Министерство образования Республики Беларусь

Учреждение образования

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ**

Кафедра интеллектуальных информационных технологий

Лабораторная работа №1

АиПОСиЗИ

Проверил: Ольга Сергеевна

Выполнил: Салюков Глеб (*121731*)

Минск

2023 г.

Лабораторная работа 5

**Общие требования:**

* Любой язык программирования;
* Допускается использование разумного количества сторонних библиотек;
* Обработка и создание HTTP пакетов без использование сторонних библиотек;
* Консольное приложение (если другое не указано в задании);
* Конфигурация всех параметров (в разумных пределах) с помощью command line аргументов (Используйте стороннюю библиотеку для парсинга). Поддержка коротких и длинных вариантов (-a, --argument), help по всем доступным аргументам.

**Требования к серверу:**

* Запись подробного лога с деталями работы протокола в log файл;
* Вывод краткой, понятной человеку информации в консоль.

**Требования к клиенту:**

* Конфигурация запросов с помощью command line аргументов;
* Поддержка выполнения запросов по шаблону из файла;
* Обработка и вывод сообщений об ошибках согласно работе/статусам ответа протокола.

**HTTP 1.1 Сервер**

* Программа должна “раздавать” файлы из указанной директории
* Конфигурация заголовков по умолчанию, которые будут отправляться с каждым запросом. Например:  
  Access-Control-Allow-Origin: https://my-cool-site.com  
  Access-Control-Allow-Methods: GET, POST, OPTIONS
* Поддерживаемые виды запросов: GET, POST, OPTIONS;
* Поддерживать необходимое для работы количество статусов ответов [HTTP response status codes - HTTP](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status).

[RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1](https://tools.ietf.org/html/rfc2616). Тестируйте с помощью браузера (Должен работать с html, css, js, svg, png)  или любых других http клиентов (postman, insomnia).

**HTTP 1.1 Клиент**

* Поддержка любого метода запроса;
* Конфигурация произвольных заголовков запроса;
* Возможность отправки тела запроса из файла или command line аргумента.

[RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1](https://tools.ietf.org/html/rfc2616). Можно вдохновляться [curl](https://curl.haxx.se/).

import http.server  
import socketserver  
import argparse  
import logging

Импортируем необходимые библиотеки для создания HTTP-сервера, обработки командной строки и ведения логов.

parser = argparse.ArgumentParser(description='HTTP 1.1 Server')  
parser.add\_argument('-p', '--port', type=int, default=8000, help='Port number')  
parser.add\_argument('-d', '--directory', default='.', help='Directory to serve')  
args = parser.parse\_args()

Этот блок кода создает парсер аргументов командной строки. Он ожидает два аргумента: **-p** или **--port** для указания порта сервера и **-d** или **--directory** для указания директории, которую сервер будет обслуживать. Если аргументы не указаны, используются значения по умолчанию (порт 8000 и текущая директория).

DEFAULT\_HEADERS = {  
 'Access-Control-Allow-Origin': 'https://my-cool-site.com',  
 'Access-Control-Allow-Methods': 'GET, POST, OPTIONS'  
}

Эта часть определяет заголовки по умолчанию для ответов сервера. В данном случае сервер разрешает доступ к ресурсам только с сайта **https://my-cool-site.com** и поддерживает методы **GET**, **POST** и **OPTIONS**.

logging.basicConfig(filename='http\_server.log', level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

Эта строка настраивает систему логирования. Логи будут записываться в файл **http\_server.log** и будут содержать информацию о времени, уровне логирования и сообщении.

class *MyHttpRequestHandler*(http.server.SimpleHTTPRequestHandler):  
 def end\_headers(self):  
 for header, value in DEFAULT\_HEADERS.items():  
 self.send\_header(header, value)  
 super().end\_headers()

Этот класс представляет собой обработчик запросов HTTP. Метод **end\_headers** переопределен, чтобы добавить к ответам сервера заголовки по умолчанию.

handler\_object = MyHttpRequestHandler  
my\_server = socketserver.TCPServer(("", args.port), handler\_object)  
logging.info(f"Server started on port {args.port}")  
my\_server.serve\_forever()

В этих строках создается сервер с указанным портом и обработчиком запросов MyHttpRequestHandler. Затем сервер запускается и начинает прослушивать соединения на указанном порту. Когда клиент отправляет запрос, сервер отвечает в соответствии с определенными в обработчике MyHttpRequestHandler правилами.

**Вывод:** Этот код представляет собой минимальный HTTP-сервер, который обслуживает файлы из указанной директории. Он добавляет заголовки CORS (Cross-Origin Resource Sharing) для разрешения доступа к ресурсам только с определенного домена (https://my-cool-site.com). Сервер будет записывать логи в файл http\_server.log. Приложение можно запустить с различными портами и директориями для обслуживания, указав соответствующие аргументы командной строки.

[**https://github.com/Salyukovgleb/AIPOSIZI**](https://github.com/Salyukovgleb/AIPOSIZI) **Ссылка на Github**