



TECHNISCHE HOCHSCHULE MITTELHESSEN

THM

**CAMPUS
GIESSEN**

MNI

Mathematik, Naturwissenschaften
und Informatik

Benutzerhandbuch

Benutzerhandbuch

Softwaretechnik-Projekt SoSe2024

Thema

Plattform zum Vergleich von Spiele-KIs: **KIMaster**



Benutzerhandbuch

Änderungshistorie				
Version	Datum	Kapitel	Änderung	Name
0.1	04.07.2024	Alle	Anlegen und Füllen	Justine Buß
0.2	20.07.2024	4	Schreiben	Thorben Jones
0.3	21.07.2024	4	Nachbessern	Alexander Roos
0.4	21.07.2024	3	Schreiben	Omar Karkotli
0.5	23.07.2024	3	Schreiben	Omar Karkotli
0.6	24.07.2024	3, 6	Nachbessern, Schreiben	Justine Buß
1.0	24.07.2024	3	Bilder einfügen	Omar Karkotli



Benutzerhandbuch

Herausgeber	Technische Hochschule Mittelhessen – FB06 Mathematik, Naturwissenschaften und Informatik	
Dateiname	Benutzerhandbuch_ver_1.0	
Dokumentenbezeichnung	Benutzerhandbuch: Plattform zum Vergleich von Spiele-KIs: KIMaster	
Version	1.0	
Stand	Donnerstag, 25. Juli 2024	
Status	Fertiggestellt	
Autoren	Justine Buß, Thorben Jones, Alexander Roos, Maximilian Bachmann, Omar Karkotli, Sven Roman Reinhard, Pascal Waldschmidt	
Freigegeben von		
Ansprechpartner	Justine Buß	justine.buss@mni.thm.de
	Thorben Jones	thorben.jones@mni.thm.de
Kurzinfo	„Technische Hochschule Mittelhessen Softwaretechnik-Projekt. Benutzerhandbuch für KIMaster.“	



Benutzerhandbuch

Inhaltsverzeichnis

1 Einführung.....	8
1.1 Kurzübersicht der Software.....	8
1.2 Zielgruppen.....	8
2 Einrichtung.....	9
2.1 Systemanforderungen	9
3 Benutzeranleitung für die Webseite	11
3.1 Webseite aufrufen	11
3.2 Startseite	12
3.2.1 Navigationsleiste	12
3.2.2 Begrüßung	13
3.2.3 Spielauswahl	13
3.2.4 Eingabe des Lobbyschlüssels	14
3.2.5 Footer.....	15
3.3 Warteraum.....	17
3.4 Lobby-Seite.....	18
3.4.1 Spielekonfiguration	19
3.4.2 Status und Lobby-Schlüssel	20
3.5 Spielseite.....	22
3.5.1 Spielbrett.....	24
3.5.4 Fertiges Spiel	29
3.5.5 Zeitleiste	30
3.5.6 Fehleranalyse	31
3.5.7 Mini-Fenster	32
4 Benutzeranleitung für die KIMaster-Schnittstelle	33
4.1 Überblick über die Schnittstelle (Python Beispiel)	33
4.2 Kommandos	34
4.3 Voraussetzungen.....	35
4.4 Einrichtung	35
4.5 Nutzung der Hauptfunktionen.....	37
5 Häufige Probleme und Lösungen	39



Benutzerhandbuch

5.1 Meine Verbindung bricht ständig ab, wenn ich etwas empfangen?	39
5.2 Ich kann mich nicht mit dem THM-Server verbinden?	39
5.3 Ich erhalte keine Antworten trotz While True loop?	40
6 Literaturverzeichnis	41



Benutzerhandbuch

Abbildungsverzeichnis

Abbildung 1: Verbindungsherstellung THM VPN	10
Abbildung 2: Verbindung hergestellt THM VPN	10
Abbildung 3: URL-Eingabe in Suchleiste des Browsers	11
Abbildung 4: Startseite	12
Abbildung 5: Navigationsleiste	12
Abbildung 6: Begrüßung	13
Abbildung 7: Spieleauswahl	13
Abbildung 8: Spieleauswahl Hovering	14
Abbildung 9: Lobby beitreten	14
Abbildung 10: Lobby beitreten - Fehleingabe	14
Abbildung 11: Footer	15
Abbildung 12: Datenschutzerklärung	15
Abbildung 13: Impressum THM	16
Abbildung 14: Warteraum: belegte Position Spieler 1	17
Abbildung 15: Lobby-Seite	18
Abbildung 16: Spielerkonfiguration	19
Abbildung 17: Spielmoduskonfiguration	19
Abbildung 18: Schwierigkeitskonfiguration	19
Abbildung 19: Spielelobbywechsel	20
Abbildung 20: Lobby-Schlüssel & QR-Code	20
Abbildung 21: QR-Code	20
Abbildung 22: Positionsanzeige	21
Abbildung 23: Lobby-Aktionen	21
Abbildung 24: Spieleseite)	22
Abbildung 25: Spielregeln	23
Abbildung 26: Zurückgenommener Zug	24
Abbildung 27: Valide Züge	24
Abbildung 28: Maus-Hoover	25
Abbildung 29: Ausgeführter Zug	25
Abbildung 30: Dame-Brett	26
Abbildung 31: Mögliche Züge anzeigen	26
Abbildung 32: Zug ausgeführt (Dame)	27
Abbildung 33: Nim-Brett	28
Abbildung 34: Ausgewählte Reihe	28
Abbildung 35: Zug bestätigt	29
Abbildung 36: Spiel vorbei	29
Abbildung 37: Spielende zusätzliche Funktionen	30
Abbildung 38: Zeitleiste - vorherig	30
Abbildung 39: Fehleranalyse	31
Abbildung 40: Mini-Fenster	32



Benutzerhandbuch

Abbildung 41: Ordnerstruktur Python Schnittstelle	34
Abbildung 42: Install-Befehl Bibliotheken	35
Abbildung 43: Zu verbindende URIs	35
Abbildung 44: connect()	36
Abbildung 45: main()	36
Abbildung 46: send_cmd()	37
Abbildung 47: receive()	38
Abbildung 48: Hauptprogramm	38
Abbildung 49: Formatüberprüfung	39
Abbildung 50: receive_handler() ohne sleep	40
Abbildung 51: send_handler() mit sleep	40



Benutzerhandbuch

1 Einführung

Dieses Benutzerhandbuch dient als Leitfaden für die Navigation und Nutzung der Plattform KIMaster. Es richtet sich sowohl an Nutzer, die über die Webseite spielen möchten, als auch an solche, die komplexere Anwendungsfälle, wie die Anbindung an die Schnittstelle und die Implementierung eigener Künstlicher Intelligenzen (KI), realisieren wollen.

1.1 Kurzübersicht der Software

Die Software stellt eine vielseitige Plattform bereit, auf der Nutzer in verschiedenen Spielen gegeneinander antreten können. Ein Alleinstellungsmerkmal ist die Möglichkeit, gegen vortrainierte KIs zu spielen, die für jedes Spiel verfügbar sind. Die Plattform kann über eine Webseite genutzt werden oder über ein Terminal, wenn die Anbindung an die bereitgestellte Schnittstelle erfolgt.

Ein zusätzliches Feature der Software ist die Option, eigene KI-Implementierungen zu integrieren. Entwickler können ihre KIs gegen andere Spieler, deren KIs oder vortrainierte KIs testen. Dies eröffnet vielfältige Möglichkeiten für Entwicklungen im Bereich der KI.

1.2 Zielgruppen

Dieses Benutzerhandbuch richtet sich an eine vielfältige Nutzergruppe:

- **Allgemeiner Spieler:** Personen, die auf der Webseite spielen und gegen andere Nutzer oder vortrainierte KIs antreten möchten.
- **Studierende:** Nutzer, die erste Erfahrungen mit KI sammeln oder spezielle KI-Module belegen und ihre eigenen KIs entwickeln möchten.
- **Dozenten:** Lehrkräfte, die die Plattform für Lehr- und Forschungszwecke im Bereich KI verwenden.
- **Entwickler:** Nutzer, die die Plattform über die Schnittstelle nutzen und eigene KI-Implementierungen integrieren möchten.



Benutzerhandbuch

2 Einrichtung

2.1 Systemanforderungen

1. Zugriff über die Webseite

Für den Zugriff über die Webseite sind folgende Anforderungen zu erfüllen:

- **Internetfähiger Browser:** Ein moderner Browser wie [Google Chrome](#), [Mozilla Firefox](#), [Apple Safari](#) oder [Microsoft Edge](#) ist erforderlich.
- **Netzwerkverbindung:** Der Zugriff muss innerhalb des Netzwerks der [Technischen Hochschule Mittelhessen](#) erfolgen. Dies kann entweder direkt vor Ort oder durch Nutzung eines [VPN-Dienstes der THM](#) geschehen.

2. Externe Anbindung

Für die externe Anbindung an die Systeme der THM sind folgende Voraussetzungen notwendig:

- **Netzwerkverbindung der THM:** Der Zugriff muss über das Netzwerk der Technischen Hochschule Mittelhessen erfolgen, was entweder direkt vor Ort oder durch die Nutzung eines [VPN-Dienstes der THM](#) möglich ist.
- **Internetverbindung:** Eine stabile Internetverbindung ist erforderlich.
- **Programmierungsumgebung:** Eine Entwicklungsumgebung (IDE) oder ein Editor, der die gewählte Programmiersprache unterstützt, muss installiert sein.
- **WebSocket-Verbindungen:** Alle Verbindungen zu den Systemen der THM werden über [WebSocket](#)-Verbindungen realisiert. Stellen Sie sicher, dass Ihre Programmierungsumgebung und die verwendete Sprache die Einrichtung und Nutzung von WebSocket-Verbindungen unterstützen.

Zusätzliche Hinweise

- Stellen Sie sicher, dass alle Sicherheitsupdates und Patches für den verwendeten Browser und die Programmierungsumgebung installiert sind.

Benutzerhandbuch

- Für die Nutzung des VPN-Dienstes der THM folgen Sie bitte den Anweisungen auf der offiziellen [Webseite](https://www.thm.de/its/campusnetz/vpn/ciscoanyconnect.html) der THM: <https://www.thm.de/its/campusnetz/vpn/ciscoanyconnect.html> oder kontaktieren Sie den IT-Support bei möglichen Problemen.

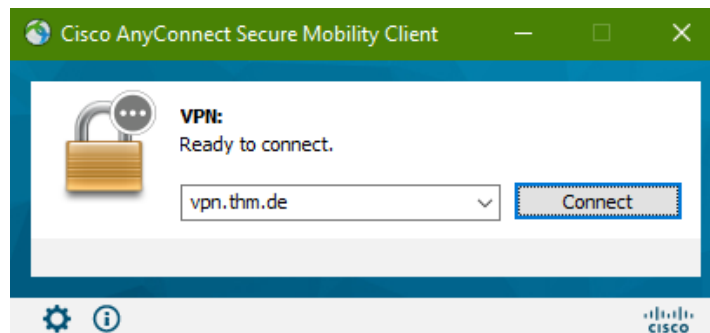


Abbildung 1: Verbindungsherstellung THM VPN

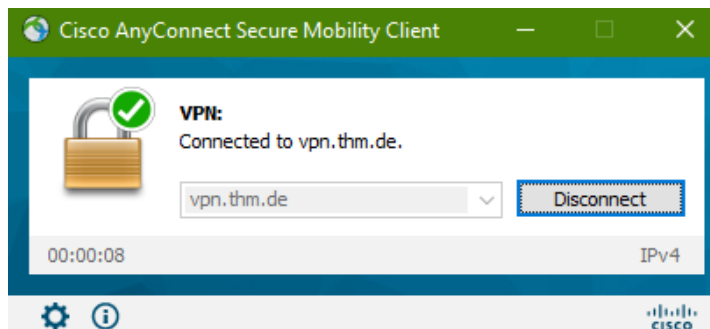


Abbildung 2: Verbindung hergestellt THM VPN

- Überprüfen Sie, ob zusätzliche Bibliotheken oder Frameworks für die Unterstützung von WebSocket-Verbindungen in Ihrer Programmierumgebung erforderlich sind, und installieren Sie diese entsprechend.
 - [Python](#): websockets, socket.io, tornado
 - [Java](#): Java.WebSocket, Tyrus, Spring WebSocket
 - [C#](#): SignalR, WebSocketSharp
 - [Rust](#): tokio-tungstenite, async-tungstenite
 - [Kotlin](#): ktor, OkHttp
- Wir bieten auf unserem [GitHub-Repository](#) ein Verbindungsbeispiel mit Python an.

Benutzerhandbuch

3 Benutzeranleitung für die Webseite

3.1 Webseite aufrufen

1. Stellen Sie sicher, dass Sie die [Systemanforderungen](#) erfüllen.
2. Öffnen Sie einen Webbrowser Ihrer Wahl.
 - Google Chrome
 - Apple Safari
 - Mozilla Firefox
 - Microsoft Edge
 - ...
3. Geben Sie die URL kimaster.mni.thm.de in die Adressleiste Ihres Browsers ein.

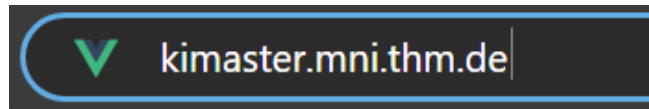


Abbildung 3: URL-Eingabe in Suchleiste des Browsers

4. Bestätigen Sie Ihre Eingabe durch Drücken der Eingabetaste (Enter).

Benutzerhandbuch

3.2 Startseite



Abbildung 4: Startseite

Die Startseite von KIMaster, kurz KIM genannt, ist übersichtlich und benutzerfreundlich gestaltet. Das Design ist minimalistisch und fokussiert sich darauf, dem Benutzer eine einfache Navigation und eine klare Struktur zu bieten.

Die Seite verwendet ein schlichtes Farbschema mit weißen Hintergründen und grünen Akzenten, die die Markenfarben der THM widerspiegeln.

Sie ist so gestaltet, dass sie auf verschiedenen Geräten gut aussieht und funktioniert. Ob auf einem Desktop-Computer, Tablet oder Smartphone, die Elemente der Seite passen sich flexibel an die Bildschirmgröße an und bleiben gut bedienbar.

Die Startseite der KIMaster-Plattform begrüßt den Benutzer und bietet eine Übersicht der verfügbaren Optionen und Spiele.

3.2.1 Navigationsleiste

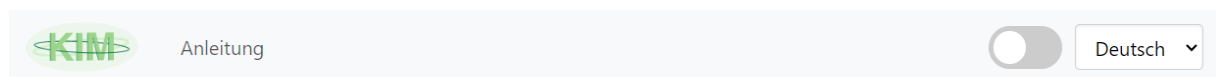


Abbildung 5: Navigationsleiste

In der linken oberen Ecke befindet sich das KIM-Logo, das auch als Button zur Rückkehr auf die Startseite dient.

Neben dem Logo ist der Link zur [Anleitung](#). Dieser führt zu Benutzeranleitung für die Schnittstelle (API).

Benutzerhandbuch

In der oberen rechten Ecke gibt es einen Sprachschalter, um die Sprache der Plattform zu ändern. Die verfügbaren Sprachen sind Deutsch, Englisch, Spanisch, Französisch.

Links neben der Sprachauswahl, befindet sich auch der Schalter für die verschiedenen Farbmodi, wie z.B. Darkmode. Diese sind zur Zeit noch nicht integriert.

3.2.2 Begrüßung

Willkommen zu KIMaster

Hier kannst du deine Spiele-KI testen

Abbildung 6: Begrüßung

Die Startseite zeichnet sich unter anderem dadurch aus, dass der Benutzer auf der Plattform willkommen geheißen wird. So hebt sich die Startseite auch von den anderen Seiten ab und man erkennt sofort, dass man sich auf dieser Seite befindet.

3.2.3 Spielauswahl

Vier Gewinnt

Tic Tac Toe

Othello

Nim

Dame

Abbildung 7: Spielauswahl

Unterhalb der Begrüßung befindet sich die Spielauswahl. Hier sind die verfügbaren Spiele in anklickbaren Buttons nebeneinander aufgelistet. Jeder Button ist mit dem Namen des Spiels beschriftet. Die aktuell verfügbaren Spiele sind: Vier gewinnt, Tic Tac Toe, Othello, Nim und Dame.

Benutzerhandbuch

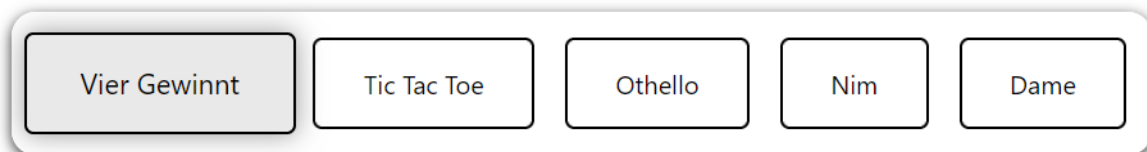


Abbildung 8: Spieleauswahl Hovering

Beim Hovern mit der Maus über die Buttons, wird der Button des Ausgewählten Spiels vergrößert und grau hervorgehoben. Hier im Beispiel am Vier Gewinnt Button dargestellt. Durch Klicken auf einen dieser Buttons gelangt der Benutzer in die jeweilige [Spielelobby](#).

3.2.4 Eingabe des Lobbyschlüssels

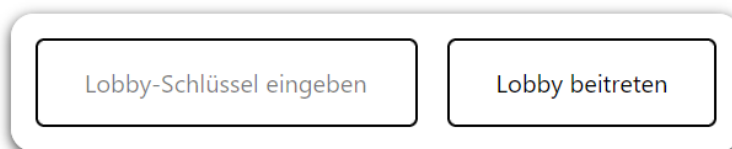


Abbildung 9: Lobby beitreten

Unter der Spielauswahl gibt es ein Eingabefeld mit der Aufschrift „Lobby-Schlüssel eingeben“. Hier kann der Benutzer einen Schlüssel eingeben, um einer bestimmten Lobby beizutreten. Direkt daneben befindet sich ein weiterer Button mit der Aufschrift „Lobby beitreten“. Dieser Button ermöglicht es dem Benutzer, einer [bestehenden Lobby beizutreten](#), nachdem ein Schlüssel eingegeben wurde.

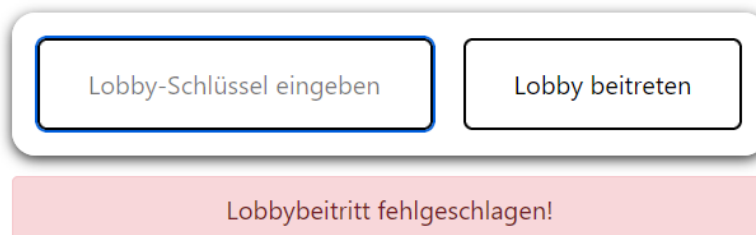


Abbildung 10: Lobby beitreten - Fehleingabe

Falls die Eingabe nicht korrekt war, erhält man eine Fehlermeldung unter dem Eingabefenster. Diese Fehlermeldung ist rot hervorgehoben und wird für 5 Sekunden eingeblendet, bevor sie wieder verschwindet.




Benutzerhandbuch

3.2.5 Footer

[THM](#) [Datenschutz](#) [Impressum](#)

Abbildung 11: Footer

Am unteren Rand der Seite befindet sich der Footer. Hier sind wichtige rechtliche Informationen verlinkt. Zum einen ist die [Webseite der THM](#) verlinkt, da dieses Projekt und die Plattform im Rahmen eines THM-Projekts entstanden sind. Außerdem finden Sie hier die Datenschutzbestimmungen und das Impressum.



☐ Deutsch ▾

Datenschutzerklärung

A. Name und Anschrift des Verantwortlichen und seines Vertreters

Der Verantwortliche im Sinne der EU-Datenschutzgrundverordnung (DSGVO) und anderer nationaler Datenschutzgesetze - insbesondere des Hessischen Datenschutz- und Informationsfreiheitsgesetzes (HDSIG) - sowie sonstiger datenschutzrechtlicher Bestimmungen ist die:

Technische Hochschule Mittelhessen (THM)
Wiesenstraße 14
Tel.: + 49 641 309-0
E-Mail: info@thm.de
Gesetzlich vertreten durch ihren Präsidenten
Prof. Dr. Matthias Willems
Wiesenstraße 14
Tel.: + 49 641 309-0
E-Mail: praesident@thm.de

B. Kontaktdaten des Datenschutzbeauftragten

Die Datenschutzbeauftragten des Verantwortlichen sind:

Abbildung 12: Datenschutzerklärung

Auf unserer Webseite legen wir großen Wert auf den Schutz Ihrer persönlichen Daten. Unsere Datenschutzrichtlinien erläutern, wie wir Ihre Daten erfassen, verwenden und schützen. Bitte lesen Sie unsere Datenschutzbestimmungen sorgfältig, um mehr darüber zu erfahren, wie wir Ihre Privatsphäre wahren.



THM

TECHNISCHE HOCHSCHULE MITTELHESSEN

CAMPUS
GIESSEN

MNI

Mathematik, Naturwissenschaften
und Informatik

Benutzerhandbuch



THM

TECHNISCHE HOCHSCHULE MITTELHESSEN

STUDIUM

WEITERBILDUNG

FORSCHUNG & TRANSFER

📍 Impressum

Impressum

Technische Hochschule Mittelhessen (THM)
University of Applied Sciences

Wiesenstraße 14
35390 Gießen
Germany

☎ +49 641 309-0

📠 +49 641 309-2901

✉ [.praesident@thm.de](mailto:praesident@thm.de)

Abbildung 13: Impressum THM

Für rechtliche Informationen und Kontaktangaben verweisen wir auf das [Impressum der THM](#) verwiesen. Dort finden Sie alle relevanten Details gemäß den gesetzlichen Anforderungen.

Benutzerhandbuch

3.3 Warteraum

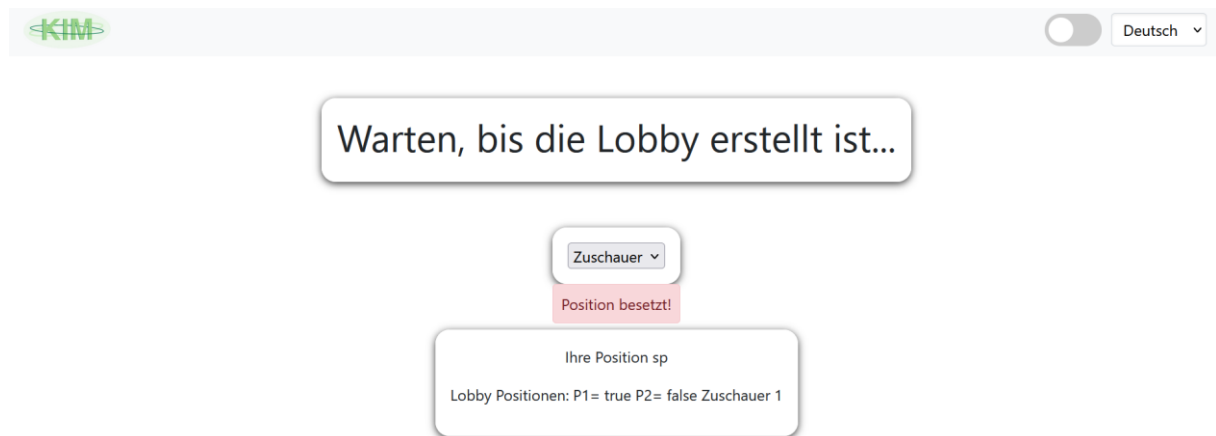


Abbildung 14: Warteraum: belegte Position Spieler 1

Nach Eingabe des Lobby-Schlüssels auf der Hauptseite gelangen Sie in den Warteraum. Hier haben Sie die Möglichkeit, ihre Position im späteren Spiel zu konfigurieren. Je nach noch unbelegten Modi ist es möglich als Spieler oder Zuschauer beizutreten. So können sie als Spieler aktiv am Spiel teilnehmen. Als Zuschauer kann das Spielgeschehen beobachtet werden, sodass neben dem Beiwohnen zu normalen Partien zwischen zwei Nutzern auch Runden gegen KI-Gegner oder unsere KI-Implementierung KIM beobachtet werden können. Sie bekommen aktiv Feedback, wenn die ausgewählte Position bereits belegt ist. Die Anzahl der Zuschauer ist dabei nicht begrenzt.

Sobald alle Teilnehmer bereit sind und das Spiel gestartet wird, gelangen Sie automatisch zur [Spieleseite](#), wo das eigentliche Spiel stattfindet.

Benutzerhandbuch

3.4 Lobby-Seite

Willkommen zur Vier Gewinnt Lobby

Spieler 1 ▾

Spieler gegen KIM ▾

Einfach ▾

Vier Gewinnt ▾

Lobby Key: a82a2

QR-Code

Ihre Position p1

Lobby Positionen: P1= true P2= false Zuschauer 1

Lobby verlassen

Spiel Starten

Abbildung 15: Lobby-Seite

Die Lobby-Seite dient als zentrale Anlaufstelle für Ihre Spielekonfiguration. Die Spielelobbys sind allgemein gleich aufgebaut, mit leichten Anpassungen bezüglich der Schriftzüge und Spielekonfigurationsmöglichkeiten.

Benutzerhandbuch

3.4.1 Spielekonfiguration

Bei der Spielekonfiguration können Sie die Parameter für Ihr Spiel festlegen. Passen Sie die Einstellungen nach Ihren Wünschen an.

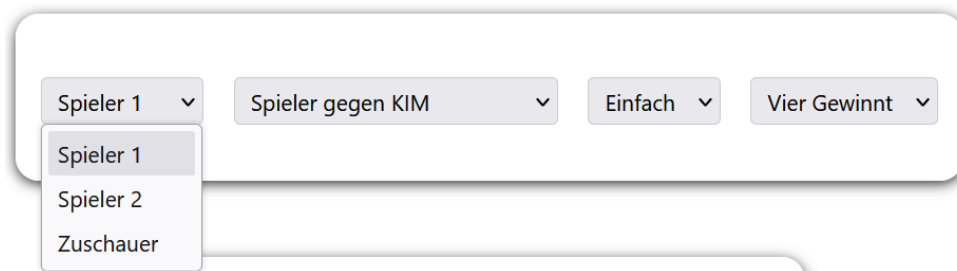


Abbildung 16: Spielerkonfiguration

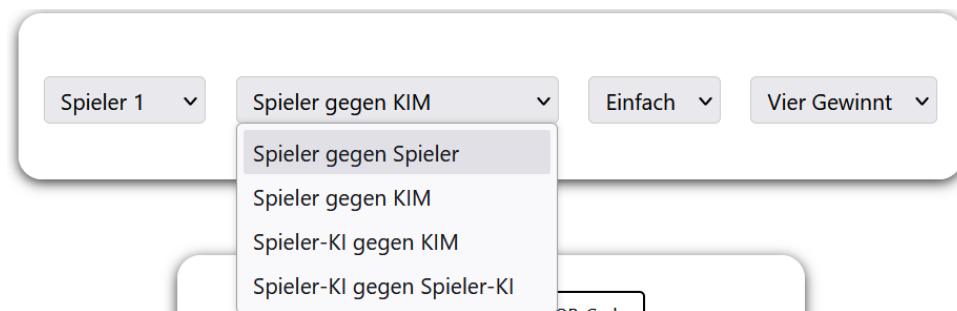


Abbildung 17: Spielmoduskonfiguration

Es besteht die Möglichkeit die eigenen Spielposition auf Spieler 1, Spieler 2 oder Zuschauer zu konfigurieren. Zudem kann der Spielmodus ausgewählt werden. So bieten wir normale „Spieler gegen Spieler“ Erlebnisse, aber auch „Spieler gegen KIM“ (unsere vortrainierte KI) oder auch das antreten mit der eigenen Spieler-KI an. Natürlich gibt es auch hier nochmal alle möglichen Konfigurationen.

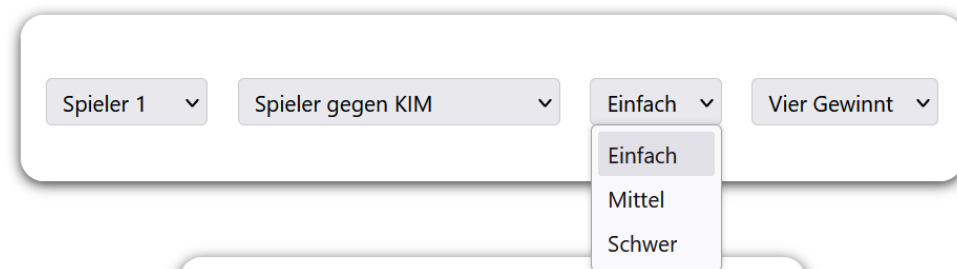


Abbildung 18: Schwierigkeitskonfiguration

Wird ein Spiel gegen KIM ausgewählt, so kann noch der Schwierigkeitsgrad von KIM zwischen einfach, mittel und schwer verstellt werden.

Benutzerhandbuch

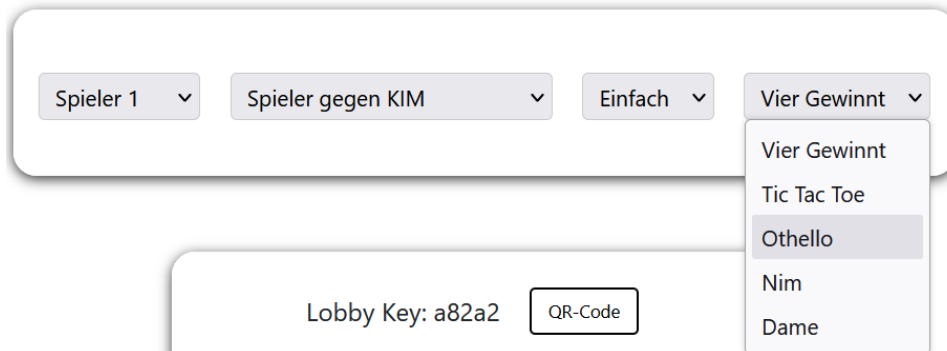


Abbildung 19: Spielelobbywechsel

Möchte man zudem vielleicht doch das Spiel, als das zuvor ausgewählte, wechseln, so ist dies über ein eigenes Drop-Down-Menü einfach möglich. Die Spiele-Lobby muss dafür nicht verlassen werden und bereits weitergegebene Schlüssel können beibehalten werden.

3.4.2 Status und Lobby-Schlüssel

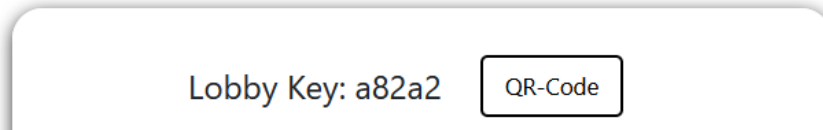


Abbildung 20: Lobby-Schlüssel & QR-Code

Der Lobbyschlüssel ist ein eindeutiger Code, der Ihnen den Zugang zu einer bestimmten Lobby ermöglicht. Geben Sie diesen Schlüssel an Freunde und Mitspielende oder Ihre eigene KI-Implementierung weiter, um der gewünschten Lobby beizutreten. Achten Sie darauf, den Schlüssel korrekt einzugeben, um Probleme beim Zugang zur Lobby zu vermeiden.

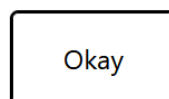


Abbildung 21: QR-Code

Der Einfachheit halber gibt es zusätzlich einen QR-Code, der das Scannen mit mobilen Geräten ermöglicht und das Beitreten in die Lobby vereinfacht.

Benutzerhandbuch

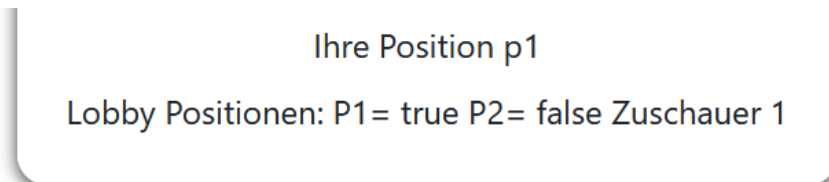


Abbildung 22: Positionsanzeige

Zusätzlich können Sie hier Ihren Status und den Status anderer Spieler einsehen, um sich einen Überblick darüber zu verschaffen, welche Positionen noch unbelegt sind und wie viele Zuschauer Sie gegebenenfalls bei Ihrem Spiel erwarten können.

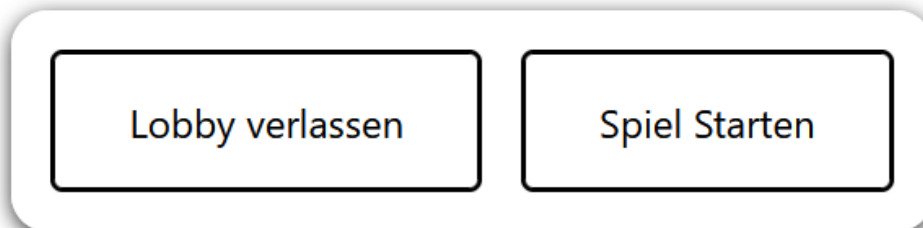


Abbildung 23: Lobby-Aktionen

Mit einem Klick auf Spiel Starten wird man auf die [Spieleseite](#) weitergeleitet. Alle weiteren Nutzer, die sich im [Warteraum](#) befunden haben, werden nun ebenfalls auf das Spiel weitergeleitet.

Benutzerhandbuch

3.5 Spielseite

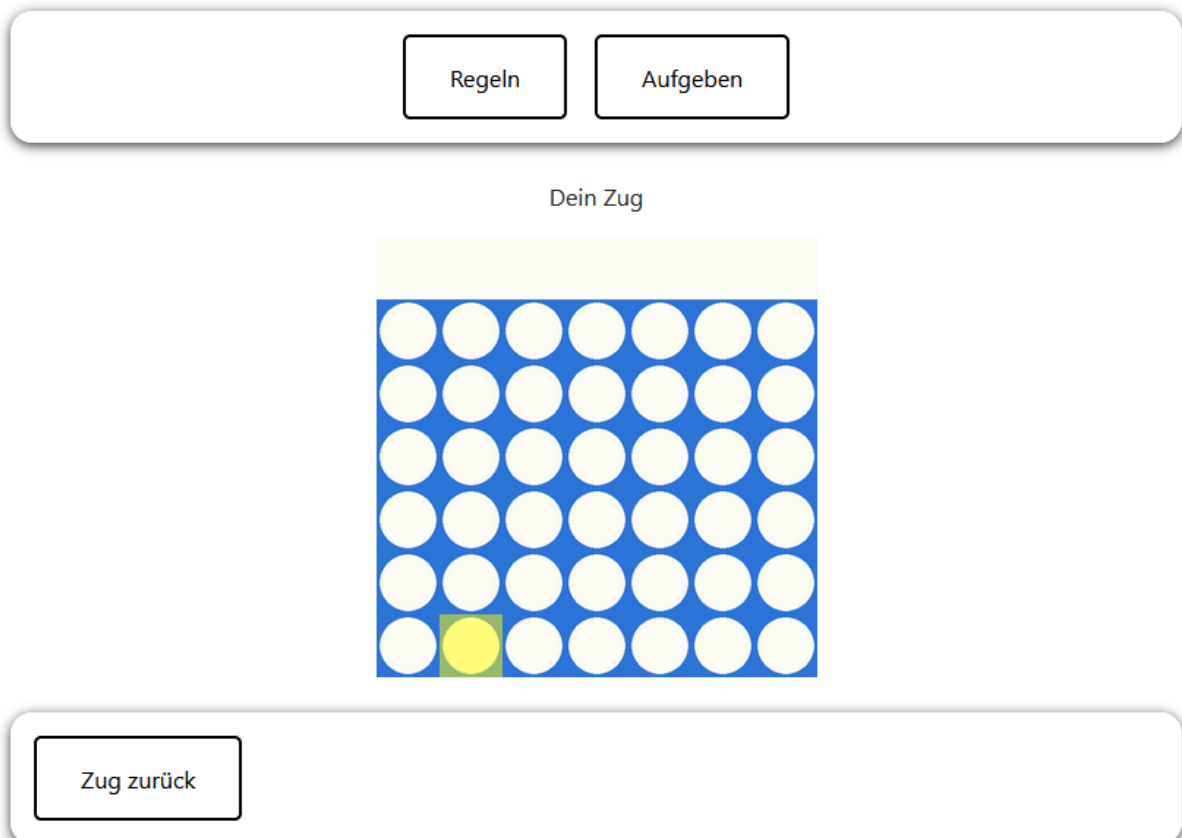


Abbildung 24: Spielseite)

Auf der Spieleseite finden Sie das zentrale Spielfeld, das sich in der Mitte der Seite befindet. Diese Ansicht zeigt nur die relevanten Schaltflächen und Funktionen an, die für das aktuelle Spiel notwendig sind. Zudem ist über dem Spielbrett angezeigt, wer gerade am Zug ist. Die Benutzeroberfläche ist so gestaltet, dass sie Ihnen eine klare und fokussierte Spielerfahrung bietet.

Benutzerhandbuch



Abbildung 25: Spielregeln

Ein Button ermöglicht es Ihnen, die Spielregeln jederzeit einzusehen. Dieser Button hilft Ihnen, sich über die Spielmechanik und spezifische Regeln zu informieren, um strategische Entscheidungen zu treffen.

Mit dem Aufgeben-Button können Sie eine laufende Partie beenden, wenn Sie sich entscheiden, das Spiel aufzugeben. Dies beendet das aktuelle Spiel und zeigt Ihnen das [Ergebnis](#) an.

Benutzerhandbuch

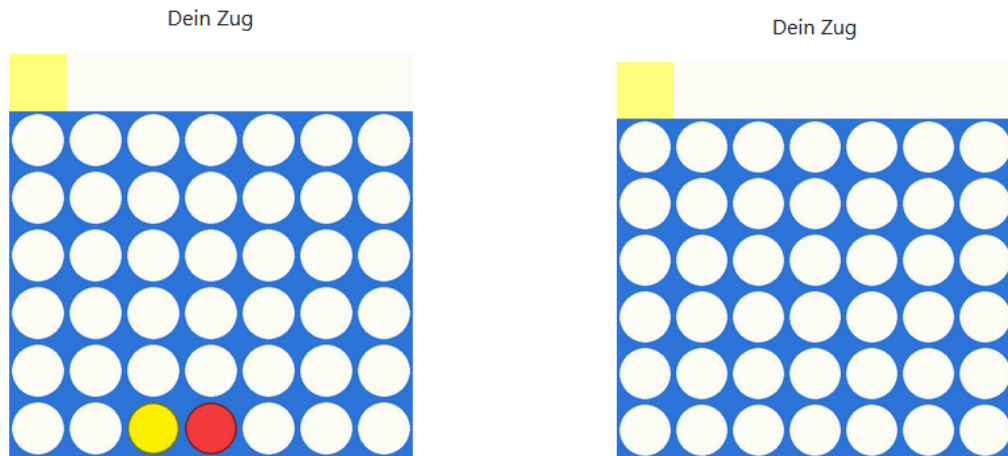


Abbildung 26: Zurückgenommener Zug

Dieser Button ist nur in der Konfiguration „Spieler gegen KIM“ verfügbar. Er erlaubt es Ihnen, den letzten Zug rückgängig zu machen, um Ihre Strategie anzupassen oder Fehler zu korrigieren. Dabei wird automatisch natürlich auch der letzte Zug von KIM rückgängig gemacht.

3.5.1 Spielbrett

Das Spielbrett ist die zentrale Fläche, auf der alle Aktionen stattfinden. Hier sind die wichtigsten Interaktionen erklärt:

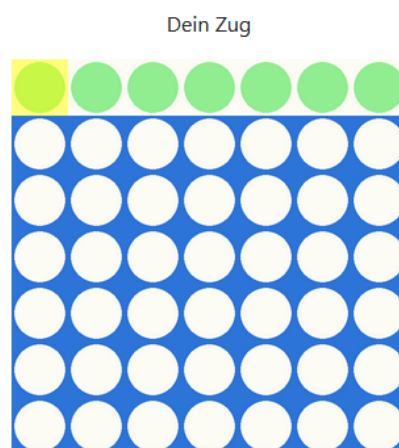


Abbildung 27: Valide Züge

- **Rechtsklick:** Wenn Sie mit der rechten Maustaste auf das Spielbrett klicken, wird eine Übersicht aller möglichen Züge angezeigt. Diese Funktion hilft Ihnen dabei, Ihre Optionen zu überprüfen und strategische Entscheidungen zu treffen.

Benutzerhandbuch

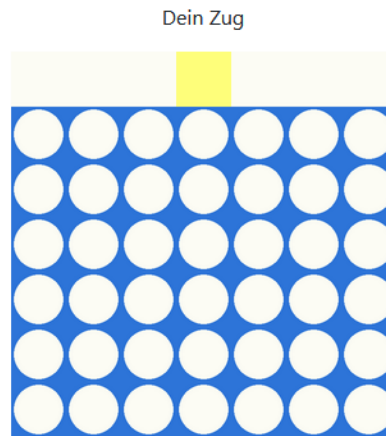


Abbildung 28: Maus-Hover

- **Maus-Hover:** Wenn Sie die Maus über ein Feld bewegen, wird dieses farbig hervorgehoben. Diese visuelle Markierung zeigt Ihnen an, welche Felder für Ihren nächsten Zug verfügbar sind und erleichtert die Orientierung auf dem Brett.

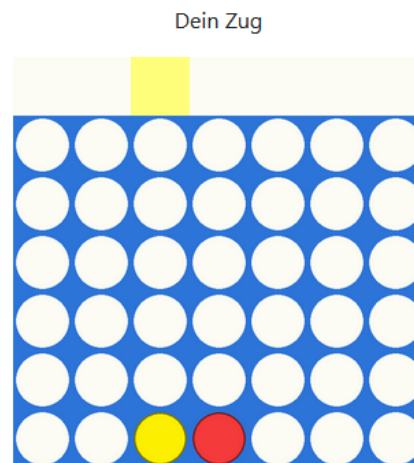
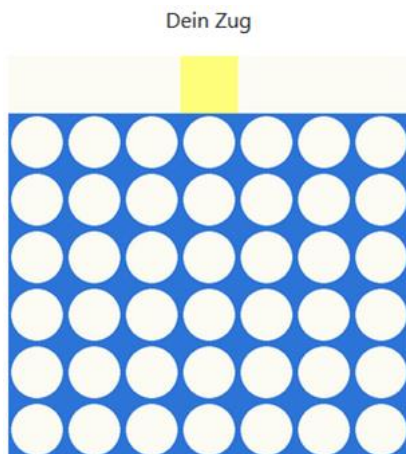


Abbildung 29: Ausgeführter Zug

- **Klick zum Ausführen:** Durch einen einfachen Klick auf ein hervorgehobenes Feld führen Sie den gewünschten Zug aus. Dieser Klick bestätigt Ihre Wahl und setzt die Spielaktion in Gang. Werden Spiele gegen KIM ausgeführt, setzt die KI daraufhin ihren Zug. Andernfalls ist der gegnerische Spieler oder dessen KI am Zug.

Diese Interaktionsmöglichkeiten sorgen für eine intuitive und benutzerfreundliche Steuerung des Spiels.

Benutzerhandbuch

Besonderheiten

Beim Spiel „Dame“ gibt es spezifische Interaktionsschritte, die Sie beachten sollten:

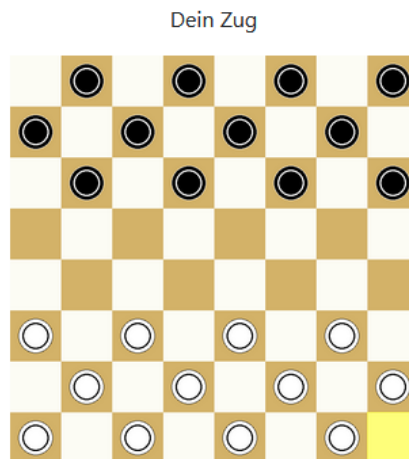


Abbildung 30: Dame-Brett

- **Spielstein auswählen:** Bevor Sie einen Zug ausführen können, müssen Sie zuerst den Spielstein anklicken, den Sie bewegen möchten. Dies aktiviert die Auswahl dieses Steins und zeigt Ihnen die möglichen Züge an.

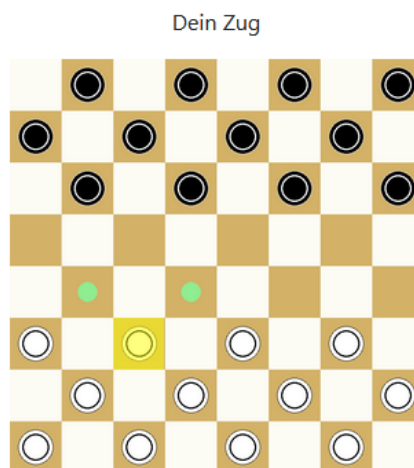


Abbildung 31: Mögliche Züge anzeigen

- **Mögliche Züge anzeigen:** Nachdem Sie einen Spielstein ausgewählt haben, werden alle gültigen Züge für diesen Stein auf dem Brett hervorgehoben. Diese Züge werden durch farbige Markierungen angezeigt, die Ihnen eine klare Übersicht über die verfügbaren Optionen bieten.

Benutzerhandbuch

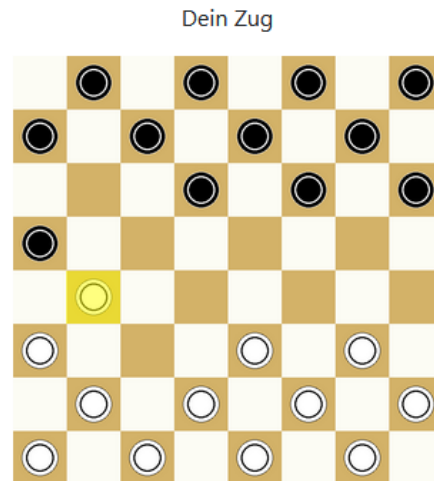


Abbildung 32: Zug ausgeführt (Dame)

- **Zug ausführen:** Wählen Sie eines der markierten Felder aus, um Ihren Zug auszuführen. Ein Klick auf das gewünschte Feld bewegt den ausgewählten Spielstein dorthin und setzt das Spiel fort.

Benutzerhandbuch

Zudem gibt es bei „Nim“ eine spezielle Vorgehensweise für das Ausführen von Zügen:

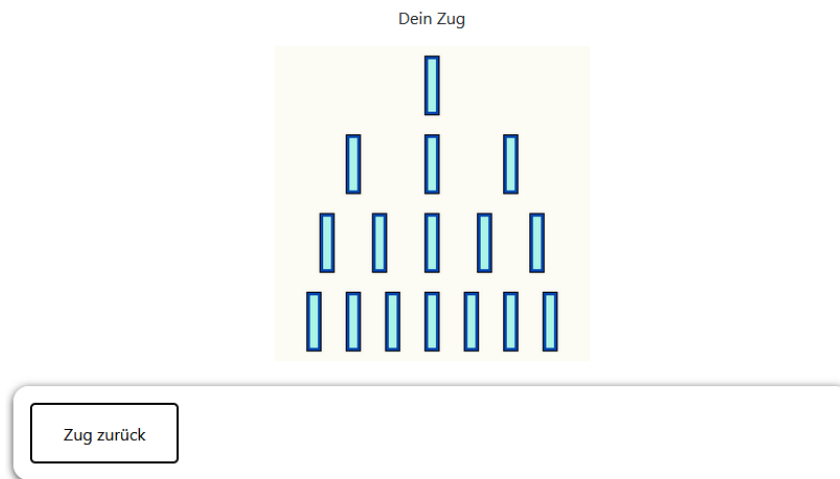


Abbildung 33: Nim-Brett

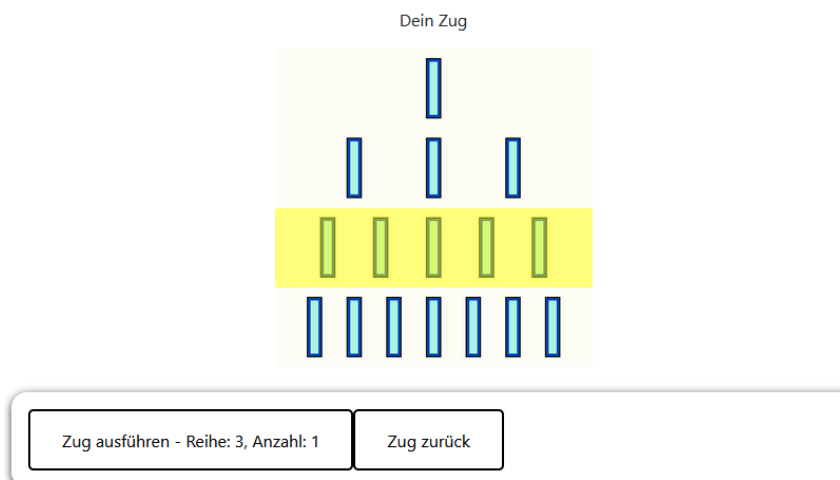


Abbildung 34: Ausgewählte Reihe

- **Reihe und Menge auswählen:** Klicken Sie auf die Reihe, aus der Sie Streichhölzer bzw. Stäbchen entfernen möchten. Sobald Sie eine Reihe ausgewählt haben, wird oben auf einem Button angezeigt, wie viele Streichhölzer oder Stäbchen Sie in dieser Reihe wegnehmen möchten. Durch mehrfaches Klicken auf die Reihe wird die Anzahl der Hölzer jedes Mal um eins erhöht.

Benutzerhandbuch

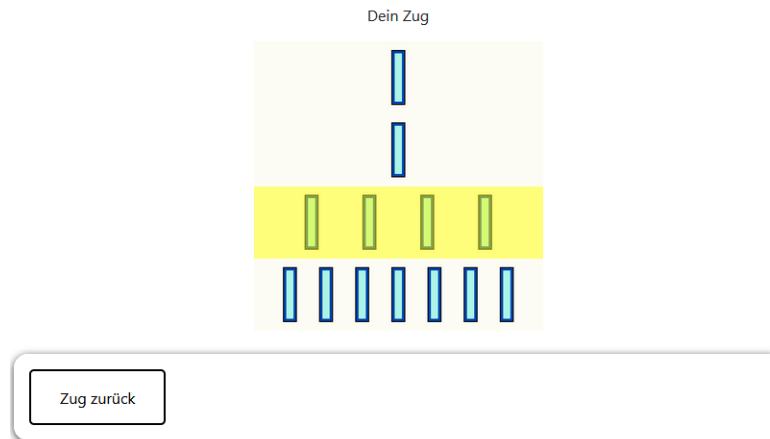


Abbildung 35: Zug bestätigt

- **Zug bestätigen:** Erst wenn Sie auf den Button mit der Anzahl der ausgewählten Streichhölzer oder Stäbchen klicken, wird der Zug ausgeführt. Dieser Klick bestätigt Ihre Wahl und entfernt die angegebenen Streichhölzer bzw. Stäbchen aus der gewählten Reihe.

3.5.4 Fertiges Spiel

Spiel vorbei

Gegner hat gewonnen

Spiel vorbei nach 18 Zügen



Abbildung 36: Spiel vorbei

Wenn das Spiel endet – sei es durch Aufgabe eines Spielers, das Fehlen weiterer gültiger Züge (wie bei Othello) oder den Sieg eines Spielers – wird ein Pop-up angezeigt. Dieses Pop-up informiert Sie darüber, wer das Spiel gewonnen hat, und gibt die Anzahl der gespielten Züge an. Sie haben dann die Möglichkeit, entweder ein neues Spiel zu starten oder durch Klicken auf „Okay“ zu den weiteren Spielfunktionen zurückzukehren. Das Pop-up ermöglicht es Ihnen, schnell zu einer neuen Runde zu wechseln oder das Spielgeschehen zu beenden und andere Optionen zu erkunden.

Benutzerhandbuch

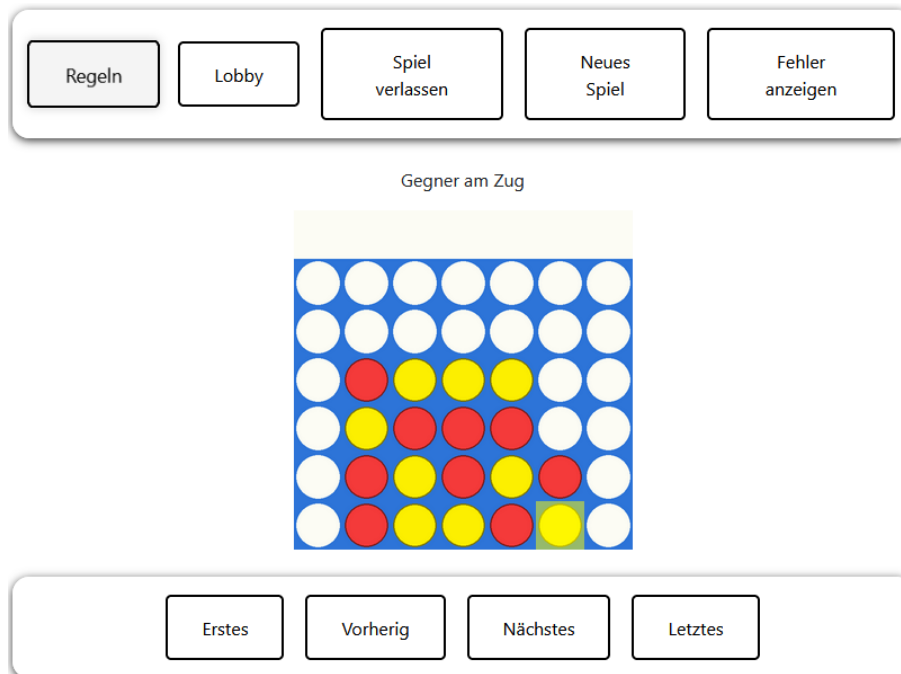


Abbildung 37: Spielende zusätzliche Funktionen

Hat man sich dazu entschieden nicht direkt eine neue Runde durch „Neues Spiel“ zu starten, so landet man wieder auf der Spielseite mit dem letzten Stand des Spielbrettes. Es sind dabei einige neue Buttons aufgetaucht, die neue Funktionen bereitstellen.

Neben „Spiel verlassen“, „Neues Spiel“ und „Lobby“ gibt es noch zwei Funktionalitäten, die genauer erläutert werden.

3.5.5 Zeitleiste

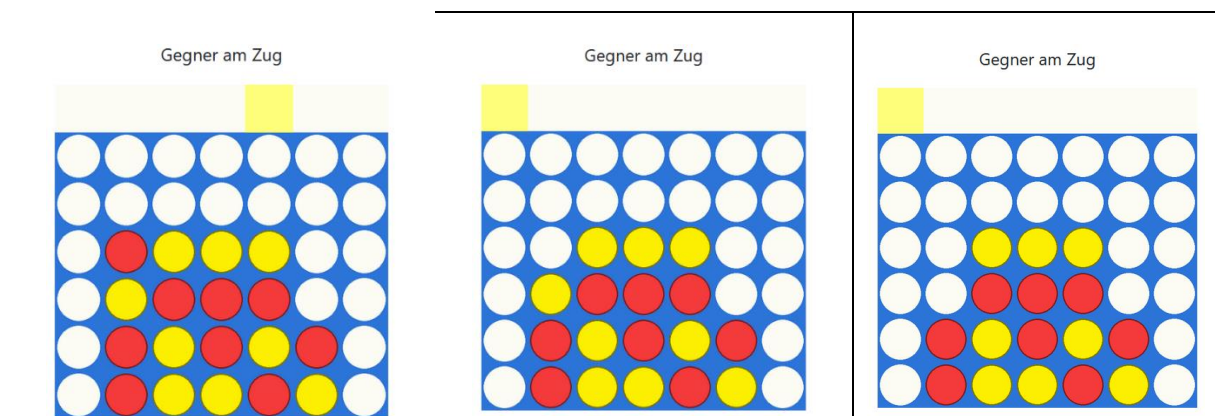


Abbildung 38: Zeitleiste - vorherig

Benutzerhandbuch

Am unteren Rand des Spielfeldes ist die Zeitleisten-Funktionalität eingeblendet worden. Durch Betätigen der Buttons navigiert man durch die einzelnen Züge des vergangenen Spiels und kann sich so jeden Zug und jede Aktion noch einmal in Ruhe anschauen.

3.5.6 Fehleranalyse

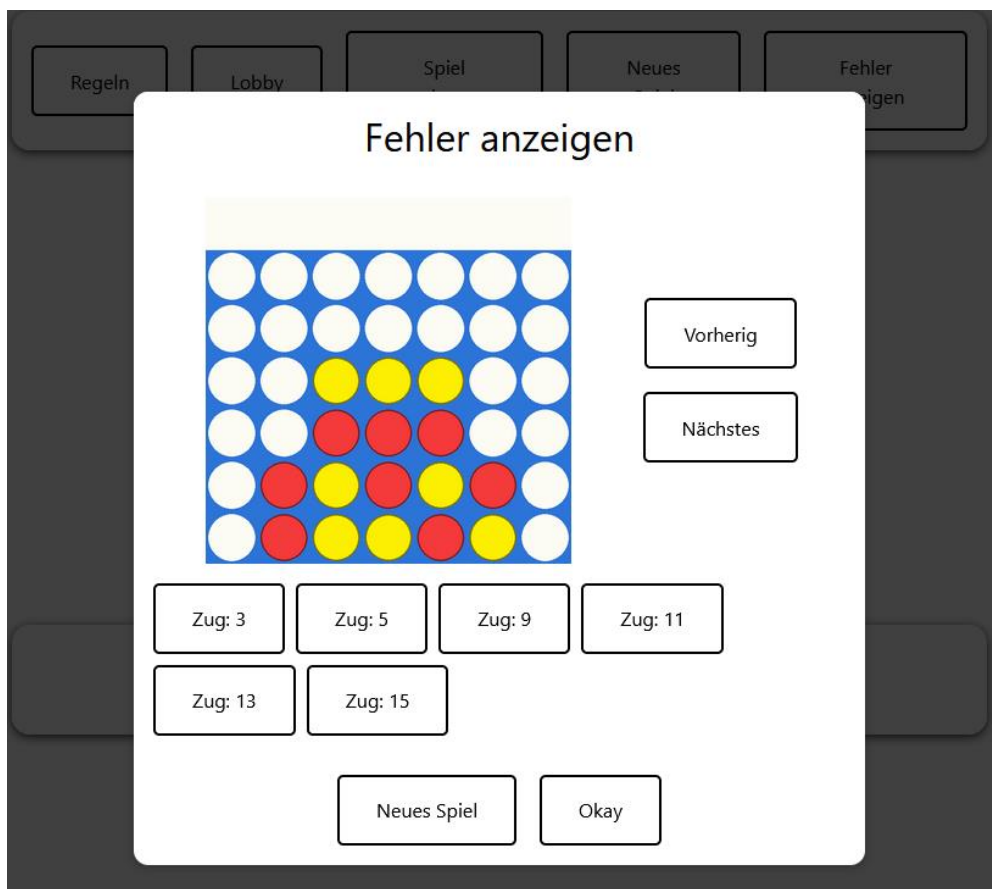


Abbildung 39: Fehleranalyse

Eine weitere wichtige Funktionalität stellt die Fehleranalyse dar. Betätigen Sie den Button am oberen Rand, so öffnet sich ein Pop-Up mit einer Auswertung von schlecht gewählten Zügen. Diese sind ebenfalls als Button realisiert, sodass ein Klick darauf das zugehörige Spielbrett mit dem Spielstand zum Zeitpunkt des schlechten Zuges anzeigt. Um sich die Situation noch einmal genauer anzuschauen kann hier auch wieder durch das Spiel iteriert werden mit „Nächstes“ und „Vorherig“.

Benutzerhandbuch

3.5.7 Mini-Fenster

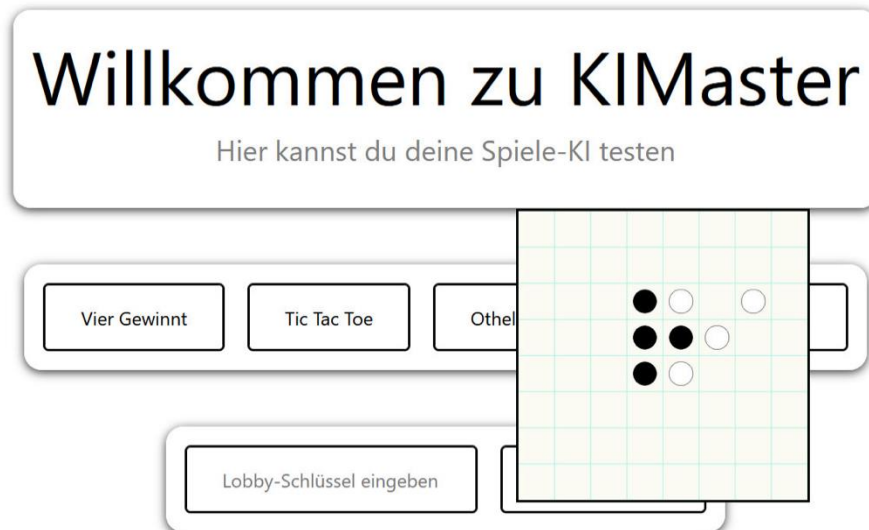


Abbildung 40: Mini-Fenster

Eine weitere spaßige Funktion stellt das Mini-Fenster da. Der Spieler hat die Möglichkeit aus der Spielseite rauszugehen, ohne dass das Spiel beendet werden muss. Man kann so durch die Plattform navigieren und jederzeit das Spiel fortsetzen.

Dieses Fenster taucht dabei automatisch auf, sobald man beispielsweise auf den Logo-Button der Webseite (oben links) klickt. Es lässt sich verschieben und ebenfalls zum Spiel zurückkehren durch einen Doppelklick auf das Fenster.

Diese Funktion ist ausschließlich plattformintern nutzbar und steht nicht mehr zur Verfügung, sobald man diese verlässt.

Benutzerhandbuch

4 Benutzeranleitung für die KIMaster-Schnittstelle

Die KIMaster-Schnittstelle ist ein Framework zur Verwaltung der Kommunikation zwischen einem Client und einem WebSocket-Server. Sie bietet Methoden zur Verbindung, zum Senden und Empfangen von Nachrichten sowie zur Verwaltung der Verbindungslebensdauer. Diese Schnittstelle wurde speziell für die Nutzung mit dem [FastAPI](#) Server des Projekts KIMaster entwickelt.

4.1 Überblick über die Schnittstelle (Python Beispiel)

Die KIMaster-Schnittstelle ermöglicht es, einfach und effizient mit einem WebSocket-Server zu kommunizieren. Es können verschiedene Befehle gesendet und Antworten erhalten werden in Echtzeit. Dabei ist zu beachten, dass ein Befehl auch mehrere in nicht geordnete Antworten mit sich bringen kann. Hier ist eine **Asynchronität** zu beachten.

Hauptfunktionen:

- Verbindung zu einem WebSocket-Server herstellen.
- Befehle an den Server senden.
- Nachrichten vom Server empfangen.

Es werden Klassen und Code-Abschnitte zur einfacheren Anbindung bereitgestellt. Dabei enthält die `KIMaster`-Klasse die grundlegende Infrastruktur für die WebSocket-Kommunikation, einschließlich Verbindungsmanagement, Befehlssendung und Nachrichtenerhalt. Die `Example`-Klasse erweitert diese Funktionalität, indem sie spezifische Befehle und Szenarien implementiert, die für die Anwendung erforderlich sind.

KIMaster-Klasse:

- `__init__`: Initialisiert die Klasse mit einer Liste von URIs.
- `connect`: Stellt eine Verbindung zu einem WebSocket-Server her.
- `send_cmd`: Sendet Befehle an den Server.
- `receive`: Empfängt Nachrichten vom Server.
- `close`: Schließt die Verbindung.
- `run`: Führt eine Coroutine aus.
- `handler`: Verwaltet das Senden und Empfangen von Nachrichten.
- `print_message`: Formatiert und druckt Nachrichten.
- `show`: Zeigt ein Bild aus einem Byte-Stream an.

Benutzerhandbuch

Example-Klasse:

- `__init__`: Initialisierung der Klasse.
- `input_thread`: Sammeln und Verarbeiten von Benutzereingaben.
- `send_handler`: Asynchrones Senden von Befehlen an den Server.
- `receive_handler`: Asynchrones Empfangen und Verarbeiten von Nachrichten vom

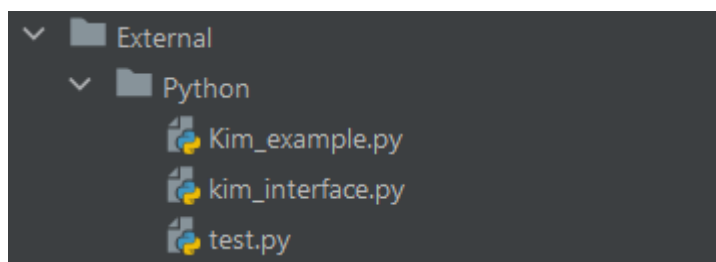


Abbildung 41: Ordnerstruktur Python Schnittstelle

4.2 Kommandos

Die Schnittstelle bietet verschiedene Befehle, die an den WebSocket-Server gesendet werden können. Diese Befehle werden als [JSON](#)-Nachrichten formatiert und enthalten spezifische Schlüssel und Daten, die vom Server verarbeitet werden. Alle Befehle sind in vollem Umfang mit ihren Möglichkeiten, Variationen und optionalen Parametern in der [commands.md](#) enthalten. Zu finden unter: Spezifikation/Dokumentation/Sonstiges/.

- Struktur einer zu sendenden Nachricht im JSON-Format:

```
{ "command": command,
  "command_key": commandy_key,
  "data": data }
```

- Struktur einer zu empfangenden Nachricht im JSON-Format:

```
{ "response_code": response_code,
  "response_msg": response_msg,
  "data": data }
```

- Beispiel für Kommandos:

- Lobby: Statusabfrage, Positionierung, Erstellung und Verlassen von Lobbys.

Benutzerhandbuch

- Play: Erstellen eines Spiels mit bestimmten Parametern (z.B. Spielname, Schwierigkeitsgrad, Modus)

4.3 Voraussetzungen

Bevor begonnen werden kann, muss sichergestellt werden, dass die folgenden Voraussetzungen erfüllt sind:

Voraussetzungen

- [Python 3.7](#) oder höher (erstellt wurde mit 3.11)
- Abhängigkeiten: asyncio, json, threading, queue, PIL, websockets

4.4 Einrichtung

1. **Installieren der notwendigen Bibliotheken:** Um die `KIMaster`-Schnittstelle zu verwenden, müssen die erforderlichen Python-Bibliotheken installiert sein. Ist dies nicht der Fall muss folgendes durchgeführt werden: Das Terminal öffnen und ausführen des folgenden Befehls zur Installation der notwendigen Bibliotheken:

```
pip install asyncio json websockets pillow
```

Abbildung 42: Install-Befehl Bibliotheken

2. **Verbindung zum WebSocket-Server:** Verwenden Sie die `KIMaster`-Klasse, um eine Verbindung zum Server herzustellen. Dies wird durch die Methode `connect()` der `KIMaster`-Klasse hergestellt. Diese Methode versucht, eine Verbindung zu einer der URIs herzustellen. Die erste erfolgreiche Verbindung wird genutzt und nicht weiter getestet. Mit der Liste können mehrere Verbindungen hinterlegt werden. Beispielsweise die localhost Adresse zum Testen und die offizielle Adresse auf der Deployt wird.

```
# List of URIs to connect to  
uri = ["wss://kimaster.mni.thm.de/ws", "ws://localhost:8010/ws"]
```

Abbildung 43: Zu verbindende URIs

Hier gut zu erkennen, die erste URI ist die der offiziellen KIMaster Adresse. Die Zweite ist die `localhost` Adresse, wenn das System bei einem Lokal in Docker gestartet wird.

Benutzerhandbuch

```

async def connect(self) -> None:
    """
    Try to connect to one of the URIs in the uri_pool.
    """
    for uri in self.uri_pool:
        print(f"Try to connect to URI: '{uri}'")
        try:
            # Attempt to establish a WebSocket connection
            self.connection = await connect(uri)
            print(f"Connected to URI: '{uri}'")
            break
        except InvalidURI:
            # Handle invalid URI exception
            print(f"URI: '{uri}' not reachable!")
  
```

Abbildung 44: connect()

3. **Einbindung der Schnittstelle:** Mit der handler-Methode, können zwei weitere Methoden verlinkt werden. Diese werden dann von der KIMaster-Klasse asynchron parallel ausgeführt.

```

async def main(self) -> None:
    """
    Main method to establish connection and start handlers.
    """
    await self.connect()

    # Start the input thread to handle user input
    input_thread = threading.Thread(target=self.input_thread, daemon=True)
    input_thread.start()

    # Link both handlers to asyncio tasks and run them
    await self.handler(self.send_handler, self.receive_handler)
  
```

Abbildung 45: main()

Benutzerhandbuch

4.5 Nutzung der Hauptfunktionen

Die KIMaster-Schnittstelle bietet mehrere Hauptfunktionen, die für die Kommunikation mit dem WebSocket-Server verwendet werden.

- **Befehle senden:** Die Methode `send_cmd()` wird verwendet, um Befehle an den WebSocket-Server zu senden. Sie nimmt den Befehl, den Befehlsschlüssel und optional zusätzliche Daten als Parameter entgegen.

```
async def send_cmd(self, command: str, command_key: str, data: dict | None = None) -> None:
    """
    Send a command to the connected WebSocket server.

    :param command: The command to send.
    :param command_key: The command key associated with the command.
    :param data: Optional additional data to send with the command.
    """
    if self.connection:
        # Prepare payload with command and command_key
        payload: dict = {"command": command, "command_key": command_key}
        if data is not None:
            # Add additional data if provided
            payload.update(data)
        # Send payload as JSON string
        await self.connection.send(json.dumps(payload))
```

Abbildung 46: `send_cmd()`

Benutzerhandbuch

- **Nachrichten empfangen:** Die Methode `receive()` wird verwendet, um Nachrichten vom WebSocket-Server zu empfangen. Sie kann Nachrichten im JSON-Format, als Zeichenkette oder als Byte-Stream zurückgeben.

```

async def receive(self) -> dict | str | bytes | None:
    """
    Receive a message from the WebSocket server.

    :return: The received message, either as a dictionary, string, or bytes.
    """
    if self.connection:
        message = None
        try:
            # Attempt to receive a message
            message = await self.connection.recv()
        except ConnectionClosedOK:
            return
        try:
            # Try to parse the message as JSON
            data = json.loads(message)
            return data
        except json.JSONDecodeError:
            # Handle binary data (e.g., PNG bytestream)
            return message
        except UnicodeDecodeError:
            # Handle non-UTF-8 encoded data
            return message

```

 Abbildung 47: `receive()`

- **Hauptprogramm:** Das Hauptprogramm initialisiert die `Example`-Klasse, startet die WebSocket-Verbindung und verwaltet die asynchronen Aufgaben für das Senden und Empfangen von Nachrichten. Dazu muss diese ausgeführt werden.

```

if __name__ == "__main__":
    # List of URIs to connect to
    uri = ["wss://kimaster.mni.thm.de/ws", "ws://localhost:8010/ws"]
    # Create an instance of the Example class with the URI list
    master = Example(uri_list=uri)
    # Run the main method
    master.run(master.main())

```

Abbildung 48: Hauptprogramm

Benutzerhandbuch

5 Häufige Probleme und Lösungen

5.1 Meine Verbindung bricht ständig ab, wenn ich etwas empfangen?

Da beim Empfangen nicht vorher bekannt ist, was das Format der empfangenen Daten ist, sollte vorerst überprüft werden, ob dieses richtig erkannt wurde.

```
print(type(payload), payload)
```

Abbildung 49: Formatüberprüfung

Bei `bytes` handelt es sich um ein PNG Bild, das interpretiert werden muss, bei `str` ein normaler String und bei `dict` um ein JSON-Objekt, von dem der Antwortcode, die -nachricht und der mögliche zusätzliche Anhang genutzt werden kann.

5.2 Ich kann mich nicht mit dem THM-Server verbinden?

Befinden Sie sich im THM-Netzwerk und können Sie sich auch per Webseite `https://kimaster.mni.thm.de` verbinden? Die URI von extern lautet: `wss://kimaster.mni.thm.de/ws`

Benutzerhandbuch

5.3 Ich erhalte keine Antworten trotz While True loop?

Da asynchrone Parallelität trotz handler-Methode nicht immer funktionieren, probieren Sie in den verlinkten Sende- oder Empfangs-Methoden ein `asyncio.sleep(0.1)` einzubauen, um dem System die Möglichkeit zu geben beide Routinen zu bearbeiten.

```
async def receive_handler(self) -> None:
    """
    Asynchronous handler to receive messages from the server.
    """
    while not self.exit:
        # Receive a message from the server
        message = await self.receive()
        if isinstance(message, dict):
            # Print the message if it is a dictionary
            self.print_message(message)
        if isinstance(message, bytes):
            # Show the image if the message is bytes
            self.show(message)
        # Close the connection when exiting
        await self.close()
```

Abbildung 50: `receive_handler()` ohne sleep

```
async def send_handler(self) -> None:
    """
    Asynchronous handler to send commands from the queue to the server.
    """
    while not self.exit:
        if not self.command_queue.empty():
            # Get the next command from the queue and send it
            command, command_key, data = self.command_queue.get()
            await self.send_cmd(command, command_key, data)
            # Sleep briefly to allow other tasks to run
            await asyncio.sleep(0.1)
        # Close the connection when exiting
        await self.close()
```

Abbildung 51: `send_handler()` mit sleep



Benutzerhandbuch

6 Literaturverzeichnis

1. Apple. *Safari*. [Online]. Verfügbar unter: <https://www.apple.com/de/safari/> [Zugriff am: 24. Juli 2024].
2. Microsoft. *Microsoft Edge*. [Online]. Verfügbar unter: <https://www.microsoft.com/de-de/edge?form=MA13FJ> [Zugriff am: 24. Juli 2024].
3. THM. *THM - Technische Hochschule Mittelhessen*. [Online]. Verfügbar unter: <https://www.thm.de/site/> [Zugriff am: 24. Juli 2024].
4. THM. *THM Campusnetz VPN - Cisco AnyConnect*. [Online]. Verfügbar unter: <https://www.thm.de/its/campusnetz/vpn/ciscoanyconnect.html> [Zugriff am: 24. Juli 2024].
5. Google. *Google Chrome*. [Online]. Verfügbar unter: https://www.google.com/intl/de_de/chrome/ [Zugriff am: 24. Juli 2024].
6. Mozilla. *Firefox*. [Online]. Verfügbar unter: <https://www.mozilla.org/de/firefox/> [Zugriff am: 24. Juli 2024].
7. Mozilla Developer Network. *WebSockets API - MDN Web Docs*. [Online]. Verfügbar unter: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API [Zugriff am: 24. Juli 2024].
8. Oracle. *Java*. [Online]. Verfügbar unter: <https://www.java.com/de/> [Zugriff am: 24. Juli 2024].
9. Microsoft. *C#-Lehrgang - Microsoft Learn*. [Online]. Verfügbar unter: <https://learn.microsoft.com/de-de/dotnet/csharp/tour-of-csharp/> [Zugriff am: 24. Juli 2024].
10. Rust Project Developers. *Rust*. [Online]. Verfügbar unter: <https://www.rust-lang.org/> [Zugriff am: 24. Juli 2024].
11. Kotlin. *Kotlin*. [Online]. Verfügbar unter: <https://kotlinlang.org/> [Zugriff am: 24. Juli 2024].
12. Python Software Foundation. *Python*. [Online]. Verfügbar unter: <https://www.python.org/> [Zugriff am: 24. Juli 2024].
13. THM. *Impressum der THM*. [Online]. Verfügbar unter: <https://www.thm.de/site/impressum.html> [Zugriff am: 24. Juli 2024].
14. Tiangolo. *FastAPI*. [Online]. Verfügbar unter: <https://fastapi.tiangolo.com/> [Zugriff am: 24. Juli 2024].
15. JSON. *JSON - JSON*. [Online]. Verfügbar unter: <https://www.json.org/json-de.html> [Zugriff am: 24. Juli 2024].
16. Python Software Foundation. *Python Downloads*. [Online]. Verfügbar unter: <https://www.python.org/downloads/> [Zugriff am: 24. Juli 2024].
17. 12ghostrider21. *KIMaster - Dokumentation*. [Online]. Verfügbar unter: <https://github.com/12ghostrider21/KIMaster/blob/main/Spezifikation/Dokumentation/sonstiges/commands.md> [Zugriff am: 24. Juli 2024].