Уральский федеральный университет

имени первого Президента России Б. Н. Ельцина

Кафедра «Интеллектуальные информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА 3**

ОТЧЁТ

Дисциплина «Web-программирование»

Преподаватель:

Д.В. Свинцов

Студент:

гр. РИ-380022

Борисова В. Б.

Екатеринбург, 2020

**Лабораторная работа №3**

**Работа с протоколом WebSocket**

1. Цель работы: получить практические навыки по работе с протоколом Websocket.
2. Задания:
   1. Написать Websocket сервер на языке python c использованием библиотеки aiohttp.
   2. Создать проект, с файлом index.html в который добавить код на javascript для получения содержимого файлов с сервера при помощи AJAX и протокола WebSocket.
   3. Добавить в index.html форму для отправки запросов на получение содержимого файлов.
3. Ход работы
   1. Написать Websocket сервер на языке python c использованием библиотеки aiohttp.

Для этого в домашней директории создадим файл с расширением py. Внутри этого файла напишем код Websocket cервера (рисунок 1, 2 , 3).

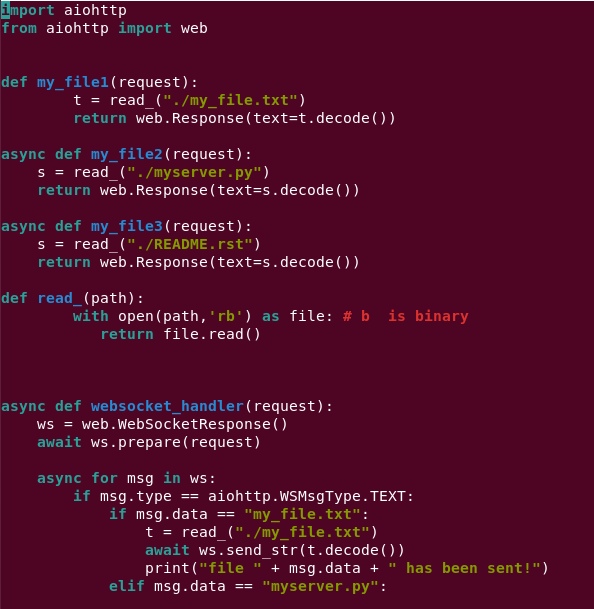


Рисунок — Исходный код Websocket сервера

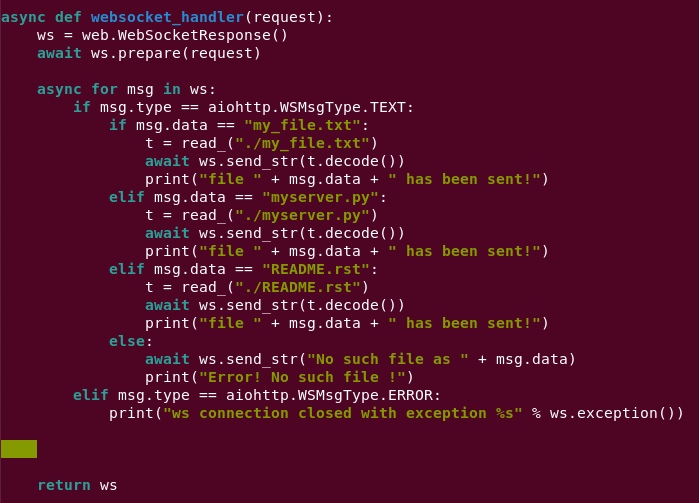


Рисунок — Исходный код Websocket сервера



Рисунок — Исходный код Websocket сервера

Теперь сохраним этот файл с названием myserver.py и попробуем его запустить при помощи команды:

**python3 myserver.py**

Терминал сообщает нам, что сервер успешно запущен по адресу 0.0.0.0 на 8080 порту (рисунок 4).



Рисунок — Запускаем созданный сервер

Теперь попробуем выполнить запрос на получение файла с сервера. Для этого откроем браузер и в адресной строке введем следующее:

**http://127.0.0.1:8080/myfile1.txt**

Получим следующий результат (рисунок 5).



Рисунок — Считывание файла myfile.txt с сервера

Как видно, файл был успешно считан и передан пользователю. Аналогичную операцию повторим с остальными файлами. А именно, введем в поисковой строке следующие адреса:

http://127.0.0.1:8080/[myserver.py](https://vk.com/away.php?to=http%3A%2F%2Fmyserver.py&cc_key=)

<http://127.0.0.1:8080/README.rst>

Результаты выполнения запросов (рисунок 5, 6, 7).



Рисунок — Считывание файла myserver.py с сервера



Рисунок — Считывание файла README.rst с сервера

Теперь необходимо изменить код так, чтобы была возможность получать эти же файлы по адресу http://127.0.0.1:8080/ws при помощи протокола WebSocket. Для этого в существующий код нашего сервера необходимо внести следующие изменения (рисунок 8):

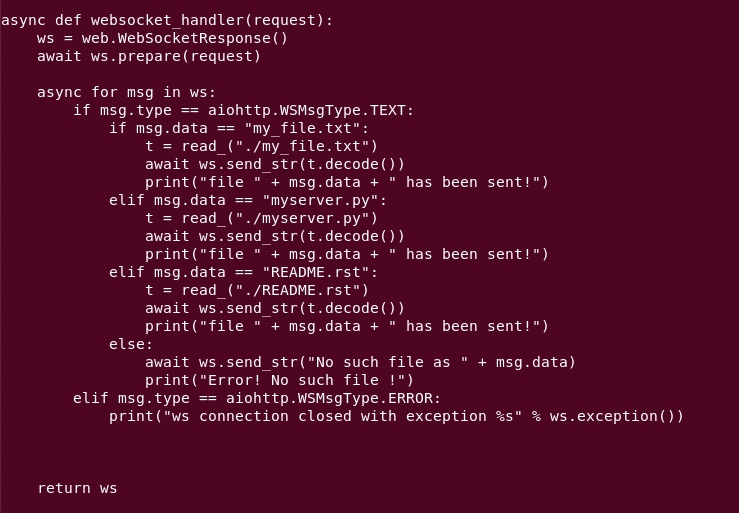


Рисунок — Вносим изменения в исходный код нашего сервера

После чего нужно сохранить файл и снова запустить сервер командой python3 myserver.py. Чтобы начать передачу файлов между сервером и нами, необходимо написать клиентскую часть. Это позволит нам установить соединение с уже созданным сервером по протоколу Websocket и получать содержимое файлов.

Для реализации вышесказанного необходимо создать новый файл с расширением py и внутрь этого файла поместить следующий код (рисунок 9).

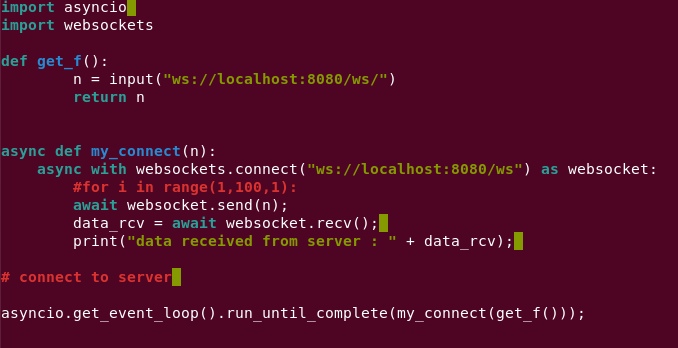


Рисунок — Исходный код клиента

После того, как файл был создан, можно инициировать соединение с сервером. Для этого в терминале нужно ввести следующую команду:

**python3 client.py**

Сервер не сообщает о том, подключился клиент или нет, поэтому попробуем запросить какой-нибудь файл (рисунок 10).

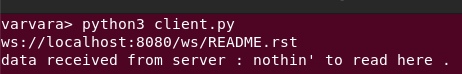


Рисунок — Запрашиваем файл README.rst

Файл README.rst был считан успешно. Похожим образом попробуем считать остальные файлы с сервера (рисунок 11).

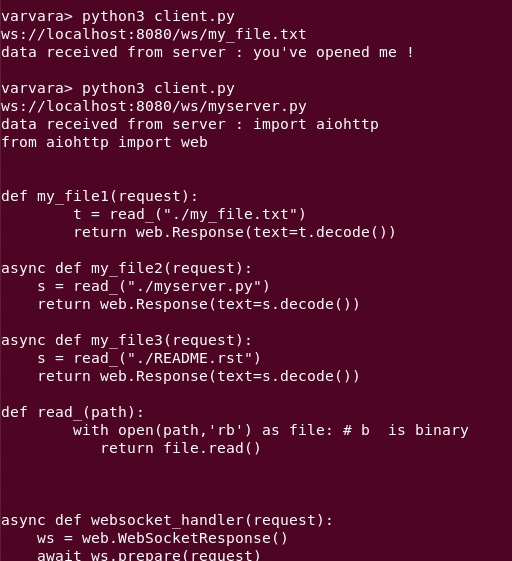


Рисунок — Запрашиваем остальные файлы с сервера

Все файлы были успешно считаны, а значит задачу, поставленную изначально, можно считать выполненной.

* 1. Создать проект, с файлом index.html в который добавить код на javascript для получения содержимого файлов с сервера при помощи AJAX и протокола WebSocket.

Создадим проект со следующей структурой (Рисунок 12):



Рисунок — Структура проекта

Внутрь файла index.html поместим следующий код на javascript. Это нужно для того, чтобы иметь возможность получать содержимое файлов с сервера при помощи AJAX запросов через WebSocket (рисунок 13).

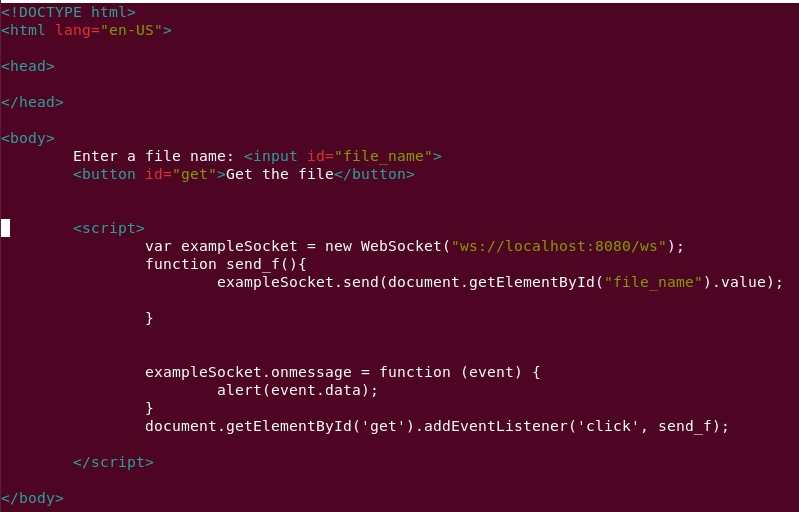


Рисунок — html-код для отправки AJAX-запросов

После чего файл нужно сохранить и отрыть в браузере (рисунок 14).

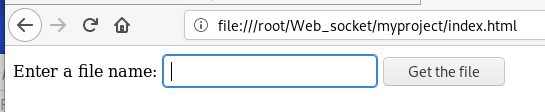


Рисунок — Файл index.html

Теперь, если мы попробуем запросить какой-нибудь файл, браузер выдаст ошибку. Потому что запросы, которые мы отправляем, некому обрабатывать. Чтобы устранить эту проблему, нужно запустить ранее созданный сервер. Для этого в терминале нужно ввести следующую команду:



Рисунок — Запускаем сервер

После чего попробуем считать один из файлов, хранящийся на сервере (рисунок 16).

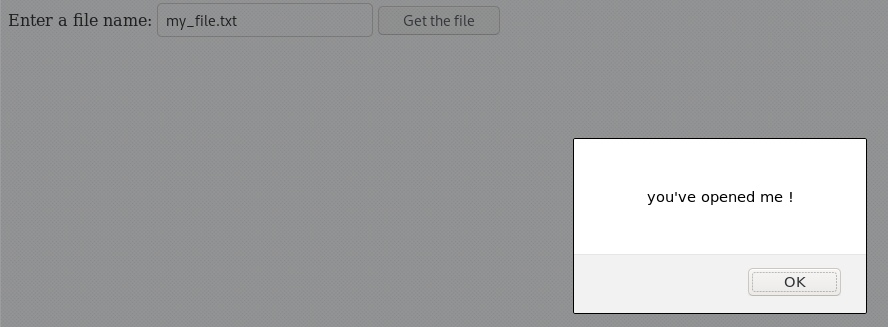


Рисунок — Считываем файл my\_file.txt

Как можно заметить, запрос был успешно отправлен и обработан сервером. Мы получили содержимое файла, хранящегося на сервере.

Теперь попробуем считать остальные файлы, вводя их имена в поле для ввода, которое мы создали (рисунок 17, 18).

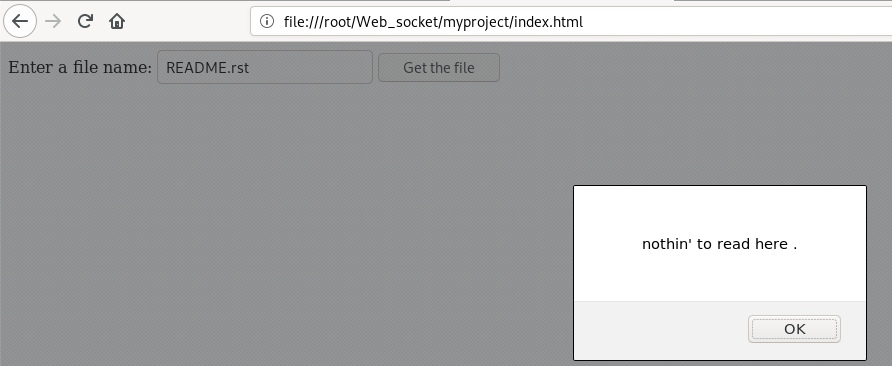


Рисунок — Считываем файл README.rst

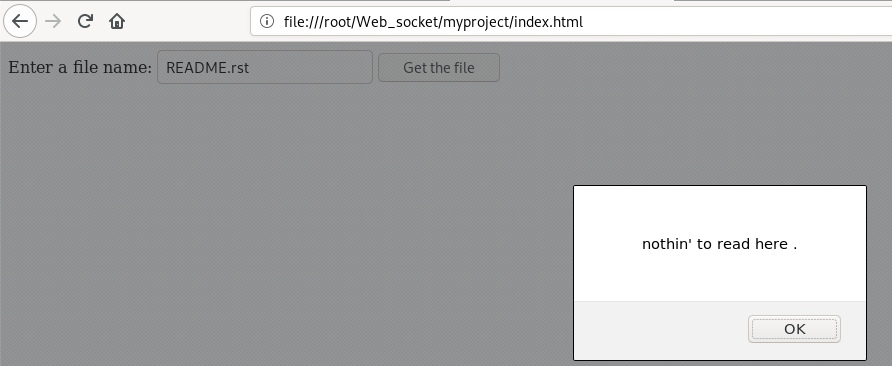


Рисунок — Считываем файл myserver.py

Все файлы были успешно считаны.

* 1. Добавить в index.html форму для отправки запросов на получение содержимого файлов.

Добавим в файл index.html форму для отправки запросов на получение содержимого файлов. Это нужно для удобства пользователя, чтобы не приходилось вручную указывать адрес запроса, и большей мобильности – считывать не какой-то определенный файл, а любой (рисунок 19).

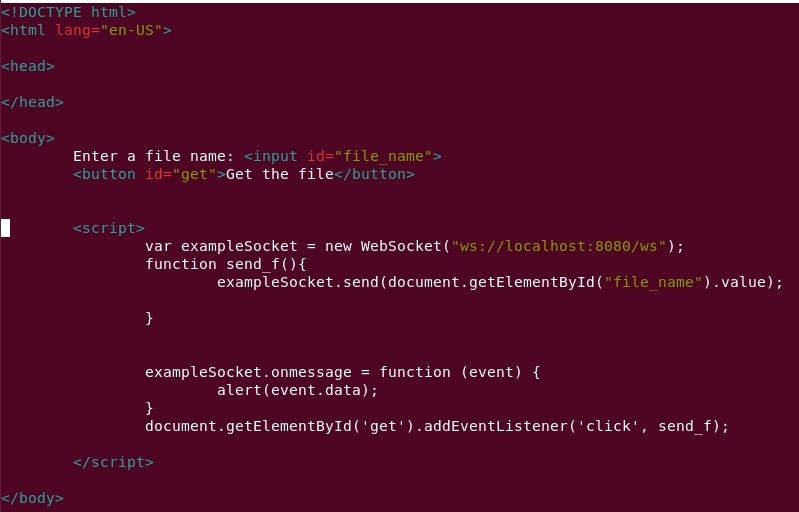


Рисунок — Создаем форму для отправки запросов

Созданная форма (рисунок 20).

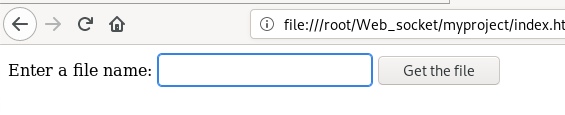


Рисунок — Созданная форма

В поле для ввода мы записываем имя файла, после чего отправляем запрос кнопкой Get the file. Запрос на считывание файла отправляется на сервер и там обрабатывается. Если запрошенный файл существует, сервер вернет его содержимое. В противном случае выдаст следующую ошибку (рисунок 21).

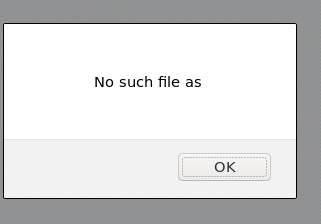


Рисунок — Ошибка считывания файла