments

Dieses Dokument dient dem Projektmanagement. Gemäß der Art eines Pflichtenhefts bestimmt dieses Dokument Ziele sowie die Wichtigkeit der Ziele, also ob das Ziel ein Muss-, ein Soll- oder ein Kann-Ziel ist. Des Weiteren werden die Ziele in Meilensteine gruppiert.

Jedes Ziel wurde mit dem gleichen Aufbau formuliert. Dazu wird eine Tabelle angeführt, welche Priorität, Abhängigkeiten von anderen Zielen, Zuständigkeiten und eine geschätzte Bearbeitungsdauer für jedes Ziel enthält. Weiterhin besteht jedes Ziel aus einer Beschreibung, welche mit einer sogenannten User Story angeführt wird.

1.2 Ziel und Zweck des Projektes

Ziel dieses Projektes ist es einen Client zu einem gegebenem Server zu implementieren, welcher selber das Brettspiel „Kingdom Builder“ implementiert.

Ein Client kann eigene Spiele erstellen und verwalten oder einem bereits erstelltem Spiel beitreten. Spiele können hierbei lokal im sog. „Hot-Seat“-Modus oder über das Netzwerk gegen andere Spieler oder Bots gespielt werden.

1.3 Nicht-Ziele des Projektes

Ziel des Projektes ist **nicht** die Implementierung eines Spieleservers.

1.4 Einsatz und Umgebung

Das Produkt ist für Endverbraucher geeignet, die einen Computer besitzen. Des Weiteren ist für die Inbetriebnahme eine Installation der „Java Runtime Environment 17“ (oder neuer) erforderlich.

Für die Onlinefunktionalität wird eine Internetverbindung benötigt.

1.5 Projektorganisation

1.5.1 Meilensteine

Die Entwicklung des Projektes wird in Meilensteinen unterteilt. Dabei verfolgen die Meilensteine aufeinander aufbauende Ziele.

Der erste Meilenstein „Chat Client“ dient der Umsetzung eines grundlegendes Systems, das ermöglicht einen Client bei dem Server zu registrieren und die dortigen Chatfunktionen zu nutzen.

Der zweite Meilenstein „Basis Spiel“ umfasst die Implementierung des Basisspiels, sodass Spieler eine Lobby erstellen oder einer bestehenden Lobby beitreten können und das Spiel gemäß der Regeln spielen können. Es wird erwartet, dass alle grundlegenden Eigenschaften umgesetzt werden. Dies umfasst sowohl die grafische Darstellung des Spiels als auch die Implementierung aller erforderlichen Spielregeln.

Der dritte und letzte Meilensteine „KI“ sieht vor, dass eine künstliche Intelligenz entwickelt wird gegen die Spieler spielen können. Des Weiteren ist auch ein „Hot-Seat“-Modus vorgesehen, welcher es ermöglicht mehreren Spielern gemeinsam an einem Computer zu spielen.

1.5.2 Entwicklerrollen

Innerhalb des Projektes nimmt jeder Teilnehmer eine spezifische Rolle an und ist somit Verantwortlicher und Ansprechpartner für einen Teilbereich des Projektes:

Build-Master (Julian Kirsch) ist verantwortlich für das Einrichten, Verwalten und Pflegen des Gradle Projekts. Diese Person sorgt dafür, dass das gesamte Projekt mittels Gradle gebaut werden kann. Ansprechpartner für alles, was mit den Bauen des Projektes zu tun hat.

Design-Pattern-Master (Linda Lang) hat die Aufgabe den Einsatz von Entwursmustern an allen Teilen des Projektes zu prüfen und den jeweiligen Modul-Mitgliedern bei der Umsetzung zu helfen.

Dokumentation-Master (Tom Weggel) ist dafür verantwortlich, dass alle relevanten (insbesondere alle sichtbaren) Methoden, Funktionen, Konstanten, etc. mittels JavaDoc dokumentiert werden und diese auf dem neuesten Stand sind.

Interface-Master (Erik Schott) ist für die Schnittstellen zwischen den einzelnen Module verantwortlich und klärt Anforderungen, Umfang und Funktion der einzelnen Komponenten des Softwareprojekts.

Test-Master (Burak Bayram) entscheidet was getestet werden soll und sorgt dafür, dass die Tests geschrieben werden.

1.5.3 Benutzerrollen

Dieses Dokument beschreibt Ziele anhand von drei Benutzerrollen. Erstere ist die Rolle des **Spielers**, welcher, wie der Name schon andeutet, der Konsument des Spiels ist und mit dem Produkt interagiert.

Die zweite Rolle, nämlich der **Root Spieler** (kurz: Root), bezieht sich auf Spieler, welche einen eigenen Spieleserver betreiben. Im Gegensatz zu normalen Spielern hat der Root administrative Freigaben und kann bspw. Spieler vom Server „kicken“.

Zuletzt gibt es die Rolle des **Entwicklers**, welcher aktiv das hier spezifizierte Produkt entwickelt.

1.5.4 Arbeitsweise

Im Folgenden sind Arbeitsweisen festgelegt, die für alle teilnehmende Entwickler verpflichtend sind.

Branches

Zu jedem Zeitpunkt im Projekt existieren zwei Branches, nämlich „master“ sowie „development“. Der master-Branch stellt den aktuellsten Zustand des Projektes **nach** Abschließen eines Meilensteins dar. Der development-Branch hingegen spiegelt den aktuellsten Entwicklungsstand wider.

Des Weiteren soll, sofern semantisch sinnvoll, für jedes Ziel, das bearbeitet wird, ein eigener, sog. „Feature-“Branch erstellt werden.

Die Branches werden der „GitFlow“-Methode folgend benannt.

Pull Requests

Ist die Entwicklung eines Zieles abgeschlossen, so muss ein „Pull Request“ angelegt und von den jeweiligen Test- sowie Dokumentation-Master auf Vollständigkeit geprüft.

Code Style

Der Code Style ist strikt an den Java Konventionen gehalten. Die GUI-Elemente werden der SnakeCase-Konvention folgenden notiert und mit einem FXML-Tag versehen.

Des Weiteren muss folgende Reihenfolge der Elemente innerhalb einer Klasse eingehalten werden:

- Statische Variablen
- Instanzvariablen
- Konstruktoren
- Methoden
- Setter
- Getter

Unit Test Style

Unit Tests werden pro Klasse mit JUnit 5 durchgeführt, wobei einzelne Testklassen, soweit es möglich ist, unabhängig voneinander sein müssen, damit fehlgeschlagene Tests eindeutig einer Klasse zugeordnet werden können.

Objekte aus anderen Klassen, die zum Testen benötigt werden als Member innerhalb der Testklasse aufgeführt und mittels einer Methode, die mit „@BeforeEach“ annotiert ist, instanziiert.

Es müssen alle Methoden außer „Getter“ und „Setter“ getestet werden.

Wenn in einer Testmethode mehrere Assertions vorhanden sind, müssen diese mit der überladenen Funktion auch eindeutig benannt werden, damit bei einem fehlgeschlagenen Test auf eine Assertion zurückgeführt werden kann. Darüber hinaus sollen mehrere Assertions nur dann verwendet werden, wenn diese alle im gleichen Kontext stehen bzw. auf die selbe Art und Weise getestet werden.

Testklassen werden mit „[KlassenName]Test“ und Testmethoden mit „test[FeatureName][optionalerSpezialFall]“ benannt.

2 Ziele

2.1 Meilenstein: Chat Client

2.1.1 Grundaufbau der Programmarchitektur

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	JK	2h	

Als Entwickler möchte ich einen Grundlage für das Programm schaffen.

Hierbei soll eine Software-Komponente entwickelt werden, welche das Redux-Architekturmuster, samt „Store“, „State“, „Reducer“ und „Actions“ umsetzt. Diese Software-Komponente fungiert als Kern der Anwendung und stellt auch die Schnittstelle zwischen den verschiedenen Komponenten der Software dar.

2.1.2 Spielerliste

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	ES, TW	4h	2.1.6; 2.1.1

Als Spieler möchte ich mir andere Spieler anzeigen lassen, die mit dem Server verbunden sind.

Hierbei muss eine GUI-Komponente erstellt werden, welche alle aktuell mit dem Server verbundenen Spieler auflistet.

Durch eine Markierung in der Auflistung müssen Spieler für eine Direktnachricht auswählbar sein. Des Weiteren muss ein Button zur Verfügung stehen, welcher die Markierung wieder aufhebt.

2.1.3 Chat-Log

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	ES, TW	20h	2.1.6

Als Spieler möchte ich mit anderen Clients per Textnachricht kommunizieren und diese angezeigt bekommen.

Hierbei muss eine GUI-Komponente implementiert werden, welche ein- und ausgehende Nachrichten in chronologischer Reihenfolge anzeigt.

Des Weiteren muss das Versenden von lokalen Nachrichten, sofern der Spieler einem Spiel beigetreten ist, an eine Lobby, sowie das Versenden von direkt adressierten und „globalen“ Nachrichten an alle Spielern auf den Server ermöglicht werden.

Nachrichten müssen visuell in sofern von einander differenziert werden, so dass...

- ...Sender- bzw. Empfängernamen **fett** angezeigt werden.
- ...Nachrichten je nach Art (zum Beispiel Textnachricht oder Statusnachricht) farblich gekennzeichnet sind.

2.1.4 Chat-Eingabe

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	ES, TW, BB	5h	2.1.6

Als Spieler möchte ich Textnachrichten für den Chat verfassen, um mit anderen Spielern zu kommunizieren.

Eine GUI-Komponente muss implementiert werden, die es ermöglicht Spieler per Tastatureingabe Text einzutippen. Weiter muss über Buttons die verfasste Nachricht „global“ also serverweit/lobbyweit als auch an ausgewählte Spieler versendbar sein.

2.1.5 Anzeigen von Statusnachrichten

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Mittel	ES, TW	3h	2.6.1

Als Spieler möchte ich benachrichtigt werden, wenn ich oder ein anderer Spieler den Server verlässt.

Hierbei muss der Chat-Log erweitert werden, so dass Statusnachrichten angezeigt werden.

2.1.6 Spielprotokoll De-/Serialisieren

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	JK	20h	-

Als Entwickler möchte ich Nachrichten de-/serialisieren um mit denen Programmieren zu können.

Hierbei muss eine Software-Komponente entwickelt werden, welche Nachrichten vom Server in eine Objektpräsentation und Befehle an den Server von der Objektpräsentation in Textform überführt. Hierfür müssen Nachrichten bzw. Befehle für folgende Aktionen oder Ereignisse unterstützt werden:

- Ein- und Ausloggen des Spielers
- Abfrage der aktuellen Spieler
- Benachrichtigung über Beitritt/Verlassen eines Spielers
- Versenden einer Nachricht
- Empfangen einer Nachricht

2.1.7 Verbindungsaufbau

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	JK	5h	-

Als Entwickler möchte ich eine Verbindung zu einem Spieleserver aufbauen, um Befehle zu versenden, Benachrichtigungen empfangen und auf diese reagieren zu können.

Hierbei muss eine Software-Komponente entwickelt werden, welche eine aktive Netzwerkverbindung aufbaut.

2.1.8 Namenseingabe

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	ES, TW	1h	2.1.1; 2.6.1

Als Spieler möchte ich meinen eigenen Namen auswählen, der anderen Spielern angezeigt wird.

Hierbei muss eine GUI-Komponente erstellt sowie die Netzwerk-Komponente erweitert werden, so dass ein Nutzer seinen Wunschnamen angeben und sich auf dem Server einloggen kann.

2.1.9 Verbindung beenden

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	Julian Kirsch	3h	2.6.1

Als Entwickler möchte ich eine bestehende Verbindung zum Server trennen um entweder das Programm zu beenden oder um auf andere Server beitreten zu können.

Hierbei muss die Netzwerk-Komponente erweitert werden, so dass diese den aktuellen Spieler vom Server ausloggt und die Verbindung schließt.

2.2 Meilenstein: Basis Spiel

2.2.1 Hauptmenü-Maske (MainView)

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	TW	2h	-

Als Spieler möchte ich die Möglichkeit haben ein lokales Spiel zu starten, mich mit einem Spieleserver zu verbinden und die Programmeinstellungen zu ändern.

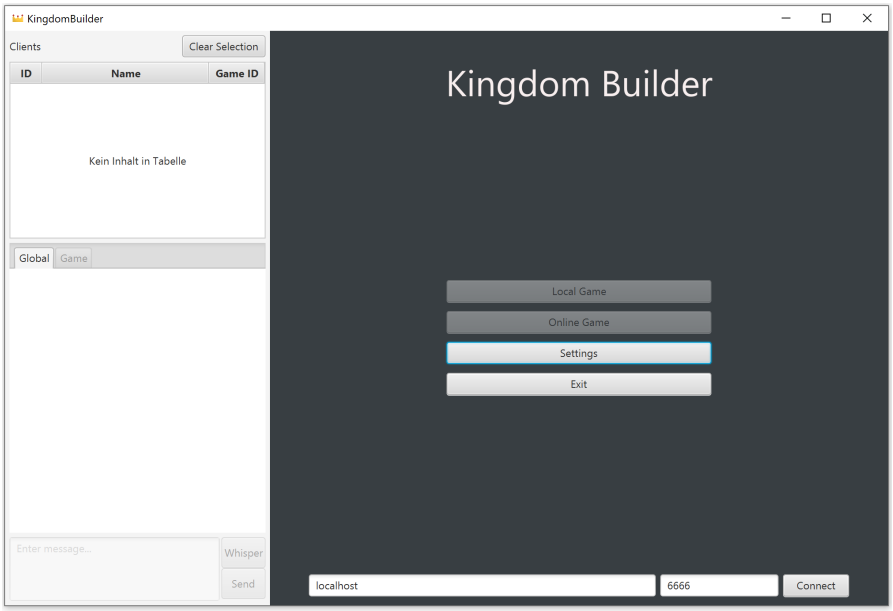
Hierbei muss eine GUI-Maske entworfen und implementiert werden, die folgende Buttons zur Verfügung stellen muss:

Local Game öffnet die Maske zum Erstellen eines Spiels („GameSettingsView“).

Online Game öffnet die Maske zur Initiierung einer neuen Netzwerkverbindung („ServerConnectView“), sofern man nicht schon verbunden ist. Ansonsten wird direkt die Maske mit dem Spielbrowser („GameBrowserView“) geöffnet.

Settings öffnet die Einstellungs-Maske („ApplicationSettingsView“).

Exit beendet das Programm.



(Skizze)

2.2.2 Einstellungs-Maske (ApplicationSettingsView)

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Mittel	TW	5h	-

Als Spieler möchte ich die Möglichkeit haben die Einstellungen des Programms zu ändern, um dieses an meine Anforderungen anzupassen.

Hierbei muss eine GUI-Maske entworfen und implementiert werden, die es einem ermöglicht die Präferenz aus einer Auswahl von verfügbaren Sprachen sowie den eigenen bevorzugten Namen zu setzen. Des Weiteren müssen zwei Buttons zur Verfügung gestellt werden, welche es ermöglichen die Änderungen anzuwenden oder direkt in die Hauptmenü-Maske „MainView“ zu wechseln.

2.2.3 Verbindungs-Maske (ServerConnectView)

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	TW	2h	-

Als Spieler möchte ich mich unter Angabe einer IP-Adresse zu einem Spieleserver verbinden.

Hierbei muss eine GUI-Maske entworfen und implementiert werden, die einem die Eingabe einer IP-Adresse ermöglicht. Weiter sollen auch zwei Buttons „Verbinden “ und „Zurück “ zur Verfügung stehen.

Wird der Button „Verbinden“ geklickt, so wird eine Verbindung erstellt und der Nutzer zum Gamebrowser („GameBrowserView“) geführt. Schlägt die Verbindung fehl, so wird der Nutzer über einen Dialog über den Fehler informiert. Wird der Button „Zurück“ geklickt, so wird der Nutzer zurück zum Hauptmenü („MainView“) geführt.

2.2.4 Gamebrowser (GameBrowserView)

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	LL, TW	7h	-

Als Spieler möchte ich ein Spiel erstellen, beobachten oder einem Spiel beitreten.

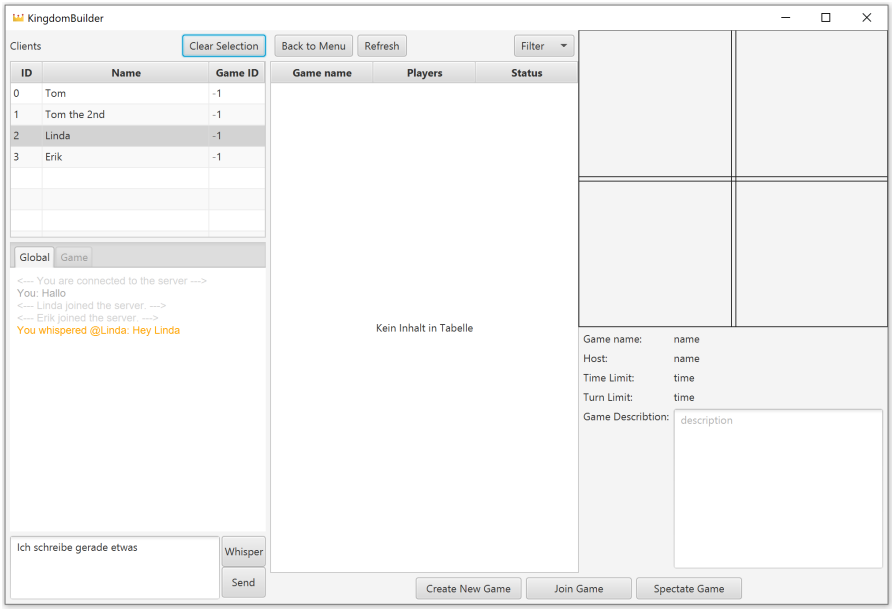
Hierbei muss eine GUI-Maske entworfen und implementiert werden, die alle Spiele des aktuellen Servers auflistet, sowie einem Spieler ermöglicht ein Spiel zu erstellen.

Es werden alle aktuell laufenden Spiele auf dem Server, zu dem der Client verbunden ist, angezeigt. Diesen Spielen kann als Spieler beigetreten werden, wenn sie nicht bereits voll besetzt sind. Es kann stets einem Spiel als Zuschauer beigetreten werden. Es kann außerdem ein neues Spiel erstellt werden.

In der oberen linken Ecke befinden sich 3 Buttons. Der erste Button „Zurück “ sorgt dafür, zurück in das Hauptmenü („MainView “) zu kommen. Daneben befindet sich ein Button „Refresh “, um die Tabelle manuell zu aktualisieren. Hierneben ist eine Combo-Box, mit der man in der Tabelle nach einem gewissen Spiel-Zustand filtern kann. Nach der Auswahl in der Combobox wird die Tabelle anhand dieser angepasst. Die laufenden Spiele werden in einer TableView angezeigt. Diese besitzt eine Spalte für den Namen des Spiels, die Anzahl der Spieler und den Status, ob das Spiel auf Spieler wartet, läuft oder beendet ist.

Auf der rechten Seite werden detaillierte Informationen zu dem in der Tabelle ausgewählten Spiel angezeigt. Diese Informationen bestehen aus dem Namen des Spiels, wer der Host des Spiels ist, dem eingestelltem Zeit- und Zuglimit. Anschließend folgt eine TextArea, die die Beschreibung des Spiels anzeigt. Die Informationen werden mit einer GridPane gegliedert.

In der untersten Zeile befinden sich drei Buttons. Der erste Button sorgt dafür, dass ein User ein Spiel erstellt und die View zu den Spiel-Settings wechselt. Mit dem Button daneben betritt der User das in der Tabelle ausgewählte Spiel. Die aktuelle View wird zur View geändert, die das Spiel anzeigt. Der letzte Button sorgt dafür, dass der User dem in der Tabelle ausgewählten Spiel als Zuschauer beitritt. Hierbei wird die View auch zur View geändert, die das Spiel anzeigt.


KingdomBuilder
Clients
Clear Selection
Back to Menu
Refresh
Filter
ID
Name
Game ID
0
Tom
-1
1
Tom the 2nd
-1
2
Linda
-1
3
Erik
-1
Global
Game
<--- You are connected to the server --->
You: Hallo
<--- Linda joined the server. --->
<--- Erik joined the server. --->
You whispered @Linda: Hey Linda
Ich schreibe gerade etwas
Whisper
Send
Game name
Players
Status
Kein Inhalt in Tabelle
Game name:
name
Host:
name
Time Limit:
time
Turn Limit:
time
Game Describtion:
description
Create New Game
Join Game
Spectate Game

(Skizze)

2.2.5 Spieleinstellungen (GameSettingsView)

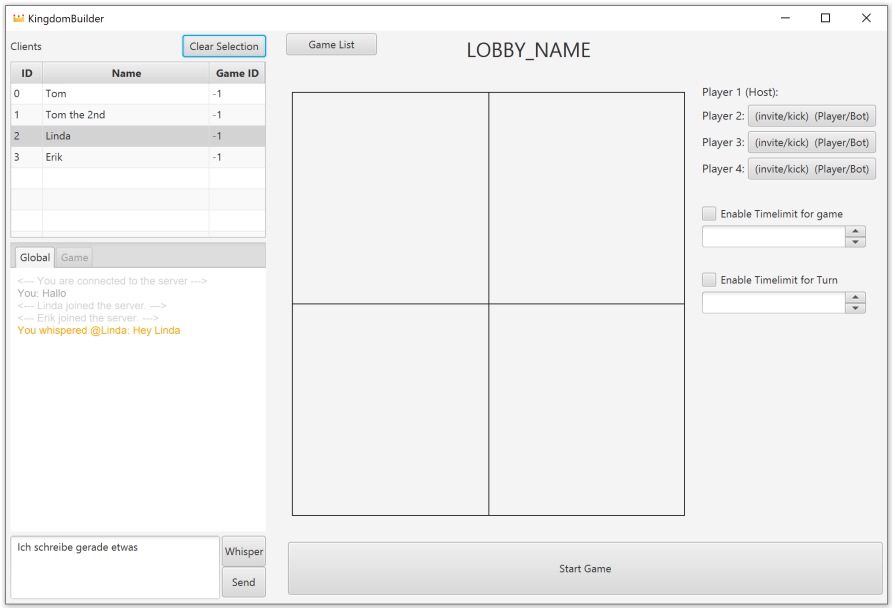
Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	BB, TW, LL	3h	-

Als Spieler möchte ich die Einstellungen für Spiel an meine Wünsche anpassen.

Hierbei soll eine GUI-Maske entworfen und implementiert werden, die einem Spieler ermöglichen folgende Einstellungen zu setzen:

- Zeitlimit
- Zuglimit
- Maximale Spielerzahl
- Bots

Des Weiteren müssen Buttons angeboten werden, die einem ermöglichen das Spiel zu „hosten“ und die Spieleinstellungen zu verlassen. Wird der Button „Hosten“ geklickt, so wird das Spiel erstellt und der Ersteller tritt automatisch bei. Anschließend wird der Spieler in die Spielinformations-Maske („GameInfoView“) geführt. Wird der Button „Zurück“ gedrückt, so wird der Spieler zurück in das Hauptmenü („MainView“) geführt.



(Skizze)

2.2.6 Spielinformationen (GameInfoView)

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	TW	6h	-

Als Spieler möchte ich Information über das aktuell beigetretene Spiel einsehen.

Hierbei muss eine GUI-Maske entworfen und implementiert werden, welche relevante Informationen über das aktuelle Spiel anzeigt.

Diese werden in zwei Bereiche unterteilt. Zunächst werden folgende Informationen angezeigt:

- Anzahl der vergangenen Züge
- Zeitlimit
- Zuglimit
- Siegbedingung
- Spielerinformationen (Name, verbleibende Siedlungen, Punkte, Farbe)

Außerdem soll das Spielfeld, die Tokens und Anzahl der auswählbaren Hexagone angezeigt werden.

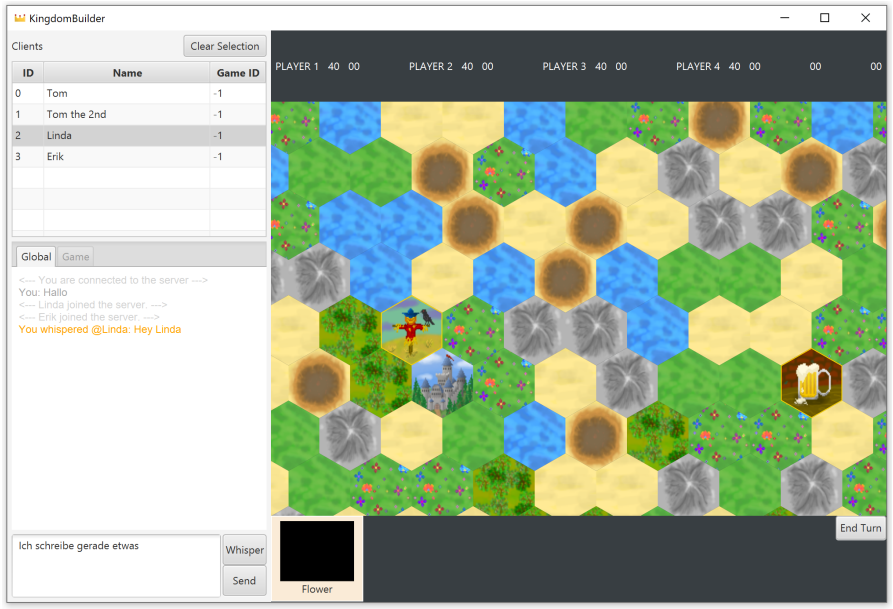
2.2.7 Spielbrett (GameView)

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	TW, LL	30h	2.2.6

Als Spieler möchte ich das Spielbrett visuell dargestellt bekommen.

Hierbei muss eine GUI-Komponente dargestellt werden, welche alle Hexagone gemäß dem aktuellen Spielfortschitts darstellt. Des Weiteren müssen Interaktionen implementiert werden, so dass die Karte bewegt und vergrößert sowie verkleinert werden kann. Es gilt zu beachten, dass Spieler die Karte nicht über den Rand hinaus bewegen dürfen sowie das Grenzen für das Verkleinern und Vergrößern der Karte gesetzt werden.

Die Karte muss über die Pfeiltasten bewegbar sein und über das Mausrad vergrößert oder verkleinert werden können.



(Skizze)

2.2.8 Spielregeln

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	ES, BB	16h	-

Als Entwickler möchte überprüfen, ob eine antizipierter Spielzug gültig ist, um fehlerhafte Züge zu unterbinden.

Hierbei soll eine Komponente entwickelt werden, welche anhand des aktuellen Spielfortschritts überprüft, ob der nächste Zug zulässig ist. Ist der Zug unzulässig, so soll dieser unterbunden werden. Die Spieler bekommen in fester Reihenfolge ihre Züge zugeteilt. Der Ablauf eines Zuges ist aufgespalten in Basis- und Bonuszüge. Basiszüge platzieren Siedlungen entsprechend der Feldkarte des Zuges und der benachbarten Siedlungen des Feldes, an dem die Siedlung platziert werden soll. Pro Zug müssen 3 Siedlungen auf diese Weise platziert werden. Bonuszüge verwenden Tokens und ermöglichen es, Siedlungen zu platzieren oder zu verschieben. Jedes Token kann einmal pro Zug verwendet werden. Ein Spieler kann einmalig von einem Token-Feld ein Token erhalten, wenn er eine Siedlung neben diesem platziert. Es gibt 10 verschiedene Win Conditions, welche definieren, wie Punkte in einem Spiel vergeben werden. Diese werden am Anfang des Spiels festgelegt. Entsprechend der Win Conditions werden die Punkte berechnet und der Punktestand aktualisiert.

2.2.9 Spielbrett 2 (GameView)

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	TW	10h	2.2.7

Als Spieler möchte ich Züge auf dem Brett ausführen können, um das Spiel zu spielen.

Ist auf einem Feld eine Siedlung platziert, so wird diese Siedlung auf dem Feld mit der entsprechenden Spielerfarbe gezeichnet.

Markierungen: Zu Beginn eines Zuges werden die Hexagone markiert, auf denen eine Siedlung entsprechend des Basiszugs platziert werden kann. Wird ein Token ausgewählt, so verändern sich die Markierungen zu den Feldern, auf denen der entsprechende Token angewendet werden kann. Beim Aufheben der Token-Auswahl ändern sich die Markierungen wieder zu den zuvor markierten Feldern. Wurde der Basiszug schon getätigt, wird nichts mehr markiert.

Basiszug: Wenn der Spieler nicht bereits alle Siedlungen für die Runde platziert hat, so kann er einen Basiszug tätigen, wobei eine Siedlung ohne Verwendung von Token platziert wird. Wird nicht ein Token in der Schaltfläche ausgewählt und ein Hexagon des Terrain Typs der Runde angeklickt, wird geprüft, ob dies ein zulässiges Feld ist. Ist dies ein valides Feld, so wird eine Siedlung platziert. Ansonsten bekommt der Spieler eine Mitteilung, dass dies kein valides Feld ist.

Terrain Karte: Am unteren Bildschirm befindet sich eine Anzeige, welche durch ein Bild und eine Beschreibung anzeigt, welche Terrain Karte gerade im Zug ausgewählt wurde. Diese wird jede Runde aktualisiert.

Token: Um einen Token zu verwenden, kann dieses an der unteren Schaltfläche neben der Terrain Karte angeklickt werden. Wenn ein Token ausgewählt ist, werden andere eigene Tokens ausgegraut und deaktiviert. Dem Spieler wird anschließend durch ein Highlight gezeigt, welche Züge möglich sind und bei Bedarf können diese durchgeführt oder die Token-Auswahl abgebrochen werden, indem das Token erneut angeklickt wird. Die Tokens werden deaktiviert, wenn der Client nicht am Zug ist. Zudem werden diese neu angezeigt, wenn sich der Hotseat Spieler wechselt.

Zum Verschieben einer Siedlung mittels Token wird die Siedlung zunächst ausgewählt und anschließend das neue Hexagon angeklickt.

Updates bei Änderungen: Bei Zügen anderer Clients wird die Kamera der Karte zu dem Feld bewegt, bei welchem sich etwas verändert hat.

2.2.10 Überarbeitung und Abschließung des Netzwerks

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	JK, LL	40h	-

Als Entwickler möchte ich alle Befehle und Benachrichtigungen des zur Verfügung gestelllten Server nutzen können.

Hierbei soll die Netzwerk-Komponente überarbeitet werden, so dass alle Benachrichtigungen und Befehle unterstützt werden.

2.3 Meilenstein: KI und Feinschliff

2.3.1 Vollständige Integration mit dem Netzwerk-Modul

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	JK, LL	10h	-

Als Spieler möchte ich mit meinen Freunden über das Netzwerk als auch lokal am selben PC gegen KIs spielen.

Hierbei soll das Netzwerk-Modul umfangreich integriert werden, so dass mehrere Spieler gemeinsam über das Netzwerk an einem Spiel teilnehmen können. Hier soll auch die Möglichkeit beachtet werden, dass KIs auch als Spieler teilnehmen. Des Weiteren soll der Hot-Seat-Modus implementiert werden, so dass mehrere Spieler am selben PC gegeneinander und gegen Spieler über das Netzwerk spielen können.

2.3.2 Siegesbedingung für Bots

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	alle	30h	-

Ich als Entwickler möchte für Bots die Siegesbedingungen zur Verfügung stellen. Ich als Spieler möchte meinen aktuellen Punktestand wissen.

Hierbei muss zwischen den 10 Siegesbedingungen unterschieden werden:

Fischer Prüft, wie viele Siedlungen angrenzend an Wasser-Feldern gebaut wurden. Pro Siedlung bekommt der Spieler, dem die Siedlung gehört, einen Punkt. Nicht beachtet werden Siedlungen, welche direkt auf Wasser liegen.

Bergleute Prüft, wie viele Siedlungen angrenzend an Bergen gebaut wurden. Pro Berg bekommt ein Spieler einen Punkt, egal wie viele Siedlungen der Spieler an diesem Berg besitzt.

Händler Prüft, ob ein Spieler ein oder mehrere Orts-/Burgfelder durch eigene Siedlungen verbunden hat.

Arbeiter Prüft für jedes Orts-/Burgfeld, wie viele Siedlungen angrenzen. Pro Siedlung erhält der Spieler einen Punkt.

Entdecker Pro Horizontaler Linie auf dem Spielbrett mit mindestens einer Siedlung bekommt der zugehörende Spieler einen Punkt.

Ritter Für jeden Spieler wird die horizontale Linie des Spielbretts mit seinen meisten Siedlungen ermittelt. Für jede Siedlung auf dieser Linie bekommt der Spieler 2 Punkte.

Einsiedler Es wird 1 Punkt für jedes separate Siedlungsgebiet pro Spieler vergeben. Ein Siedlungsgebiet wird definiert durch eine Menge von angrenzenden Siedlungen.

Lords Pro Quadrant werden Punkte vergeben. Spieler mit den meisten Siedlungen erhalten 12 Punkte. Spieler mit den zweitmeisten Siedlungen erhalten 6 Punkte.

Bürger Jeder Spieler erhält 1 Punkt für je 2 seiner Siedlungen in seinem größten Siedlungsgebiet.

Bauern Jeder Spieler erhält 3 Punkte für jede Siedlung in dem Quadranten, in dem er die wenigsten Siedlungen besitzt. Bei 2 Quadranten mit der gleichen geringsten Anzahl an Siedlungen wird nur ein Quadrant gewertet.

2.3.3 Bots

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Hoch	Alle	20h	2.3.2

Ich als Spieler möchte Bots zu meinem Spiel hinzufügen und somit jederzeit beliebig viele Gegenspieler für mein Spiel haben können.

Ermittlung strategisch günstiger Züge mittels der bereitgestellten Funktionen in der Spiellogik.

Dies erfolgt, indem mittels der Win Conditions der Zug geplant wird. Die Berechnung erfolgt in jedem Zug, da Vorausschauen nicht funktioniert, weil zukünftige Karten unbekannt sind.

Das Ziel liegt darin, den Spielzug zu optimieren und die meisten Punkte zu bekommen. Dabei werden alle möglichen platzierbaren Felder betrachtet und das beste Feld ausgesucht. In diesem Moment wird jedoch noch keine Siedlung platziert, sondern die Felder angeschaut, die nun möglich sind. Dies wird in einem Graphen aufgenommen. Hiernach werden eine oder mehrere Alternativen betrachtet, damit die KI prüft, ob die erste Siedlung wo anders platziert werden kann, damit die Punkte maximiert werden können. Alle Alternativen werden in dem Graphen aufgenommen und letztendlich der beste Zug ausgewählt und durchgeführt. Dieses Konzept ist bekannt unter dem Namen Zielorientierte Aktionsplanung (GOAP).

Die optionale Schwierigkeitseinstellung wird umgesetzt, in dem die KI für Ihre Planung für den Zug lediglich eine Teilmenge der Win Conditions auswählt. Die anspruchsvolle KI wird anhand aller Win Conditions ihren Zug planen.

Ein Bot wird mit Hilfe eines internen Client implementiert, der erstellt wird, wenn in den Spieleinstellungen ein Bot ausgewählt wird. Es wird für jeden Bot ein eigenes Client Objekt erstellt, damit dieser ein eigenen Socket zum Server besitzt und somit auch eine eigene ID. Dies ist notwendig, damit ein Bot eigene Spielzüge unabhängig vom User Client durchführen kann.

2.3.4 Root

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Mittel	tba.	2h	3.2.1

Ich als Spieler möchte zum Root Spieler werden.

Der Chat hat eine Funktionalität, die erlaubt, mit einem Befehl "/root" und einem folgenden String als Passwort sich im Server als Root Spieler anzumelden. Die Eingabe des Befehls erfolgt im Eingabefeld für Chatnachrichten und wird bestätigt durch Absenden der Nachricht. Nach der Anmeldung sind die folgenden administrativen Befehle des Root Spieler im Chat auf die gleiche Weise ausführbar:

Mit "/kick ID" kann ein Spieler anhand der eingegebenen ID vom Server gekickt werden.

Mit "/shutdown server" kann der Server beendet werden.

2.3.5 Dekorationen

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Mittel	tba.	10h	2.2.9

Als Entwickler möchte ich eine „schöne “ GUI haben um den Spieler das Spiel schmackhaft zu machen.

Dieses Ziel befasst sich mit dem Design der GUI. Hierbei sollen GUI-Element dekoriert und angepasst werden. Dies erfolgt unter Anderem durch die Auswahl geeigneter Farben, Schriftarten und Formen sowie Anpassungen am Layout.

2.3.6 Animationen

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Gering	tba.	10h	2.2.9

Hierbei sollen Züge anschaulicher über Animationen dargestellt werden. Dabei werden insbesondere die Veränderungen am Spielbrett wie z.B. das Platzieren oder Verschieben von Siedlungen flüssig und klar verfolgbar animiert.

Optional können Wolken animiert dargestellt werden, welche sich über das Feld bewegen.

2.3.7 3D-Darstellung

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Gering	tba.	8h	2.2.9

Hierbei soll das Spiel nicht mehr durch einfach 2D-Bilder sondern durch einfache 3D-Modelle dargestellt werden.

2.3.8 Benutzerhandbuch

Priorität	Zuständiger	Arbeitsaufwand	Abhängigkeiten
Mittel	alle	10h	2.3.5

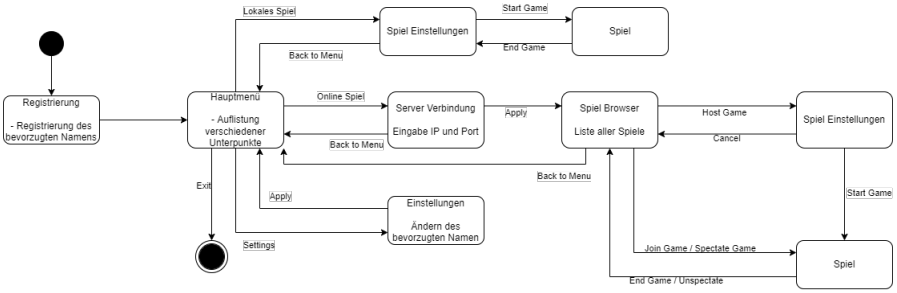
Erstellen eines Benutzerhandbuchs, welche dem Endanwender die Bedienung der Anwendung näher erklärt.

Der Design-Stil wird anhand unseres Programm-Stils gewählt und anschaulich mit Bildern dargestellt.

Dies wird wie im Lastenheft als Portable Document Format (pdf) in der Papiergröße DIN A5 erstellt.

3 Anhang

3.1 Menüführung



3.2 Architektur: Redux

Die Redux Architektur besitzt verschiedene Bereiche. Es existiert der Store, mehrere Reducers, die Views und Actions.

Der Mittelpunkt ist hierbei der Store. In Redux existiert immer nur ein Store-Objekt. Dieser beinhaltet den State des ganzen Programms. Dies hat den Vorteil, dass alle Daten des Programms an einem Ort gespeichert sind und Undo und Redo Funktionen leichter umzusetzen sind. Hierbei ist jedoch zu beachten, dass Funktionen nur Leserechte auf den State haben und keine Schreibrechte. Hier kommen die Actions in Einsatz. Interagiert der Nutzer mit der View, wird eine Action erstellt und dem Dispatcher des Stores übergeben. Dieser ruft den Reducer mit dem State und der Action auf. Nur der Reducer kann nun anhand der Action den State verändern.

Wenn der Reducer erfolgreich den State verändert hat, werden anhand der Änderung Methoden aufgerufen, die die View updaten. Dabei schreibt sich die View beim Initialisieren in eine OnChange-Methode ein und teilt dieser mit, welche Methoden aufgerufen werden müssen, wenn eine Änderung vorliegt.

3.3 Glossar

Action Objekt anhand dessen der State/Store der Application aktualisiert wird.

Application Anwendung auf der das Spiel abläuft.

Assertion Wird bei Tests verwendet, um einen erwarteten Wert mit einem tatsächlichen zu Vergleichen.

Basiszug Platzieren der 3 Siedlungen auf dem gegebenen Feldtyp. Pflichtaktion pro Zug.

Benutzer Anwender des Programms.

Bonuszug Zusätzliche Aktion, wie Platzierung einer Siedlung auf einem vorgeschriebenen Feld oder Verschieben bereits gesetzter Siedlungen. Ausgelöst werden diese durch die Verwendung von Tokens.

Bots Computer (Künstliche Intelligenz) die anstelle eines Menschen spielt.

Bump Map Verbessert den Detailreichtum von 3D Objekten.

Burgfeld Nicht bebaubares Feld. Bei angrenzend gebauten Siedlungen erhält der Spieler 3 Punkte.

Button Ein Knopf, der betätigt werden kann.

Chat Bereich in den Clients Nachrichten miteinander austauschen können.

CheckBox Steuerelement grafischer Benutzeroberflächen. Auch bekannt als: Auswahlkasten/Kontrollkästchen/Markierungsfeld.

Client Eine Instanz, die sich mit dem Server verbindet.

Client-Message Message, die ein Client an den Server schickt.

ComboBox Element der Oberfläche, mit der man unterschiedliche gegebene Eigenschaften auswählen kann.

CSS-Stylesheet Wird für optische Verbesserungen in GUI verwendet.

Deserialisieren Umwandlung eines Bytestroms, der durch Serialisierung eines Objekts gewonnen wurde. Der Bytestrom wird zurück in das Objekt umgewandelt.

Dialogfenster Fehlermeldungen, die vom Server ausgegeben werden.

Direktnachricht Schicken einer Nachricht, die nur bei einer Teilmenge an Nutzern ankommt.

Einzelspieler-Modus Modus, in dem ein Spieler alleine spielt.

Feature Die Funktionalität einer Software.

Fehlermeldung Eine Benachrichtigung, wenn etwas schief geht.

Feldtyp Typ eines Feldes (bsp.: Wüste, Wasser, Gras).

FXML Tag Markierung, dass der folgende Code Teil ein FXML Bereich ist.

Gradle Ist ein auf Java basiertes Build-Management-Automatisierungs-Tool. Es dient zur Beschreibung der zu bauenden Projekte.

GridPane Layout, welche Inhalte in Tabellen Form anzeigt.

HBox Layout, welches Inhalte nebeneinander anzeigt.

Hexagon Ein Feld auf das ein Spieler oder Token stehen kann.

Highlight Visuelle Darstellung bestimmter Objekte, die dadurch hervorgehoben sind.

Hosten Erstellen eines Spiels.

Hotseat An einem PC können mehrere Spieler gleichzeitig Spielen.

html Hypertext Markup Language. Code welcher benötigt wird, um Webinhalte darzustellen.

ID Eindeutige Nummer die einem Client zugeordnet wird.

Initialisieren Das erstmalige Festlegen einer Variablen auf einen bestimmten Wert.

Interessierte / Observer / Subscriber Methoden, die erfahren wollen, ob Veränderungen geschehen sind.

IP Die IP des Servers an den sich der Client verbindet.

Kamera Der bewegbare, sichtbare Ausschnitt der Karte.

Karte Das Spielfeld für das Spiel.

KI (Künstliche Intelligenz) Computer als nicht-menschlicher Spieler.

Kicken Einen Client vom Server trennen.

Label Ein GUI Element, welches Text anzeigt.

Leserecht Liest nur Daten und kann diese nicht modifizieren.

Markierung Visuelle Darstellung bestimmter Objekte, die dadurch hervorgehoben sind.

modular Ein ganzer Block aufgeteilt in kleinere Module, Komponenten oder Bausteine.

Netzwerknachricht Eine Nachricht, wenn der Server eine Message an den Client sendet.

Ortsfeld Unbebaubare Felder. Angrenzende Siedlungen bekommen einen Token.

Parsen Zerlegung und Umwandlung einer Eingabe in ein für die Weiterverarbeitung geeigneteres Format.

Protokoll Gegebenes Server-Protokoll, in dem jede Server-/Client-Message aufgeführt ist.

Port Adresse am Server an der sich ein Client verbinden kann.

Quadrant Das Spielfeld wird in vier Teile aufgeteilt. Ein Quadrant ist eines davon.

quittieren Synonym für beenden.

Record(-Datentyp) Spezielle Klasse zur Speicherung unveränderlicher Daten, wobei Accessor-Methoden, hashCode(), toString() und equals() implizit implementiert sind.

Rectangle Java FX Klasse für Rechtecke.

Redo Etwas wiederholen.

Reducer Teil der Redux Architektur. Ist zuständig, die durch Actions aufgerufenen Änderungen am State durchzuführen.

rendern Zeichnen und Anzeigen einer Grafik.

Repository Stellt ein verwaltetes digitales Verzeichnis zur Speicherung digitaler Objekte dar.

Root Spieler Benutzer mit administrativen Rechten.

Schreibrecht Kann Daten lesen und modifizieren.

Server Repräsentiert ein Computerprogramm oder ein Gerät, welches Daten, Funktionalitäten oder Dienstprogramme bereitstellt, damit andere Computer oder Programme darauf zugreifen können. Dies geschieht meist über ein Netzwerk.

Server-Message Eine Nachricht, die vom Server gesendet wird und Informationen für Clients enthält.

Siedlung Die Spielfigur des Spieles, welche auf ein Hexagon platziert werden kann.

Siedlung platzieren Eine Spielfigur auf einem Feld platzieren.

Siedlung verschieben Eine Spielfigur von einem Feld entfernen und auf ein anderes Feld platzieren.

Siedlungsfelder Felder auf dem Spielbrett, welche mit einer Siedlung bebaut sind.

Sieges-Karten / Siegesbedingungen Drei von zehn ausgewählten Siegesbedingungen. Diese beschreiben, unter welchen Bedingungen ein Spieler Punkte bekommt.

Snake-Case Notation von Variablen mit Unterstrichen.

Spiel beitreten Ein Benutzer tritt einem Spiel bei, um in diesem zu spielen oder diesem zuzuschauen.

Spielbrett Das Spielfeld, das aus den 4 Quadranten beziehungsweise deren einzelnen Feldern besteht.

Spielfunktion Die Funktionen, die benötigt werden, ein Spiel als Datenstruktur zu verwalten.

Spielerfarbe Die eindeutige Farbe, die einem Spieler am Anfang des Spiel zugeordnet wird.

Spielstatus Der Status eines Spiels. Ein Spiel kann auf Spieler warten, gestartet sein und laufen oder beendet sein.

Spielzug Ein Zug eines Spielers in einem Spiel. Dieser besteht aus Pflichtaktionen wie dem Basiszug und optionalen Aktionen wie dem Spielen von Tokens.

State Teil der Redux Architektur. Die State speichert die Daten des Programms.

Store Teil der Redux Architektur. Der Store bildet die Schnittstelle zur State.

SubScene Eine abstrahierte Teilmenge der gesamten Szene, um nur bestimmte Daten zu lesen.

subscribe Sich als Teilprogramm einschreiben, um benachrichtigt zu werden, wenn ein Kriterium erfüllt ist.

Subscriber Ein Teilprogramm, welches benachrichtigt wird, wenn ein Kriterium erfüllt ist.

Substate Teil der Redux Architektur. Einzelne Unterkategorien der State, welchen separat subscribed werden kann.

Socket Die lokale Anbindung der direkten Verbindung zum Server.

SpinBox Element der Oberfläche, mit welchem eine Zahl spezifiziert werden kann.

Szene Die Benutzeroberfläche, welche im Fenster angezeigt wird.

Tab Element, welches eine Registerkarte darstellt.

TableView Element, welches eine Tabelle anzeigt.

Terrain Karte Element der Oberfläche, welches das gegebene Terrain der Runde anzeigt in Form eines Bildes und einer Beschreibung.

Terrain Typ Bebaubaren Feldtypen wie Gras, Blumen etc.

TextArea Element der Oberfläche, welches Text anzeigt.

TextField Element der Benutzeroberfläche, welches erlaubt, eine Texteingabe zu tätigen.

Textur Darstellung von Spielobjekte als Bilder (bsp.: Gras oder Settlement).

Tile Ein Feld bzw. einzelnes Hexagon der Spielkarte.

Token Ein Ortsplättchen, welche dem Spieler ermöglicht, Bonuszüge zu tätigen.

Token-Typ Die Art des Token die man erhalten kann, wenn man auf einen besonderen Ort anbaut.

Undo Etwas Rückgängig machen.

User Anwender des Programms.

WebView Darstellung des Chatfensters in HTML.

Win Conditions Drei von zehn ausgewählten Siegesbedingungen. Diese beschreiben, unter welchen Bedingungen ein Spieler Punkte bekommt.

Zeitlimit Zeitbegrenzung für eine Runde in einem Spiel.

Zoom Karte vergrößern oder verkleinern.

Zug Erste bis letzte Aktion während ein Spieler dran ist.

Zug beenden Beendet seinen Zug, wenn die Pflichtaktion der 3 Siedlungen abgeschlossen ist und überspringt die Verwendung von Tokens.

Zuglimit Maximale Anzahl an Runden in einem Spiel.