



ITMO UNIVERSITY

Факультет ПИиКТ

Тестирование программного обеспечения

Лабораторная работа 4

Выполнил: Давтян Д. А.

Группа: Р33022

Преподаватель: Исаев И. В.

Санкт-Петербург

2021 г.

Задание

С помощью программного пакета Apache JMeter провести нагрузочное и стресс-тестирование веб-приложения в соответствии с вариантом задания.

В ходе нагрузочного тестирования необходимо протестировать 3 конфигурации аппаратного обеспечения и выбрать среди них наиболее дешёвую, удовлетворяющую требованиям по максимальному времени отклика приложения при заданной нагрузке (в соответствии с вариантом).

В ходе стресс-тестирования необходимо определить, при какой нагрузке выбранная на предыдущем шаге конфигурация перестаёт удовлетворять требованиям по максимальному времени отклика. Для этого необходимо построить график зависимости времени отклика приложения от нагрузки.

Параметры тестируемого веб-приложения:

- URL первой конфигурации (\$ 2300) - <http://aqua:8080?token=466666545&user=1964509753&conf=1>;
- URL второй конфигурации (\$ 4200) - <http://aqua:8080?token=466666545&user=1964509753&conf=2>;
- URL третьей конфигурации (\$ 4700) - <http://aqua:8080?token=466666545&user=1964509753&conf=3>;
- Максимальное количество параллельных пользователей - 14;
- Средняя нагрузка, формируемая одним пользователем - 40 запр. в мин.;
- Максимально допустимое время обработки запроса - 480 мс.

Отчёт по работе должен содержать:

1. Текст задания.
2. Описание конфигурации JMeter для нагрузочного тестирования.
3. Графики пропускной способности приложения, полученные в ходе нагрузочного тестирования.
4. Выводы по выбранной конфигурации аппаратного обеспечения.
5. Описание конфигурации JMeter для стресс-тестирования.
6. График изменения времени отклика от нагрузки для выбранной конфигурации, полученный в ходе стресс-тестирования системы.
7. Выводы по работе.

Выполнение

Конфигурация

Всё начинается с конфигурации JMeter. В нём были выставлены пользовательские переменные HOST, PORT и CONF, чтобы было удобнее настраивать всё остальное.

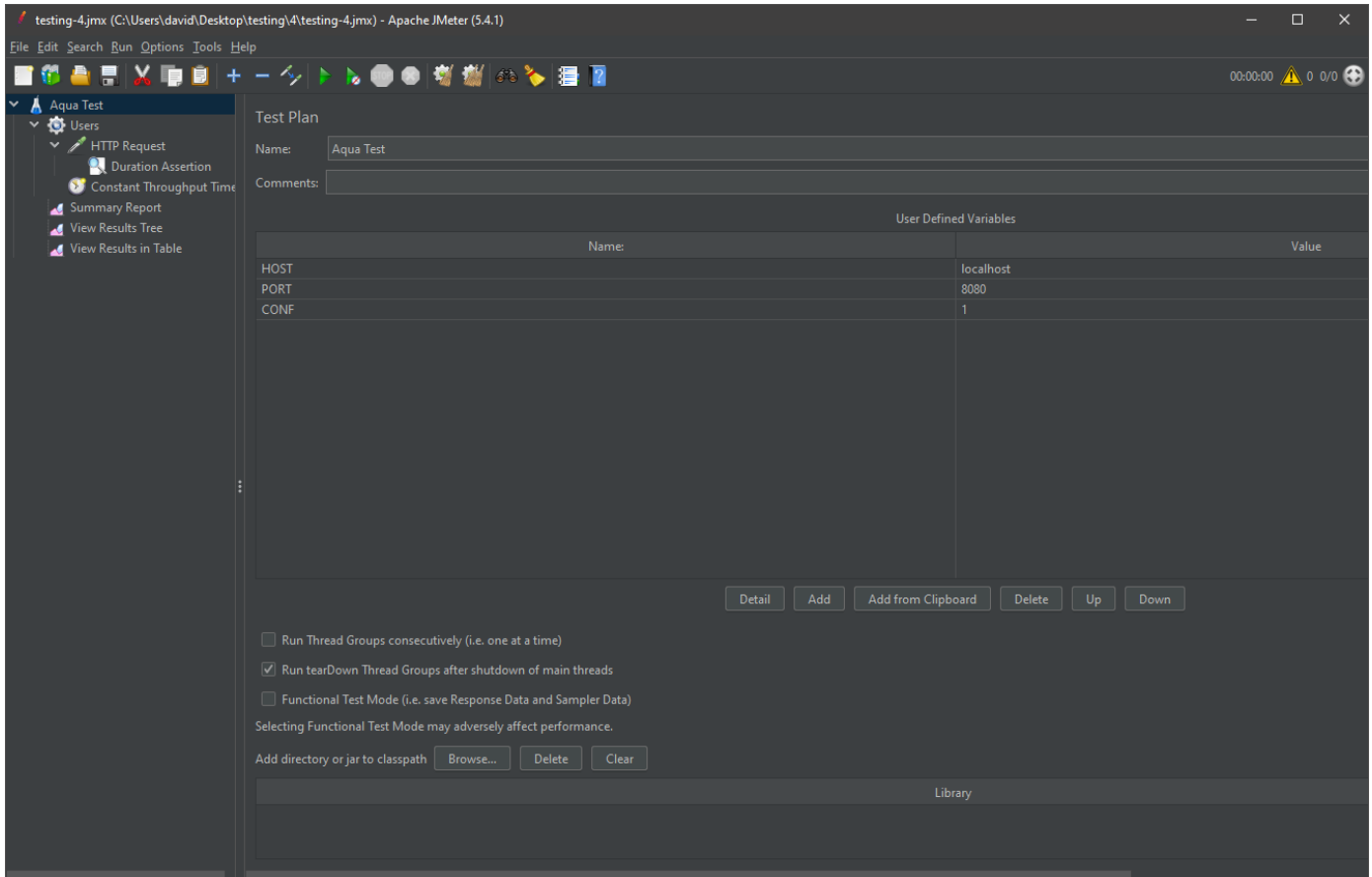


Рисунок 1. Структура тестового плана

Период наращивания в секундах равен количеству пользователей.

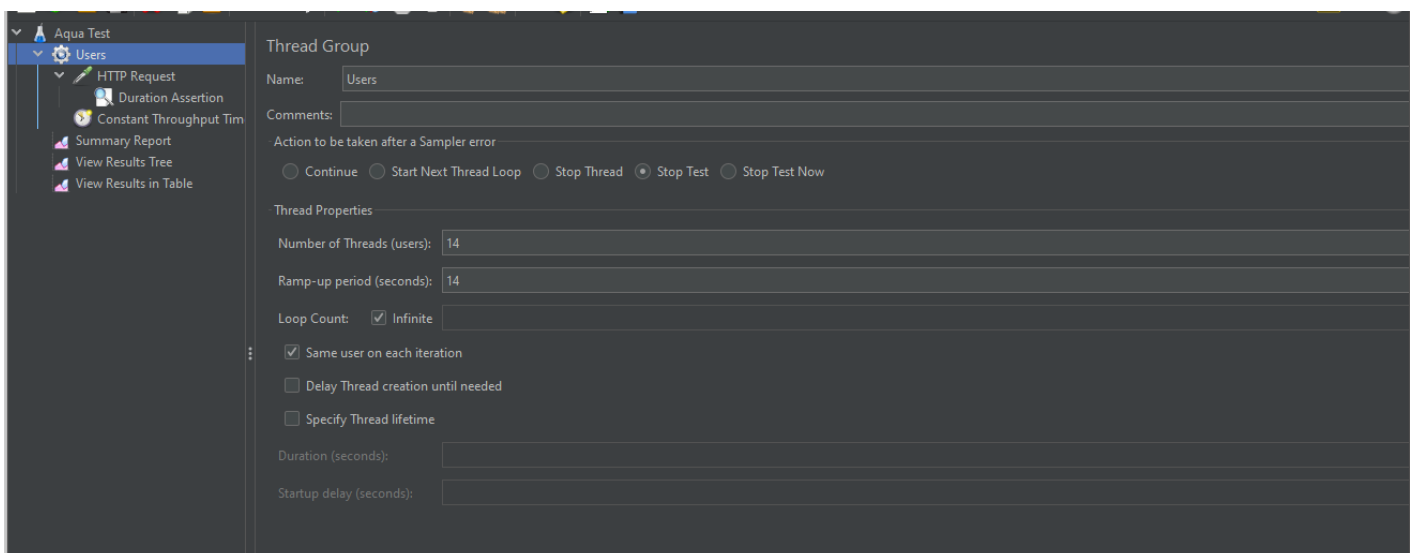


Рисунок 2. Настройка количества потоков и периода наращивания

Период наращивания – время, за которое запустятся все потоки. Так сделано для того, чтобы каждую секунду тестирования прибавлялось по одному потоку (до 14) и на выходе получился красивый график зависимости количества пользователей от времени отклика. Если период наращивания будет 0, все потоки начнут работу сразу и вместо красивого графика мы получим невзрачную ярмую линию.

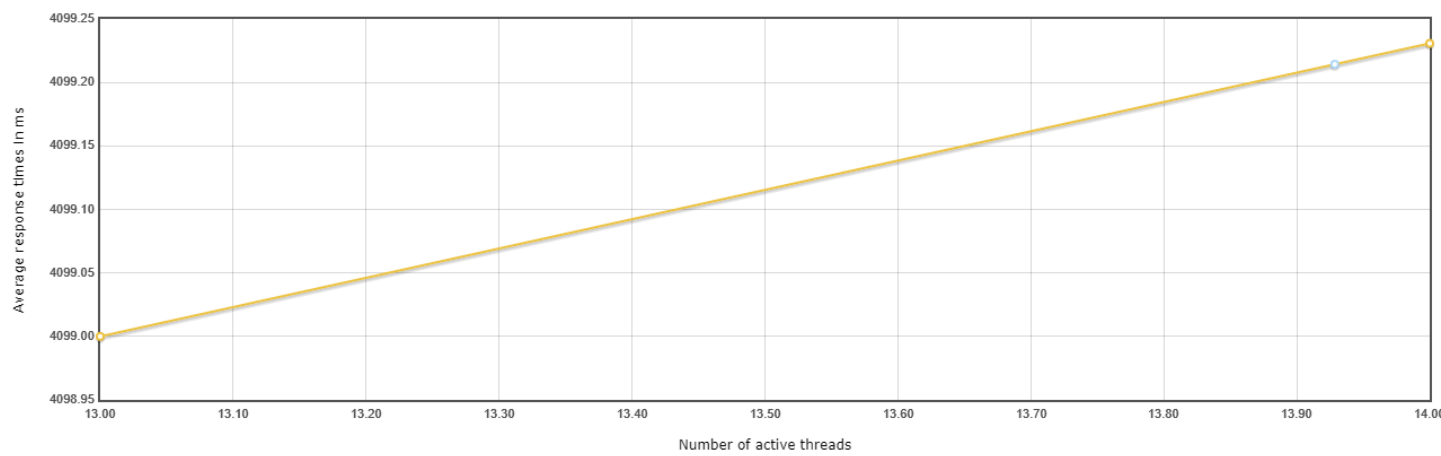


Рисунок 3. Невзрачная прямая линия

Далее нужно создать и настроить запрос, который будет отправляться.

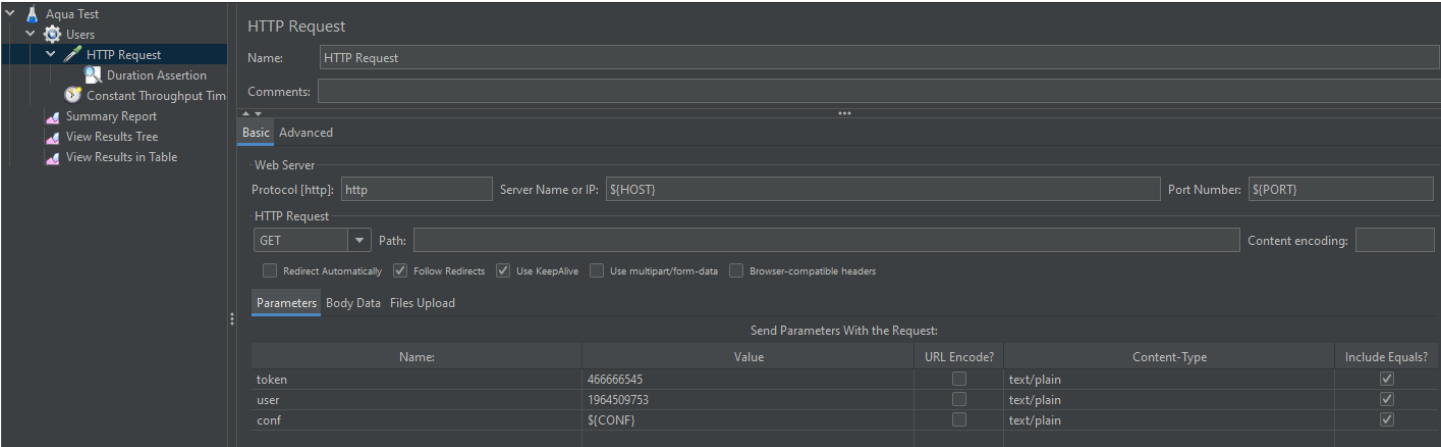


Рисунок 4. Настройка запроса

Добавим ещё и ограничение на максимальное время обработки запроса.

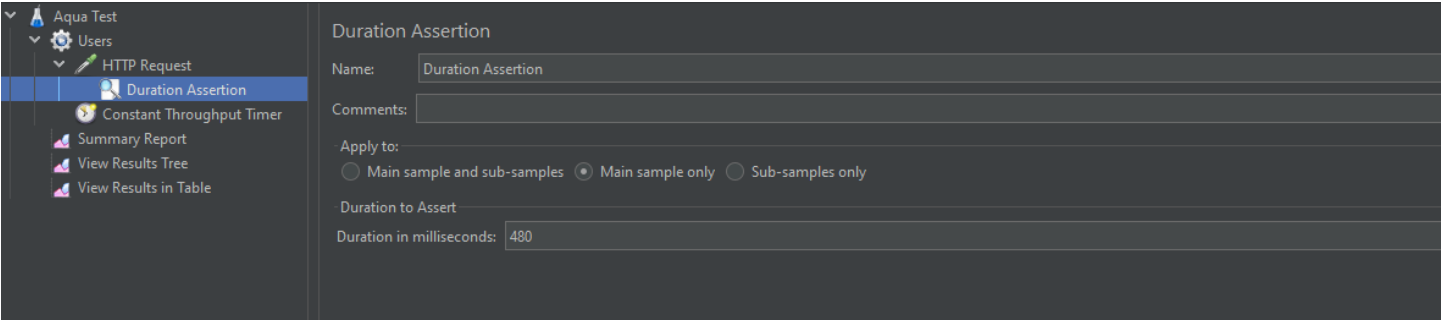


Рисунок 5. Настройка ограничения на время выполнения запроса

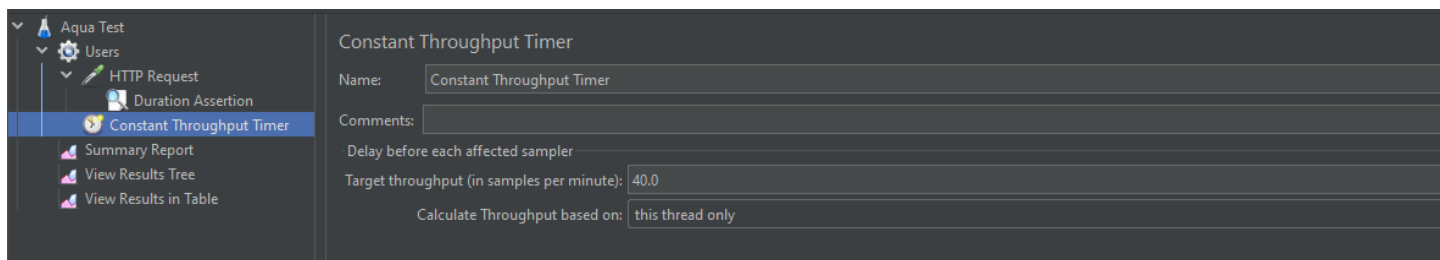


Рисунок 6. Настройка количества запросов в минуту, отправляемых одним пользователем

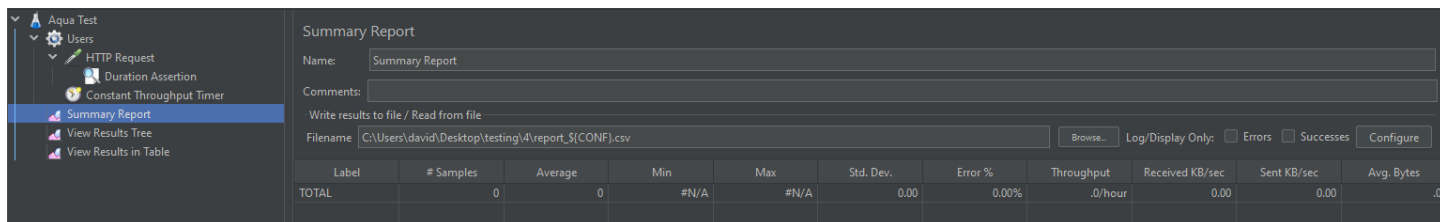


Рисунок 7. Настройка отчёта

Нагрузочное тестирование

Чтобы провести нагрузочное тестирование, например, с первой конфигурацией, нужно выставить CONF=1 и нажать кнопку Start. После генерации csv-файла нужно очистить записи, нажав кнопку Clear All.

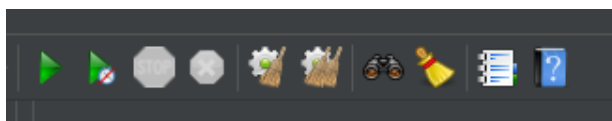


Рисунок 8. Кнопка Start с изображением зелёного треугольника (первая слева) и кнопка Clear All с изображением шестерни и двух венков (шестая слева).

Нагрузочное тестирование следует провести с конфигурациями 1, 2 и 3. После этого будут получены csv-файлы с записями о том, как прошло тестирование.

report_1.csv	24-May-21 7:30 PM	Microsoft Excel C...	1 KB
report_2.csv	24-May-21 7:31 PM	Microsoft Excel C...	33 KB
report_3.csv	24-May-21 8:33 PM	Microsoft Excel C...	136 KB

Рисунок 9. Три csv-файла, получившиеся в результате нагрузочного тестирования

Если очень интересно, можно заглянуть в файлы и посмотреть, что внутри.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	timeStamp	elapsed	label	response	responseTime	threadName	dataType	success	failureMessage	bytes	sentBytes	grpThread	allThread	URL	Latency	IdleTime	Connect
2	1.62E+12	221	HTTP Request	200 OK		Users 1-1	text	TRUE		367	155	1	1	http://localhost	221	0	1
3	1.62E+12	223	HTTP Request	200 OK		Users 1-2	text	TRUE		367	155	2	2	http://localhost	223	0	0
4	1.62E+12	215	HTTP Request	200 OK		Users 1-1	text	TRUE		367	155	2	2	http://localhost	215	0	0
5	1.62E+12	227	HTTP Request	200 OK		Users 1-3	text	TRUE		367	155	3	3	http://localhost	227	0	1
6	1.62E+12	216	HTTP Request	200 OK		Users 1-2	text	TRUE		367	155	3	3	http://localhost	216	0	0
7	1.62E+12	216	HTTP Request	200 OK		Users 1-1	text	TRUE		367	155	4	4	http://localhost	216	0	0
8	1.62E+12	237	HTTP Request	200 OK		Users 1-4	text	TRUE		367	155	4	4	http://localhost	237	0	1
9	1.62E+12	223	HTTP Request	200 OK		Users 1-3	text	TRUE		367	155	4	4	http://localhost	223	0	0
10	1.62E+12	231	HTTP Request	200 OK		Users 1-2	text	TRUE		367	155	5	5	http://localhost	231	0	0
11	1.62E+12	238	HTTP Request	200 OK		Users 1-5	text	TRUE		367	155	5	5	http://localhost	238	0	0
12	1.62E+12	241	HTTP Request	200 OK		Users 1-1	text	TRUE		367	155	5	5	http://localhost	241	0	0
13	1.62E+12	248	HTTP Request	200 OK		Users 1-4	text	TRUE		367	155	5	5	http://localhost	248	0	0
14	1.62E+12	219	HTTP Request	200 OK		Users 1-3	text	TRUE		367	155	6	6	http://localhost	219	0	0
15	1.62E+12	230	HTTP Request	200 OK		Users 1-6	text	TRUE		367	155	6	6	http://localhost	230	0	0
16	1.62E+12	228	HTTP Request	200 OK		Users 1-5	text	TRUE		367	155	6	6	http://localhost	228	0	0

Рисунок 10. Содержимое report3.csv

В файлах лежат сырые данные, на которые смотреть не очень полезно. Чтобы превратить их во что-то более удобное, для всех csv-отчётов нужно выполнить следующую команду.

```
jmeter.sh -g <source_csv> -o <out_dir_name>
```

В данном случае получилось вот так:

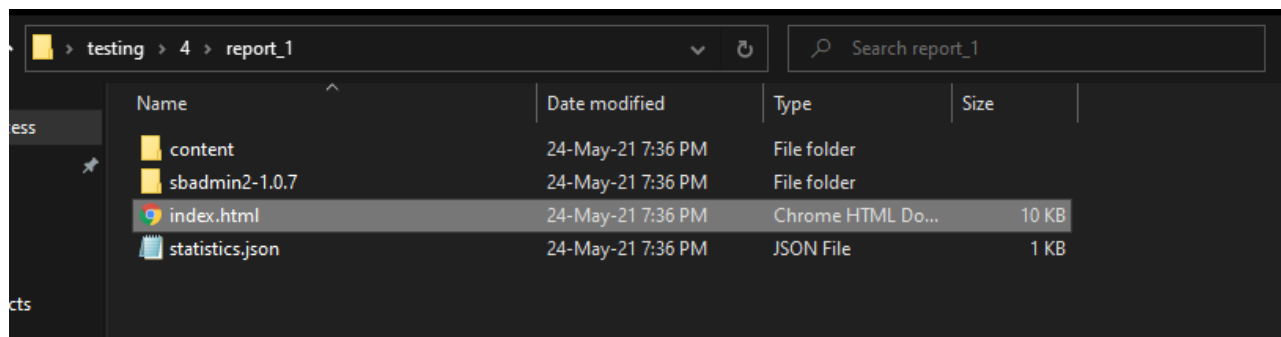
```
jmeter.sh -g report1.csv -o report1
jmeter.sh -g report2.csv -o report2
jmeter.sh -g report3.csv -o report3
```

После этих команд появятся три директории.

report_1	24-May-21 7:36 PM	File folder
report_2	24-May-21 7:36 PM	File folder
report_3	24-May-21 7:36 PM	File folder

Рисунок 11. Три директории

В каждом из них есть файл index.html.



Name	Date modified	Type	Size
content	24-May-21 7:36 PM	File folder	
sbadmin2-1.0.7	24-May-21 7:36 PM	File folder	
index.html	24-May-21 7:36 PM	Chrome HTML Do...	10 KB
statistics.json	24-May-21 7:36 PM	JSON File	1 KB

Рисунок 12. Файл index.html

Этот файл можно (и даже нужно) открыть в браузере. Нас интересует раздел Charts > Response Times > Time Vs Threads.

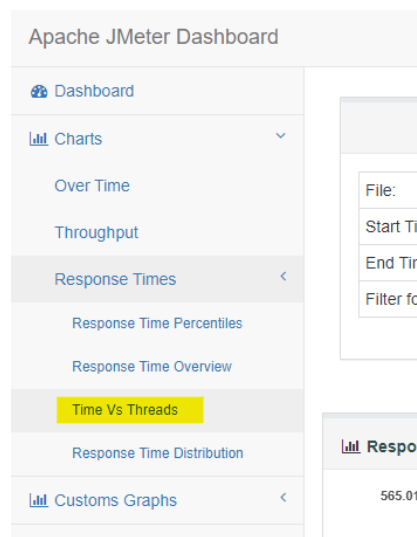


Рисунок 13. Раздел, в котором содержится нужный график.

Вот так выглядит график для первой конфигурации.

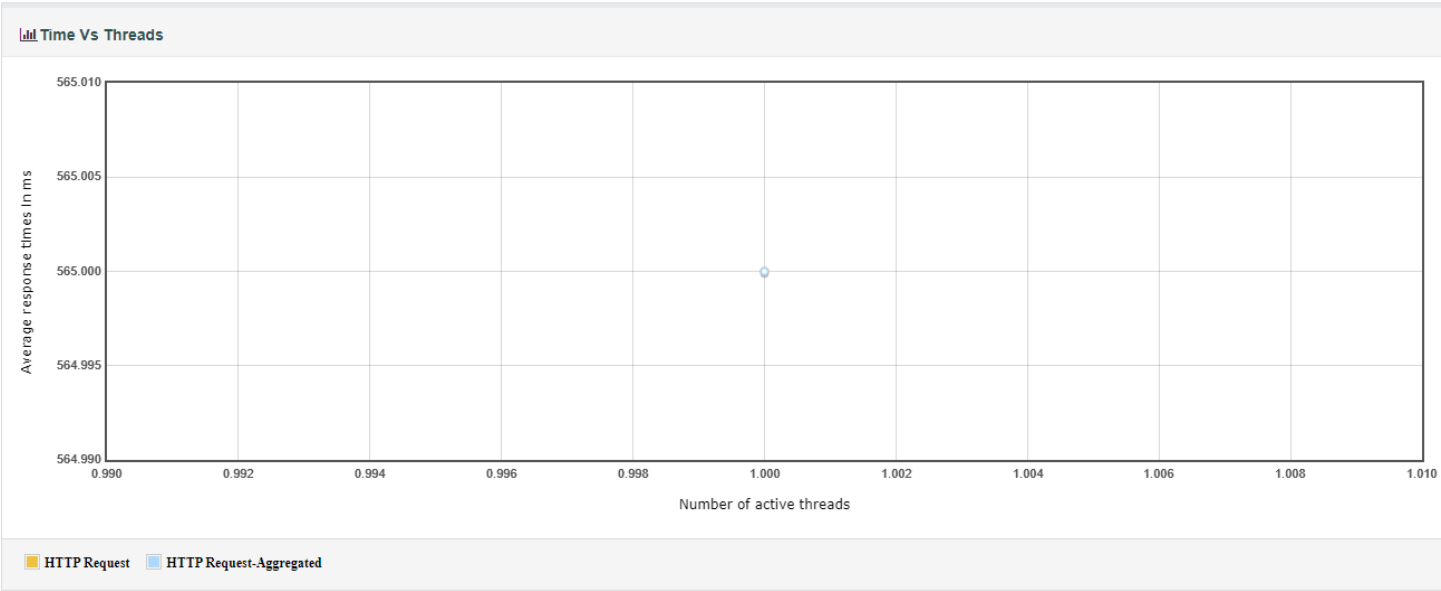


Рисунок 14. График зависимости среднего времени отклика от количества потоков при конфигурации 1.

На график не очень похоже, потому что время отклика при конфигурации 1 такое большое, что тестирование автоматически остановилось после запуска первого потока. Такая конфигурация не удовлетворяет требованиям.

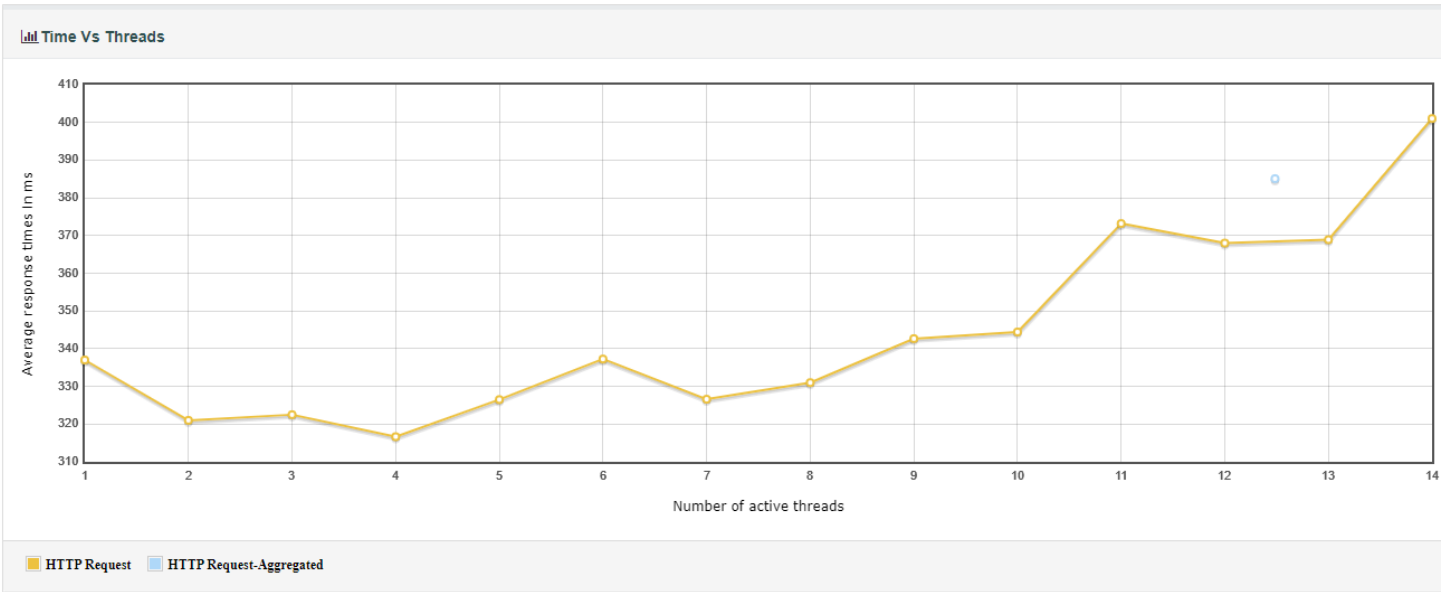


Рисунок 15. График зависимости среднего времени отклика от количества потоков при конфигурации 2.

На графике при конфигурации 2 можно видеть, что среднее время отклика увеличивается с увеличением количества потоков. Время отклика при конфигурации 2 превышает 480 мс, поэтому эта конфигурация тоже не удовлетворяет требованиям.

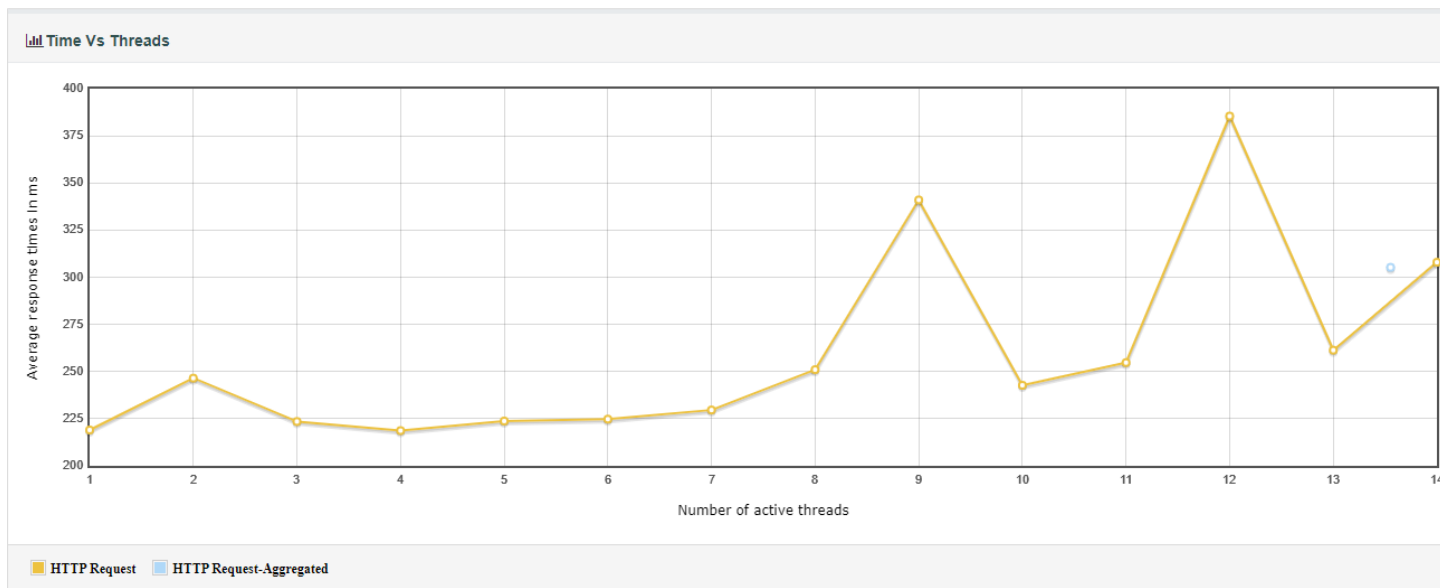


Рисунок 16. График зависимости среднего времени отклика от количества потоков при конфигурации 3.

На графике при конфигурации 3 значительно проявляется шум, обусловленный случайностью задержки, с которой пакеты от JMeter доходят до сервера, потому что для подключения к интернету компьютер с запущенным JMeter использует публичную университетскую сеть Wi-Fi. Вообще, время отклика и здесь иногда превышает 480 мс, но, учитывая, что на это время влияет в т.ч. задержка публичной сети Wi-Fi, была выбрана третья, самая дорогая, конфигурация.

Стресс-тестирование

Для стресс-тестирования выбрана конфигурация 3. Чтобы провести стресс-тестирование, нужно немного изменить конфигурацию.

Для начала удалим Duration Assertion, потому что теперь не нужно проверять время отклика.

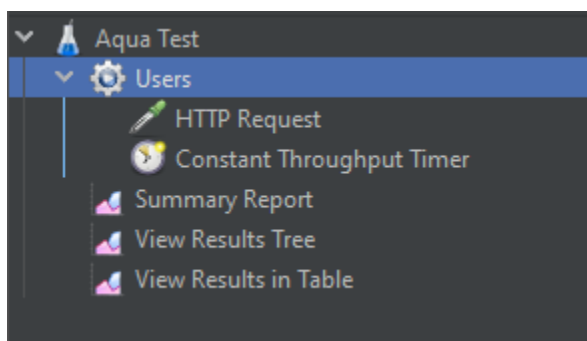


Рисунок 17. Здесь больше нет Duration Assertion

Затем нужно сделать так, чтобы JMeter отправлял запросы, постепенно увеличивая количество потоков. Из предположения, что с 300-400 потоками сервер с высокой вероятностью не справится, было выставлено 1000 потоков. Время наращивания равно 4000 секунд. Это значит, что каждые 4 секунды количество работающих потоков будет увеличиваться на 1. Кроме того, нужно настроить JMeter не прекращать тестирование в случае ошибки: нужно узнать, при каком количестве потоков сервер вернёт HTTP 503.

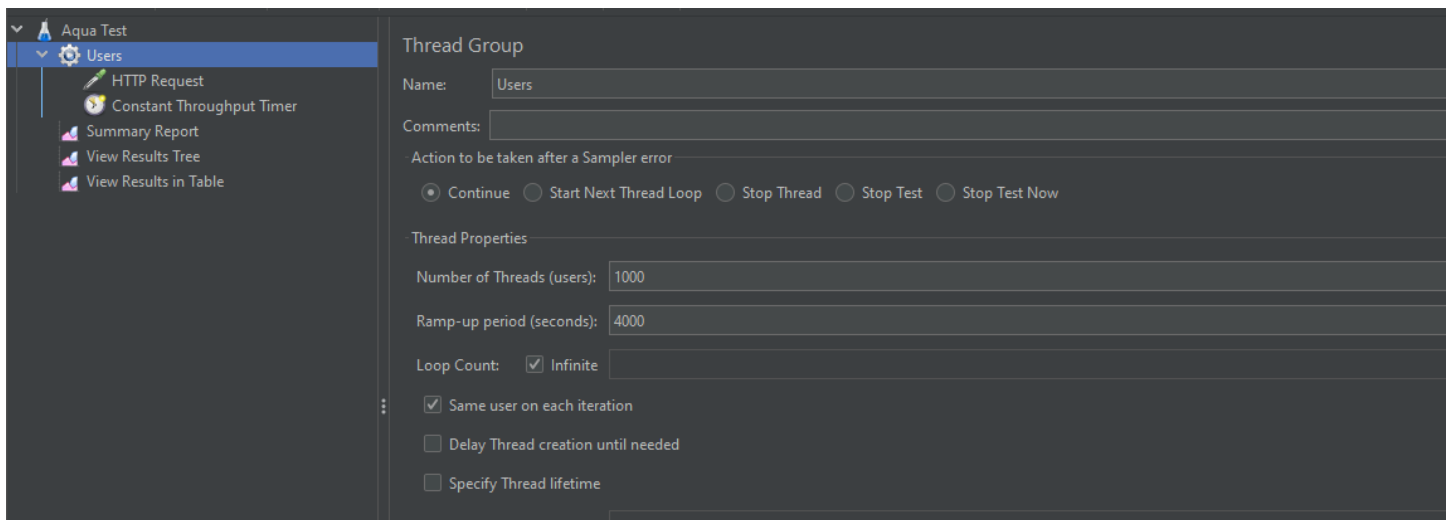


Рисунок 18. Новая конфигурация для стресс-тестирования

Запускаем JMeter, переходим в View Results Tree и наблюдаем за происходящим.

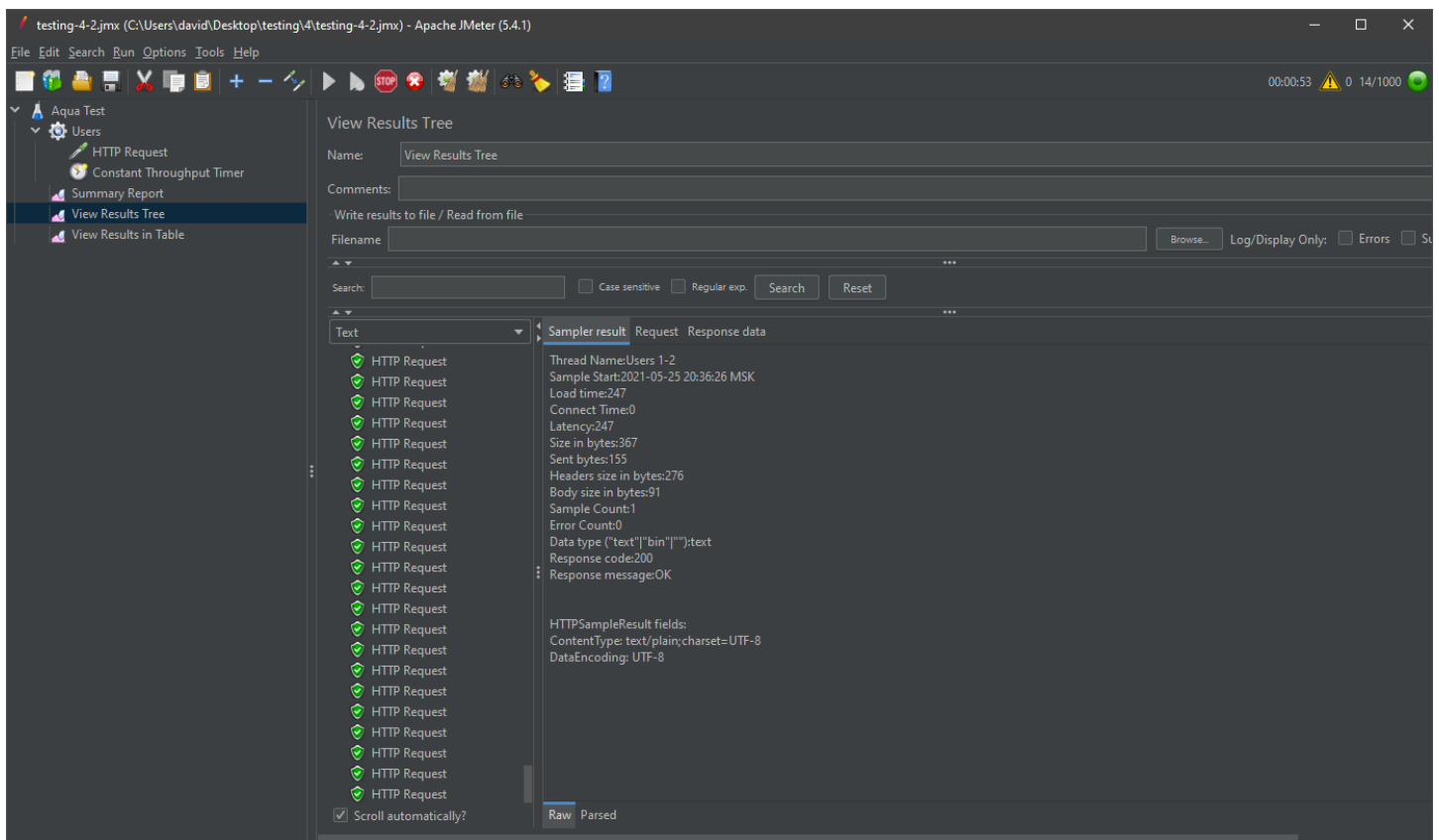


Рисунок 19. Стресс-тестирование

Через некоторое время в ответ на очередной запрос сервер вернёт HTTP 503.

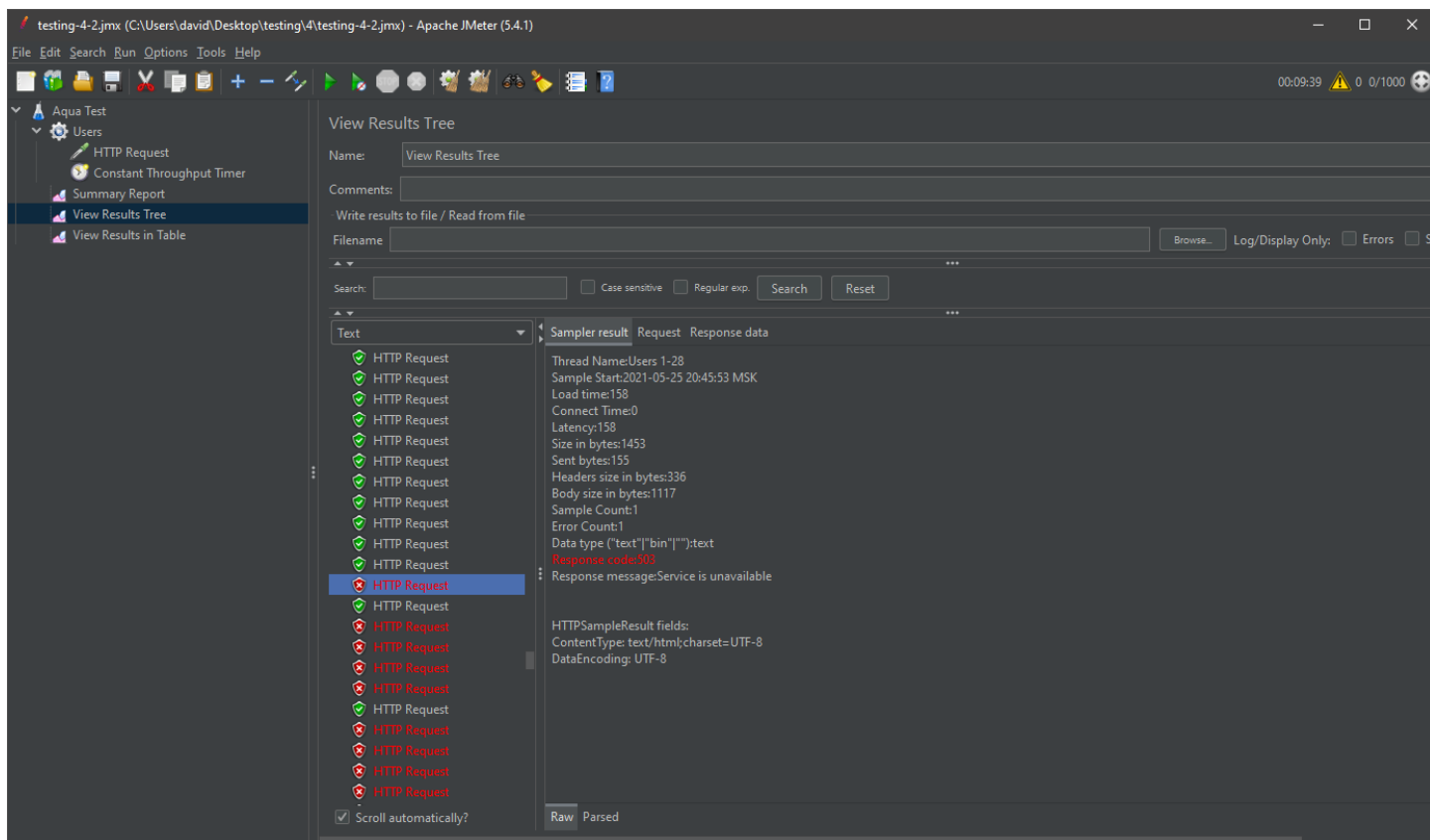


Рисунок 20. Сервер вернул HTTP 503

Это произошло в момент, когда работало 145 потоков.

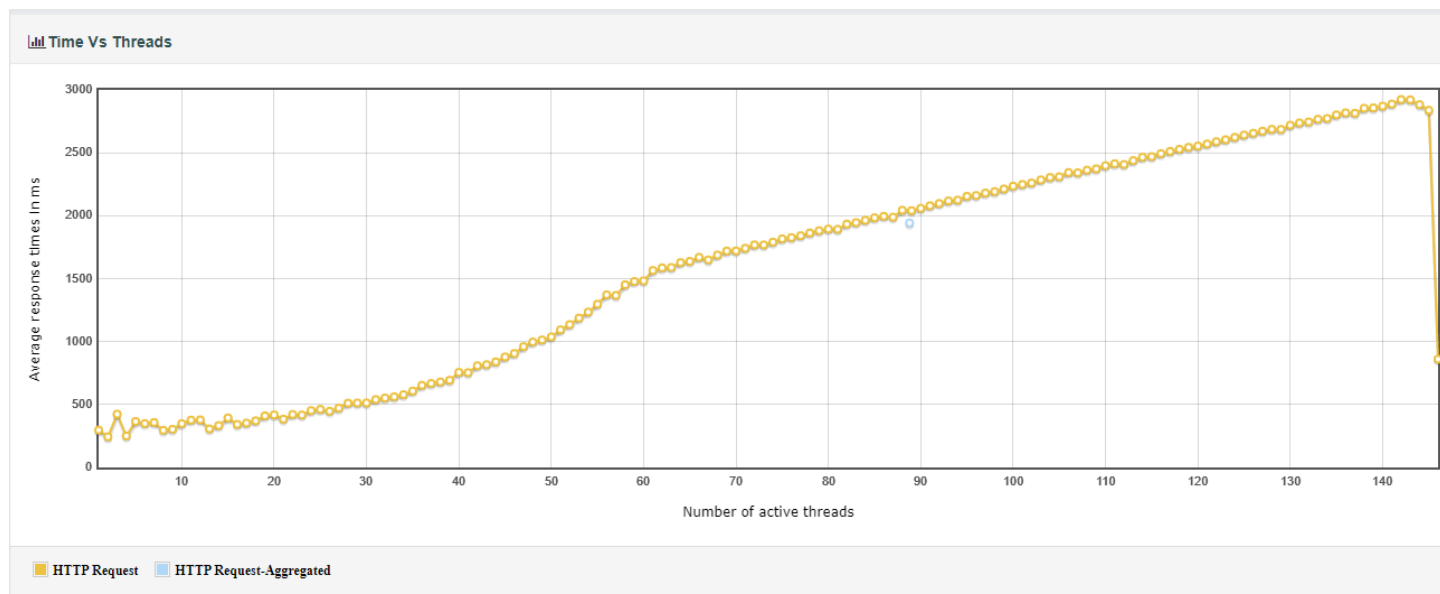


Рисунок 21. График зависимости времени обработки запроса от количества потоков

Репозиторий с конфигурациями и прочим

<https://github.com/david-d25/testing-lab4>

Вывод

Сделав эту лабораторную работу, я освоил работу с JMeter, провёл нагрузочное тестирование, провёл стресс-тестирование, сгенерировал красивые отчёты с графиками, начал пользоваться университетским Wi-Fi вместо того, чтобы раздавать интернет самому себе с телефона (потому что ping мобильной сети, как оказалось, ещё больше, чем ping университетского Wi-Fi) и получил представление о том, как проверяют соответствие продукта требованиям производительности со стороны его API. Думаю, этот инструмент может пригодиться мне в будущем, если я захочу сделать какой-нибудь веб-проект.