**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный университет**

**Кафедра прикладной математики и процессов управления.**

**отчет**

**по лабораторной работе №3**

**по дисциплине «Функциональное программирование»**

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

|  |  |  |
| --- | --- | --- |
| Студент гр. 22.Б16-пу |  | Поляков Л.С. |
| Преподаватель |  | Киямов Ж. У. |

**Санкт-Петербург**

**2023 г.**

**Оглавление**

[**1.** **Введение** 3](#_Toc144847601)

[**2.** **Цель работы** 3](#_Toc144847602)

[**3.** **Задача** 3](#_Toc144847603)

[**4.** **Описание программы** 3](#_Toc144847604)

[**5.** **Рекомендации пользователю** 4](#_Toc144847605)

[**6.** **Рекомендации программисту** 4](#_Toc144847607)

[**7.** **Контрольный пример** 4](#_Toc144847608)

[**8.** **Вывод** 6](#_Toc144847611)

**1. Введение:**

Целью работы является создание асинхронного чат-сервера, который можно запустить в браузере, и клиентского приложения для авторизации пользователей.

**2. Цель работы:**

Разработка функционального асинхронного чат-сервера с поддержкой авторизации и клиентского приложения для взаимодействия с сервером.

**3. Задачи:**

• Реализация функций для добавления и удаления пользователей, создания и удаления комнат, добавления и удаления пользователей из комнат, сохранения и отправки сообщений в комнатах.

• Создание пользовательского интерфейса для взаимодействия с сервером.

**4. Описание программы:**

**4.1 server.py:**

|  |  |
| --- | --- |
| addUser(websocket, user\_name) | Добавляет пользователя в словарь пользователей. |
| removeUser(websocket) | Удаляет пользователя из словаря пользователей. |
| createRoom(room\_name) | Создает новую комнату. |
| addUserToRoom(websocket, room\_name) | Добавляет пользователя в комнату. |
| removeUserFromRoom(websocket, room\_name) | Удаляет пользователя из комнаты. |
| saveRoomData(room\_name, room\_messages) | Сохраняет сообщения комнаты в файл. |
| sendSavedMessagesToUser(websocket, room\_name) | Показывает сохраненные сообщения пользователю. |
| socket(websocket, path) | Основная функция чата, обрабатывает сообщения и управляет подключением и отключением пользователей. |

**4.2 socket.html:**

|  |  |
| --- | --- |
| connectToWebSocket() | Устанавливает соединение с сервером, отправляет имя пользователя и обрабатывает события клавиатуры. |
| Button Click Listeners | Обрабатывают события кликов по кнопкам для создания, входа и выхода из комнат, а также для запроса списка комнат. |
| Message Input Listener | Отправляет сообщение при нажатии клавиши Enter в поле ввода сообщения. |

**5. Рекомендации пользователю:**

• Python версии 3.8-3.11.

• Входные данные: взаимодействие через HTML-форму с полем ввода имени пользователя и кнопками для управления комнатами и сообщениями.

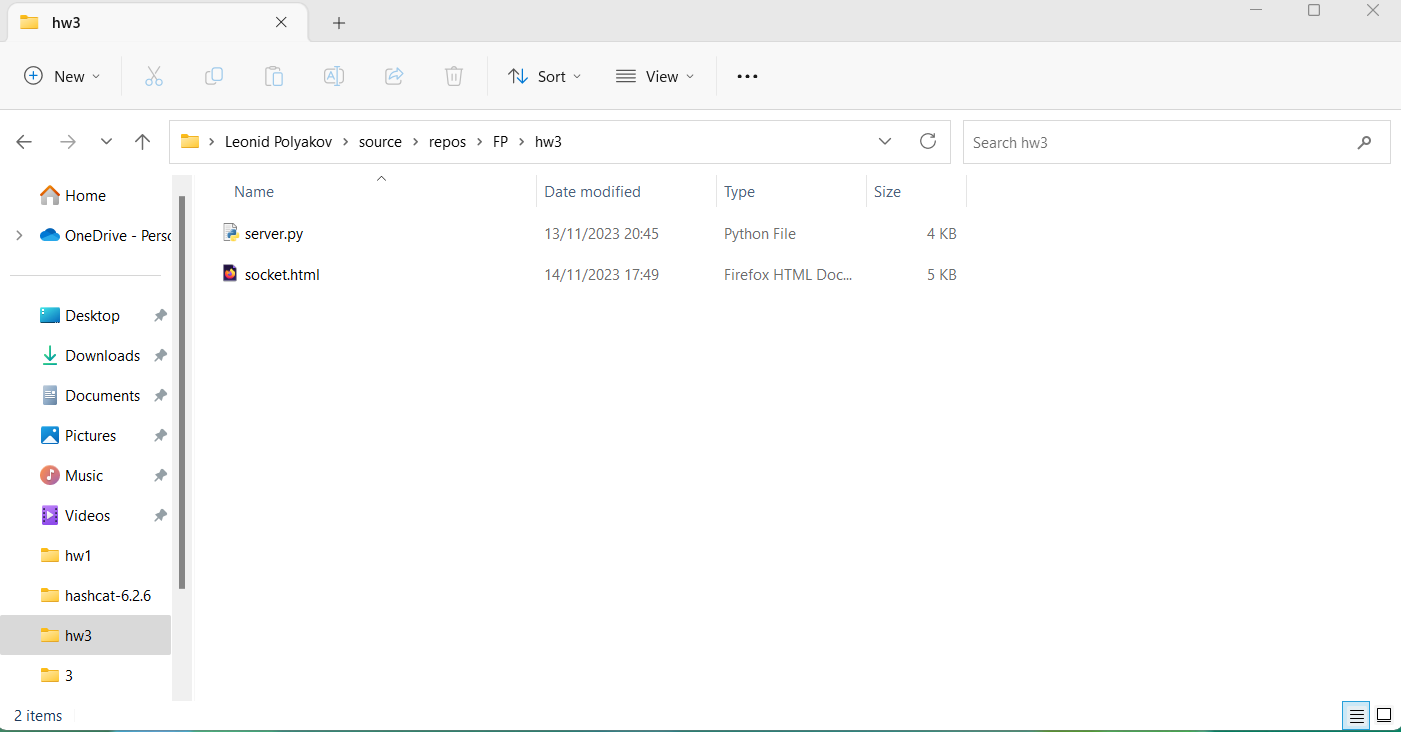
**6. Рекомендации программисту:**

• Убедитесь, что у вас установлены библиотеки asyncio и websockets (установка: *pip install asyncio websockets*).

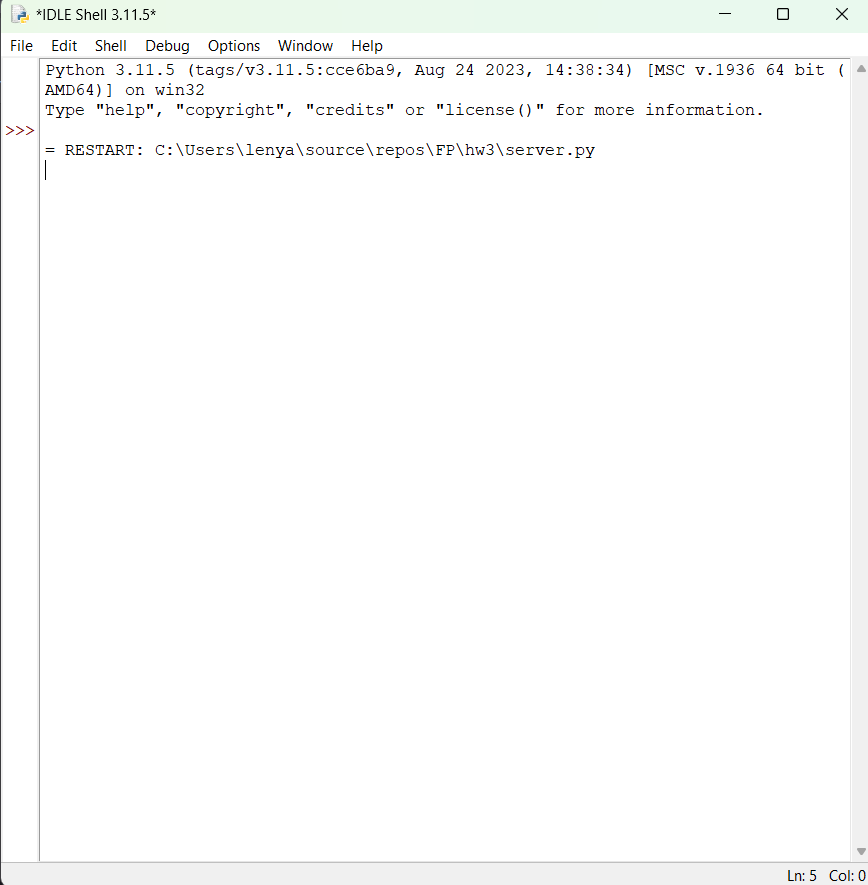
• Проверьте корректность путей и наличие необходимых файлов.

• При желании можно добавить дополнительные функции и улучшения, такие как логирование и защиту от некорректных данных.

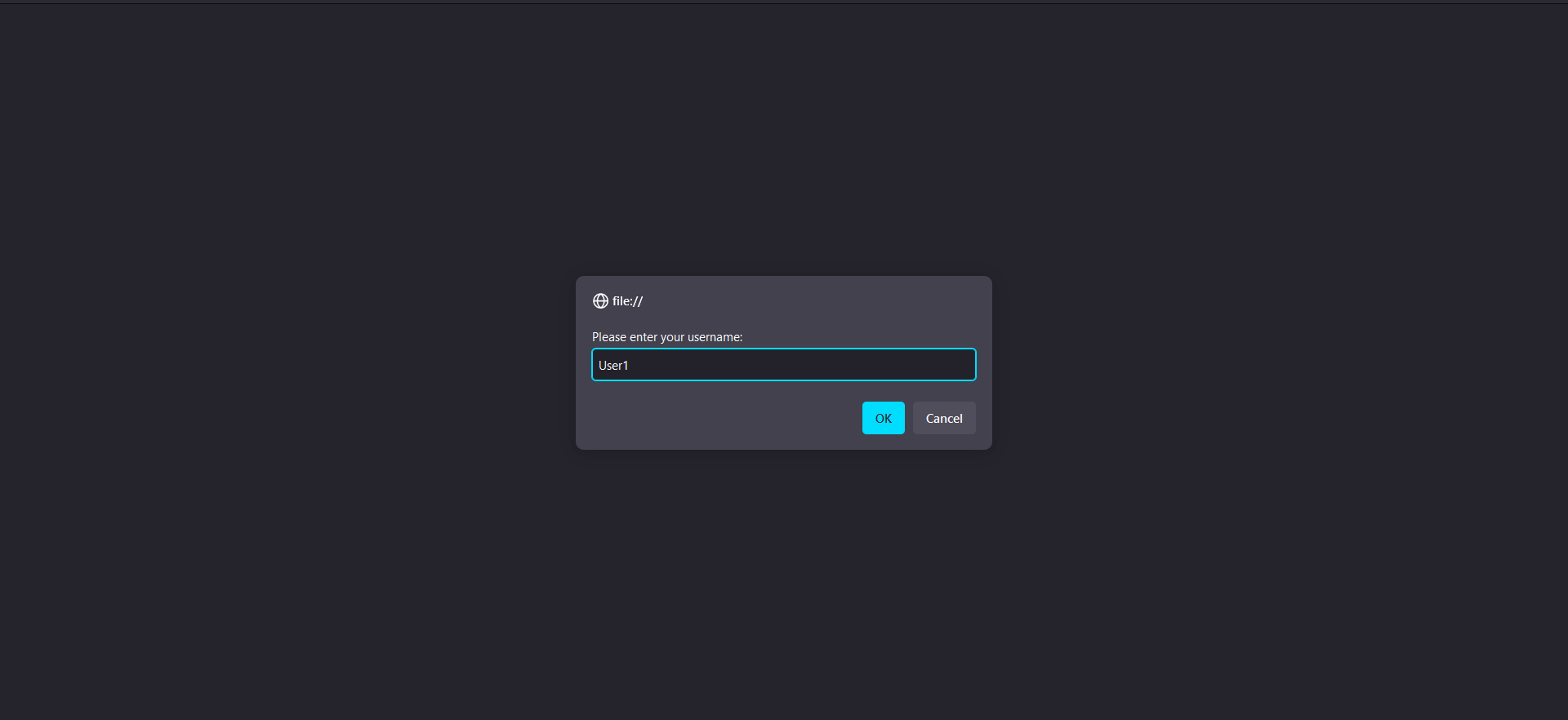
**7. Контрольный пример:**

****

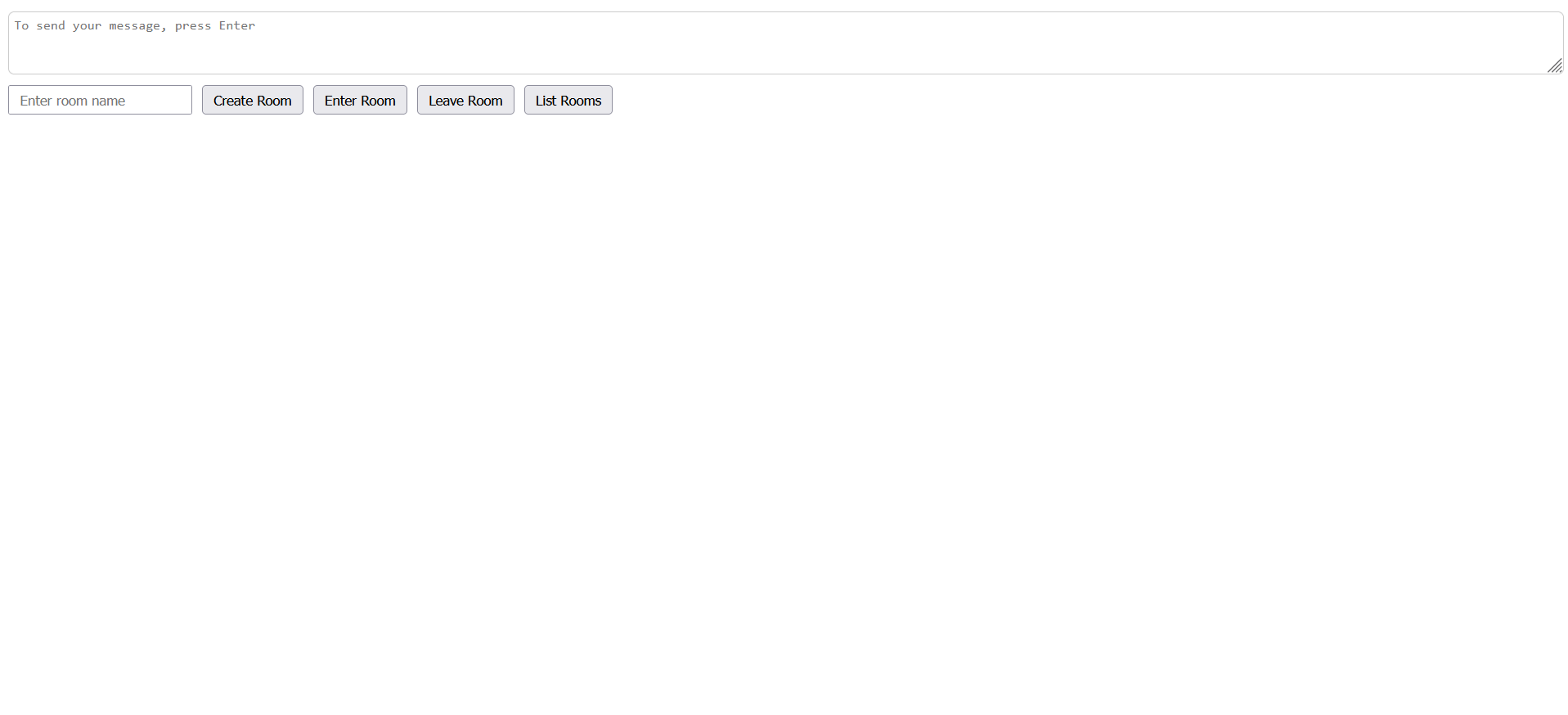
*Рис. 1 (Перед запуском сервера убедитесь, что файлы находятся в одной директории для удобства использования программы.)*

**

*Рис. 2 (После запуска server.py откроется сервер на указанных IP и порте (по умолчанию – localhost:5678) В терминале будет отображатся вывод всех сообщений, поступающих на сервер.)*

**

*Рис. 3 (После запуска socket.html в браузере откроется следующее окно. Оно предложит ввести имя пользователя (любое на выбор) и авторизоваться, чтобы получить доступ к чат-комнатам.)*

**

*Рис. 4 (После авторизации во вкладке браузера откроется данное окно. Здесь можно управлять комнатами, то есть создавать и удалять их, а также отправлять сообщения в выбранной комнате (изначально все пользователи попадают в комнату “hub”).)*

**8. Вывод:**

Проект предоставляет функциональный и удобный способ обмена сообщениями в реальном времени через асинхронный чат-сервер и клиентский интерфейс в браузере. Серверная часть реализована на языке Python с использованием библиотеки *websockets*, что обеспечивает эффективное взаимодействие между клиентами и сервером. Клиентская часть, в свою очередь, реализована при помощи HTML файла, содержащего функции для отправки пакетов на сервер.

Ссылка на GitHub: https://github.com/p0le0n/fp\_3