Plauderkiste

2025

Erzeugt von Doxygen 1.14.0

Kapitel 1

Plauderkiste

1.1 Plauderkiste – Peer-to-Peer Chat für lokale Netzwerke

1.1.1 Team

- Mahir Ahmad
- Sena Akpolad
- Onur Ücelehan
- Meriam Lakhrissi
- Najiba Sulaimankhel

1.1.2 Projektbeschreibung

Plauderkiste ist eine eigenständige Peer-to-Peer-Chatsoftware, die speziell für lokale Netzwerke (z.B. Uni, WG, Büro) entwickelt wurde. Plauderkiste ermöglicht es, direkt und ohne zentralen Server mit anderen Nutzern zu chatten und Bilder zu teilen. Das System ist robust, einfach zu bedienen und benötigt keinerlei Abhängigkeiten.

1.1.3 Hauptfunktionen

- **Direkte Textnachrichten** zwischen allen verbundenen Nutzern
- **Bildversand** mit Dateigrößenanzeige und Speicherung in Benutzerordnern
- **Automatisches Discovery**: Neue Nutzer werden im Netzwerk ohne zentrale Verwaltung gefunden
- **Statusanzeige** und **Nicht-stören-Modus**
- **Speicherung** und **Anzeige** des gesamten Chatverlaufs
- **Konfigurations-Reload** im laufenden Betrieb möglich

1.1.4 Technischer Ansatz

- *Peer-to-Peer Architektur*: Jeder Nutzer ist gleichberechtigt, es gibt keinen Server
- *Kommunikation*: UDP-Broadcast für Discovery, TCP für Messaging/Bilder
- *Prozessmodell*: Jeder Hauptteil (Discovery, Messaging, UI) läuft als separater Prozess
- *Prozesssynchronisation*: Über Python multiprocessing.Manager und Queues
- *Modular*: Die Software ist modular und rein funktionsbasiert geschrieben, für maximale Nachvollziehbarkeit

1.1.5 Bedienung

- Gestartet wird mit `python start.py config1.toml` (bzw. mit einer anderen Konfiguration für weitere Nutzer)
- Im Chat stehen intuitive Befehle wie `msg`, `img`, `who`, `contacts`, `status`, `help` etc. bereit
- Bei Beenden wird alles sauber geschlossen und der Verlauf kann gespeichert werden

1.1.6 Architekturdiagramm

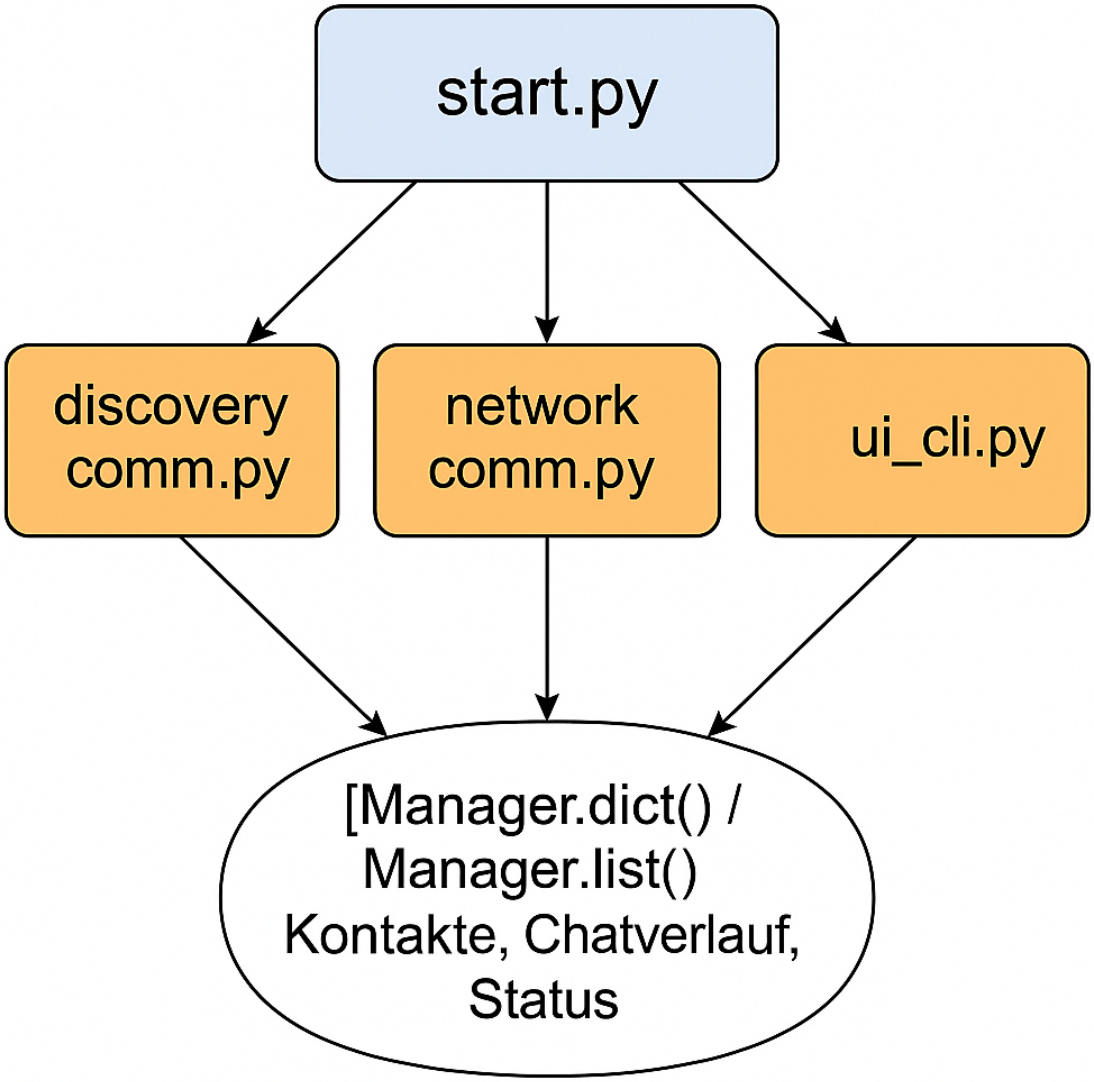

Abbildung 1.1 Architekturdiagramm

1.1.7 CLI-Screenshots

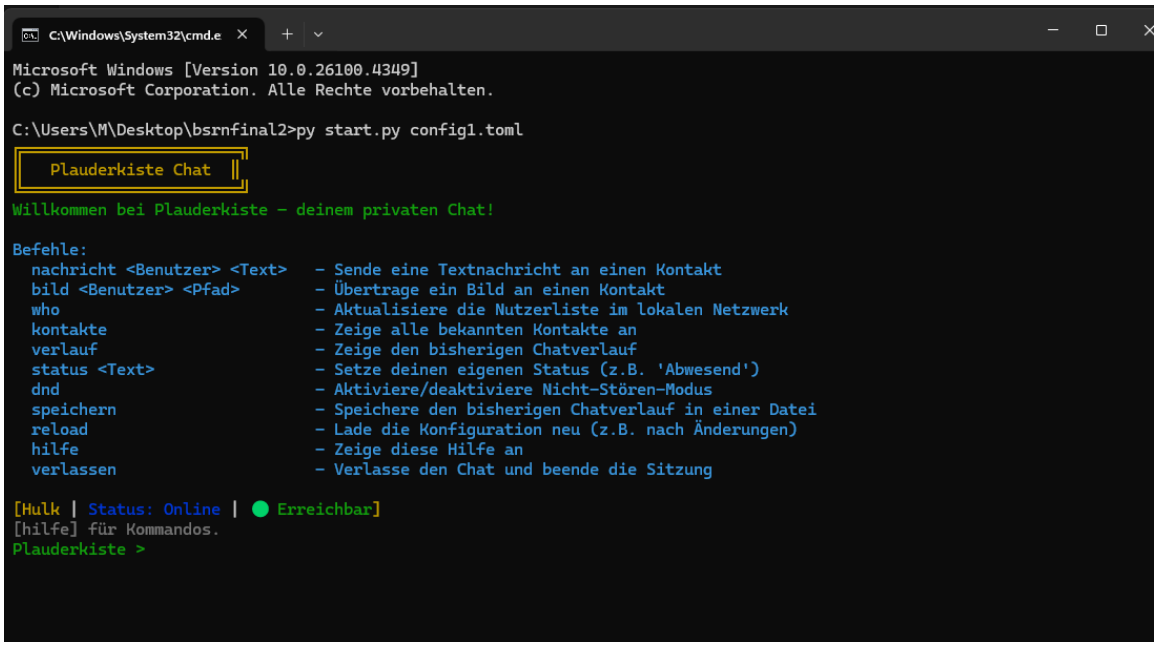

Abbildung 1.2 CLI Screenshot 1

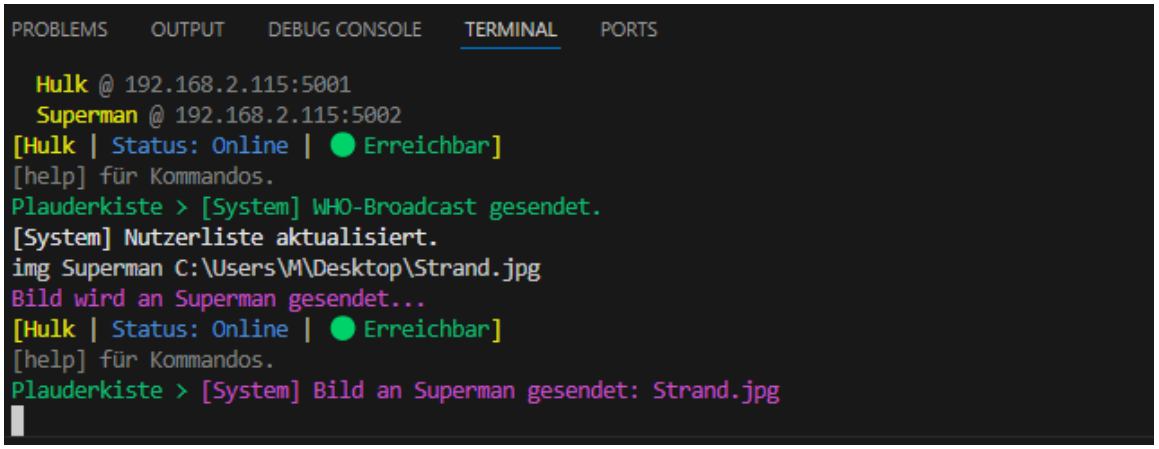

Abbildung 1.3 CLI Screenshot 2

1.1.8 Detaillierter Aufbau

Plauderkiste besteht aus den folgenden Modulen:

- **start.py:** Initialisiert alles, startet Prozesse, hält alles zusammen
- **discovery_comm.py:** Findet andere Nutzer im Netzwerk (UDP-Broadcast)
- **network_comm.py:** Übernimmt das Versenden und Empfangen von Nachrichten/Bildern (TCP)
- **ui_cli.py:** Das Kommandozeilen-Interface für Eingabe, Status und Darstellung
- **data_manager.py:** Speichern und Nachladen des Verlaufs und der Konfiguration

1.1.9 Wichtige Designentscheidungen und Herausforderungen

- *Synchronisierung ohne Datenverlust*: Durch periodisches "WHO"/"SEEN"-Verfahren und klare Fehlerbehandlung
- *Unabhängige Prozesse*: Prozesse laufen unabhängig, Datenaustausch nur über Queues und Manager-↩ Objekte
- *Modularität*: Jede Funktionalität (Discovery, Netzwerk, UI, Daten) ist ein eigenes Modul – verständlich für neue Entwickler
- *Benutzerfreundlichkeit*: Einfache, farbige CLI mit klaren Kommandos und Fehlerausgaben

1.1.10 Bekannte Probleme & Hinweise

1.1.10.1 Windows-spezifische Einschränkungen

- **UDP-Broadcast auf localhost**: Unter Windows funktioniert UDP-Broadcast mit mehreren Instanzen auf derselben Maschine nur eingeschränkt. Wenn Sie mehrere Clients auf einem Gerät starten, kann es sein, dass nicht alle sich gegenseitig sehen. Im realen Netzwerk (mehrere Geräte) oder unter Linux/Mac gibt es dieses Problem nicht.
- **Firewall blockiert Kommunikation**: Windows-Firewall (oder Antivirus) kann sowohl UDP- als auch TCP-↩ Verbindungen blockieren. Geben Sie im Zweifel dem Python-Interpreter (z.B. python.exe) Zugriffsrechte für das lokale Netzwerk.
- **Port bereits belegt**: Falls beim Starten die Fehlermeldung `OSError: [WinError 10048]` erscheint, ist der gewünschte Port schon in Benutzung. Ändern Sie in der jeweiligen Konfigurationsdatei (`configX.↩ toml`) den Wert für `port` und `whoisport` auf einen anderen, noch freien Port.

1.1.10.2 Allgemeine Hinweise

- **Images werden nicht immer automatisch angezeigt**: Das Öffnen empfangener Bilder funktioniert nur, wenn das Betriebssystem das zugehörige Standardprogramm richtig hinterlegt hat.
- **Verlorene Nachrichten bei Netzwerkproblemen**: Da Discovery über UDP läuft, kann es in seltenen Fällen passieren, dass Nachrichten zur Nutzererkennung (WHO/SEEN) verloren gehen und temporär nicht alle Nutzer angezeigt werden. Ein erneutes Ausführen von `who` löst das Problem meist.
- **Erreichbarkeit im WLAN/Netzwerk**: In manchen WLANs blockiert der Router lokale Broadcasts. Prüfe ggf. deine Router-Einstellungen oder wechsel in ein anderes Netzwerk.

Diese Einschränkungen liegen **außerhalb des Einflusses der Software** und betreffen grundlegende Eigenschaften von Betriebssystemen, Netzwerken oder Geräteeinstellungen.

Autor

Team Plauderkiste: Mahir Ahmad, Sena Akpolad, Onur Ücelehan, Meriam Lakhrissi, Najiba Sulaimankhel

Datum

2025

Kapitel 2

Verzeichnis der Namensbereiche

2.1 Liste aller Namensbereiche

Liste aller Namensbereiche mit Kurzbeschreibung:

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

Kapitel 4

Dokumentation der Namensbereiche

4.1 data_manager-Namensbereichsreferenz

Funktionen

- save_history (chat_history, filename="verlauf.txt")

  Speichert den aktuellen Chatverlauf in eine Textdatei.
- add_message (chat_history, msg)

  Fügt eine neue Nachricht dem Chatverlauf hinzu.
- reload_config (config_path)

  Lädt die Konfiguration aus einer TOML-Datei neu.

4.1.1 Ausführliche Beschreibung

```
@file data_manager.py
@brief Modul zur Verwaltung von Chatverlauf und Konfiguration im Peer-to-Peer-Chat "Plauderkiste".

Dieses Modul übernimmt das Speichern und Nachladen von Chatverläufen sowie das Nachladen der
Konfiguration zur Laufzeit. Alle Funktionen sind als Einzeloperationen ohne Klassen implementiert.

Das Datenmanagement sorgt dafür, dass Nutzer jederzeit ihren bisherigen Verlauf sichern und
wiederherstellen können. Außerdem kann die Konfiguration (z.B. Nutzername oder Port) zur Laufzeit neu geladen

@author Mahir Ahmad, Sena Akpolad, Onur Ücelehan, Meriam Lakhrissi, Najiba Sulaimankhel
@date 2025
```

4.1.2 Dokumentation der Funktionen

4.1.2.1 add_message()

```
data_manager.add_message (
              chat_history,
              msg)
```

Fügt eine neue Nachricht dem Chatverlauf hinzu.

```
Hängt eine neue Nachricht an die Chat-Historie an.

@param chat_history: Chatverlauf als Manager Liste
@param msg: Text der neuen Nachricht
@return: None
```

4.1.2.2 reload_config()

```
data_manager.reload_config (
              config_path)
```

Lädt die Konfiguration aus einer TOML-Datei neu.

```
Lädt alle Einstellungen (z.B. Nickname, Port, Bildordner) aus der angegebenen Konfigurationsdatei nach.

@param config_path: Pfad zur TOML-Konfigurationsdatei
@return: config: Dict mit neuen Konfigurationseinstellungen
```

4.1.2.3 save_history()

```
data_manager.save_history (
              chat_history,
              filename = "verlauf.txt")
```

Speichert den aktuellen Chatverlauf in eine Textdatei.

```
Speichert alle bisher gesendeten und empfangenen Nachrichten in eine Textdatei.

@param chat_history: Liste aller bisherigen Chat-Nachrichten
@param filename: Name der Zieldatei (Standard: "verlauf.txt")
@return: None
```

4.2 discovery_comm-Namensbereichsreferenz

Funktionen

- discovery_service (ui_queue, disc_queue, config, kontaktbuch)

  Discovery-Service: Findet andere Nutzer im lokalen Netzwerk und verwaltet die Kontaktliste.

4.2.1 Ausführliche Beschreibung

```
@file discovery_comm.py
@brief Verantwortlich für das Auffinden und die Verwaltung von Chat-Teilnehmern im lokalen Netzwerk.

Dieses Modul übernimmt die Discovery-Funktion im Peer-to-Peer-Netzwerk.
Nutzer werden über UDP-Broadcast gefunden und bekannt gemacht (JOIN, WHO, LEAVE).
Die Nutzerliste wird regelmäßig synchronisiert, damit immer die aktuellen Teilnehmer angezeigt werden.

Optimiert für Tests auf localhost (mehrere Instanzen auf einem PC).
@author Mahir Ahmad, Sena Akpolad, Onur Ücelehan, Meriam Lakhrissi, Najiba Sulaimankhel
@date 2025
```

4.2.2 Dokumentation der Funktionen

4.2.2.1 discovery_service()

```
discovery_comm.discovery_service (
              ui_queue,
              disc_queue,
              config,
              kontaktbuch)
```

Discovery-Service: Findet andere Nutzer im lokalen Netzwerk und verwaltet die Kontaktliste.

```
Erkennt andere Nutzer im lokalen Netz per UDP-Broadcast und synchronisiert die Kontaktliste.
Wartet nach WHO kurz auf alle SEEN-Antworten und merged sie.

@param ui_queue: Queue für System-/Statusnachrichten
@param disc_queue: Queue für eingehende Discovery-Kommandos aus dem CLI
@param config: Dictionary mit Konfiguration (Nickname, Ports, etc.)
@param kontaktbuch: Manager Dictionary zur Verwaltung aller bekannten Kontakte
@return: None
```

4.3 network_comm-Namensbereichsreferenz

Funktionen

- open_image (filepath)

- network_service (ui_queue, net_queue, config, peers)

  TCP-Server & Client für Nachrichten- und Bildübertragung.

4.3.1 Ausführliche Beschreibung

```
@file network_comm.py
@brief Modul für den Austausch von Nachrichten und Bildern im Peer-to-Peer-Chat "Plauderkiste".

Dieses Modul kümmert sich um das Senden und Empfangen von Textnachrichten und Bilddateien zwischen den Teilneh
Die Kommunikation läuft über TCP - jedes Plauderkiste-Programm ist zugleich Client und Server.
Nachrichten und Bilder werden von diesem Modul empfangen, verarbeitet und an das User-Interface weitergegeben.

Es werden keine Klassen verwendet. Die Anbindung an das CLI und die Discovery-Komponente erfolgt über Queues.

@author Mahir Ahmad, Sena Akpolad, Onur Ücelehan, Meriam Lakhrissi, Najiba Sulaimankhel
@date 2025
```

4.3.2 Dokumentation der Funktionen

4.3.2.1 network_service()

```
network_comm.network_service (
              ui_queue,
              net_queue,
              config,
              peers)
```

TCP-Server & Client für Nachrichten- und Bildübertragung.

```
Startet den TCP-Server und verarbeitet sowohl eingehende als auch ausgehende Nachrichten und Bilder.

@param ui_queue: Queue für Status- und Chatnachrichten an die Benutzeroberfläche (multiprocessing.Queue)
@param net_queue: Queue für ausgehende Befehle/Sendewünsche aus dem CLI (multiprocessing.Queue)
@param config: Konfigurationsdaten (dict) mit Nutzername, Port, Bildordner etc.
@param peers: Kontaktbuch (Manager.dict), aktuelle Peer-Liste mit IP und Port
@return: None
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

4.4 start-Namensbereichsreferenz

Funktionen

- main ()

  Hauptfunktion: Initialisiert Konfiguration und startet alle Komponenten.

4.4.1 Ausführliche Beschreibung

```
@file start.py
@brief Startet alle Prozesse und das CLI (oder optional GUI) für den Peer-to-Peer-Chat "Plauderkiste".

Dieses Modul ist der Einstiegspunkt der Anwendung. Es kümmert sich um das Laden der Konfiguration,
startet alle benötigten Prozesse (Discovery, Netzwerk, UI) und verwaltet die Kommunikation zwischen ihnen.
Es kann über Kommandozeilenargumente die zu verwendende Konfigurationsdatei wählen.

Die einzelnen Komponenten kommunizieren über multiprocessing.Queue und teilen Daten über Manager-Objekte.

@author Mahir Ahmad, Sena Akpolad, Onur Ücelehan, Meriam Lakhrissi, Najiba Sulaimankhels
@date 2025
```

4.4.2 Dokumentation der Funktionen

4.4.2.1 main()

```
start.main ()
```

Hauptfunktion: Initialisiert Konfiguration und startet alle Komponenten.

```
Initialisiert das Plauderkiste-System: Lädt die Konfiguration, erstellt Datenstrukturen,
startet Discovery-, Netzwerk- und UI-Prozess.

Kann die Konfigurationsdatei als Argument entgegennehmen, ansonsten wird 'config1.toml' verwendet.
Am Ende werden die Hintergrundprozesse sauber beendet.

@return: None
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft: Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

4.5 ui_cli-Namensbereichsreferenz

Funktionen

- print_help ()

  Gibt einen Hilfetext für alle verfügbaren Kommandos aus.
- watcher (ui_queue)

  Gibt alle neuen Nachrichten aus der UI-Queue farbig auf dem Terminal aus.
- print_status (handle, status, dnd)

  Zeigt aktuelle Nutzerinformationen (Nickname, Status, DND) im Terminal an.
- start_cli (ui_queue, discovery_queue, network_queue, config, contacts, chat_history, dnd, status)

  Hauptfunktion für die farbige Kommandozeilensteuerung von Plauderkiste.

Variablen

- autoreset

4.5.1 Ausführliche Beschreibung

```
@file ui_cli.py
@brief Farbige, interaktive Kommandozeilen-Benutzeroberfläche für den Peer-to-Peer-Chat "Plauderkiste".

Dieses Modul bietet eine textbasierte, farbige Chat-Oberfläche. Nutzer können Nachrichten senden,
Bilder übertragen, Kontakte verwalten und ihren Status setzen. Die Oberfläche zeigt alle wichtigen
Informationen wie Status, DND (Nicht-stören), Kontakte und Chatverlauf an und gibt nützliche Hilfetexte aus.

Das Interface kommuniziert über Queues mit den Discovery- und Netzwerkmodulen.
Für die Farbdarstellung wird das Paket colorama verwendet.

Das CLI ist so gestaltet, dass es ohne Klassen auskommt und auch für technisch weniger erfahrene
Nutzer leicht zu bedienen ist.

@author Mahir Ahmad, Sena Akpolad, Onur Ücelehan, Meriam Lakhrissi, Najiba Sulaimankhel
@date 2025
```

4.5.2 Dokumentation der Funktionen

4.5.2.1 print_help()

```
ui_cli.print_help ()
```

Gibt einen Hilfetext für alle verfügbaren Kommandos aus.

```
Gibt eine Übersicht über alle unterstützten Chat-Befehle und deren Anwendung aus.
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

4.5.2.2 print_status()

```
ui_cli.print_status (
              handle,
              status,
              dnd)
```

Zeigt aktuelle Nutzerinformationen (Nickname, Status, DND) im Terminal an.

```
Zeigt in einer farbigen Zeile an, unter welchem Nickname der Nutzer aktiv ist,
welchen Status er gesetzt hat und ob der Nicht-stören-Modus aktiv ist.

@param handle: Aktueller Nutzername
@param status: Aktueller Statuswert
@param dnd: DND-Modus (Nicht stören)
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

4.5.2.3 start_cli()

```
ui_cli.start_cli (
              ui_queue,
              discovery_queue,
              network_queue,
              config,
              contacts,
              chat_history,
              dnd,
              status)
```

Hauptfunktion für die farbige Kommandozeilensteuerung von Plauderkiste.

```
Startet die textbasierte, farbige Oberfläche des Plauderkiste-Chats.
Nutzer können Kommandos eingeben, Nachrichten versenden und Systeminformationen einsehen.

Die Funktion läuft in einer Schleife bis der Nutzer den Chat verlässt.

@param ui_queue: Queue für neue System- und Chatnachrichten
@param discovery_queue: Queue für Kommandos an die Discovery-Komponente
@param network_queue: Queue für Kommandos an das Netzwerkmodul
@param config: Konfiguration (dict), enthält Nutzername, Port, u.a.
@param contacts: Gemeinsame Kontaktliste
@param chat_history: Gemeinsame Chat-Historie
@param dnd: Status für Nicht-stören
@param status: Freitext-Statusanzeige
@return: None
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

4.5.2.4 watcher()

```
ui_cli.watcher (
              ui_queue)
```

Gibt alle neuen Nachrichten aus der UI-Queue farbig auf dem Terminal aus.

```
Gibt alle eingegangenen System- oder Chatnachrichten, die sich in der Queue befinden, farbig aus.

@param ui_queue: Queue mit neuen Nachrichten
```

4.5.3 Variablen-Dokumentation

4.5.3.1 autoreset

```
ui_cli.autoreset
```

Kapitel 5

Datei-Dokumentation

5.1 architektur.png-Dateireferenz

5.2 data_manager.py-Dateireferenz

Namensbereiche

- namespace data_manager

Funktionen

- data_manager.save_history (chat_history, filename="verlauf.txt")

  Speichert den aktuellen Chatverlauf in eine Textdatei.
- data_manager.add_message (chat_history, msg)

  Fügt eine neue Nachricht dem Chatverlauf hinzu.
- data_manager.reload_config (config_path)

  Lädt die Konfiguration aus einer TOML-Datei neu.

5.3 discovery_comm.py-Dateireferenz

Namensbereiche

- namespace discovery_comm

Funktionen

- discovery_comm.discovery_service (ui_queue, disc_queue, config, kontaktbuch)

  Discovery-Service: Findet andere Nutzer im lokalen Netzwerk und verwaltet die Kontaktliste.

5.4 Strand.jpg-Dateireferenz

5.5 mainpage.dox-Dateireferenz

5.6 network_comm.py-Dateireferenz

Namensbereiche

- namespace network_comm

Funktionen

- network_comm.open_image (filepath)
- network_comm.network_service (ui_queue, net_queue, config, peers)

  TCP-Server & Client für Nachrichten- und Bildübertragung.

5.7 screenshot_cli.png-Dateireferenz

5.8 screenshot_cli_2.png-Dateireferenz

5.9 start.py-Dateireferenz

Namensbereiche

- namespace start

Funktionen

- start.main ()

  Hauptfunktion: Initialisiert Konfiguration und startet alle Komponenten.

5.10 ui_cli.py-Dateireferenz

Namensbereiche

- namespace ui_cli

Funktionen

- ui_cli.print_help ()

  Gibt einen Hilfetext für alle verfügbaren Kommandos aus.
- ui_cli.watcher (ui_queue)

  Gibt alle neuen Nachrichten aus der UI-Queue farbig auf dem Terminal aus.
- ui_cli.print_status (handle, status, dnd)

  Zeigt aktuelle Nutzerinformationen (Nickname, Status, DND) im Terminal an.
- ui_cli.start_cli (ui_queue, discovery_queue, network_queue, config, contacts, chat_history, dnd, status)

  Hauptfunktion für die farbige Kommandozeilensteuerung von Plauderkiste.

Variablen

- ui_cli.autoreset