

DevOps Dokument

Camel Up

Softwaretechnikpraktikum Gruppe 06

08. Dezember 2024



Inhaltsverzeichnis

1 Development	1
1.1 Software Architektur	1
1.1.1 Backend	1
1.1.2 Frontend	1
1.2 Modularisierung des Projektes	2
1.2.1 Interface-Definition	2
1.2.2 Core-Modul	2
1.2.3 Admin-Modul	2
1.2.4 Server-Modul	2
1.2.5 Engine-Modul	3
1.2.6 Observer-Modul	4
1.3 Spieler-Teilnehmer	4
1.4 Grafische Benutzeroberfläche	4
1.5 Möglichkeiten der Erweiterung	5
2 Operations	6
2.1 CI/CD-Pipeline	6
2.2 Entwicklerhandbuch	6
2.3 Konfiguration des Spiels	7
2.4 Spielerhandbuch	8
2.5 Eingesetzte Technologien	11
3 Änderungsverlauf	13
3.1 22.12.2024	13

1 Development

1.1 Software Architektur

1.1.1 Backend

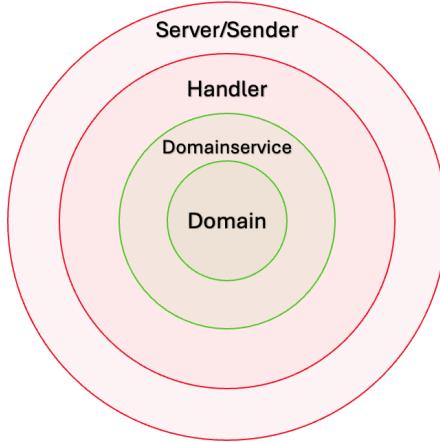


Abbildung 1: Onion-Architektur

Die oben dargestellte Architektur unserer Software für das Spiel *Camel Up* basiert auf dem Prinzip der Onion-Architektur. Dieses Architekturprinzip wird speziell auf der Serverseite umgesetzt und ermöglicht eine klare Trennung der Verantwortlichkeiten und Abhängigkeiten. Die Onion-Architektur besteht aus mehreren konzentrischen Schichten, wobei die Abhängigkeiten stets von den äußeren Schichten zu den Inneren verlaufen, also jede Schicht nur auf die inneren Schichten zugreifen kann.

Der innere Kern der Architektur ist die Domain-Schicht, welche die zentralen Geschäftsregeln und Datenmodelle enthält. Darauf folgt die Domain-Service-Schicht, welche als zentraler Zugriffspunkt für konsistentes Lesen und Modifizieren der Domänenobjekte verantwortlich ist. Diese beiden Schichten bilden zusammen die Domäne, die unabhängig von äußeren Komponenten arbeitet.

Die äußeren Schichten bilden die Application-Schicht, welche die Verarbeitung von Anfragen und die Interaktion mit externen Systemen übernimmt. Dazu gehören die Handler, die eingehende Anfragen bearbeiten und an die zugehörigen Domainservices delegieren, sowie die Server-/Sender-Komponenten, welche die Kommunikation mit den Clients sicherstellen. Diese klare Trennung ermöglicht es, Änderungen an der Infrastruktur vorzunehmen, ohne die Geschäftslogik zu beeinflussen.

1.1.2 Frontend

Im Gegensatz zur Onion-Architektur, die auf der Serverseite verwendet wird, basiert die Architektur der Benutzeroberfläche (UI) auf dem Model-View-Controller (MVC)-Pattern. Dieses Pattern wurde verwendet, um eine klare Trennung zwischen Daten (Model), Darstellung (View) und Benutzerinteraktion (Controller) zu schaffen. Der Vorteil dieser Struktur liegt in der Modularität, die die Wartung und Erweiterung der Benutzeroberfläche erleichtert. Darüber hinaus ermöglicht das MVC-Pattern eine flexible Anpassung der UI, ohne die zugrunde liegende Logik oder Datenstruktur verändern zu müssen.

1.2 Modularisierung des Projektes

Neben der logischen Struktur der Onion-Architektur beschreibt das folgende Diagramm (Abbildung 2) die Aufteilung der Software in Module und deren Beziehungen. Durch die Verwendung von Maven-Modulen lassen sich Abhängigkeiten zentral verwalten und Code in verschiedenen Teilen des Projekts wiederverwenden.

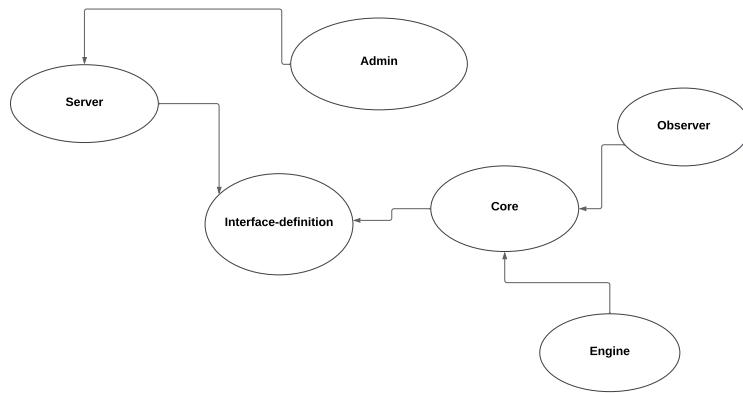


Abbildung 2: Module

1.2.1 Interface-Definition

Die Interface-Definition definiert die Datenstrukturen für die Kommunikation zwischen den Modulen (Java-Datentransfer-Objekte). Dieses Modul ist essenziell für die Konsistenz und Standardisierung der Daten, die zwischen Server und Clients ausgetauscht werden. Deswegen wird es sowohl vom Server als auch von den Clients benutzt.

1.2.2 Core-Modul

Das Core-Modul gehört zu den Client-Modulen und wird zwischen dem Observer und der Engine geteilt. Es beinhaltet Code, der für alle Clients relevant ist, wie beispielsweise eine API, um mit dem Server zu kommunizieren. Um eine ordnungsgemäße Kommunikation zu gewährleisten, greift dieses auf die Datenstrukturen des Interface-Definition-Moduls zu.

1.2.3 Admin-Modul

Das Admin-Modul bietet eine Benutzeroberfläche für die Verwaltung und Konfiguration des Servers. Es ermöglicht das Starten des Servers sowie die Administration und Überwachung der Spiele.

1.2.4 Server-Modul

Der Server kann, wie in der Product Vision angefordert, vom Ausrichter gestartet, konfiguriert und bedient werden. Die gesamte Implementierung des Servers ist in Java geschrieben und der Server verwendet Maven für die Verwaltung von Abhängigkeiten und den Build-Prozess, was eine einfache Integration und Erweiterung ermöglicht. Bei der Entwicklung des Servers wurde auf einen einwandfreien Ablauf des Spiels geachtet, welcher durch folgende technische Implementierungen sichergestellt wird:

- **Paketverarbeitung:** Pakete werden über das Transmission Control Protocol (TCP) empfangen. Um den Zugriff mehrerer Nutzer zu ermöglichen, erlaubt der Server verschiedene Verbindungen gleichzeitig. Dieser ist hierbei auf einem definierten Port verfügbar, um Verbindungen von Clients anzunehmen. Eingehende Pakete werden an spezialisierte Packet-Handler übergeben, die je nach Paket-Typ spezifische Aktionen auslösen, z. B. Spielstatus-Meldungen oder Anmeldungen von Spielern. Änderungen werden dann an den betroffenen Client und/oder alle anderen Clients zurückgemeldet. Jeder Client des Servers läuft auf einem eigenen Thread, wodurch ein paralleler Ablauf von mehreren Spielen gewährleistet wird.
- **Spiellogik-Synchronisation:** Die Spiellogik-Synchronisation gewährleistet, dass alle vom Server empfangenen Spielzüge und Interaktionen korrekt verarbeitet und konsistent an alle verbundenen Clients weitergegeben werden. Dadurch wird sichergestellt, dass alle Spieler stets den aktuellen Spielstatus sehen und gleichzeitig agieren können, ohne Inkonsistenzen oder Verzögerungen in der Darstellung zu erleben.
- **Thread-Sicherheit und Konsistenz:** Es werden Locking-Mechanismen genutzt, um Datenkonsistenz bei parallelen Zugriffen sicherzustellen. Dies ist besonders wichtig, wenn mehrere Clients gleichzeitig auf gemeinsame Ressourcen (z. B. Spielstatus) zugreifen. Hierdurch werden Konflikte und inkonsistente Zustände vermieden.
- **Server-Resilienz:** Der Server ist so konzipiert, dass er selbst bei Fehlern auf der Client-Seite stabil bleibt, diese effektiv handhabt und seinen Betrieb nahtlos fortsetzt.

1.2.5 Engine-Modul

Das Engine-Modul, oder auch Engine-Teilnehmer, wurde entwickelt, um die Rolle eines menschlichen Spielers in einem Wettbewerb zu simulieren. Es basiert auf einem Algorithmus, der strategische Züge berechnet und dabei die Spielregeln einhält. Eine grafische Benutzeroberfläche ist nicht vorhanden, da der Fokus auf der Effizienz und der Validität der Berechnungen liegt. Die wichtigsten Merkmale und Funktionen sind wie folgt:

- **Algorithmische Zugberechnung:** Der Engine-Teilnehmer verwendet heuristische Ansätze und Bewertungsmodelle, um aus einer gegebenen Spielsituation den optimalen Zug zu ermitteln. Dies geschieht unter Berücksichtigung der aktuellen Spiellage und der gültigen Regeln.
- **Simulation von Spielverläufen:** Der Engine-Teilnehmer ermöglicht die Simulation vollständiger Spiele. Dies erfolgt durch die Integration in eine Server-Umgebung, die den Austausch von Spielzügen zwischen Teilnehmern steuert.
- **Validierung und Testverfahren:** Aktuell befindet sich der Teilnehmer in der Entwicklungs- und Testphase. Hierbei werden folgende Methoden angewendet:
 - Manuelle Konstruktion von Spielszenarien zur Validierung der Zugberechnung.
 - Einsatz von Unit-Tests mithilfe von JUnit zur Sicherstellung der korrekten Implementierung der Spielregeln.
 - Simulation von Spielen und Nutzen des Spieler-Beobachters, um das Verhalten des Engine-Teilnehmers zu analysieren.
- **Fehlermodellierung und Ausnahmebehandlung:** Randfälle werden durch definierte Aktionen behandelt. Dadurch wird ein stabiler Spielverlauf gewährleistet.

1.2.6 Observer-Modul

Das Observer Modul ist eine Komponente des Systems, die es einem Spieler ermöglicht, im Spielstart-Menü suchende, laufende und beendete Spiele auszuwählen und diese Spiele in Echtzeit zu beobachten. Hier wurde, wie im Kapitel 1.1.2 beschrieben, mit dem MVC-Pattern gearbeitet und die Benutzeroberfläche mit JavaFX gestaltet.

1.3 Spieler-Teilnehmer

Der Spieler-Beobachter wird auf der Messe am 11.12.2024 eingekauft, und auf Basis dieses Beobachters wird ein Spieler-Teilnehmer entwickelt, der aktiv an dem Spiel teilnehmen kann.

1.4 Grafische Benutzeroberfläche

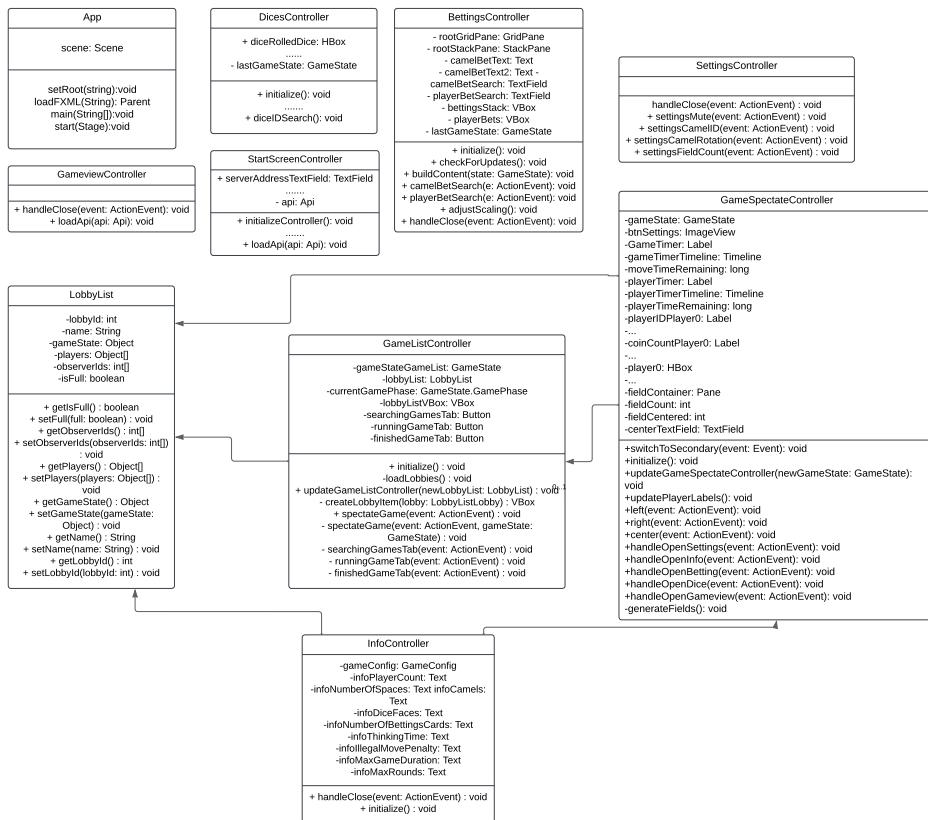


Abbildung 3: GUI

- **App:** Die Klasse App ist der Einstiegspunkt der Anwendung und verantwortlich für das Starten der JavaFX-Benutzeroberfläche. Sie initialisiert das Grundgerüst der Anwendung und lädt die erste Szene (Startbildschirm). Zudem steuert sie den Wechsel zwischen verschiedenen Szenen (Views) im Spiel.
- **GameListController:** Die Klasse GameListController verwaltet die Übersicht über die verfügbaren Lobbys. Sie zeigt Lobbys basierend auf ihrem Status (z. B. laufend, abgeschlossen) an und ermöglicht Benutzern, Lobbys beizutreten oder Details zu den Lobbys anzuzeigen. Dabei verwendet sie JavaFX-Komponenten, um die Liste interaktiv darzustellen.

- **GameSpectateController:** GameSpectateController steuert die Beobachtungsansicht eines laufenden Spiels. Es zeigt Spielstände, Timer und Spielfeldinformationen in Echtzeit an. Die Klasse ist zentral für Beobachter, da sie den aktuellen Zustand des Spiels darstellt und regelmäßige Updates ermöglicht.
- **InfoController:** Die Klasse InfoController zeigt Konfigurationsdetails eines Spiels an, wie z. B. die Anzahl der Spieler, Kamele und Felder. Sie liest diese Daten aus dem aktuellen Spielstatus und verwendet JavaFX-Elemente, um die Informationen übersichtlich darzustellen.
- **LobbyList:** LobbyList ist eine Datenklasse, die Informationen zu einzelnen Lobbys speichert, wie deren ID, Namen, Spieler, Beobachter und den Spielstatus. Sie wird verwendet, um Daten zwischen verschiedenen Controllern zu übertragen und die Lobbys zu verwalten.
- **StartScreenController:** Der StartScreenController verwaltet den Startbildschirm der Anwendung. Er bietet eine einfache Benutzeroberfläche mit einem Button, um das Spiel zu starten, und leitet den Benutzer zur Lobby-Übersicht (GameList) weiter.

1.5 Möglichkeiten der Erweiterung

Die Architektur des Spiels und seine einzelnen Komponenten sind darauf ausgelegt, Erweiterungen in der Zukunft effizient und unkompliziert umzusetzen:

- Einzelspieler-Modus: Erweiterung der KI-Spieler und das Schaffen eines benötigten Spielumfeldes für den Einzelspieler-Modus.
- Cross-Plattform-Support: Erweiterung des Clients für andere Endgeräte, bspw. für Smartphones (iOS und Android).
- Erweiterte Statistiken: Hinzufügen einer Analyse der Spielergebnisse und Leaderboards (Bestenlisten).
- GUI-Erweiterung: Erweiterung der grafischen Benutzeroberfläche und Verfeinerung der grafischen Elemente sowie eine mögliche Skalierung für höhere Auflösungen.
- Engine-Teilnehmer Schwierigkeit: Die Möglichkeit für Spieler-Teilnehmer und Administratoren, das Schwierigkeitslevel des Engine-Teilnehmers anzupassen, um ein anspruchsvolleres Spiel für die Spieler-Teilnehmer zu schaffen.

Hinweis zur Übersicht

Alle Darstellungen bieten nur einen groben Überblick und konzentrieren sich auf die wichtigsten Module und Schichten. Die tatsächliche Software enthält zahlreiche weitere Submodule und Details, die im Rahmen dieser Beschreibung nicht berücksichtigt werden können. Ziel ist es, die grundlegende Struktur und die zentralen Komponenten der Software hervorzuheben.

2 Operations

2.1 CI/CD-Pipeline

In GitLab wurde eine Pipeline eingerichtet, welche automatisiert Prozesse nach dem Pushen eines Commits ausführt:

1. Wurde ein LaTeX Dokument bearbeitet, wird dieses kompiliert und als PDF als herunterladbares Artefakt der Pipeline bereitgestellt.
2. Wurde Java-Code verändert, wird das Projekt kompiliert und die Tests durchlaufen. Die Testergebnisse sind in GitLab unter der jeweiligen Pipeline-ID unter dem Reiter Tests einsehbar.
3. Sind alle Tests erfolgreich durchgelaufen, werden Server, Engine und Observer als ausführbare .jar-Dateien durch verschiedene Maven-Plugins gebaut und als Artefakt des Pipelineschrittes *build-and-test* unter dem Ordner *camelup-artifacts/jars* hochgeladen.
4. Der Test-Coverage-Report befindet sich in den Artefakten unter *camelup-artifacts/test-report/index.html* und kann im Browser geöffnet werden.
5. Durch das Taggen eines Commits wird automatisch ein neuer Release des Projekts erstellt und hochgeladen: Git Repository Releases

2.2 Entwicklerhandbuch

Voraussetzungen

Die Software-Komponenten sind unter Mac OS X, Ubuntu und Windows lauffähig. Dafür muss die aktuellste Version von Java SE 21 auf dem Gerät installiert sein.

Installation

Laden Sie das neueste Release von Git Repository Releases herunter und entpacken Sie die Datei lokal auf Ihrem Gerät.

Starten des Servers

Der Server kann über die Kommandozeile gestartet werden. Führen Sie hierzu den folgenden Befehl aus:

```
java -jar server-1.0.0.jar <port>
```

Alternativ kann der Server auch über die Benutzeroberfläche des Admin-Clients gestartet werden. Sobald der Server gestartet ist, können sich auch die Clients zu ihm verbinden.

Starten weiterer Komponenten

- Beobachter: Beobachter können über die Kommandozeile mit folgendem Befehl gestartet werden:

```
java -jar observer-1.0.0.jar
```

- Engine: Um die Engine zu starten, die automatisierte Spielzüge berechnet, verwenden Sie:

```
java -jar engine-1.0.0.jar <server-ip> <server-port>
```

- Admin-Client: Der Admin-Client kann für die Konfiguration und Verwaltung des Servers gestartet werden:

```
java -jar admin-1.0.0.jar
```

Um den Server, einen Beobachter und sechs Engines zu starten, können die beigefügten Skripte aus dem Terminal gestartet werden:

- Windows-Nutzer: *start-all.bat*
- Mac/Linux-Nutzer: *start-all.sh*

Der Server ist in den Skripten so konfiguriert, dass automatisch Spiele mit maximal sechs Spielern erstellt werden. Sobald die maximale Spieleranzahl eines Spiels erreicht wurde, treten die sechs Engines gegeneinander an. Ist das Spiel vorbei, wiederholt sich dieser Prozess, bis die Programme geschlossen werden.

2.3 Konfiguration des Spiels



Abbildung 4: Lobby

Wie bereits beschrieben, kann der Server alternativ über den Admin-Client und die benutzerfreundliche GUI gestartet werden. Dort können Spiele erstellt, konfiguriert und verwaltet werden. Nach dem Start des Clients wird eine Übersicht der laufenden und anstehenden Spiele sowie der aktuellen Teilnehmer angezeigt (siehe Abbildung 4). Über Buttons können Spiele erstellt, gestartet, pausiert oder abgebrochen werden.

Die Konfiguration eines Spiels erfolgt direkt über den integrierten Spiel-Konfigurator im Admin-Client (siehe Abbildung 5). Dabei können folgende Parameter angepasst werden:

- Spieler: Anzahl der Spieler.
- Spielfeld: Anzahl der Felder.
- Kamele: Anzahl der vorwärts- und rückwärtslaufenden Kamele sowie die Anzahl der Wertscheine pro Kamel.

Abbildung 5: Konfiguration



- Spielmechanik: Maximale Augenzahl der Würfel, Bedenkzeit pro Spielzug, maximale Spieldauer.
- Spielregeln: Konsequenzen für regelwidrige Spielzüge.

Nach Abschluss der Konfiguration können die Einstellungen gespeichert oder direkt ein neues Spiel gestartet werden. Bereits gespeicherte Konfigurationen können jederzeit wieder importiert werden.

2.4 Spielerhandbuch



Abbildung 6: Startbildschirm

Beim Starten des Spiels wird der **Startbildschirm** geöffnet. Nach erfolgreicher Eingabe der korrekten Serveradresse gelangt man durch das Klicken auf den Button „**Verbinden**“ zum Hauptmenü.

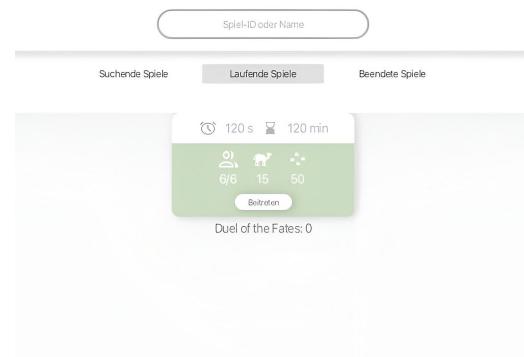


Abbildung 7: Hauptmenü

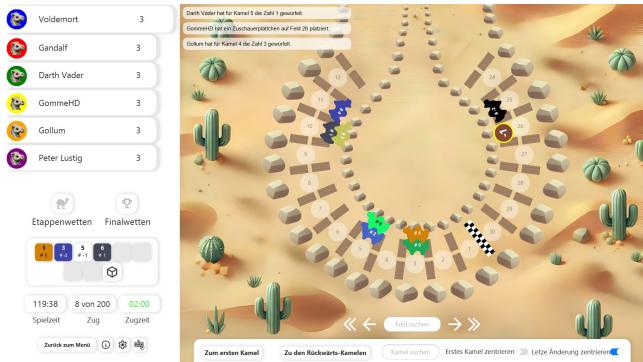
Im **Hauptmenü** ist es möglich, zwischen drei Kategorien zu wählen, die oben im Menü sichtbar sind: „**Suchende Spiele**“, „**Laufende Spiele**“ und „**Beendete Spiele**“. Durch Anklicken einer der Kategorien werden die entsprechenden Spiele angezeigt. Unterhalb der Kategorien werden die verfügbaren Spiele in einzelnen Kacheln übersichtlich dargestellt. Jede Kachel enthält die folgenden Informationen:

- Spielzeit: Die Dauer des Spiels.
- Rundenzeit
- Spieleranzahl: Die aktuelle Anzahl der Spieler im Verhältnis zur maximal möglichen Spieleranzahl.
- Anzahl der Kamele

- Anzahl der Spielfelder

Um einem Spiel beitreten zu können, klickt man auf den Button „**Beitreten**“ innerhalb der entsprechenden Box. Zusätzlich befindet sich oben im Menü eine Suchleiste, mit der Sie gezielt nach einem Spiel anhand des Spielnamens suchen können. Die klare und intuitive Anordnung erleichtert Ihnen die Navigation und Auswahl eines Spiels.

Abbildung 8: Spiel



Im Spiel angelangt ist das **Spielfeld** mit den teilnehmenden Spielern und die Zuschauerplättchen zu sehen. Bei hoher Felderanzahl wird das Feld ausgeweitet, und es ist möglich, einzelne Felder in „**Feld suchen**“ zu suchen. Unabhängig von der Anzahl der Kamele auf einem Feld besteht die Möglichkeit, auf das Feld zu klicken, um alle Kamele darauf einzusehen.

Auf der linken Seite der Benutzeroberfläche befinden sich die Spielstatistiken und Informationen zu den Spielern:

- Eine Liste der Spielernamen wird zusammen mit ihrem jeweiligen Punktestand angezeigt. Die Spielerfarben sind dabei den Farben der Kamele zugeordnet.
- Die Buttons „**Etappenwetten**“ für die Wetten der aktuellen Runde und „**Finalwetten**“ für die Wettscheine des gesamten Spiels.
- Die Spielzeit und Zugzeit
- Die aktuelle Runde

Außerdem besteht die Möglichkeit, auf die Wett-Buttons zu klicken. Beim Klicken auf die „**Etappenwetten**“ können die Wettscheine pro Kamel eingesehen werden. Diese können auch mithilfe der ID gesucht oder die Wettscheine eines Spielers angezeigt werden. Beim Klicken von „**Finalwetten**“ besteht die Möglichkeit, die Wetten für das Sieger-Kamel und das letzte Kamel einzusehen.

Zusätzlich befinden sich am unteren Rand der Benutzeroberfläche weitere Funktionen:

- Pfeile, um durch die Felder zu navigieren.
- Doppelter Pfeil, um zum nächsten Feld zu gehen, auf dem ein Kamel oder ein Zuschauerplättchen liegt.
- Buttons, um das erste Kamel oder die rückwärts laufenden Kamele einmalig zu zentrieren.
- Eine Suchfunktion, um ein bestimmtes Kamel zu finden.
- Ein Umschalter, um fortlaufend die letzte Änderung oder das erste Kamel zu zentrieren.



Der Spielverlauf wird nach der Auswahl des Spiels in Echtzeit übertragen.



Abbildung 9: Würfeln

Es besteht die Möglichkeit, bereits gewürfelte Würfel und weitere Statistiken im Würfelfenster einzusehen. Dieses Fenster kann durch einen Klick auf das entsprechende Symbol geöffnet werden. Sollte die Anzahl der gewürfelten Würfel zu hoch werden, ist es möglich gezielt nach bestimmten Würfeln mithilfe der **Suchfunktion** im **Würfelfenster** zu suchen, indem die Kamel-ID eingegeben wird.



Abbildung 10: Info-Bereich



Abbildung 11: Settings-Bereich



Abbildung 12: Hotkeys-Übersicht

Der **Info-Bereich** enthält zahlreiche Informationen zum Spielverlauf, wie z. B. die Anzahl der Spieler, maximale Rundenanzahl, Bestrafungen und vieles mehr. Im **Settings-Bereich** kann man Lautstärke, maximal angezeigte Spielfelder und Sichtbarkeit der Kamel-ID einstellen. In der **Hotkeys-Übersicht** können die zur Verfügung stehenden Hotkeys angezeigt werden.

2.5 Eingesetzte Technologien

Libraries und Frameworks

Technologie	Beschreibung und Ziel
Overleaf/LaTeX	Overleaf ist eine Plattform zur kollaborativen Erstellung von hochwertigen LaTeX-Dokumenten, die simultanes Arbeiten mehrerer Nutzer mit Echtzeitaktualisierungen ermöglicht. Sie kombiniert eine benutzerfreundliche Oberfläche mit Vorlagen und Tools, die den LaTeX-Schreibprozess unterstützen. LaTeX selbst zeichnet sich durch eine markierungsbasierte Struktur aus, die professionelle Formatierung mathematischer Ausdrücke, automatische Typografie und eine klare Trennung von Inhalt und Design ermöglicht.
Lucidchart	Lucidchart wurde als Hilfsmittel zur Erstellung des Klassendiagramms verwendet, welches die einzelnen Komponenten, deren Attribute und Methoden sowie die Zusammenhänge der Komponenten darstellt.
Maven	Maven dient als Werkzeug für Projektmanagement und wird genutzt, um das Projekt mit Java zu bauen und um Abhängigkeiten innerhalb des Projektes zu verwalten.
GitLab und GitLab CI/CD	GitLab ist ein Versions-Kontroll-System, das genutzt wird, um Änderungen im Quellcode zu dokumentieren. Dies ermöglicht es, unserem Projektteam parallel den Code zu bearbeiten und jede Änderung direkt mit den anderen Teammitgliedern zu teilen. Dabei automatisiert die GitLab-Pipeline die Entwicklungsprozesse (siehe Kapitel 2.1) .
Gson	Gson ist eine Java-Bibliothek, die JSON-Objekte in Java-Objekte und umgekehrt konvertiert. Sie wird für die Kommunikation zwischen Server und Clients verwendet, indem JSON-Daten verarbeitet werden.
JUnit	JUnit ist ein Testtreiber. Mit ihm lassen sich automatisiert Tests durchführen, nachdem neue Units an Code implementiert wurden. Neue Units können beispielsweise einzelne Methoden oder Klassen sein, die programmiert worden sind.
JavaFX	JavaFX ist ein Framework zur Erstellung plattformunabhängiger grafischer Benutzeroberflächen (GUIs) für Java. Es unterstützt FXML für deklarative UI-Entwicklung und CSS für Styling. Der Scene Builder von JavaFX ist ein Werkzeug, das die visuelle Gestaltung von FXML-Dateien ermöglicht. Es erleichtert die Erstellung von Benutzeroberflächen, die später in JavaFX-Anwendungen verwendet werden können.
Javadoc	Javadoc dient dem Erstellen von Dokumentationen im HTML-Format. Diese werden von Javadoc automatisch aus den Deklarationen und Kommentaren im Java Source Code generiert.

Tabelle 1: Übersicht über verwendete Technologien und Frameworks

Pattern	Beschreibung
Builder	Das Builder-Pattern wird verwendet, um Objekte Schritt für Schritt zu erstellen. Es ist besonders nützlich, wenn ein Objekt komplexe Initialisierungen oder viele optionale Parameter hat.
Inversion of Control (IoC)	IoC ist ein Prinzip, bei dem die Kontrolle über die Erstellung und Verwaltung von Objekten an ein Framework oder eine externe Instanz übergeben wird. Es wird oft in Kombination mit Dependency Injection genutzt, um lose Kopplung zwischen Komponenten zu fördern.
Factory	Das Factory-Pattern ermöglicht die Erstellung von Objekten, ohne deren konkrete Implementierungen direkt zu instanziieren. Es bietet eine zentrale Stelle zur Erzeugung von Objekten basierend auf Eingaben oder Kontext.

Tabelle 2: Übersicht über Design Patterns

3 Änderungsverlauf

3.1 22.12.2024

- Abbildung 8 erneuert.
- Hinzufügen einer Übersicht der zur Verfügung stehenden Hotkeys.
- Hinzufügen von Pfeiltasten, um zum nächsten nicht-leeren Feld zu kommen.