**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 |  | Агишев А.Б. |
| Преподаватель |  | Киямов Ж.У. |

**Санкт-Петербург**

**2023 г.**

**Содержание**

[1. Цель работы 2](#_Toc153975680)

[2. Задача 2](#_Toc153975681)

[3. Теоретическая часть 2](#_Toc153975682)

[4. Алгоритм метода 3](#_Toc153975683)

[5. Описание программы 4](#_Toc153975684)

[6. Рекомендации пользователю 5](#_Toc153975685)

[7. Рекомендации программисту 5](#_Toc153975686)

[8. Контрольный пример 6](#_Toc153975687)

[9. Заключение 7](#_Toc153975688)

[10. Литература 7](#_Toc153975689)

# **Цель работы**

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

# **Задача**

1. Приобрести и закрепить знания в области асинхронности, создании чат-серверов.
2. Разработать программное обеспечение для чат-сервера.
3. Создать интерфейс для программы.

# **Теоретическая часть**

*Асинхронность*:

Асинхронность в программировании — это подход, при котором задачи могут выполняться параллельно без блокировки основного потока выполнения. В асинхронном программировании используются асинхронные функции, которые могут быть приостановлены во время выполнения, позволяя другим задачам выполняться. Это повышает эффективность программ и позволяет обрабатывать множество задач одновременно.

*JSON (JavaScript Object Notation)*:

JSON — это формат обмена данными, основанный на подмножестве синтаксиса языка JavaScript. Он предоставляет простой и удобный способ представления структурированных данных в виде текста. JSON поддерживает объекты (ассоциативные массивы), массивы, строки, числа, логические значения и значения null. JSON широко используется для передачи данных между клиентом и сервером в веб-приложениях.

*Чат-сервер*:

Чат-сервер — это программное обеспечение, предназначенное для обеспечения обмена сообщениями между пользователями в режиме реального времени. Чат-серверы могут поддерживать один на один чаты, групповые чаты, а также функции, такие как передача файлов, аудио- и видеозвонки. Они играют важную роль в создании средств связи, таких как мессенджеры, социальные сети и внутрикорпоративные системы обмена сообщениями. Реализация чат-сервера может быть синхронной или асинхронной в зависимости от требований и планирования разработчиков.

# **Алгоритм метода**

Программа состоит из следующих частей:

1) Асинхронный клиент на Python, который обрабатывает следующие запросы: создание комнаты, отправка и восстановление сообщений, рассылка сообщений, вход в комнату.

2) Веб-клиент, который отправляет запросы на сервер через протокол WebSocket.

# **Описание программы**

В программе используется 2 класса, в таблице 5.1 представлено их описание.

*Таблица 5.1. Описание классов*

|  |  |
| --- | --- |
| Имя класса | Описание |
| Room | Класс, описывающий комнату |
| ChatServer | Класс, описывающий простой чат-сервер |

Описание переменных класса *«Room»* представлено в таблице 5.2.

*Таблица 5.2. Описание переменных класса «Room»*

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Описание |
| set | members | Участники комнаты |
| List[] | message\_history | История сообщений в комнате |

Описание функций класса *«Room»* представлено в таблице 5.3.

*Таблица 5.3. Описание функций класса «Room»*

|  |  |  |
| --- | --- | --- |
| Входные данные | Имя | Описание |
| — | \_\_init\_\_ | Инициализация переменных класса |
| Участник комнаты | add\_member | Добавление участника комнаты в случае его вступления |
| Участник комнаты | remove\_member | Удаление участника комнаты в случае его выхода |
| Отправитель, сообщение, имя пользователя | broadcast | Отправка сообщения всем участникам комнаты |
| Сообщение | add\_message\_to\_history | Добавление сообщения в историю сообщений |

Описание функций класса *«ChatServer»* представлено в таблице 5.4.

*Таблица 5.4. Описание функций класса «ChatServer»*

|  |  |  |
| --- | --- | --- |
| Входные данные | Имя | Описание |
| — | \_\_init\_\_ | Инициализация переменных класса |
| Вебсокет, порт | handle\_client | Обработка нового клиентского подключения |
| Отправитель, сообщение, имя пользователя | broadcast | Передача сообщения во все соответствующие комнаты |
| Клиент, имя пользователя, комната | join\_room | Присоединение к комнате, создание ее в случае, если её нет, оповещение об этом всем участникам комнаты |
| Клиент, имя пользователя | leave\_room | Выход из комнаты и оповещение об этом всем участникам комнаты |

# **Рекомендации пользователю**

Введите имя пользователя и название комнаты, а затем нажмите кнопку *«Join»* для того, чтобы подключиться к комнате или выберете доступную комнату из списка предложенных. Смена темы приложения происходит по нажатию в верхнем правом углу экрана соответствующего значка. Для того чтобы отправить сообщение, введите его в соответствующее поле, и нажмите кнопку *«Send»*. Сообщение отобразится в верхнем поле. Для выхода из комнаты нажмите кнопку *«Back»*.

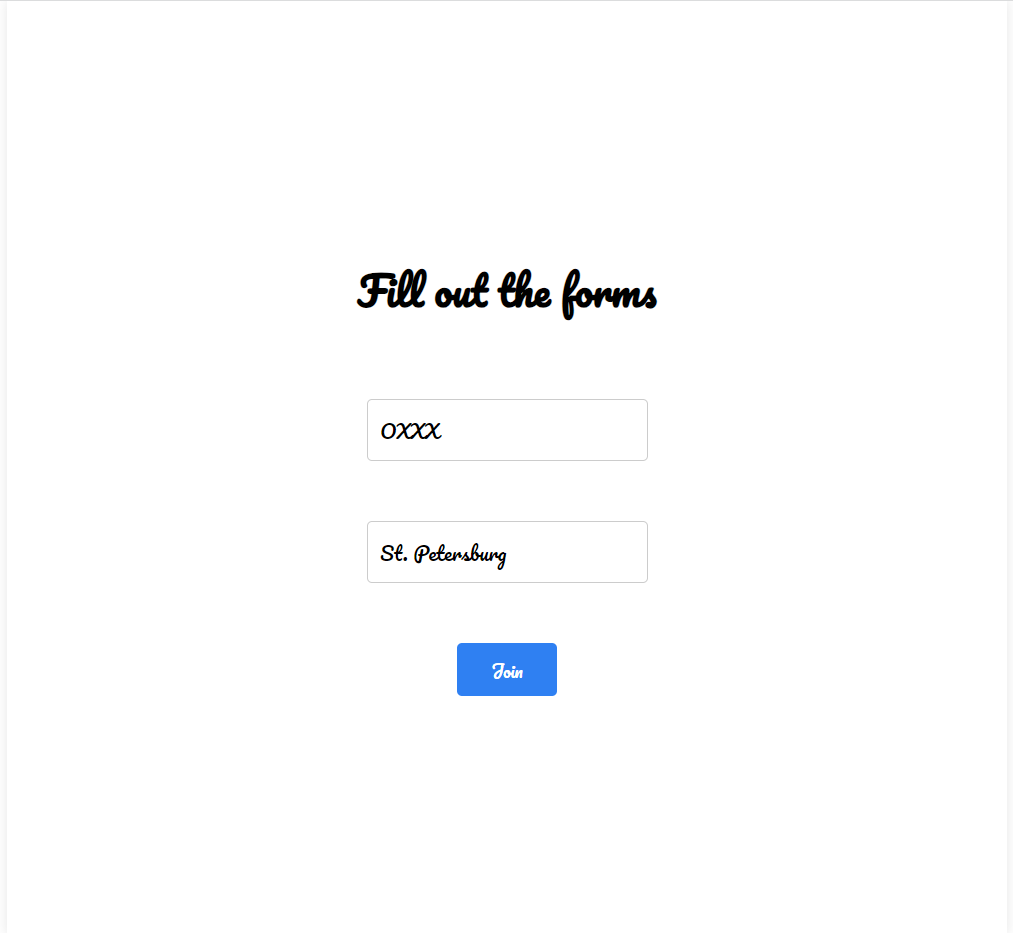
# **Рекомендации программисту**

Для запуска программы необходим Python, а также 64-битная операционная система Windows, или Linux, или macOS, а также любой доступный браузер. Необходимые библиотеки: json версии 4.17.3, asyncio версии 3.4.3, websockets версии 12.0.

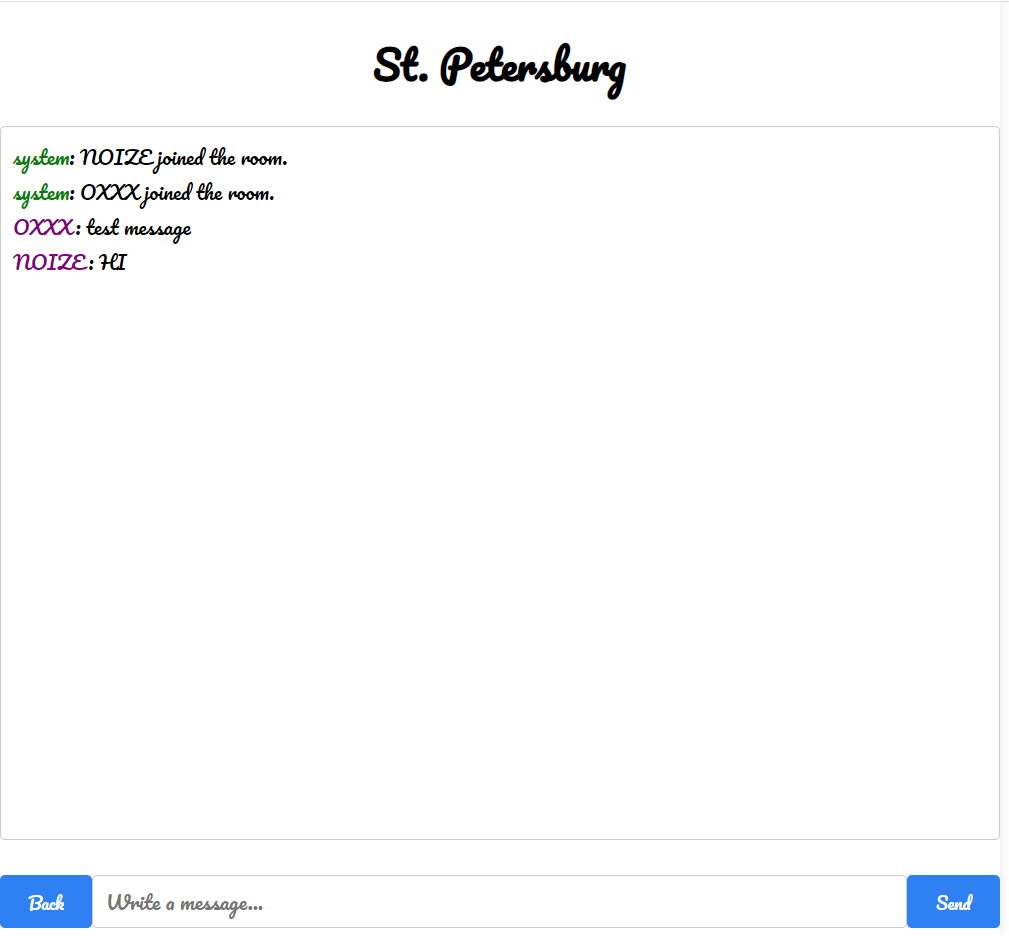
Минимальное необходимое место на диске: 0.5 МБ. Минимальное необходимое количество оперативной памяти: 50 МБ.

# **Контрольный пример**

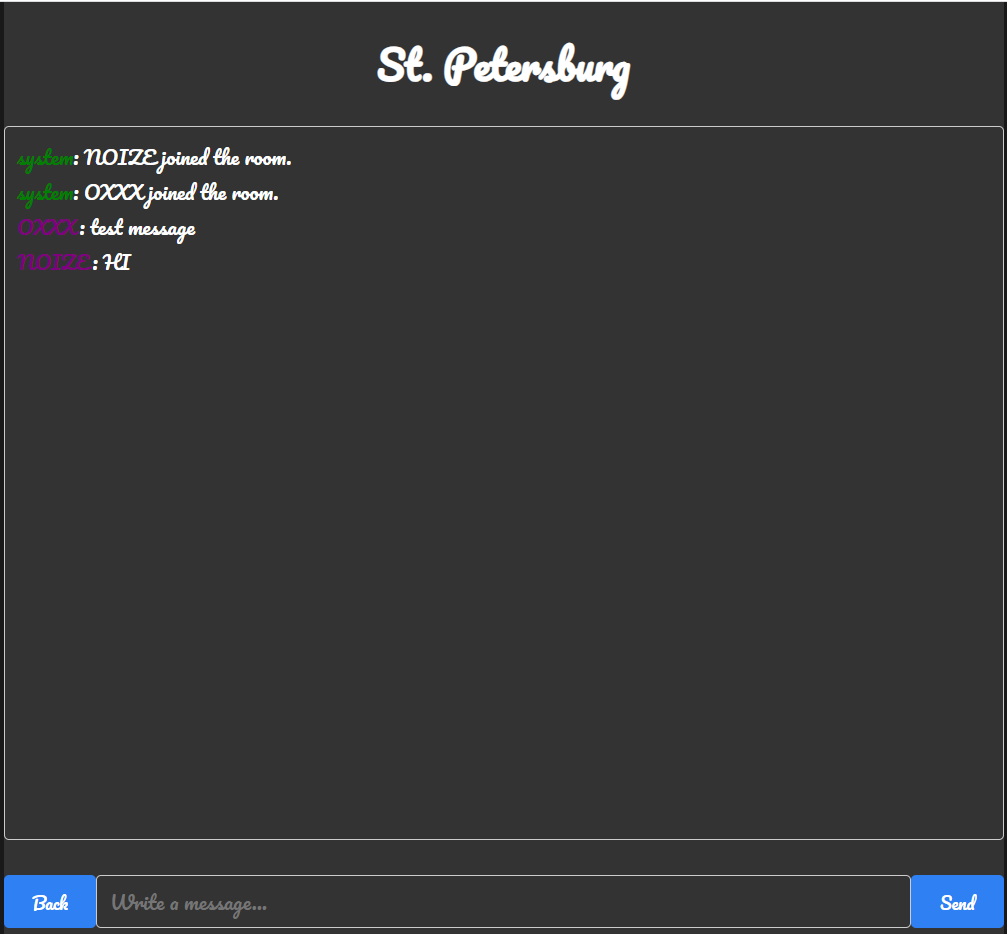
В данном разделе представлены контрольные примеры, демонстрирующий способность чат-сервера выполнять свои функции.

*Рисунок 8.1. Интерфейс программы. Вход в комнату*



*Рисунок 8.2. Интерфейс программы во время общения*



*Рисунок 8.3. Смена темы приложения*

# **Заключение**

В результате работы была успешно создана функциональная система для обмена сообщениями через чат-сервер и веб-клиент. Разработанный чат обеспечивает стабильное и безопасное взаимодействие между пользователями, а веб-клиент предоставляет удобный и интуитивно понятный интерфейс. Важным аспектом является также оптимизация системы для обеспечения эффективного обслуживания большого числа пользователей. Эта работа дает возможность пользователям обмениваться сообщениями в реальном времени, создавая удобное и привлекательное средство коммуникации.

# **Литература**

1. <https://docs.python.org/3/library/asyncio.html>
2. <https://github.com/finstape/functional-programming/tree/main/practicum_3>