**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**Факультет прикладной математики – процессов управления**

**отчет**

**по лабораторной работе**

**по дисциплине «Функциональное программирование»**

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

|  |  |  |
| --- | --- | --- |
| Студент гр. 22.Б15 |  | Добренкова Л.С. |
| Преподаватель |  | Киямов Ж.У. |

**Санкт-Петербург**

**2023 г.**

**Содержание**

[1. Цель работы 2](#_Toc149462504)

[2. Задача 2](#_Toc149462505)

[3. Теоретическая часть 2](#_Toc149462506)

[4. Описание программы 4](#_Toc149462507)

[5. Рекомендации пользователю 6](#_Toc149462508)

[6. Рекомендации программисту 6](#_Toc149462509)

[7. Контрольный пример 6](#_Toc149462510)

[8. Заключение 7](#_Toc149462511)

# **Цель работы**

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

# **Задача**

* 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 |
| 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 пользователю**

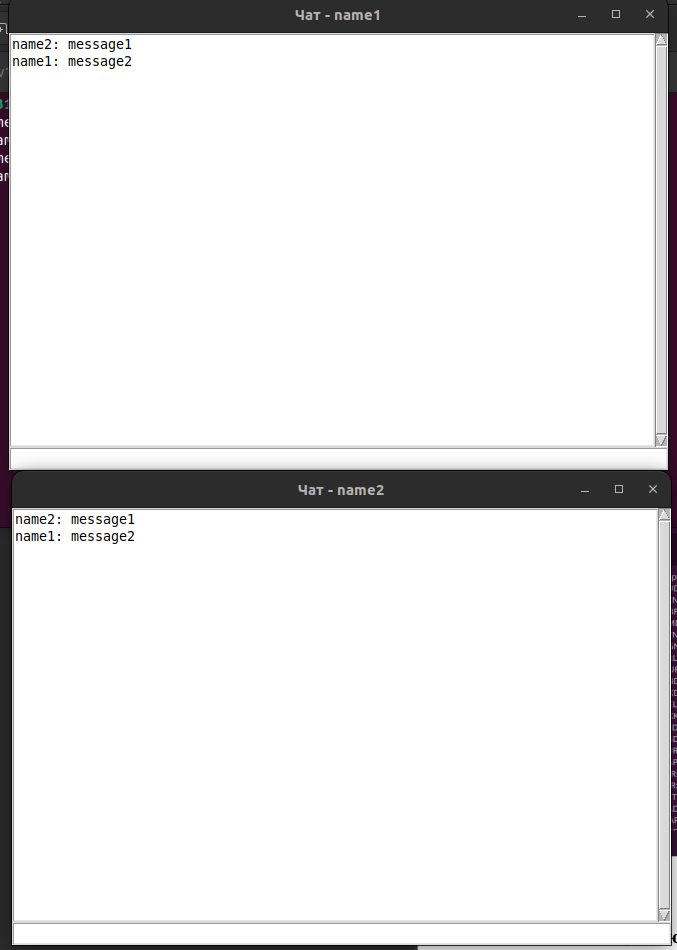
https://github.com/v131v/functional\_programming/blob/main/lab4/README.md

# **Рекомендации программисту**

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

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

# **Контрольный пример**

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

*Рисунок 7.1. «client.py»*

# **Заключение**

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.

Код доступен по ссылке: https://github.com/v131v/functional\_programming/tree/main/lab4