**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики - процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №3**

**по дисциплине «Функциональное программирование»**

**на тему «Разработка асинхронного чат-сервера»**

**Студент гр. 23Б15-пу**

**Кубякин Н.А.**

**Преподаватель**

**Киямов Ж. У.**

**Санкт-Петербург**

**2024 г.**

***Оглавление***

[1. Цель работы 3](#_Toc178797423)

[2. Описание задачи (формализация задачи) 3](#_Toc178797424)

[3. Теоретическая часть 3](#_Toc178797431)

[4. Описание программы 4](#_Toc178797442)

[5. Рекомендации пользователя 6](#_Toc178797443)

[6. Рекомендации программиста 6](#_Toc178797444)

[7. Исходный код программы 6](#_Toc178797445)

[8. Контрольный пример 7](#_Toc178797447)

[9. Вывод 9](#_Toc178797447)

[10. Источники 9](#_Toc178797448)

# Цель работы

# Создать асинхронный чат-сервер, который может обслуживать множество клиентов одновременно и позволяет им обмениваться сообщениями в режиме реального времени.

# Описание задачи (формализация задачи)

# Обеспечение взаимодействия между асинхронным кодом и циклом событий Tkinter

# Разработка интерфейса Tkinter

# Реализация асинхронного сервера с использованием asyncio

# Парсинг входящих сообщений и извлечение информации

# Теоретическая часть

Асинхронное программирование:

Определение: Асинхронное программирование – это методология разработки программного обеспечения, основанная на концепции асинхронных операций, которые выполняются независимо друг от друга, без блокировки выполнения программы.

Применение в Python: Язык программирования Python предоставляет библиотеку asyncio для удобной реализации асинхронных приложений.

Библиотека asyncio:

Определение: asyncio – это стандартная библиотека Python для написания асинхронного кода. Она предоставляет средства для организации асинхронных задач, управления событиями и выполнения ввода/вывода без блокировки.

Ключевые компоненты: Цикл событий (event loop), корутины (coroutines), Future, Task.

Чат-протокол:

Определение: Чат-протокол – это соглашение о взаимодействии между клиентами и сервером для обмена текстовыми сообщениями.

Использование в лабораторной работе: В данной лабораторной работе чат-протокол простой, где клиенты отправляют и принимают текстовые сообщения через сервер.

# 

# Описание программы

Описание основных компонентов программы *«server.py»* представлено в таблице 4.1, программы *«client.py»* представлено в таблице 4.2.

*Таблица 4.1. Описание переменных программы «server.py»*

|  |  |  |
| --- | --- | --- |
| Имя | Параметры | Описание |
| find\_writer\_by\_username | username: Строка содержащая имя пользователя отправителя | Метод для поиска имени отправителя для отправки личного сообщения. |
| handle | reader: Объект asyncio.StreamReader для чтения данных от клиента.  writer: Объект asyncio.StreamWriter для записи данных клиенту. | Метод для обработки подключенных клиентов. Он принимает комнату от клиента, добавляет клиента в соответствующую комнату и обрабатывает обмен сообщениями между клиентами в этой комнате. |
| broadcast | message: Сообщение, которое будет рассылаться.  sender: Объект asyncio.StreamWriter отправителя сообщения.  room\_name: Имя комнаты, в которой происходит рассылка. | Метод для рассылки сообщения всем клиентам в указанной комнате. Он использует объекты asyncio.StreamWriter для отправки сообщения каждому клиенту в комнате, за исключением отправителя. |
| run | host: Хост, на котором будет запущен сервер.  port: Порт, на котором будет запущен сервер. | Метод для запуска сервера на указанном хосте и порте. Создает серверное соединение с использованием asyncio.start\_server и запускает его на бесконечное обслуживание клиентов. |

*Таблица 4.2. Описание переменных программы «client.py»*

|  |  |  |
| --- | --- | --- |
| Имя | Параметры | Описание |
| initialize\_gui | - | Инициализация графического интерфейса пользователя. Создает окно, текстовую область и поле ввода для отправки сообщений. |
| connect | - | Метод для подключения к серверу. Отправляет имя и комнату на сервер и запускает асинхронную задачу для приема сообщений от сервера. |
| start\_receive | - | Метод для асинхронного приема сообщений от сервера. Бесконечный цикл, читающий данные от сервера и отображающий их в текстовой области Tkinter. |
| send | - | Метод для отправки сообщений на сервер. Извлекает текст из поля ввода и отправляет его на сервер. |

# Рекомендации пользователя

Использование:

Начать сервер - `python server.py`

Начать клиент - `python client.py`

Команды:

`/get\_rooms` - Вывести список комнат

`/change\_room [room] [password]` - присоединиться к комнате  
 `/pm [user] [message]` - личное сообщение

# Рекомендации программиста

# Для запуска программы необходим Python, а также 64-битная операционная система Windows, или Linux, или macOS.

# Библиотеки: tkinter [[1]](#tkinter), asyncio [[2]](#asyncio).

# Минимальное необходимое место на диске: 1 МБ. Минимальное необходимое количество оперативной памяти: 25 МБ.

# Исходный код программы

# Исходный код программы и необходимые текстовые файлы доступны по ссылке: <https://github.com/NikiTaku1/spbu_funcprog/tree/main/lab3>

# Контрольный пример

В данном разделе представлены контрольные примеры для каждой из задач, демонстрирующие способность выполнять комплексную обработку данных.

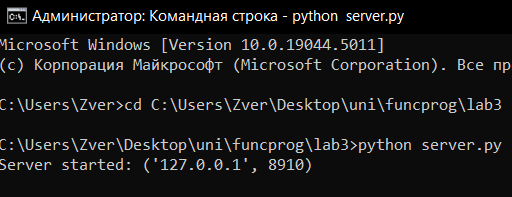


Рис. 9.1 Запуск сервера

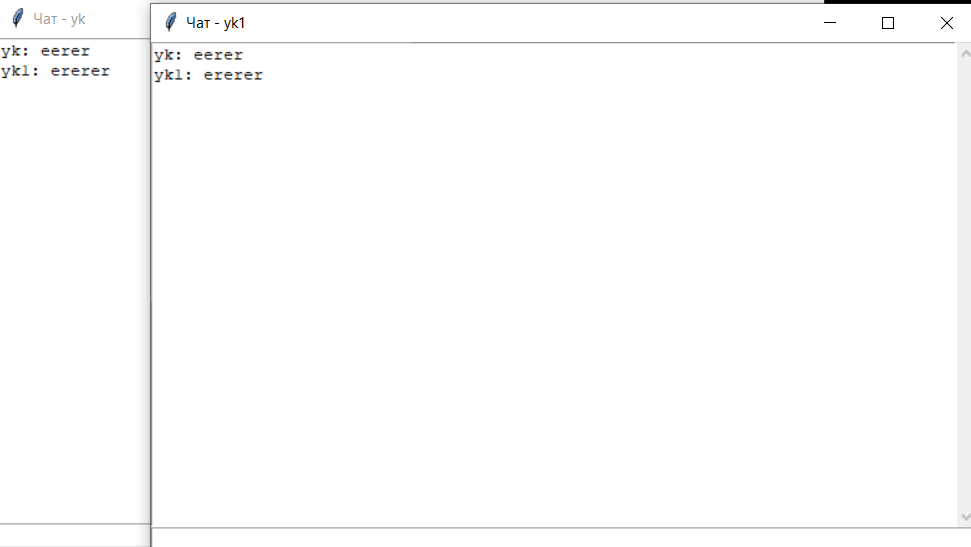


Рис. 9.2 Работа чата с двумя клиентами

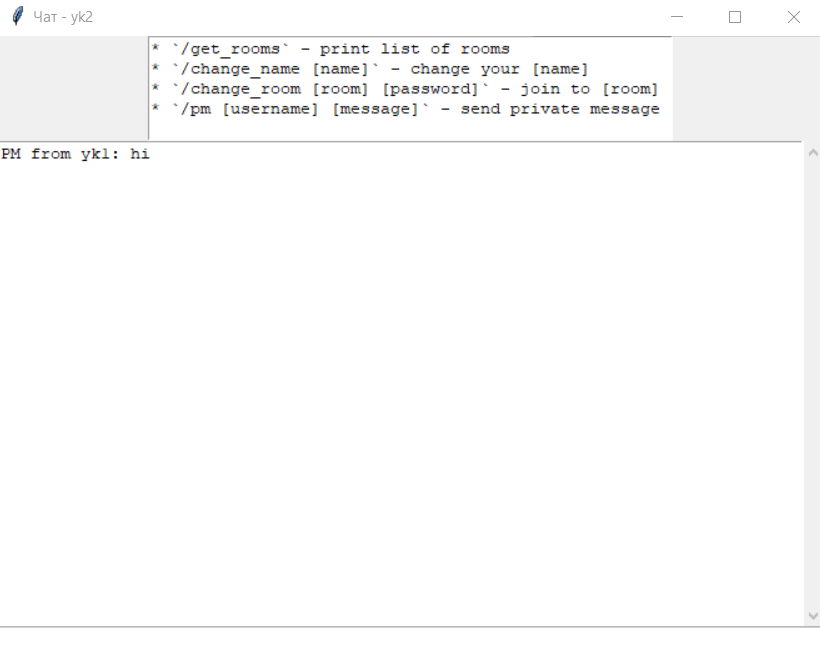


Рис. 9.3 Пересылка личного сообщения

# Вывод

# В ходе разработки чат-системы с использованием библиотеки asyncio для асинхронного программирования и Tkinter для создания графического интерфейса пользователя были успешно реализованы основные функциональности сервера и клиента. Сервер способен обслуживать множество клиентов одновременно, поддерживать чат-комнаты и рассылать сообщения в реальном времени. Клиентская часть предоставляет удобный графический интерфейс, позволяющий пользователям взаимодействовать с чатом.

# Источники

# 1. tkinter — tkinter documentation // Documentation URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 25.11.2024).

# 2. asyncio — asyncio documentation // Documentation URL: <https://docs.python.org/3/library/asyncio.html> (дата обращения: 25.11.2024).