Benutzerhandbuch

Softwaretechnik-Projekt SoSe2024

Thema

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 | 24.07.2024 | 3 | Schreiben | Omar Karkotli |

|  |  |  |
| --- | --- | --- |
| Herausgeber | [Technische Hochschule Mittelhessen – FB06 Mathematik, Naturwissenschaften und Informatik](https://www.thm.de/mni/) | |
| Dateiname | Benutzerhandbuch\_ver\_0.5 | |
| Dokumentenbezeichnung | Technische Dokumentation: Plattform zum Vergleich von Spiele-KIs | |
| Version | 0.5 | |
| Stand | Mittwoch, 24. Juli 2024 | |
| Status | In Bearbeitung | |
| 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](mailto:justine.buss@mni.thm.de) |
|  | Thorben Jones | [thorben.jones@mni.thm.de](mailto:thorben.jones@mni.thm.de) |
| Kurzinfo | „Technische Hochschule Mittelhessen Softwaretechnik-Projekt. Technische Dokumentation.“ | |

Inhaltsverzeichnis

[1 Einführung 7](#_Toc172680327)

[1.1 Kurzübersicht der Software 7](#_Toc172680328)

[1.2 Zielgruppen 7](#_Toc172680329)

[2 Einrichtung 8](#_Toc172680330)

[2.1 Systemanforderungen 8](#_Toc172680331)

[3 Benutzeranleitung für die Webseite 10](#_Toc172680332)

[3.1 Webseite aufrufen 10](#_Toc172680333)

[3.2 Startseite 11](#_Toc172680334)

[3.2.1 Navigationsleiste 11](#_Toc172680335)

[3.2.2 Begrüßung 12](#_Toc172680336)

[3.2.3 Spielauswahl 12](#_Toc172680337)

[3.2.4 Eingabe des Lobbyschlüssels 13](#_Toc172680338)

[3.2.5 Footer 13](#_Toc172680339)

[3.3 Datenschutz und Impressum 15](#_Toc172680340)

[3.4 Warteraum 16](#_Toc172680341)

[3.5 Lobby-Seite 17](#_Toc172680342)

[3.5.1 Spielekonfiguration 17](#_Toc172680343)

[3.5.2 Status und Lobby-Schlüssel 17](#_Toc172680344)

[3.6 Spielseite 18](#_Toc172680345)

[3.6.1 Regeln 18](#_Toc172680346)

[3.6.2 Spielfunktionen 18](#_Toc172680347)

[3.6.3 Spielbrett 18](#_Toc172680348)

[3.6.4 Fertiges Spiel 18](#_Toc172680349)

[3.6.5 Fehleranalyse 18](#_Toc172680350)

[3.6.6 Mini-Fenster 18](#_Toc172680351)

[4 Benutzeranleitung für die KIMaster-Schnittstelle 19](#_Toc172680352)

[4.1 Überblick über die Schnittstelle (Python Beispiel) 19](#_Toc172680353)

[4.2 Kommandos 20](#_Toc172680354)

[4.3 Voraussetzungen 21](#_Toc172680355)

[4.4 Einrichtung 21](#_Toc172680356)

[4.5 Nutzung der Hauptfunktionen 23](#_Toc172680357)

[5 Häufige Probleme und Lösungen 25](#_Toc172680358)

[5.1 Meine Verbindung bricht ständig ab, wenn ich etwas empfange? 25](#_Toc172680359)

[5.2 Ich kann mich nicht mit dem THM-Server verbinden? 25](#_Toc172680360)

[5.3 Ich erhalte keine Antworten trotz While True loop: 25](#_Toc172680361)

Abbildungssverzeichnis

[Abbildung 1: Verbindungsherstellung THM VPN 9](#_Toc172680372)

[Abbildung 2: Verbindung hergestellt THM VPN 9](#_Toc172680373)

[Abbildung 3: URL-Eingabe in Suchleiste des Browsers 10](#_Toc172680374)

[Abbildung 4: Startseite 11](#_Toc172680375)

[Abbildung 5: Navigationsleiste 11](#_Toc172680376)

[Abbildung 6: Begrüßung 12](#_Toc172680377)

[Abbildung 7: Spieleauswahl 12](#_Toc172680378)

[Abbildung 8: Spieleauswahl Hovering 13](#_Toc172680379)

[Abbildung 9: Lobby beitreten 13](#_Toc172680380)

[Abbildung 10: Lobby beitreten - Fehleingabe 13](#_Toc172680381)

[Abbildung 11: Footer 13](#_Toc172680382)

[Abbildung 12: Ordnerstruktur Python Schnittstelle 20](#_Toc172680383)

[Abbildung 13: Install-Befehl Bibliotheken 21](#_Toc172680384)

[Abbildung 14: Zu verbindende URIs 21](#_Toc172680385)

[Abbildung 15: connect() 22](#_Toc172680386)

[Abbildung 16: main() 22](#_Toc172680387)

[Abbildung 17: send\_cmd() 23](#_Toc172680388)

[Abbildung 18: receive() 24](#_Toc172680389)

[Abbildung 19: Hauptprogramm 24](#_Toc172680390)

[Abbildung 20: Formatüberprüfung 25](#_Toc172680391)

[Abbildung 21: receive\_handler() ohne sleep 26](#_Toc172680392)

[Abbildung 22: send\_handler() mit sleep 26](#_Toc172680393)

# 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.

# 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 Chrome, Firefox, Safari oder Edge ist erforderlich.
* **Netzwerkverbindung**: Der Zugriff muss innerhalb des Netzwerks der THM 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 THM erfolgen, was entweder direkt vor Ort oder durch die Nutzung eines VPN-Dienstes der THM möglich ist.
* **Internetverbindung**: Eine stabile Internetverbindung ist erforderlich.
* **Programmierumgebung**: 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 Programmierumgebung 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 Programmierumgebung installiert sind.
* Für die Nutzung des VPN-Dienstes der THM folgen Sie bitte den Anweisungen auf der offiziellen Webseite der THM oder kontaktieren Sie den IT-Support <https://www.thm.de/its/campusnetz/vpn/ciscoanyconnect.html>.

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 1: Verbindungsherstellung THM VPN

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte Beschreibung

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-Repsoitory](https://github.com/12ghostrider21/KIMaster/tree/main/External/Python) ein Verbindungsbeispiel mit Python an.

# 3 Benutzeranleitung für die Webseite

## 3.1 Webseite aufrufen

1. Stellen Sie sicher, dass Sie die [Systemanforderungen](#_2.1_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

1. Bestätigen Sie Ihre Eingabe durch Drücken der Eingabetaste.

## 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

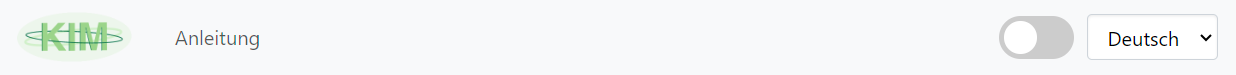


Abbildung 5: Navigationsleiste

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](#_4_Benutzeranleitung_für). Dieser führt zu Benutzeranleitung für die Schnittstelle (API).

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.)

### 3.2.2 Begrüßung

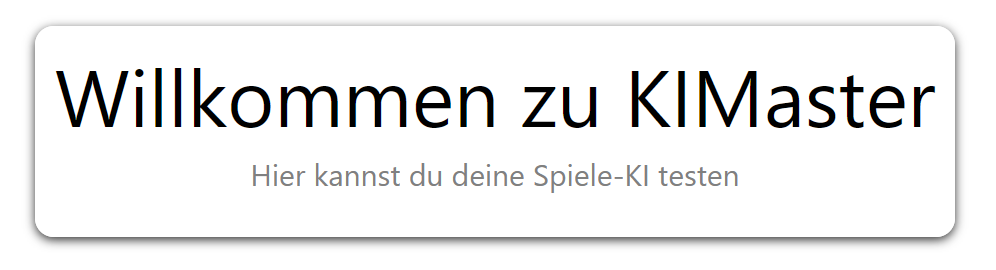


Abbildung 6: Begrüßung

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

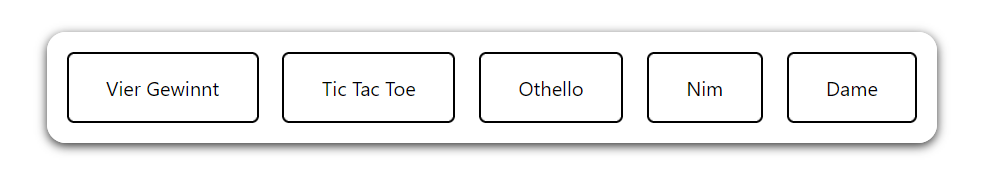


Abbildung 7: Spieleauswahl

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.

Durch Klicken auf einen dieser Buttons gelangt der Benutzer in die jeweilige Spielelobby.

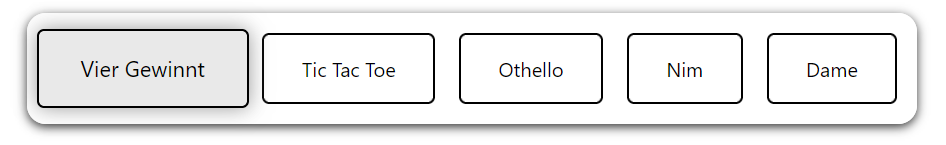


Abbildung 8: Spieleauswahl Hovering

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.

### 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.

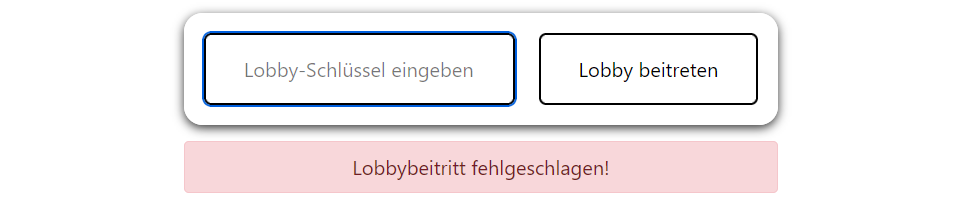


Abbildung 10: Lobby beitreten - Fehleingabe

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.

### 3.2.5 Footer



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 ist. Außerdem finden Sie hier die Datenschutzbestimmungen und das Impressum.

## 3.3 Datenschutz und Impressum

## 3.4 Warteraum

## 3.5 Lobby-Seite

### 3.5.1 Spielekonfiguration

### 3.5.2 Status und Lobby-Schlüssel

## 3.6 Spielseite

### 3.6.1 Regeln

### 3.6.2 Spielfunktionen

### 3.6.3 Spielbrett

### 3.6.4 Fertiges Spiel

### 3.6.5 Fehleranalyse

### 3.6.6 Mini-Fenster

# 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 FastAPIServer 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 dieKIMaster-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.

**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

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 12: 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](https://github.com/12ghostrider21/KIMaster/blob/main/Spezifikation/Dokumentation/Sonstiges/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.
  + 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:

Ein Bild, das Text, Schrift, Screenshot, Grafiken enthält.

Automatisch generierte Beschreibung

Abbildung 13: Install-Befehl Bibliotheken

1. **Verbindung zum WebSocket-Server**: Verwenden Sie die KIMaster-Klasse, um eine Verbindung zum Server herzustellen. Dies wird durch die Methodeconnect() 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.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 14: 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.

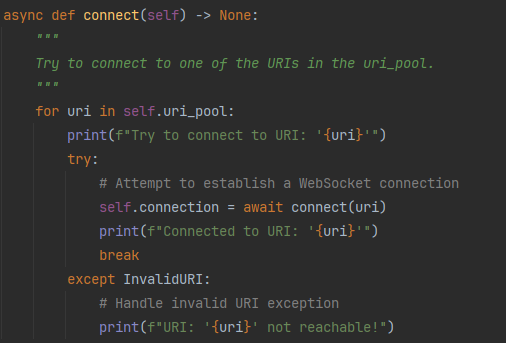


Abbildung 15: connect()

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

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 16: main()

## 4.5 Nutzung der Hauptfunktionen

DieKIMaster-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.

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

Abbildung 17: send\_cmd()

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

Ein Bild, das Text, Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung 18: 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 dieses ausgeführt werden.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 19: Hauptprogramm

# 5 Häufige Probleme und Lösungen

## 5.1 Meine Verbindung bricht ständig ab, wenn ich etwas empfange?

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



Abbildung 20: 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

## 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.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 21: receive\_handler() ohne sleep

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 22: send\_handler() mit sleep