DevOps-Dokument

"Dog Royale"

Softwaretechnikpraktikum Gruppe 01

10. Dezember 2023

Product Owner: Luca Timo Eisen
E-Mail: luca-eisen@nexusvision.com

Inhaltsverzeichnis

1 Development

1.1 Architektur

Ein Architekturpattern definiert die grundlegenden Strukturen einer Softwareanwendung und legt fest, wie ihre Komponenten miteinander interagieren. Diese Patterns helfen dabei, eine klare Trennung von Verantwortlichkeiten zu schaffen, den Code wiederverwendbar zu machen und die Flexibilität zu erhöhen. Im Softwarebereich gibt es verschiedene Architekturpattern. Im Rahmen dieses Projekts wurden folgende Pattern angewendet:
Die Projektarchitektur wird mit Hilfe von Maven als Multi-Modul-Projekt umgesetzt. Ein Maven Multi-Modul-Projekt ist eine Struktur, bei der ein übergeordnetes Projekt mehrere Untermodule enthält. Jedes Untermodul ist dabei ein eigenständiges Maven-Projekt mit seinem eigenen Verzeichnis und seiner eigenen POM (Project Object Model)-Datei. Das übergeordnete Projekt koordiniert diese Untermodule und ermöglicht eine gemeinsame Verwaltung, Build-Prozesse und Abhängigkeiten.

1.1.1 Modell-View-Controller Pattern

In diesem Projekt wird das etablierte Architekturmuster MVC (Model-View-Controller) zur Programmierung angewendet. Dabei werden die Klassen in drei zentrale Typen unterteilt: Model (Datenrepräsentation), View (Benutzeroberfläche) und Controller (Steuerungslogik).
Es wird implementiert, um eine klare Trennung von Datenmodell, Benutzeroberfläche und Steuerung zu gewährleisten. Dies fördert die Wartbarkeit und Erweiterbarkeit der Anwendung.

Modell
Das Modell repräsentiert die Datenstruktur und die Spiellogik der Anwendung.

View (Sicht)
Die View ist für die Darstellung der Benutzeroberfläche verantwortlich.

Controller (Steuerung)
Der Controller handhabt Benutzerinteraktionen und aktualisiert das Modell entsprechend.

Die klare Struktur des MVC-Patterns erleichtert die kontinuierliche Integration und Bereitstellung von Dog Royale. In jedem Entwicklungszyklus werden Änderungen im Model, der View oder dem Controller automatisch durch Tests überprüft und in den Gesamtprozess integriert. Das MVC-Pattern fördert eine konsistente und strukturierte Entwicklung, was die Zusammenarbeit im Team und die Integration von Komponenten in DevOps-Umgebungen erleichtert.

1.1.2 Singleton-Pattern

Zu erwähnen ist die Implementierung des Singleton-Pattern in der Klasse ServerController. Hierbei wird sichergestellt, dass lediglich eine einzige Instanz dieses Controllers existiert. Diese Struktur gewährleistet nicht nur eine ressourceneffiziente Handhabung, sondern sichert auch eine klare und effiziente Kontrolle über den Server, da nur eine eindeutige Instanz aktiv ist.

1.2 Softwarekomponenten

Das Projekt beinhaltet die Entwicklung des Brettspiels Dog Royale als funktionierende App für Android-Smartphone und PC. Das Projekt wird in folgende 5 Komponenten aufgeteilt:

- Server
- Smartphone-Beobachter
- PC-Beobachter
- PC-Teilnehmer
- KI-Teilnehmer

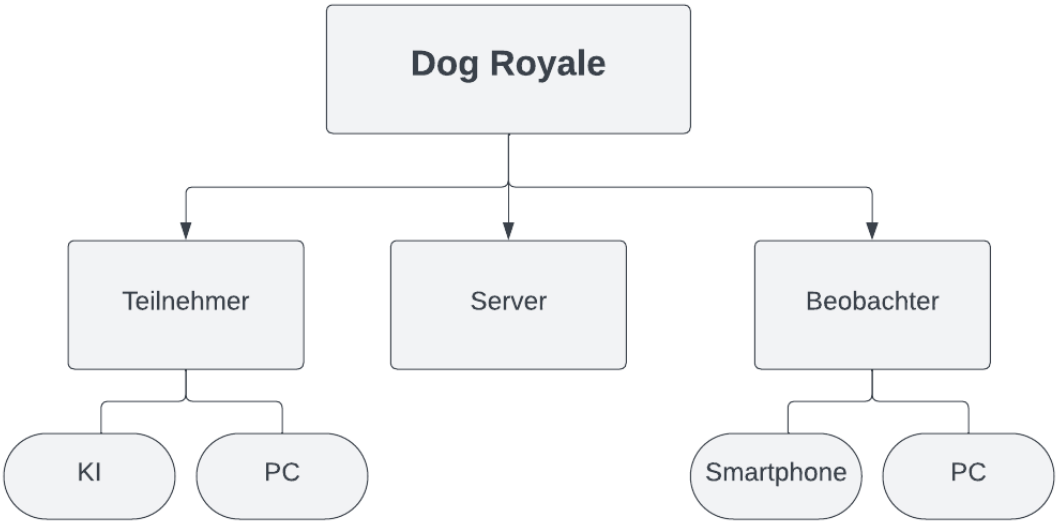


Abbildung 1: Bestandteile des Gesamtprojekts

Der Server wird vom Ausrichter betrieben und dient zur Verwaltung von Spielen und Turnieren, Konfiguration von Spieleinstellungen sowie zur Validierung von Spielzügen. Neben der Spiel-und Turnierverwaltung enthält der Server auch eine Konfigurationsdatei, in der Spielparameter sowie die Spiellogik angepasst werden können.

Der Smartphone-Beobachter ist ein Client, der für das Betriebssystem Android implementiert wurde. Wie aus dem Namen hervorgeht, kann der Client lediglich Spiele und Turniere beobachten und nicht aktiv an diesen teilnehmen. Er ist in dem Sinne also kein Spielteilnehmer.

Der PC-Beobachter-Client wird auf einer Messe erworben. Dieser soll die nötigen Anforderungen an Code-Verständlichkeit und Qualität bestmöglich erfüllen, sodass während der zweiten Entwicklungsphase eine Weiterentwicklung zu einem PC-Teilnehmer problemlos erfolgen kann.

Der PC-Teilnehmer fungiert ebenfalls als Client und ist wie der Smartphone-Teilnehmer zur aktiven Teilnahme an Spielen und Turnieren berechtigt. Dieser Client wird für alle gängigen Desktop-Betriebssysteme implementiert.

Der KI-Teilnehmer wird im weiteren Verlauf der Entwicklungsphase ebenfalls entwickelt. Dieser ist aktiver Teilnehmer an einem Spiel oder Turnier, stellt jedoch keine grafische Schnittstelle bereit. Grund dafür ist, dass er Entscheidungen ohne menschliche Eingabe anhand Algorithmen/künstlicher Intelligenz trifft.

Die Entwicklung des KI-Teilnehmers hat noch nicht begonnen. Es gibt zwei unterschiedliche Vorgehensweisen, von der eine in naher Zukunft ausgewählt wird: Die erste Möglichkeit ist das Verwenden des Monte-Carlo Algorithmus, der auf Zufallsproben basiert. Im Kontext des Projekts wird er verwendet, um die Entscheidungen des KI-Teilnehmers beim Spiel zu berechnen.
Eine Alternative stellt das Entwickeln neuronaler Netze dar. Neuronale Netzwerke sind eine Art von maschinellem Lernen, die von der Funktionsweise des menschlichen Gehirns inspiriert ist. Sie bestehen aus künstlichen Neuronen, auch als Perzeptronen oder künstliche Neuronen bezeichnet.

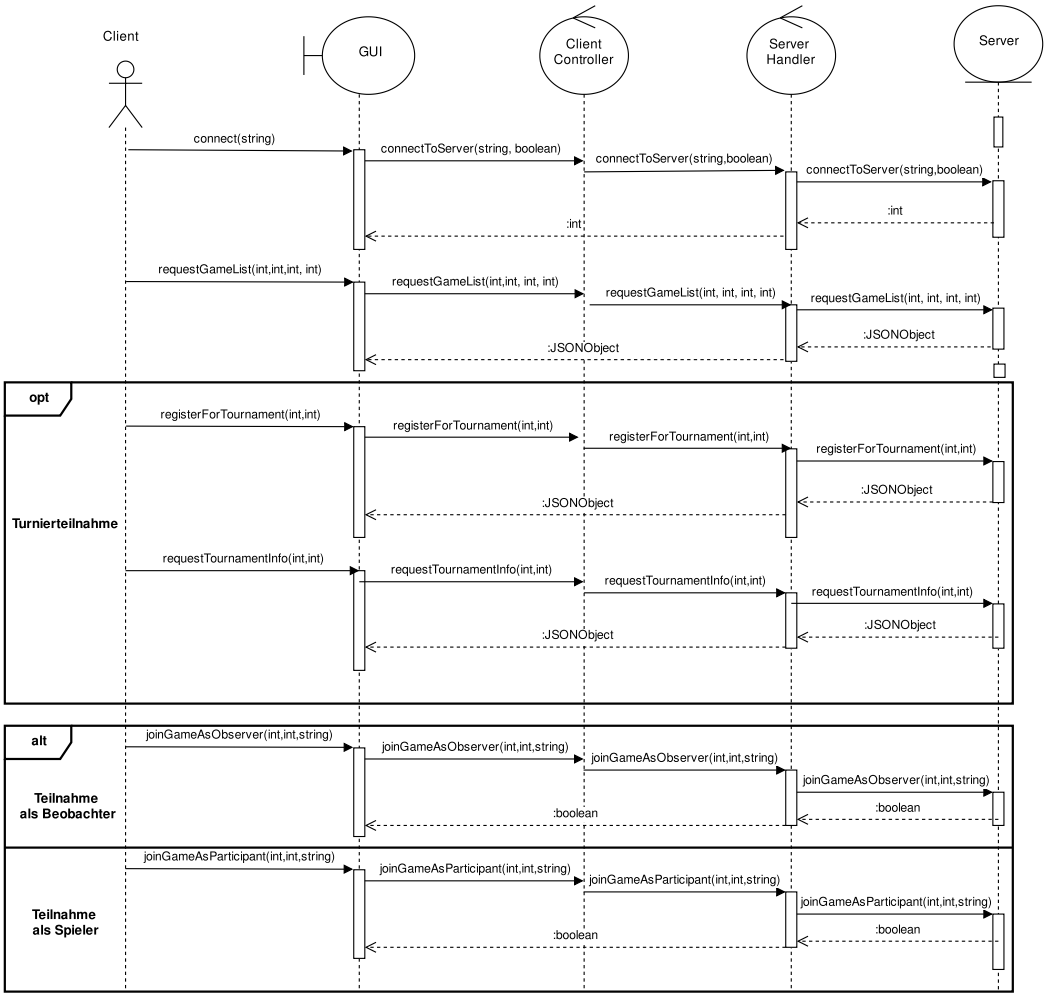


Abbildung 2: Sequenzdiagramm der Standardkommunikation für Benutzer

1.2.1 Server

Der Server entspricht dem Kern der Software. Dieser stellt für den Client Funktionalitäten bereit, mit dieser am Spielgeschehen teilnehmen kann. Nachdem Clients dem Server beigetreten sind, können sie über den Server Spielen zuschauen oder auch Spielen als Spieler beitreten. Sobald sich ein neuer Client beim Server anmeldet, erstellt der Server einen neuen Serversocket für diesen Client, welcher auf seiner Seite einen Clientsocket zur Verfügung stellt. Dadurch entsteht eine durchgehende Verbindung zwischen Client und Server, welche bei fehlendem Bedarf geschlossen werden kann. Zusätzlich erhält jeder Client einen ClientHandler, der die Nachrichten des Clients zum Server verarbeitet.

MVC Pattern für den Server:

- **Modell**: Hier werden Daten wie Spielerinformationen und Spielregeln verwaltet. Es repräsentiert den Zustand und die Logik des Spielservers. Die Spiellogik umfasst die Spielregeln des Spiels Dog, die aus der Product Vision implementiert wurden. Die Nachrichten enthalten die Messages vom Client und vom Server. Dazu zählt jede Art der Anfrage, die ein Client an den Server stellen kann und die Rückgabe auf diese Anfrage von dem Server. Die Messages werden auf Nachrichten in einem Spiel (z.B das Legen einer Karte) und Nachrichten im Menü (z.B. Verbinden mit einem Spiel oder Turnierinformationen) unterteilt.

- **View**: Auf dem Server ist die View nicht direkt sichtbar, könnte aber als eine Art Protokollierung oder Statusbericht betrachtet werden, die auf Anfragen und Ereignisse reagiert. Der Ausrichter hat die Befugnis, Spielparameter zu konfigurieren oder auf vordefinierte Konfigurationen zurückzugreifen. Zusätzlich kann er Turniere verwalten. Teilnehmer, die sich für ein Spiel angemeldet haben, können dem Spiel hinzugefügt werden, und der Ausrichter kann bei Fehlverhalten Strafen verhängen. Aktuell verfügt der Ausrichter noch nicht über alle geplanten Funktionen und befindet sich in der Entwicklungsphase.

- **Controller und Handler** : Der Controller verarbeitet Benutzereingaben und eingehende Anfragen. Auf Serverebene steuert der Controller, wie die Spiellogik auf Aktionen der Spieler reagiert und wie der Zustand des Modells aktualisiert wird.
  Auf Clientebene sorgt der Controller gemeinsam mit einzelnen Klassen, die Anfragen repräsentieren für die korrekte Bearbeitung der eingehenden Anfragen.
  Die Controller übernehmen die Verwaltung spezifischer Bereiche, so leitet der ServerController den Server im Allgemeinen, während der GameController eine Spiellobby betreut. Dabei bieten sie Methoden für Operationen auf anderen Objekten an.

1.2.2 Smartphone-Beobachter

MVC Pattern für den Smartphone-Beobachter-Client:

- **GUI (View):** Die GUI repräsentiert die Benutzeroberfläche des Smartphone-Beobachter-Clients und zeigt Informationen an.

- **Client Controller (Controller):** Dieser reagiert auf Benutzereingaben, leitet sie an den Server weiter und aktualisiert die GUI entsprechend.

- **Server Handler (Model):** Der Server Handler repräsentiert den Zustand und die Funktionalität des Smartphone-Beobachter-Clients. Er empfängt Informationen vom Server und aktualisiert das Modell.

1.2.3 PC-Beobachter

Die Wahl des PC-Beobachters unterliegt einigen internen Kriterien. Zum einen die Qualität der Software, zum anderen die Integration zum eigenen Server. Um dem Kunden das bestmögliche Produkt anzubieten, wird bei der Auswahl des PC-Beobachter stark auf die Qualität der Softwarestruktur geachtet. Ebenso wird der ausgewählte PC-Beobachter auf Integrität mit dem eigenen Server und Ausrichter überprüft um eine effiziente Weiterentwicklung des PC-Teilnehmers zu gewährleisten.

1.2.4 PC-Teilnehmer

Die Entwicklung des PC-Teilnehmers wird nach dem Einkauf des PC-Beobachters aufbauend beginnen. Dieser soll wie der Smartphone-Beobachter laufende und startende Spiele sehen und die Match Historie einsehen können. Darüber hinaus, kann der Teilnehmer den laufenden und startenden Spielen als Spieler beitreten.

1.2.5 KI-Teilnehmer

Um ein Spiel mit nicht-menschlichen Akteuren zu ermöglichen, wird fortlaufend an der Entwicklung eines KI-Teilnehmers gearbeitet. Dieser kommuniziert auf dieselbe Art und Weise wie ein PC-Teilnehmer mit dem Server, macht seinen nächsten Zug jedoch nicht von einer Benutzereingabe abhängig, sondern von internen Kalkulationen. Monte-Carlo ist eine Methode, die auf Zufallsproben basiert.Im Kontext des Projekts wird es verwendet, um die Entscheidungen des KI-Teilnehmers beim Spiel zu berechnen, ohne den Einsatz von Machine Learning-Algorithmen.

1.3 Eingesetzte Development-Tools

1.3.1 Entwicklungsumgebung

Als IDE wurde IntelliJ Ultimate gewählt. Diese IDE unterstützt Maven und Git nativ und bietet darüber hinaus mit IntelliSense eine hervorragende Autocompletion, welche den Workflow erheblich vereinfacht.

1.3.2 Logging-Framework

Als Logging-Framework für die Entwicklungsumgebung wird Log4J2 genutzt, da es bis dato das am weitesten entwickelte Framework ist mit den meisten zu bietenden Funktionalitäten. Zusätzlich bietet das Framework eine Lizenz zum genutzten Build Automation Tool.

1.3.3 Build Automation Tool

Als Build Automation werden sowohl Maven als auch Gradle verwendet. , welches auch mit dem Logging-Framework kompatibel ist. Mit Maven ist es möglich die Software plattformübergreifend starten und bedienen zu können, wodurch eine hohe Portabilität möglich ist. Dies hilft besonders in einer weniger benutzerfreundlichen Umgebung durch das Starten der App aus externen Quellen.

1.3.4 Test-Framework

Zum Durchführen von Unit-Tests um einzelne Komponenten der Software zu testen wird das Test-framework JUnit5 verwendet. Inklusive der Programmbibliothek Mockito, wodurch das Erstellen von gemockten(aus dem Original simulierten) Objekten ermöglicht wird. aus

1.3.5 Versionsverwaltung

Zur Versionsverwaltung wurde die Webplattfrom GitLab und folglich das verteilte Versionskontrollsystem Git eingesetzt, da GitLab eine umfassende Lösung für das Repository-Management bietet.

1.4 Weiterentwicklung

Im Laufe der Entwicklung wurden hohe Anforderungen an die Lesbarkeit und Wartbarkeit des Codes gestellt. Zusätzlich werden die genannten Design Patterns und ein streng modulares Design verwendet. Dies ermöglicht ein schnelles Verständnis der Arbeitsweise des Projekts sowie eine rasche Weiterentwicklung. Die interne Spiellogik kann schnell weiterentwickelt werden, da sie aus Klassen besteht, die stets ein reales Pendant haben, sodass Sinn und Funktionalität einer Klasse schnell einzusehen sind.

Die Server-Client Kommunikation funktioniert strikt nach dem durch das Interface-Protokoll spezifizierten Format. Im Bezug auf den verkauften Smartphone-Beobachter Client wird empfohlen, die Weiterentwicklung zum Smartphone-Teilnehmer Client durch Vererbung zu implementieren. Wichtige, bereits vorhandene Funktionalität wird so optimal wiederverwertet.
Zusätzlich müssen weitere Integrationstests durchgeführt werden, um eine bessere Kompatibilität zwischen Komponenten zu gewährleisten.

2 Operation

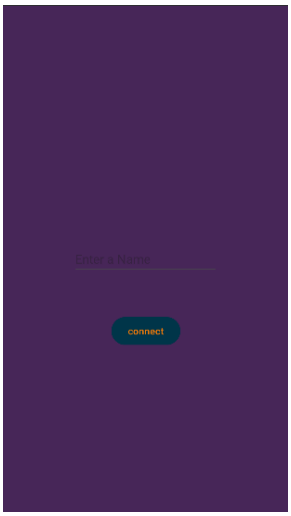
2.1 Benutzerhandbuch

Login
Nach dem Start der App gelangt der User zum Startbildschirm(Abbildung 1). In diesem wird der User nach einem Benutzernamen gefragt. Dieser Benutzername wird nur für die aktuelle Sitzung gespeichert. Es kann jeder beliebige Name gewählt werden und es können mehrere User denselben Namen wählen.

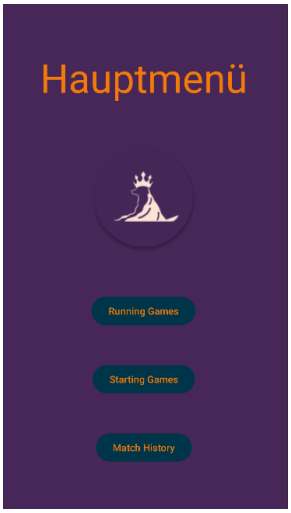
Hauptmenü
Nachdem der Benutzername gewählt wurde, gelangt der Nutzer in das Hauptmenü. Im Hauptmenü sind 3 Buttons zu sehen, durch die der Nutzer Spielen beitreten kann oder die Ergebnisse der zuletzt gespielten Spiele sehen kann.

- Running Games
- Starting Games
- Match History

In jedem dieser Menüs existiert ein Button oben links auf dem Bildschirm, der es ermöglicht zum Hauptmenü zurückzukehren.



(a) Startmenü mit Formular für den Spielernamen



(b) Hauptmenü mit drei Buttons

Abbildung 3: Die beiden ersten Ansichten der Benutzeroberfläche

Running Games

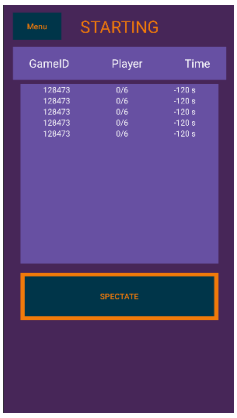
Wird auf den Button Running Games gedrückt, gelangt der Nutzer in den Spectate Games Bildschirm. Zu sehen in Abbildung 4b In diesem Menü befindet sich eine Liste mit den momentan laufenden Spielen. In einem Spiel werden die SpielID, die Spieler, die momentane Spielzeit,

Starting Games

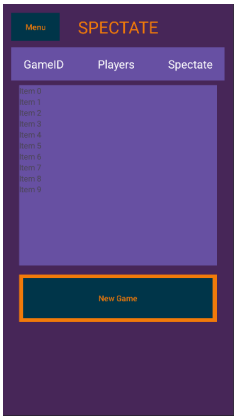
Durch den Button Starting Games gelangt der Nutzer in das Starting Games Menü. Zu sehen in Abbildung 4a. In diesem Menü sind alle Spiele aufgelistet, die ein Ausrichter erstellt hat, aber noch nicht gestartet sind. Der Nutzer kann auf ein Spiel drücken um diesem beizutreten.

Match History

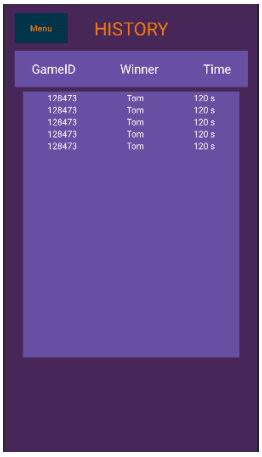
Ist es gewollt die Ergebnisse der zuletzt gespielten Spiele zu betrachten, kann der Nutzer durch den Button Match History in das Match History Menü (Abbildung4c) gelangen. Im Match History Menü sind die Ergebnisse der zuletzt gespielten Spiele aufgelistet. Die Ergebnisse beinhalten die SpielID, den Gewinner des Spiels und die gespielte Zeit.



(a) Liste an startenden Spielen



(b) Liste an laufenden Spielen



(c) Liste von zuletzt gespielten Spielen

Abbildung 4: Zu sehen sind die drei Menüs: a) Starting Games b) Spectate Games c) Match History

2.2 Entwicklerhandbuch

2.2.1 Server

Im folgenden Abschnitt werden die Hintergrundprozesse erläutert, die beim Starten eines Spiels Dog Royale laufen. Dazu zählt das Konfigurueren als Ausrichter, Spielen als Teilnehmer und zuschauen als Beobachter.

Installation
Für die Serverinstallation ist es erforderlich, dass der Rechner über Java 11 in der aktuellen Version verfügt und die entsprechende Datei lokal vorhanden ist. Die Auswahl des Betriebssystems ist dabei nicht relevant, solange Java installiert ist. In diesem Fall funktioniert der Serverstart problemlos auf gängigen Systemen wie Windows, Linux und macOS.

Konfiguration
Die Konfiguration des Servers obliegt dem Ausrichter. Dieser interagiert mit einer grafischen Oberfläche, um Lobbies zu eröffnen und zu schließen.
Darüber hinaus nutzt der Ausrichter die Oberfläche, um Spielparameter zu setzen und anzupassen, wie in Abbildung 5 zu sehen ist.

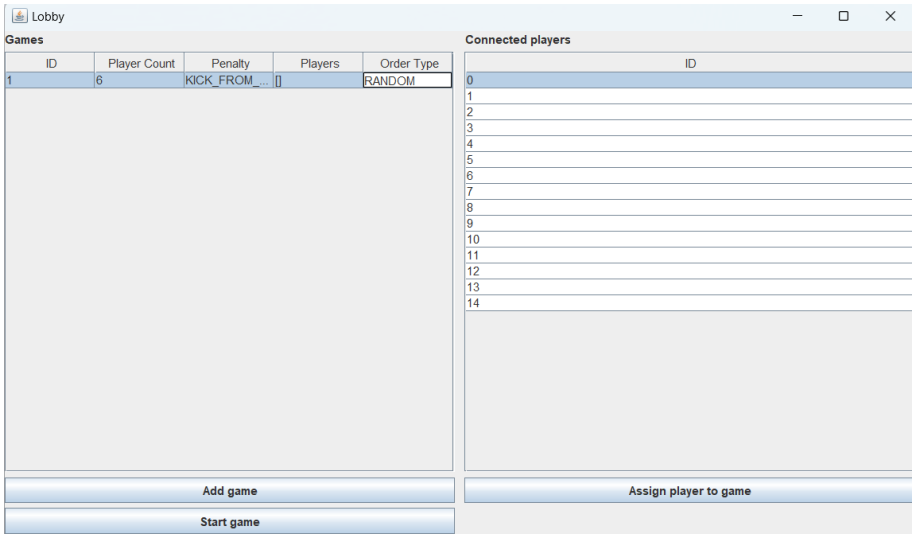


Abbildung 5: Oberfläche zur Übersicht der Lobbies

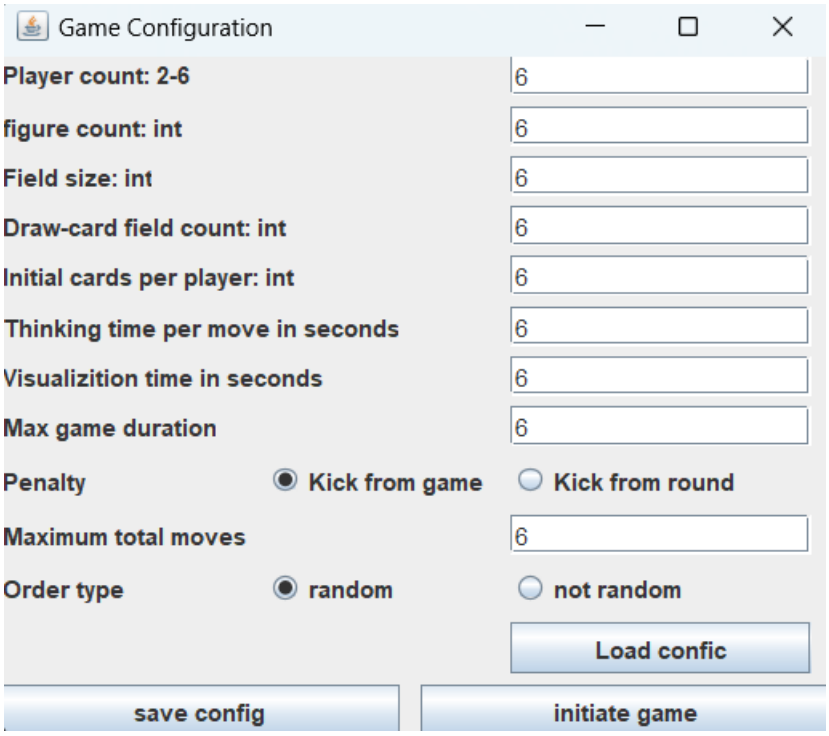


Abbildung 6: Konfigurationsoberfläche des Ausrichters

Ausführung
Die Inbetriebnahme des Servers erfolgt mittels der Kommandozeile oder des Terminals, abhängig vom Betriebssystem. Der Startbefehl (java -jar /path/to/jarfile) wird verwendet. Die Konfiguration wird durch die Datei ‘config.properties‘ vorgenommen, in der der Port über entsprechende Einträge festgelegt ist. Alternativ lässt sich der Server ebenso durch den Befehl (java -jar /path/to/jarfile port=8082) initiieren.

2.2.2 Smartphone-Beobachter

Die folgende Sektion enthält eine detaillierte Beschreibung für die Bedienung des Beobachter-Clients als Smartphone-Beobachter.

Installation
Um das Spiel auf dem Smartphone installieren zu können, wird ein Android-Handy, mit einem Betriebssystem von Android 8.0 oder höher vorraussgesetzt. Die Software wurde nicht auf öffentlichen Applikation Stores veröffentlicht, sondern wurde nur für den privaten Gebrauch unseres Kunden entwickelt. Dementsprechend muss die Software aus externen Geräten auf das Smartphone geladen werden.

Konfiguration

Mit der entsprechenden .jar-Datei wird der Server gestartet. Nach der Installation und dem Start des Servers wird der gewünschten Betriebsmodus und den Port angegeben. Der Server ist standardmässig auf Port 8082 eingestellt, um Kompatibilitätsprobleme mit den Betriebssystemen zu vermeiden, und die IP-Adresse entspricht der des Startgeräts. Die Applikation öffnet entsprechend dem Modus die passende Schnittstelle für Spiele (oder Turniere). Die Serverkonfiguration erfolgt über eine JSON-Datei. Bei Aktivierung öffnet sich automatisch die erste grafische Benutzeroberfläche (GUI).

Ausführung
Sobald die Spiele vom Server erstellt wurden, besteht für den Client die Option, eine Verbindung

zum Server herzustellen und einem dieser Spiele beizutreten.

2.3 Betrieb des KI-Teilnehmers

Nutzung
Um den KI-Teilnehmer möglichst einfach zu nutzen, genügt es, das Programm mit dem Teilnehmer zu starten. Die Steuerung erfolgt bequem von der Kommandozeile aus, indem der Nutzer angibt, welchem Spiel der Teilnehmer beitreten soll und welchen Namen er verwenden möchte. Gegenwärtig nutzt der KI-Teilnehmer den Monte-Carlo-Search-Algorithmus zur Bestimmung seiner nächsten Züge, jedoch fehlt ihm noch eine erfolgreiche Strategie.
Die Implementierung des KI-Teilnehmers ist nicht fortgeschritten. Geplant ist diesen neben dem PC-Teilnehmer zu implementieren, da dieser ebenfalls die Funktionen eines Teilnehmers nutzt.

2.4 Entwicklungsprozess

Um einen agilen Entwicklungsprozess zu gewährleisten, wird die Methode Scrum für das Projektmanagement genutzt.

2.5 Entwicklungstools

2.5.1 Kommunikation

Discord
Für die Kommunikation innerhalb des Entwicklerteams wird der Onlinedienst Discord verwendet. Dadurch können die Funktionen Sprach- und Video-Konferenzen des Dienstes genutzt werden um die Kommunikation zwischen den Entwicklern außerhalb der wöchentlichen Meetings zu gewährleisten. Für eine bessere Strukturierung innerhalb des Dienstes wurde ein eigener Server erstellt. Im Server wurde für jedes Projekt ein eigener Channel erstellt um eine zusätzliche visuelle Übersicht neben einer Versionsverwaltung zu besitzen. In dieser können über interne Fragen und Themen disskutiert werden.

2.5.2 Dokumentation

LaTeX
Um die Professionalität unserer Dokumente zu gewährleisten, wurde die Markup-Sprache LaTeX mithilfe des kollaborativen OpenSource-Programms Overleaf verwendet. Die erstellten Projekte durch diese Software wurden zusätzlich in ein Git-Repository hochgeladen.

DrawIO
Die UML-Diagramme wurden mit der Software DrawIO erstellt.

GitMind
Für die Erstellung der Sequenzdiagramme wurde das OpenSource-Programm GitMind verwendet. Die beiden Programme zur Erstellung der Diagramme sind Öffentlich zugänglich und kostenfrei. So können in der Entwicklung des Projekts Kosten gespart werden.