Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи і засоби тестування веб-додатків

(тема)

Виконав:
студент II курсу, групи КСМм-19-1
Євсюков В.В.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Горбачов В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Євсюкову Владиславу Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи і засоби тестування веб-додатків

затверджена наказом по університету від " 30 " жовтня 2020 р. № 1487 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи

1. Розробка сценаріїв тестування веб-додатків
2. Використання засобу тестування Jmeter
3. Автоматизація тестування

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд методів тестування веб-додатків
2. Вибір та обґрунтування методів та засобів дослідження
3. Розробка автоматизованих сценаріїв тестування веб-додатків
4. Проведення автоматизованих сценаріїв тестування
5. Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд презентація – 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд методів та засобів тестування	03.11.20-09.11.20	
	веб-додатків		
2	Вибір та обґрунтування методики дослідження	10.11.20-17.11.20	
3	Вибір інструментальних засобів	18.11.20-23.11.20	
4	Проведення експериментів	24.11.20-01.12.20	
5	Оформлення матеріалів атестаційної роботи	02.12.20-07.12.20	
6	Подання атестаційної роботи керівникові та	08.12.20-09.12.20	
	її попередній захист		
7	Подання атестаційної роботи на рецензування	10.12.20-11.12.20	

Дата видачі завдання 2 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Горбачов В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 102 с., 51 рис., 3 табл., 1 дод., 19 джерел.

ВЕБ-ДОДАТОК, МЕТОДИ ТЕСТУВАННЯ, ЗАСОБИ ТЕСТУВАННЯ, ТЕСТУВАННЯ ПРОДУКТИВНОСТІ, АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ, JMETER, ХАМРР, ВЕБ-СЕРВЕР, HTTP.

Метою атестаційної роботи є визначення методів та засобів тестування, та автоматизація методів тестування для своєчасного виявлення помилок в роботі сайту.

У ході виконання атестаційної роботи були розглянуті, засоби та методи тестування веб-додатків.

Виконано підбір програмного забезпечення необхідного для виконання роботи. Розроблено сценарії автоматизації тестування продуктивності веб-додатків та проведено аналіз результатів тестування, для виявлення можливостей веб-додатка.

ABSTRACT

Master's thesis 102 pages, 51 figures, 3 tables, 1 appendices, 19 sources.

WEB APPLICATION, TESTING METHODS, TESTING TOOLS, PERFORMANCE TESTING, TESTING AUTOMATION, JMETER, XAMPP, WEB SERVER, HTTP.

The major goal of this thesis is to determine the methods and resources of testing and automation of testing methods for timely detection of errors in the site.

Web application testing tools and methods were reviewed and analyzed. Selection of the software necessary for performance of work is executed. Scenarios for automating web application performance testing have been developed and testing results have been analyzed to identify web application capabilities.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

CPU – центральний процесор (англ., Central processing unit)

FTP – протокол передачі файлів (англ., File Transfer Protocol)

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

IDE – інтегроване середовище розробки (англ., Integrated development environment)

JSON – запис об'єктів JavaScript (англ., JavaScript JavaScript Object Notation)

TCP – протокол керування передачею (англ., Transmission Control Protocol)

ВСТУП

Метою атестаційної роботи є визначення методів та засобів тестування, та автоматизація методів тестування для своєчасного виявлення помилок в роботі сайту.

Для початку розглянемо, чому необхідне тестування. Тестування є важливим, оскільки всі ми допускаємо помилки. Деякі з цих помилок не важливі, але деякі є дорогими або можуть загрожувати життю. Тому ми повинні перевіряти все, що виробляємо, для того щоб усунути помилки.

Тестування має багато переваг, і одна з найважливіших - це економічна ефективність. Тестування може заощадити гроші в довгостроковій перспективі. Розробка програмного забезпечення або веб-додатків складається з багатьох етапів, і якщо помилки виявляються на попередніх етапах, їх виправлення коштує набагато менше. Ось чому важливо якомога швидше провести тестування.

Питання, що стосуються тестування веб-додатків є дуже актуальними. Це пов'язано з тим що на сьогоднішній день веб-додатки використовуються в кожній сфері життя. Тестування продуктивності не обов’язково має відображати дефекти програми. Воно повинно забезпечити, щоб веб-додаток працював належним чином, незалежно від коливань мережі, наявності смуги пропускання або навантаження трафіку.

Отже, проектування та виконання цих тестів є актуальним та критичним для забезпечення стабільності веб-додатку. Уявіть, що ви відкрили веб-додаток онлайн банку, і він має одну з найменших комісій на ринку за швидке відправлення грошей. Ви спробували “Зареєструватися”, і з’явилося повідомлення “Помилка”, або ви намагаєтеся скористуватися веб-додатком, на який одночасно заходить велика кількість людей, і доступ до цього сайту став обмежений. Зараз через цю проблему не тільки ви, але й багато інших користувачів не може "зареєструватися" та користуватися цим продуктом.

Отже, їхній бізнес вже втратив гроші, оскільки транзакції не укладаються і не вирішують проблему з самого початку. Таким чином, користувачі знайдуть інший подібний веб-додаток, який працює, і, можливо, ніколи не повернеться до нього через поганий досвід. Тестування, по суті, допомагає заощадити час і гроші в довгостроковій перспективі, оскільки проблеми вирішуються до виникнення більших проблем. Витрати на технічне обслуговування також нижчі, і, зрештою, якщо продукт працює на 100% як слід, експоненціальна шкода не завдається вашому бізнесу з точки зору витрат.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Актуальність проблеми

Проживаючи в цифровому світі, використання веб-додатків суттєво спрощує життя. Веб-додатки використовуються в різних сферах життя від розважальної сфери до сфери охорони здоров'я. Веб-сайт може бути використаний для реалізації багатьох різних маркетингових стратегій, які допоможуть розвитку бізнесу. Коли надається хороший сервіс або товар, позитивний відгук може поширити бізнес або сервіс. Веб-сайт не лише забезпечує надійність, але й допомагає скласти позитивне враження про те, що компанія успішніша.

Маючи популярний веб сайт можна зіткнуться з різними проблемами, такими як: помилки в коді, вразливості і так далі. В нашому випадку ми будемо розглядати тестування продуктивності веб-додатку. Як і більшість стратегій тестування, метою тестування навантаження для веб-додатків є виявлення вразливостей. Тим не менше, тестування навантаження – це спеціалізований метод виявлення проблем, безпосередньо пов'язаних із завантаженням веб-додатку користувачами.

Бувають випадки, коли веб-сайт має більше трафіку, ніж зазвичай, і ми хочемо переконатися, що сервер може обробляти швидкість та обсяг клієнтів. Ми повинні знати, з якою кількістю клієнтів може працювати веб-сайт, поки він не перестане працювати. В розділі “огляд літератури” будуть більш детально розглянуті приклади веб-додатків, для яких час є критичним параметром.

1.2 Основні поняття і визначення

Наступні визначення описують термінологію, яка використовується в атестаційній роботі. Визначення взяті з обраної літератури.

Веб-сервер – це комп’ютерна система, що розміщує веб-сайти. Веб-сервер обробляє вхідні мережеві запити через HTTP та кілька інших пов'язаних протоколів[6].

Тестування продуктивності – це тип тестування який визначає або перевіряє швидкість, масштабованість або характеристики стійкості системи чи програми, що тестується[1].

Тестування витривалості – це вид перевірки продуктивності, на який орієнтовано визначення або перевірка експлуатаційних характеристик продукту, що випробовується, під впливом моделей робочого навантаження на тривалий проміжок часу[1].

Тестування динамічного навантаження – це тип тестування продуктивності, орієнтований на визначення або підтвердження експлуатаційних характеристик продукту під робочим навантаження, яке неодноразово збільшується понад очікуваних параметрів на короткі проміжки часу[1].

Метрики – це вимірювання, отримані в результаті роботи тестів[1].

Час відповіді – це показник того, наскільки реагує веб-додаток чи підсистема на запит клієнта[1].

Пропускна здатність – це кількість одиниць роботи, яку можна виконати за одиницю часу, наприклад, кількість запитів на секунду[1].

1.3 Важливість проведення тестування продуктивності

На сьогодні багато підприємств електронної комерції покладаються на кіберпростір, щоб отримати левову частку свого доходу в тому, що є найбільш конкурентним середовищем, яке тільки можна собі уявити.

Якщо кінцевий користувач сприймає погану ефективність веб-сайту, то скоріше за всього користувач більше не відкриє цей веб-додаток. Додатки електронної комерції також дуже вразливі до раптових стрибків попиту, як це виявили не лише кілька гучних роздрібних компаній в пікові періоди року [2].

Тестування продуктивності допомагає виявити вузькі місця в системі, встановити базову лінію для майбутнього тестування, підтримати зусилля з налаштування продуктивності та визначити відповідність цілям та вимогам щодо ефективності.

Розробники системи, спонсори та власники мають певний інтерес до веб-сайту. Очікування цих зацікавлених сторін веб-сайту є більшим, ніж очікування користувачів веб-сайту.

Їхні очікування можуть полягати в таких аспектах[12]:

- доступність веб-сайту 24 години на добу;
- швидкий час відгуку;
- висока пропускна здатність, коли користувач бере участь у декількох транзакціях;
- належне використання пам'яті як клієнта, так і сервера;
- адекватне використання процесора різними системами, що використовуються для транзакцій;
- належне використання пропускної здатності мережі;

Відповідно до статті Jérôme Loisel за статистикою 43% користувачів настільних комп'ютерів здаються, коли завантаження сторінки займає більше 3 секунд. Користувачі мобільних пристроїв трохи терплячіші, але не набагато більше. Вони йдуть, якщо час відгуку перевищує 5 секунд. Майже 9 з 10 людей ніколи не повертаються, якщо відчувають повільність. Вся робота, витрачена на маркетинг та інтерфейс веб-сайту, може бути зіпсована, якщо Інтернет-магазин не швидко реагує[4].

Власні дослідження підрозділу Gomez виявляють відсутність лояльності відвідувачів. Аналізуючи дані про залишення сторінок на більш ніж 150 веб-сайтах і 150 мільйонах переглянутих сторінок, Gomez виявив, що збільшення

час відгуку сторінки з 2 до 10 секунд збільшив рівень залишення сторінки на 38%. На рисунку 1.1 зображена відповідність користувачів які покинули сторінку до зміни часу відгуку[13].

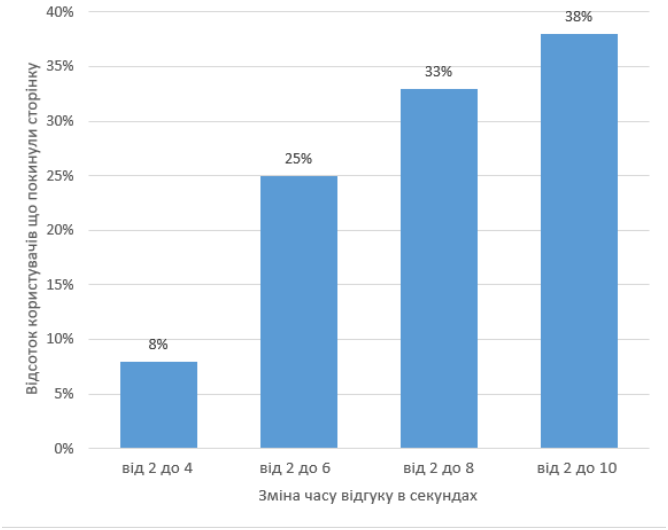

Рисунок 1.1 – Час відгуку сприяє збільшенню залишення сторінки

Кожен веб-додаток повинен мати нормальний час відгуку до 3 секунд. Швидкий час відгуку, нижче ніж 1,5 секунди, є дуже критичним для таких сайтів як: Google.com, Bing.com, yahoo.com. Це пов’язано з тим що люди замість довгого очікування можуть вибрати інший пошукової сервіс.

В дослідженні Gomez дослідженні наводиться наступна інформація про важливість кожної секунди для веб-додатків. Навіть невеликі цифри можуть мати великий вплив на ваш бізнес. Згідно з дослідженнями Абердинської дослідницької групи, середній показник доступності веб-додатку становить 97,8%. з першого погляду це число має хороший показник, але розглянемо те як дійсно виглядають ці два відсотки, це означає, що веб-додаток не працює протягом 8 днів щороку. Для сайту електронної комерції, який приносить 100000 доларів на день, це призводить до втрати 800000 доларів щорічного доходу. Те саме дослідження Абердіна показало, що середній вплив затримки на 1 секунду означав зменшення конверсії на 7%. Для сайту електронної

комерції на суму 100000 доларів на день затримка в одну секунду означає втрату доходу на 2,5 мільйона доларів за рік[13].

Microsoft Bing, і Google провели тести, в яких штучно збільшили час відповіді сервера. Bing збільшив час відгуку з 50мс. до 2000мс. Результати тесту були настільки чіткими, що закінчили його раніше, ніж планувалося спочатку. Показних часу на клік користувача збільшився до 3100мс. Теорія полягає в тому, що поки сторінка завантажується, користувач відволікається і не займається сторінкою. Іншими словами, вони втратили повну увагу користувача. Google провів подібний експеримент, коли перевіряв затримки від 50 мс до 400 мс. Щоденний пошук на кожного користувача на період проведення тесту зменшився на 1%. Найцікавіше в цьому тесті – це тривалий вплив експерименту на користувачів навіть після його закінчення. Деякі користувачі так і не відновились, особливо ті, що мали більшу затримку в 400 мс[14].

У 2009 році сайт електронної комерції та сайт засобів масової інформації взяли участь в інноваційному дослідженні, яке випадковим чином розділило їх 14000 відвідувачів на дві групи: одна відвідала оптимізований веб-сайт, а інша – сайт, який залишився без поліпшень. Поведінка клієнтів і бізнес результати бізнесу – були чітко пов’язані з роботою веб-сайту. Результати тесту знаходяться в таблиці 1.1[13].

Таблиця 1.1 – Результат поведінки користувача

Поведінка користувача	Показники	
	оптимізований сайт	неоптимізований сайт
Показник відмов, %	13.38	14.35
Кількість переглянутих сторінок	15.64	11.04
Середній час перебування на сайті, хв	0:30:10	0:23:25

Як випливає з таблиці, оптимізований сайт перевершує неоптимізований сайт за ключовими показниками залучення відвідувачів.

Найважливіший показник для сайту електронної комерції стосується результатів бізнесу. Оптимізація продуктивності покращила коефіцієнт конверсії на 16,07%, а кількість замовлень – на 5,51%[13]. Очевидно, що покращення веб-додатка в технічному плані суттєво вплинуло на його ефективність.

Jérôme Loisel визначає що поліпшуючи час завантаження сторінок, не тільки покращується взаємодія з користувачем, але й збільшуєте шанси отримати кращий рейтинг у пошукових системах. Google зменшить кількість сканерів, які він надсилає на сайт, якщо сервер працює повільніше, ніж дві секунди. Amazon виявив, що затримка 100 мс коштує їм 1% продажів. У 2006 році Google експериментував, та показував на головній сторінці 30 результатів, замість 10. Час відгуку збільшився з 400 до 900 мілісекунд. Це спричинило падіння трафіку на 20%[4].

На рисунку 1.2 за даними дослідницької групи Gomez зображений вплив однієї секунди на веб-додаток. Як видно з цього рисунку навіть одна секунда час має сильний вплив на успіх веб-додатку[13].

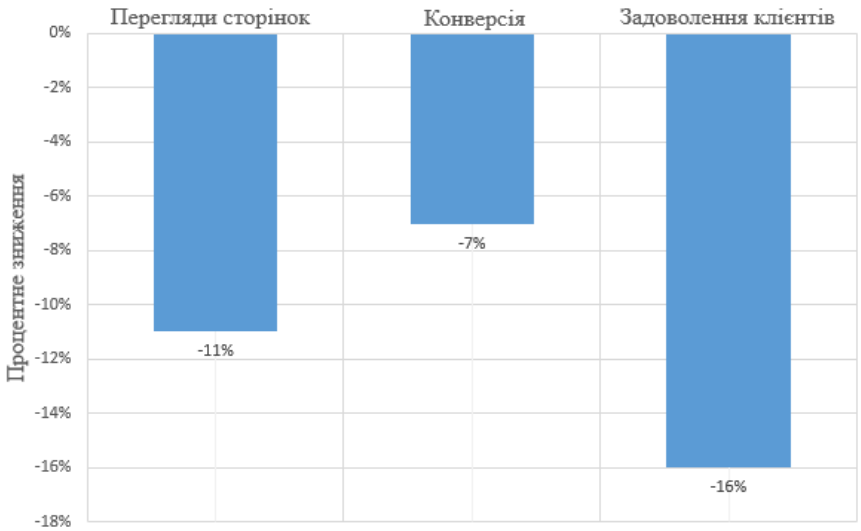

Рисунок 1.2 – Вплив часу відгуку на веб-додаток

Наступний приклад стосується веб-додатка під назвою Shopzilla. З 20 – 29 мільйонами унікальних відвідувачів на місяць, 100 мільйонами показів на день та 8000 пошукових запитів на секунду, Shopzilla, безумовно, відповідає вимогам Інтернет-мережі. Хоча його статистика вражала, Shopzilla розпочала 16-місячну програму поліпшення технічної складової, спрямовану на підвищення продуктивності. В результаті його роботи доступність зросла з 99,65% до 99,97%. Середній час завантаження сторінки значно покращився з 6 секунд до надзвичайно швидких 1,2 секунди. Неймовірно, що капітальний ремонт насправді вимагав меншої фізичної інфраструктури, та успішний сайт збільшив дохід на 5 – 12%[13]. Із цього прикладу можна зрозуміти те що правильний підхід до розробки сайту може зменшити вимоги до сервера на якому розміщений сайт, і зменшити час відповіді, що призведе до збільшення зручності користування веб-додатком, і в майбутньому до збільшення конверсії.

Все більше і більше споживачів отримують доступ до веб-сайтів через смартфони та інші мобільні пристрої. Хоча мобільний пристрій сильно відрізняється від комп’ютера, очікування споживачів залишаються незмінними. Згідно нещодавнього дослідження Gomez, 58% користувачів мобільних пристроїв очікують, що сайти завантажуватимуться так само швидко, як на домашніх комп’ютерах. Гірше того, переважна більшість, 61%, сказали, що низька продуктивність зробить їх менш вірогідними знову відвідувати мобільний веб-сайт[13].

1.4 Визначення причин тестування

Основні причини тестування продуктивності певного проекту не завжди очевидні ґрунтуючись лише на баченні та контексті. Правильність визначання причин перевірки продуктивності має вирішальне значення для можливості визначити, які дії з тестування продуктивності додадуть найбільшої цінності проекту.

Причини проведення тестування продуктивності автори Sai Matam та Jagdeep Jain показують на прикладі серверів Netflix. Дуже рідко регіон AWS стає недоступним, але це трапляється. 20 вересня 2015 року Amazon DynamoDB зіткнувся з проблемою доступності у своєму регіоні US-EAST-1. Ця нестабільність спричинила вихід із ладу понад 20 додаткових служб AWS, які залежали від DynamoDB. Деякі найбільші веб-сайти та програми Інтернету були періодично недоступними протягом шести-восьмигодинного вікна того дня. Це мало мінімальний вплив на Netflix, завдяки сервісу Chaos Kong, який симулює раптові навантаження. Netflix справді відчув короткий проміжок доступності у постраждалому регіоні, але це уникнуло значного впливу, оскільки Chaos Kong підготував їх до подібних інцидентів. Регулярно проводячи тести, що імітують відключення на рівні регіону, вони змогли завчасно виявити будь-які системні слабкі місця та виправити їх. Коли US-EAST-1 фактично став недоступним, їх система вже була достатньо надійною, щоб справлятись із збоєм трафіку[9].

Причини проведення тестування ефективності часто виходять за рамки списку результатів критерії приймання. J.D. Meier та Carlos Farre визначають що кожен проект має різні причини для прийняття рішення про включення або не включення тестування продуктивності як частини його процесу. Приклади можливих причин інтегрованої роботи тестування частини проекту включає наступне[1]:

- покращити тестування продуктивності, поєднавши тестерів продуктивності з розробниками;
- оцінити ефективність алгоритму;
- відстежити тенденції використання ресурсів;
- виміряти час відгуку;
- зібрати дані для масштабованості та планування потужності.

Gomez Peak Time використали дослідження проведене Equation Research на 1500 користувачів (лютий 2010), що підтверджує негативний вплив низької продуктивності на користувачів[13]:

- у часи піку більше 75% онлайн-споживачів переходили на сайт конкурента, а не терпіли затримки;

- 88% онлайн-споживачів рідше повертаються на сайт після поганого досвіду;

- майже половина з них висловили менш позитивне сприйняття компанії в цілому після одного невдалого досвіду;

- більше третини розповіли іншим про свій невтішний досвід.

Як видно з цього дослідження мати хороші показники продуктивності, дуже важливо для успіху веб-додатка і задоволення досвіду використання сайту користувачами.

Одна із причин тестування полягає в тому що менеджери сайтів повинні мати видимість сайту, як його бачать користувачі, щоб контролювати ефективність своєї програми. Знайомі функції електронної комерції, такі як кошики для покупок та інтерактивні каталоги товарів, тепер залежать від складних програм, які працюють у браузерах кінцевих користувачів, щоб зробити їх більш динамічними та цікавими.

Ian Molyneaux пише що багато компаній недооцінюють популярність своїх нових веб-додатків. Це частково тому, що вони розгортають їх, не беручи враховуючи фактор новизни веб-додатку[2]. Коли щось блискуче та нове, людям загалом це цікаво, і тому вони з’являються натовпами, особливо там, де запуск програми супроводжується пресою та рекламою у ЗМІ. Тому 10 000 звернень, які ретельно розрахувалися протягом першого дня розгортання, раптом перетворюються в 1 000 000 звернень, і інфраструктура веб-додатка занепадає

Загалом корисно виявити причини проведення тестування продуктивності дуже рано на початку проекту. Оскільки ці причини неодмінно будуть змінюватимуться в міру прогресування проекту то слід регулярно проводити тестування продуктивності, для своєчасного виявлення вузького місця в продуктивності веб-додатку.

1.5 Автоматизація тестування

У багатьох компаніях тестування продуктивності не є пріоритетом, доки воно не стане критичним. Інженерна команда повинна пришвидшити тестування продуктивності в загалом та за допомогою засобів тестування продуктивності за дуже короткий час. Незважаючи на розуміння того, наскільки критичним є тестування продуктивності веб-додатків.

Як пише Jonathan Rasmusson, для розробників автоматичне тестування стосується швидкості веб-додатка.[7]. Автоматизація тестів, як правило, передбачає як ручну, так і інструментальну діяльність. Інструменти не допоможуть без належних даних для тестування продуктивності та вичерпного планування.

Ми не можемо провести ефективне тестування продуктивності без використання автоматизованих інструментів тестування. Ian Molyneaux каже що навіть якщо залучити 100 співробітників для проведення тесту і стратегічно розташувати людей із секундоміром, то з цього нічого не вийде. Ви ніколи не зможете повторити один і той же тест двічі пише автор. Більше того, змусити працівників працювати протягом доби, якщо ми виявимо проблеми, є порушенням прав людини[2].

Для проведення автоматичних сценаріїв тестування був обраний додаток Jmeter. Зі слов Sai Matam та Jagdeep Jain JMeter – це дуже здібний інструмент. Цей інструмент постійно переглядався з моменту його створення в 2001 році. Комерційні інструменти можуть мати кращі користувальницькі інтерфейси або більш красиві звіти, але JMeter можна налаштувати для забезпечення тих самих можливостей. JMeter можна запускати в «розподіленому режимі» в хмарі, створюючи навантаження на тисячі користувачів. JMeter постачається з ліцензією Apache, яка не має жодних обмежень щодо використання, розповсюдження або модифікації[9].

Успішна автоматизація тестування продуктивності повинна забезпечити миттєві результати роботи та повинна зменшити людські помилки, наскільки

це можливо, ізолюючи людські втручання. Крім того, перевірка результативності повинна бути автоматизована таким чином, щоб новачок або менш кваліфікована людина могла провести тест пише B. M. Subraya[12].

Як і аналіз у реальному часі, можливості інструменту тестування продуктивності значною мірою визначатимуть, наскільки буде легко розшифрувати результати проведеного тестування.

Автори Vinayak Hegde та Pallavi M.S. вказують на деякі типові проблеми, з якими стикаються команди з контролю якості протягом тестування продуктивності[10]:

- закупівля дорогих ліцензій на інструменти;
- витрати та технічні проблеми, з якими стикалися під час оновлення до новіші версії інструментів тестування продуктивності;
- недоступність версії сумісного інструменту тестування продуктивності;
- строгі терміни проекту.

Для виконання тестування продуктивності Ian Molyneaux пропонує наступні вказівки[2]:

- створення та перевірка сценаріїв тестування продуктивності;
- виконання тесту продуктивності;
- збір даних та видалення програмного забезпечення;
- проведення остаточного аналізу та складання звітності;
- проведення повторного тестування, якщо потрібно.

Автори Ms. S. Sharmila1 та Dr. E. Ramadevi пишуть що більшість тестів продуктивності залежать від набору заздалегідь визначених, та узгоджених кінцевих метрик. Знаючи цілі тестування з самого початку допомагає зробити процес тестування більш ефективним. Ми можемо оцініть ефективність нашої програми, порівнявши її з цілями тестування. Цілі діяльності часто включають наступне[11]:

- час відгуку;
- пропускна здатність;

- використання ресурсів (процесор, дисковий ввід та вивід, та пам'ять);
- кількість віртуальних користувачів;

До цього списку можна додати такий параметр, як кількість помилкових запитів отриманих від протестованого веб-додатку.

Ian Molyneaux писав що час відгуку має дуже важливу роль в продуктивності веб додатку. Після досягнення цільової паралельності або пропускної здатності, програма повинна дозволити кінцевим споживачам своєчасно виконувати свої завдання[2].

Доступність теж має важливу роль в роботі веб-додатка. Вона визначає який відсоток користувачів може успішно отримати доступ до сторінки або здійснити транзакцію в розумні терміни, не зазнаючи помилок. Послідовність визначає наскільки послідовною є ефективність веб-додатку за географічними показниками, мережами та часом доби.

Jonathan Rasmusson писав що автоматизовані тести які ми проводимо, повинні бути дуже детермінованими[7]. Це означає послідовність. Тести повинні працювати точно так само та надійно. Кожного разу при повторному проведенні одного і того ж тесту ми повинні отримувати такі ж результати як і раніше.

Одна з величезних переваг використання автоматизованих засобів тестування продуктивності – це те що результати кожного тестового запуску зберігаються для подальшого використання. Це означає, що ми можемо легко порівняти два або більше наборів результатів тесту і побачити, що змінилося. Насправді багато інструментів надають можливість шаблонування, щоб ми могли заздалегідь визначити види порівняння, які хочемо бачити.

1.6 Кроки тестування

J.D. Meier разом з великою кількістю інших авторів вказують на наступні кроки тестування продуктивності які зображені на рисунку 1.3[1]. Наведений нижче зразок є загальним серед організацій, що використовують

каскадну модель розробки. Такі самі практики зазвичай зустрічаються в інших моделях розвитку, але можуть бути не такими чіткими чи явними.

Перш ніж розглядати будь-яке тестування продуктивності, потрібно переконатися, що ваша програма функціонально стабільна. Ian Molyneaux пише що це може здатися очевидним, але занадто часто тестування продуктивності перетворюється на неприємні вправи з виправлення помилок, при цьому час, виділений на проект, швидко скорочується[2].

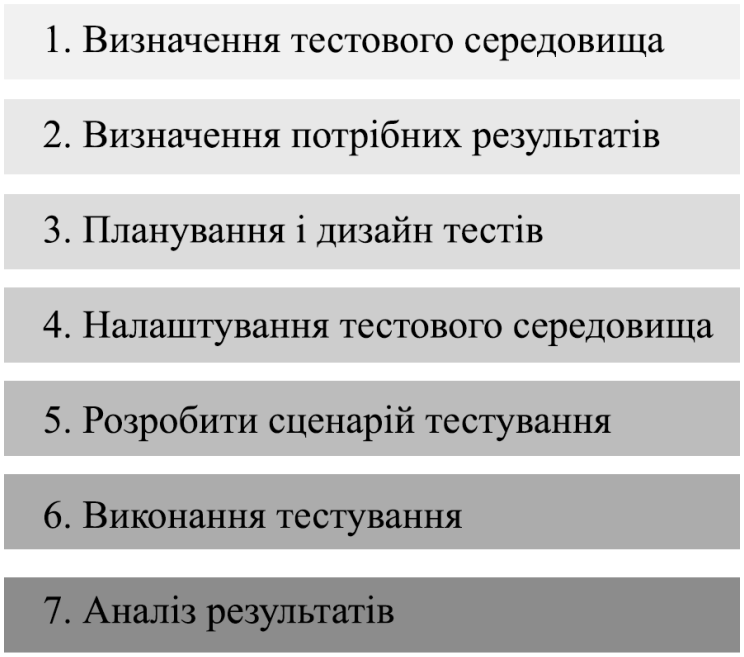

Рисунок 1.3 – Кроки тестування

Визначення тестового середовища: Потрібно визначити фізичне тестове середовище та виробниче середовище, а також інструменти та ресурси, доступні тестовій групі. Фізичне середовище включає апаратне, програмне забезпечення та конфігурації мережі.

J.D. Meier пише що в ідеальних умовах, якщо метою є визначення робочих характеристик програми у виробництві, тестове середовище повинно бути точною копією виробничого середовища, але з додаванням інструментів генерації навантаження та моніторингу ресурсів[1].

Автори Vinayak Hegde та Pallavi M. S. описують наступні критичні характеристики для тестового середовища[10]:

- апаратне забезпечення: апаратне забезпечення комп’ютера (процесор, оперативна пам’ять, тощо);
- мережа: архітектура мережі та розташування кінцевого користувача, доменне ім’я;
- інструменти: генерації навантаження;
- програмне забезпечення: інше програмне забезпечення, встановлене або запущене у спільному доступі або віртуальному середовищі;
- зовнішні фактори: Обсяг і тип додаткового трафіку в мережі, заплановані або пакетні процеси, взаємодія з іншими системами.

Визначення потрібних результатів: потрібно визначити час відгуку, пропускну здатність, цілі використання ресурсів та обмеження.

Як правило, має сенс починати виявлення або, принаймні, оцінку бажаних характеристик продуктивності програми на початку життєвого циклу розробки. Цього можна досягти найпростіше, відзначивши характеристики продуктивності, які користувачі та зацікавлені сторони ототожнюють з хорошими показниками.

Загалом, час відгуку – це проблема користувачів, пропускна спроможність справа бізнесу, а використання ресурсів – це проблема системи[1].

Як було сказано раніше класи характеристик, які часто співвідносяться із задоволенням користувача чи зацікавлених сторін зазвичай включають:

- час відгуку: наприклад, каталог адресна книга повинна відображатися менше ніж три секунди;
- пропускна здатність: наприклад, система повинна підтримувати додавання 30 адрес в секунду;
- використання ресурсів: наприклад, використання процесора становить не більше 80 відсотків;
- кількість помилкових запитів 0%.

Планування і дизайн тестів. Планування та проектування тестів продуктивності включає визначення ключових сценаріїв використання, визначення відповідної унікальності між користувачами, ідентифікацію та генерацію тестових даних та визначення метрик, що збираються. Зрештою, ці елементи забезпечать основу для робочих навантажень та профілів робочого навантаження.

Розробляючи та плануючи випробування з метою характеристики виробничих показників, метою має бути створення реальних моделей, щоб забезпечити надійні дані, які дозволять організації приймати зважені ділові рішення. Реальний дизайн тестів значно підвищує актуальність та корисність даних результатів. Реалістичні тести можуть бути більш витратними та трудомісткими для впровадження, але вони забезпечують набагато більшу точність для бізнесу та зацікавлених сторін. [1].

Реалістична поведінка може проявлятися в наступних параметрах:

- реалістичне моделювання затримок користувачів та часу роздумів, що є вирішальним для точності тесту;
- відмова користування, якщо користувачі з будь-якої причини можуть відмовитись від користування веб-додатком.

Налаштування тестового середовища. Потрібно підготувати тестове середовище, інструменти та ресурси, необхідні для виконання кожного сценарію, коли функції та компоненти стануть доступними для тестування. Також потрібно переконайтеся, що тестове середовище підготовлено для моніторингу ресурсів, якщо це необхідно[10].

Крім того, потрібно періодично переналаштовувати, оновлювати, додавати або іншим чином покращувати середовище генерації навантаження та пов'язані з ним інструменти протягом усього проекту. Навіть якщо протестована програма залишається незмінною, а інструмент генерації навантаження працює належним чином, ймовірно, показники, які потрібно зібрати, зміняться.

Потрібно виконувати моніторинг використання ресурсів (центральний

процесор, мережа, пам’ять, диск) на серверах у конфігурації із збалансованим навантаженням під час перевірки навантаження для перевірки того, що навантаження розподіляється рівномірно[2].

Розробити сценарії тестування. Потрібно розробити тести продуктивності відповідно до кожного запланованого проекту. Найбільша проблема, пов’язана з проектом тестування продуктивності, полягає в тому, щоб здійснити свій перший відносно реалістичний тест, в якому користувачі, як правило, змодельовані таким чином, що веб-додаток на якому проводиться тест не може законно визначити різницю між модельованими користувачами та реальними користувачами[2].

Переконатися, що перевірка транзакцій здійснюється правильно. Багато операцій про які веб-сервер повідомив, що вони успішні, можуть виконуватися не правильно.

Прикладами перевірки будуть записи бази даних, які вставляються з правильною кількістю рядків або інформація про продукт, що повертає правильний вміст, тощо.

Виконання тестування. Виконання тестування, це не просто запуск тесту. Насправді ця діяльність значно складніша, ніж просто натискання кнопки та контроль комп’ютерів.

Виконання тесту можна розглядати як комбінацію підзадач[1]:

- узгодження виконання та моніторингу тесту з командою;
- перевірка тестів, конфігурація та стан середовищ та даних;
- безпосереднє виконання тесту;
- відстежувати та перевіряти сценарії, системи та дані під час виконання тесту;
- після завершення тесту швидко переглянути результати на наявність очевидних ознак, якщо тест завершився некоректно;
- зберегти тести, дані тестів, результати та іншу інформацію, необхідну для повторення тесту пізніше, якщо потрібно;
- записати час початку та закінчення тесту, назву даних результатів

тощо, що дозволить ідентифікувати дані послідовно після завершення тесту.

Аналіз результатів. Проведення аналізу результатів тестування, для розуміння продуктивності веб-додатку та серверу, та знаходження вузького місця. Менеджерам та зацікавленим сторонам потрібні не лише результати різних тестів – їм потрібні висновки, а також консолідовані дані, що підтверджують ці висновки. Технічній команді також потрібні не лише результати, їм потрібен аналіз, порівняння та подробиці того, як були отримані результати[1]

Перш ніж передавати результати, дані необхідно проаналізувати. Розглянемо наступні важливі моменти при аналізі даних, які отримуються в результаті тестування продуктивності:

- аналіз отриманих даних та порівняння результатів з прийнятним або очікуваним рівнем метрики, для визначення відповідності результатів проведення тестів з очікуваними результатами тестування[2]:

- якщо тест не пройдений успішно, діагностувати та виправити причину;

- при виправленні будь-якого вузького місця в системі або веб-додатку, потрібно повторити тест, щоб перевірити продуктивність після виправлення помилок;

- змінити параметри тесту, щоб отримати нову, кращу або іншу інформацію, якщо результати не відповідають визначеному тесту.

Тестування продуктивності включає в себе набір загальних основних видів діяльності, які відбуваються на різних етапах проектів. Кожен вид діяльності має конкретні характеристики та завдання, які потрібно виконати.

1.7 Методи тестування

Точна конструкція тесту продуктивності спирається на поєднання обговорюваних дотепер вимог у узгоджений набір сценаріїв тесту продуктивності, що точно відображає паралельність та пропускну здатність.

Визначивши ключові випадки використання та вимоги до даних, наступний крок полягає в створенні ряду різних типів тестів продуктивності. Остаточний вибір буде в значній мірі визначатися характером застосування і кількістю часу на тестування продуктивності.

Найпоширенішими питаннями продуктивності, пов'язаними з веб-додатками, є: "Чи буде це досить швидко?", "Чи підтримуватиме веб-додаток всіх моїх клієнтів?", "Що станеться, якщо щось піде не так?" та "Що мені потрібно планувати, коли я отримаю більше клієнтів?". У невимушеній розмові більшість людей пов’язують "досить швидко" з тестуванням продуктивності, "пристосувати поточну або очікувану базу користувачів" з тестуванням навантаження, "щось йде не так" із стрес-тестуванням та "планування майбутнього зростання" з тестуванням потужності. [1].
У сукупності ці питання складають основу для чотирьох ключових типів тестів продуктивності веб-додатків.

Наступні терміни тестування загалом добре відомі в галузі, хоча часто виникає плутанина щодо того, що вони насправді означають[2]:

- об'ємний тест: класичний тест продуктивності, де веб-додаток завантажується до цільових задач, але, як правило, не далі. Метою є досягнення цілей ефективності щодо доступності, паралельності або пропускної здатності та часу відгуку. Об’ємне тестування є найближчим наближенням реального використання додатків, і воно, як правило, включає моделювання ефектів взаємодії користувача з клієнтом програми. Сюди входять затримки та паузи під час введення даних, а також (людські) відповіді на інформацію, що повертається із програми:

- стрес-тест: цей тест має зовсім іншу мету від об'ємного. Стрес-тест намагається викликати збій програми або якоїсь частини допоміжної інфраструктури. Метою є визначення порогу пропускної здатності веб-додатку під максимальним навантаженням. Таким чином, стрес-тест триває до тих пір, поки щось не зламається: наприклад користувач не може ввійти, час відгуку перевищує значення, яке ми визначили як прийнятне, або програма

стає недоступною. Обґрунтування стрес-тестування полягає в тому, що якщо наша цільова група паралельних користувачів становить 1000, але інфраструктура не перестає працювати лише на 1005 користувачів, то це варто знати, оскільки це наочно демонструє, що додаткових можливостей у веб-додатка більше немає;

- тест на витривалість призначений для виявлення проблем, які можуть виникнути лише через тривалий проміжок часу. Класичним прикладом може бути повільний розвиток витоку пам'яті або якесь непередбачене обмеження кількості використання деяких елементів веб-додатка. Такого роду тестування не може бути проведено ефективно, якщо воно передбачений відповідний моніторинг інфраструктури. Проблеми які зазвичай проявляються при виконанні тесту на витривалість можуть бути як поступове уповільнення часу відгуку, або як раптова втрата доступності програми. Для забезпечення точного діагнозу життєво важливим є співвідношення даних від введеного навантаження та інфраструктури в момент відмови або сприйманого уповільнення;
- тестування динамічного навантаження орієнтоване на визначення або перевірку експлуатаційних характеристик веб-додатку при впливі на моделі робочого навантаження та обсяги навантаження, які неодноразово збільшуються за межі передбачуваних виробничих операцій протягом коротких періодів часу. Під час проведення тесту динамічного навантаження тестувальники можуть виявити, що продуктивність програми погіршується, сповільнюється або повністю зупиняється. Визначення того, що і де відбувається збій веб-додатка під час проведення тестування, дозволяє розробникам краще підготуватися до несподіваних стрибків навантаження під час виробничого середовища.

J.D. Meier та Carlos Farre вважають, що завжди потрібно проводити тести на, об'ємність, витривалість, стрес-тест та тест динамічного навантаження. Інші типи тестів більше залежать від програми та кількості часу, доступного для тестування та якщо були знайдені помилки в роботі веб-додатку[1].

Тестування продуктивності – це широка та складна діяльність, яка може приймати різні форми, усувати багато ризиків та надати широкий спектр цінності для організації. Важливо розуміти різні типи тестів продуктивності, щоб зменшити ризики, мінімізувати витрати та знати, коли застосовувати відповідний тест протягом певного проекту тестування продуктивності.

1.8 Результати і моніторинг тестування

Віддалений моніторинг – це технологія яка забезпечує дані про продуктивність сервера (разом з іншими показниками) з віддаленої системи; тобто тестований сервер передає дані через мережу тій частині інструменту тестування продуктивності, яка запускає програмне забезпечення для моніторингу. Велика перевага використання віддаленого моніторингу полягає в тому, що вам зазвичай не потрібно встановлювати будь-яке програмне забезпечення для моніторингу системи.

Далі будуть наведені елементи статистики, які мають значення для збору результатів тестування продуктивності[2]:

- стандартне відхилення і нормальний розподіл: іншим загальним і корисним показником є стандартне відхилення, яке відноситься до середньої дисперсії від розрахункового середнього значення. Віно базується на припущенні, що більшість даних про випадкові події в реальному житті мають нормальний розподіл. Чим вище стандартне відхилення, тим далі елементи даних, як правило, лежать від середнього. З точки зору тестування продуктивності, високе стандартне відхилення може свідчити про непостійний досвід користувачів. Наприклад, у випадку використання може бути розрахований середній час відгуку 40 секунд, але стандартне відхилення 30 секунд. Це означало б, що кінцевий користувач має високі шанси відчути час відгуку від 25 до 55 секунд для тієї ж активності. Слід прагнути досягти невеликого стандартного відхилення;

- пропускна здатність та потужність: поряд із часом відгуку

тестувальників продуктивності, як правило, найбільше цікавить, скільки даних або скільки випадків використання можна обробити одночасно. Ми можете сприймати це вимірювання як пропускну здатність, щоб підкреслити, наскільки швидко обробляється певна кількість випадків використання, або як здатність підкреслити, скільки випадків використання можна обробити певний період часу. Раптове зменшення пропускної здатності незмінно вказує на проблеми і може збігатися з помилками, з якими стикаються один або кілька віртуальних користувачів. Ian Molyneaux пише що це часто трапляється, коли рівень веб-сервера досягає своєї точки насиченості для вхідних запитів. Віртуальні користувачі починають зупинятися, чекаючи відповіді веб-серверів, що призводить до зменшення пропускної спроможності. Приклад графіка пропускної здатності та потужності зображений на рисунку 1.4;

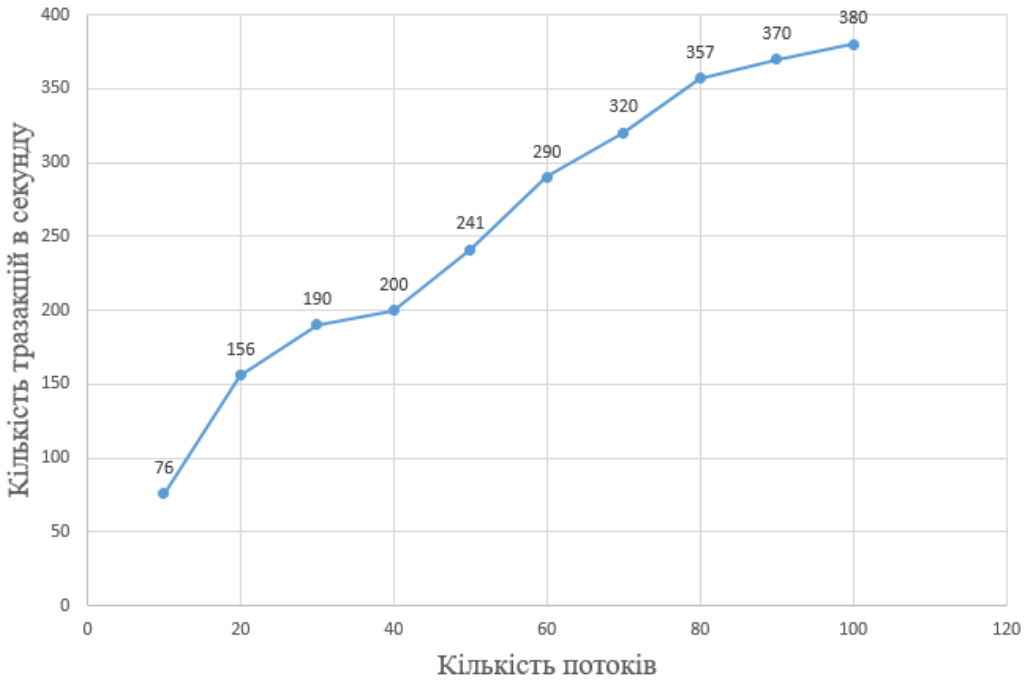

Рисунок 1.4 – Відношення пропускної здатності до потужності

- розподіл часу відгуку заснований на нормальній моделі розподілу, це спосіб агрегування часу відгуку на основі кількості запитів, зібраних під час тесту продуктивності, у ряд груп або сегментів. В статті аналізу розподілу часу

відгуку пишуть, що розуміючи розподіл часу відповіді на всі запити, ми можемо зосередитись на тих запитах, які мають найповільніший час відповіді[15]. Приклад графіка розподілу часу відгуку зображений на рисунку 1.5;

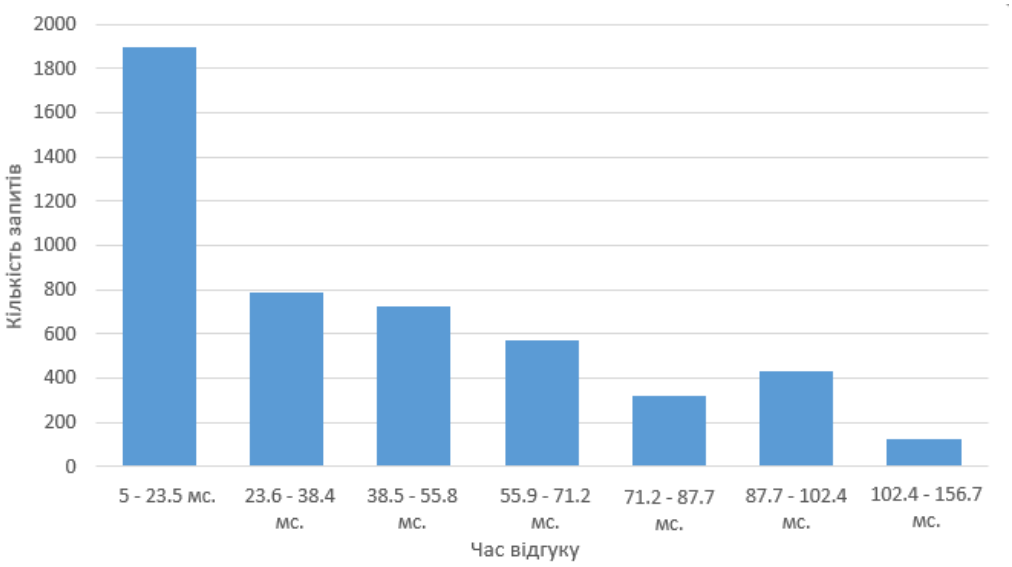

Рисунок 1.5 – Розподіл часу відгуку

Зміцнення впевненості є важливою діяльністю при виконанні тесту на ефективність. Ніщо не забезпечує більшої впевненості в успішному виконанні тесту, як перегляд журналів виконання. Журнали виконання забезпечують прозорість виконання тесту та визначають, наскільки здоровим було виконання тесту. Спостереження журналів виконання надає статус відмови тестового сценарію під час виконання, тривалості тестових запусків, відмови рівня ОС, та збоїв рівня програми, якщо такі є[12].

Швидкість веб-сайту залежить від різних факторів, таких як вміст на сторінках, браузер, географічне розташування доступу, пропускна здатність тощо. Часто можна зробити так, щоб вміст веб-сторінки займав менше байт, не змінюючи зовнішній вигляд або функції сторінки. Зменшення кількості байтів, які повинен завантажити клієнт, пришвидшує завантаження сторінки[16].

Jmeter має тестовий журнал, який записує кожну віртуальну тестову дію, якщо це спеціально не вимкнено користувачем. Тестові журнали містять деталі запису та відтворення, повну інформацію про тестовий цикл, тестові запуски та стан результатів усіх команд. З цього журналу можна зробити висновок, чи правильно виконані тестові запуски чи ні. Це забезпечує успішне завершення пробного запуску.

Коли неможливо використовувати віддалений моніторинг – можливо, через обмеження мережевого брандмауера або політики безпеки – рішення для тестування продуктивності може надавати компонент агент, який можна встановити безпосередньо на сервери, які ми хочете контролювати. Ian Molyneaux визначає що ми все ще можемо зіпсувати внутрішню безпеку та запити на зміни, спричинивши затримки або запобігаючи встановленню програмного забезпечення агента, але це корисна альтернатива, якщо ваше рішення для тестування продуктивності пропонує цю можливість, а опція віддаленого моніторингу недоступна[2].

1.9 Постановка завдань дослідження

На підставі попередніх розділів, були виявлені наступні завдання дослідження:

- розробка автоматизованих сценаріїв тестування продуктивності на основі головних методів тестування продуктивності;
- визначення тестового сайту і проведення його установки на локальний сервер;
- проведення тестування веб-додатка на основі головних методів тестування продуктивності;
- проведення аналізу результатів тестування веб-додатка, визначення основних метрик, та знаходження вузького місця в роботі веб-додатка.

2 МЕТОДИ ТЕСТУВАННЯ ТА ОЦІНКИ ПРОДУКТИВНОСТІ

2.1 Тестування веб-додатку

Тестування веб-додатку – це спосіб перевірити, чи відповідає фактичний програмний продукт очікуваним вимогам, і забезпечити відсутність дефектів цього продукту. Він передбачає виконання програмних та системних компонентів за допомогою ручних або автоматизованих інструментів для оцінки працездатності.

Існує чотири рівня тестування, які допомагають перевірити поведінку та ефективність тестування програмного забезпечення. Ці рівні зображені на рисунку 2.1.

Рисунок 2.1 – Рівні тестування

Модульне тестування – це найменша частина тестування системи або програми, яку можна скомпілювати, завантажити та виконати. Цей вид

тестування допомагає протестувати кожен модуль окремо. Мета цього рівня протестувати кожну частину програмного забезпечення, відокремивши її. Він перевіряє, чи компонент відповідає функціональним можливостям чи ні. Таке тестування проводять розробники.

Інтеграційне тестування – це поєднання. Наприклад, на цьому етапі тестування різні програмні модулі поєднуються та тестуються як група, щоб переконатися, що інтегрована система готова до тестування цілої системи. Інтеграційне тестування перевіряє потік даних від одного модуля до інших модулів. Цей вид тестування проводять тестери.

Системне тестування – це тестування системи яке проводиться на повній, інтегрованій системі. Це дозволяє перевірити відповідність системи відповідно до вимог. Воно перевіряє загальну взаємодію компонентів та передбачає тестування навантаження, продуктивності, надійності та безпеки. Системне тестування найчастіше фінальне тестування для перевірки відповідності системи специфікації. Віно оцінює як функціональну, так і нефункціональну потребу в тестуванні[5].

2.2 Тестування функціональності

Тестування функціональності веб-сайту – це процес, що включає декілька параметрів тестування, таких як користувальницький інтерфейс, API, тестування баз даних, тестування безпеки, тестування клієнта та сервера та основні функціональні можливості веб-сайту. Функціональне тестування є дуже зручним і дозволяє користувачам проводити як ручне, так і автоматизоване тестування. Він проводиться для перевірки функціональних можливостей кожної функції на веб-сайті.

Засоби веб-тестування включає тестування всіх посилань на веб-сторінках на працездатність. Посилання для перевірки включатимуть:

- вихідні посилання;
- внутрішні посилання;

- якірні посилання;
- тестові форми.

Тестові форми включають:

- перевірку сценаріїв форм, чи працюють належним чином. Наприклад, якщо користувач не заповнює обов’язкове поле у формі, відображається повідомлення про помилку;
- перевірку, чи заповнюються значення за замовчуванням;
- після подання, дані у формах подаються до бази даних або пов’язуються з діючою електронною адресою;
- форми оптимально відформатовані для кращої читабельності.

Тестові файли cookie (сеанси) видаляються або після очищення кеш-пам’яті, або після закінчення терміну їх дії.

Тестування HTML та CSS, щоб пошукові системи могли легко сканувати сайт, це включатиме:

- перевірка синтаксичних помилок;
- читаємі кольорові схеми;
- відповідність стандартам. Забезпечте дотримання стандартів, таких як W3C, OASIS, IETF, ISO, ECMA або WS-I.

Перевірка практичності – це процес, за допомогою якого вимірюються характеристики взаємодії людини і комп'ютера в системі та виявляються слабкі місця для виправлення:

- простота використовування веб-додатку;
- навігація;
- суб’єктивне задоволення користувачів;
- загальний вигляд.

Тест навігації означає, як користувач переглядає веб-сторінки, різні елементи керування, такі як кнопки, рамки або як користувач використовує посилання на сторінках для перегляду різних елементів сайту.

Тестування зручності використання включає наступне:

- веб-сайт повинен бути простим у користуванні;

- надані інструкції повинні бути дуже чіткими;
- відповідність наданих інструкцій з їх призначенням;

Зміст повинен бути логічним і простим для розуміння. Зміст повинен бути осмисленим. Усі прив'язні текстові посилання повинні працювати належним чином.

При веб-тестуванні слід протестувати серверний інтерфейс. Це виконується шляхом перевірки правильності спілкування. Слід перевірити сумісність сервера із програмним, апаратним забезпеченням, мережею та базою даних. Основними інтерфейсами є:

- веб-сервер та інтерфейс сервера додатків;
- сервер додатків та інтерфейс сервера баз даних.

Потрібно перевірити, чи всі взаємодії між цими серверами виконуються, і помилки обробляються належним чином. Якщо база даних або веб-сервер повертає повідомлення про помилку для будь-якого запиту сервером додатків, тоді сервер додатків повинен ловити та відображати ці повідомлення про помилки користувачам.

Тестування на сумісність – Тестування на сумісність проводиться на основі контексту програми:

- сумісність браузера;
- сумісний з різними пристроями, такими як ноутбук, мобільний телефон тощо.

Тест на сумісність браузерів: однаковий веб-сайт у різних браузерах відображатиметься по-різному. Потрібно перевірити, чи правильно відображається веб-додаток в браузерах, JavaScript, AJAX і автентифікація працює нормально.

2.3 Поширені проблеми продуктивності

Більшість проблем з продуктивністю пов'язані зі швидкістю, часом відгуку, часом навантаження та поганою масштабованістю. Швидкість часто є

одним з найважливіших атрибутів програми. Повільно запущений додаток втратить потенційних користувачів. Тестування продуктивності проводиться для того, щоб переконатися, що програма працює досить швидко, щоб утримати увагу та інтерес користувача. Типові проблеми з продуктивністю будуть наведені далі.

Тривалий час завантаження, як правило, є початковим часом, який потрібен для запуску програми. Цей час як правило, має бути мінімальним. Незважаючи на те, що деякі програми неможливо завантажити менше ніж за хвилину, час завантаження слід тримати за кілька секунд, якщо це можливо.

Час відгуку – це час, з якого користувач вводить дані у програму, поки програма не видасть відповідь на цей вхід. Як правило, це повинно бути дуже швидко. Знову ж таки, якщо користувачеві доведеться чекати занадто довго, він втрачає інтерес.

Погана масштабованість – програмний продукт страждає від поганої масштабованості, коли він не може обробити очікувану кількість користувачів або коли він не вміщує досить широкого кола користувачів. Перевірку навантаження слід проводити, щоб програма могла обробляти очікувану кількість користувачів.

Вузьке місце – це коли помилки кодування або проблеми з обладнанням спричиняють зменшення пропускної здатності під певними навантаженнями. Ключ до виправлення проблеми з вузьким місцем полягає у пошуку розділу коду, що спричиняє уповільнення, та спроби виправити його там. Вузькі місця, як правило, виправляються шляхом виправлення незадовільних запущених процесів або додавання додаткового обладнання. Деякі загальні вузькі місця в роботі:

- використання процесора;
- використання пам'яті;
- використання мережі;
- використання диска.

2.4 Методи виміру продуктивності і тестування

Тестування продуктивності можна застосувати для розуміння масштабованості веб-сайту або для порівняння продуктивності в середовищі сторонніх продуктів, таких як сервери та проміжне програмне забезпечення, для потенційної покупки.

Навантаження на веб-додаток:

- кількість користувачів за раз;
- перевірка пікового навантаження та поведінки системи;
- великий обсяг даних, до яких користувач отримує доступ.

Веб-програма повинна витримувати велике навантаження. Тестування веб-продуктивності повинно включати:

- тестування веб-навантаження;
- веб-стрес тестування;
- тестування динамічного навантаження;
- знаходження точки насичення.

Тестування веб-навантаження: потрібно перевірити, чи багато користувачів отримують доступ до однієї сторінки або вимагають її. Чи може система витримувати час пікового навантаження? Сайт повинен обробляти багато одночасних запитів користувачів, великі вхідні дані від користувачів, одночасне підключення до БД, велике навантаження на певні сторінки тощо.

2.4.1 Використання Jmeter в тестуванні

Тестування навантаження JMeter – це процес тестування, який виконується за допомогою інструменту тестування навантаження під назвою Apache JMeter. JMeter для тестування на навантаження є вирішальним інструментом, який визначає, чи може веб-програма, що перевіряється, задовольнити високі вимоги щодо завантаження чи ні. Це також допомагає аналізувати загальний сервер під великим навантаженням.

За допомогою цієї програми ми будемо проводити вимір продуктивності веб-додатку під різними умовами. Алгоритм роботи Jmeter зображений на рисунку 2.2.

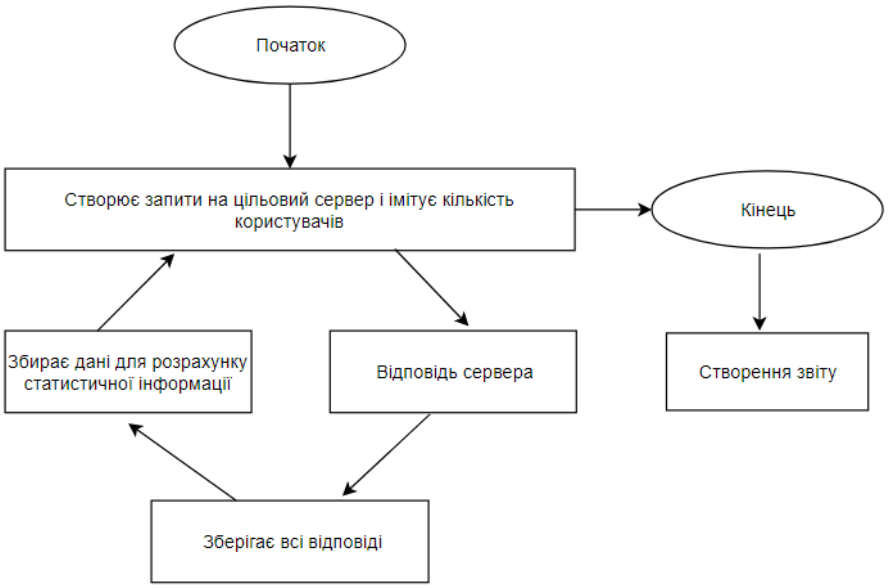

Рисунок 2.2 – Алгоритм роботи Jmeter

На початку алгоритму Jmeter створює HTTP запит на необхідний веб-сервер. Кількість одночасних користувачів та інші параметри вказуються заздалегідь в налаштуваннях запиту. Після цього веб-додаток(сервер) надає відповідь на HTTP запити, у вигляді наступних цифрових значень:

- інформаційні відповіді (100–199);
- успішні відповіді (200–299);
- переспрямування (300–399);
- помилки клієнта (400–499);
- помилки сервера (500–599).

Всі відповіді незалежно від того помилкові вони чи ні програма записує в звіт для подальшого розподілу інформації. Наступним етапом в алгоритмі програми буде зберігання даних у вигляді статистичної інформації. Алгоритм

виконується поки не закінчиться заздалегідь встановлений час або все користувачі не закінчать роботу. Кінцевим етапом буде створення звіту на основі збереженої інформації.

2.4.2 Тестування продуктивності

У контексті веб-розробки тестування продуктивності передбачає використання програмних засобів для імітації роботи програми за певних обставин. Кількісне тестування продуктивності розглядає такі показники, як час відгуку, тоді як якісне тестування стосується масштабованості, стабільності та сумісності. Коли звучить слово «продуктивність», більшість людей відразу замислюється про швидкість. Швидкий час завантаження та час відгуку є абсолютно необхідними в наші дні, але також слід подумати про загальну картину, яка вимагає не просто перегляду всіх ваших посилань, щоб переконатися, що вони працюють[2]. Те, що під час виробничого тестування все працює ідеально, ще не означає, що це буде так, коли веб-сайт буде заповнений трафіком. Для цього тесту буде виконуватися тестування в звичайних умовах використання веб-додатку. Приклад зображений на рисунку 2.3.

Рисунок 2.3 – Тестування в звичайних умовах

Тестування продуктивності веб-сайту або програми дозволяє виявляти проблеми та покращувати загальну ефективність, що може призвести до поліпшення взаємодії з користувачами та збільшення доходу. Є багато загальних проблем, які може виявити тестування продуктивності, наприклад, вузькі місця. Переривання потоку даних через обмежену потужність називається вузьким місцем. Вузькі місця можуть виникнути, наприклад, якщо раптово зростає трафік, який не можуть витримати сервери. Вузькі місця – це лише одна з багатьох проблем, які можуть виникнути, коли веб-сайт не є масштабованим.

2.4.3 Стрес тестування

Стрес-тестування – це тип тестування програмного забезпечення, який перевіряє стабільність та надійність програмного забезпечення. Метою стрес-тестування є вимірювання програмного забезпечення на його надійність та можливості обробки помилок в умовах надзвичайно великих навантажень, а також забезпечення того, щоб програмне забезпечення працювало в аварійних ситуаціях. На рисунку 2.4 наведений приклад графіку тестування знаходження точки насичення веб-додатку.

Рисунок 2.4 – Знаходження точки насичення

Під час стрес-тестування веб-додаток піддається стресу протягом короткого періоду часу, щоб дізнатися про його здатність витримувати навантаження. Найбільш відомим використанням стрес-тестування є визначення межі, при якій система, програмне чи апаратне забезпечення почне працювати некоректно[2].

Потреба в стрес-тестуванні обумовлюється різними сценаріями, наприклад:

- під час розпродажі на сайті інтернет-магазину може спостерігатися сплеск відвідуваності;
- коли сайт згадується другим популярним сайтом.

Стрес зазвичай надається полям введення, областям входу та реєстрації. Що дає стрес тестування:

- визначає, чи можуть бути пошкоджені дані через надмірне напруження системи;
- надає оцінку того, наскільки далеко за межі цільового навантаження може перейти програма, перед тим, як викликати помилки та впливати на повільність;
- дозволяє встановити тригери моніторингу додатків, щоб попередити про майбутні збої;
- гарантує, що вразливі місця безпеки не відкриваються в стресових умовах;
- визначає побічні ефекти загальних апаратних засобів або підтримку відмов додатків;

2.4.4 Динамічне навантаження

Мета динамічного навантаження – побачити, як система реагує на несподіваний підйом та падіння навантаження користувача. У програмній інженерії тестування Spike допомагає визначити погіршення продуктивності системи при раптовому великому навантаженні.

Сценарії відновлення при динамічному завантаженні. Для захисту від стрибків можна налаштувати три основні сценарії відновлення:

Використовувати хмарні платформи, такі як AWS, Azure, щоб динамічно збільшувати пропускну здатність сервера в тандемі із завантаженням користувача. На рисунку 2.5 зображений графік тестування динамічного навантаження.

Рисунок 2.5 – Графік динамічного навантаження

Не давати доступ деяким користувачам, щоб система не стикалася з великим навантаженням. Таким чином захищає систему від загрози надмірного навантаження.

Адміністратор сайту дозволяє користувачам приєднуватися до системи. Однак з попередженням, що вони можуть зіткнутися з повільною реакцією через велике навантаження.

2.4.5 Функціональне тестування в Jmeter

Функціональне тестування – це той тип тестування, коли ми дійсно стурбовані тим, що система робить те, що вона повинна робити з функціональної точки зору. Наприклад введення логіну та паролю, додання

або видалення інформації з сервера і так далі. Функціональне тестування передбачає проведення набору завдань і порівняння результату одного і того ж із очікуваним результатом та можливість повторити один і той же набір завдань кілька разів з різним введенням даних та однаковим рівнем точності. Веб-функціональне тестування може виконуватися як вручну за допомогою людського тестера, так і автоматично з використанням програмного забезпечення. В Jmeter цю частину тестування можна зробити за допомогою тестового плану та різних HTTP запитів.

2.4.6 Тестування на витривалість

Тестування на витривалість – це тип тестування веб-додатку, де програмне забезпечення тестується з великим навантаженням, розширеним протягом значної кількості часу, щоб оцінити поведінку програмного забезпечення при тривалому використанні. Основна мета випробувань на витривалість – забезпечити достатню здатність програми справлятися з тривалим навантаженням без будь-якого погіршення часу відгуку. Приклад графіка тестування на витривалість зображений на рисунку 2.6.

Рисунок 2.6 – Тест на витривалість

Цей вид тестування має різні цілі. Одна із головних цілей тестування на витривалість – перевірити витік пам'яті та дізнатися, як працює система при тривалому використанні. Щоб перевірити, що після тривалого періоду час реакції системи залишатиметься однаковим або кращим, ніж початок тесту.

Для визначення кількості користувачів та транзакцій яких певна система буде підтримувати та відповідати цілям ефективності.

Для управління майбутніми навантаженнями нам потрібно зрозуміти, скільки додаткових ресурсів (наприклад, ємність процесора, ємність диска, використання пам’яті або пропускна здатність мережі) потрібно для підтримки використання в майбутньому. Тестування на витривалість зазвичай проводиться або шляхом перевантаження системи, або за рахунок зменшення певних системних ресурсів та оцінки наслідків.

2.5 Тестування безпеки

Тестування безпеки є дуже важливим для веб-сайту електронної комерції, який зберігає конфіденційну інформацію про клієнтів, наприклад, кредитні картки. Тестова діяльність включатиме:

- несанкціонований доступ до захищених сторінок не повинен бути дозволений;
- файли з обмеженим доступом не можна завантажувати без відповідного доступу;
- сеанси перевірки автоматично вбиваються після тривалої бездіяльності користувача;
- при використанні сертифікатів SSL веб-сайт повинен перенаправляти на зашифровані сторінки SSL.

Основною причиною тестування безпеки Інтернету є виявлення потенційних вразливих місць та подальше їх усунення. Таких як:

- мережеве сканування;
- сканування вразливості;

- злом пароля;
- огляд журналу;
- перевірка цілісності;

2.6 Ручне та автоматичне тестування

Тестування вручну – це тестування програмного забезпечення, де тести виконуються аналітиком з контролю якості вручну. Він виконується для виявлення помилок у програмному забезпеченні, яке розробляється. Під час тестування вручну тестер перевіряє всі основні особливості даного додатку чи програмного забезпечення. У цьому процесі тестувальники програмного забезпечення виконують тестові кейси та формують звіти про тестування без допомоги будь-яких засобів тестування програмного забезпечення для автоматизації. Це класичний метод усіх типів тестування та допомагає знаходити помилки в програмних системах. Зазвичай його проводить досвідчений тестер, щоб виконати процес тестування програмного забезпечення. Коротше кажучи, ручне тестування найкраще підходить для наступних областей або сценаріїв:

- дослідницьке тестування: Цей тип тестування вимагає знань, досвіду, аналітичних та логічних навігаторів, творчості та інтуїції тестувальника;
- тестування зручності використання: Це область, в якій вам потрібно виміряти, завдяки зручному, ефективному чи зручному програмному забезпеченню продуктів для кінцевих користувачів. Тут людське спостереження є найважливішим фактором, перед переважно ручним підходом;
- спеціальне тестування: У цьому сценарії немає конкретного підходу. Абсолютно незапланований метод тестування, де розуміння у тестера є єдиним важливим фактором.

Плюси ручного тестування:

- отримаємо швидкий і точний візуальний відгук;

- це дешевше, оскільки нам не потрібно витрачати свій бюджет на засоби автоматизації та процес;

- людське судження та інтуїція завжди приносять користь ручному елементу;

- під час тестування невеликої зміни, тест автоматизації вимагав би кодування, яке може зайняти багато часу, поки можна було тестувати вручну на льоту.

Мінуси ручного тестування:

- менш надійний метод тестування, оскільки його проводить людина. Тому він завжди схильний до помилок та помилок;

- процес ручного тестування неможливо записати, тому повторне використання ручного тестування неможливе;

- у цьому методі тестування певні завдання важко виконати вручну, що може зажадати додаткового часу етапу тестування програмного забезпечення.

В автоматизованому тестуванні програмного забезпечення тестувальники тестові скрипти для автоматизації виконання тесту. Тестери використовують відповідні засоби автоматизації для розробки тестових сценаріїв та перевірки програмного забезпечення. Мета полягає у виконанні тесту за менший проміжок часу. Автоматизоване тестування повністю покладається на попередньо виконаний сценарій, який запускається автоматично для порівняння фактичних результатів із очікуваними результатами. Це допомагає тестувальникові визначити, чи працює програма належним чином чи ні. Автоматизоване тестування дозволяє виконувати повторювані завдання та регресійний тест без участі ручного тестера. Незважаючи на те, що всі процеси виконуються автоматично, автоматизація вимагає певних зусиль вручну для створення сценаріїв початкового тестування.

Автоматизоване тестування є найкращим варіантом у наступних областях або сценаріях:

- тестування регресії: тут підходить автоматичне тестування через час зміни, коли та дозволяється своєчасне запуску регресії;

- тестування завантажень: Автоматизоване тестування – це також найкращий спосіб зробити тестування, коли йдеться про тестування завантажень;

- повторне виконання: Тестування, яке вимагає багаторазового виконання завдання, найкраще автоматизоване;

- тестування продуктивності: аналогічне тестування, яка вимагає моделювання тисяч одночасних користувачів, вимагає автоматизації.

Плюси автоматизованого тестування:

- автоматизоване тестування допомагає знайти більше помилок порівняно з людським тестером;

- оскільки більша частина процесу тестування автоматизована, ми можемо мати швидкий та ефективний процес;

- процес автоматизації можна записати. Це дозволяє повторно використовувати та виконувати однакові операції тестування;

- автоматизоване тестування проводиться за допомогою програмних засобів, тому воно працює без втоми та втоми, на відміну від людей при ручному тестуванні;

- це може легко підвищити продуктивність, оскільки забезпечує швидкий і точний результат тестування;

- автоматизоване тестування підтримує різні програми.

Мінуси автоматизованого тестування: без людського елемента важко отримати уявлення про візуальні аспекти інтерфейсу, такі як кольори, шрифт, розміри, контраст чи розмір кнопок. Інструменти для запуску автоматичного тестування можуть бути дорогими, що може збільшити вартість проекту тестування. Засіб автоматизації тестування ще не є надійним. Кожен інструмент автоматизації має свої обмеження, що зменшує сферу автоматизації.

3 МОДЕЛЬ СЕРВЕРА

3.1 Модель продуктивності веб-сервера

Для моделі продуктивності веб-сервера була обрана абстрактна модель, змодельована як відкрита мережа масового обслуговування, яка абстрагує всі деталі апаратного та програмного забезпечення, але є досить конкретною для отримання значних результатів роботи щодо взаємозв'язку між швидкістю сервера та мережі. Структура моделі сервера являє собою СМО. Моделі мереж розглянуті в [Ponomarenko O., Gorbachov V., Abdulrahman Kataeba Batiaa., Kotkova O. The Software Platform for Evaluation of Effectiveness of Network Systems Analysis Technologies Proceedings of IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 2019, pp. 513-516]

Схема мережі масового обслуговування моделі веб-сервера представлена на рисунку 3.1.

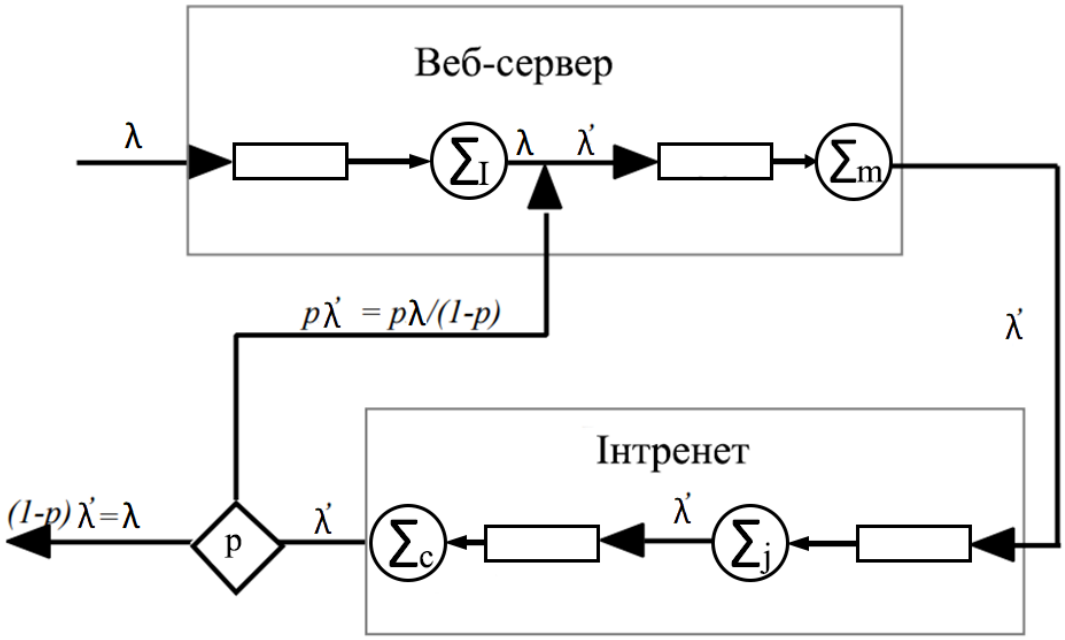

Рисунок 3.1 – Мережева модель масового обслуговування веб-сервера

Ця мережа складається з чотирьох вузлів: дві моделі самого веб-сервера та дві моделі інтернет-зв'язку. Запити на файли (тобто "завдання") надходять на веб-сервер з частотою λ. Вся одноразова обробка "ініціалізації" виконується на вузлі Σ_I. Потім завдання переходить до вузла Σ_m, де дані одного буфера зчитуються з файлу, обробляються та передаються в мережу. На вузлі Σ_j цей блок даних передається в інтернет за швидкістю передачі даних сервером (наприклад, 10 мегабіт/с.). Ці дані передаються через Інтернет і отримуються браузером клієнта, представленим вузлом Σ_C. Якщо файл передано не повністю, "завдання" розгалужується і повертається до вузла Σ_m для подальшої обробки. Розгалуження є імовірнісним, з урахуванням середнього розміру файлу F та розміру буфера B, ймовірність повного передавання файлу дорівнює p = B / F. В іншому випадку завдання завершується і виходить із мережі. Крім того, швидкість прибуття на вузол Σ_m (λ') – це сума швидкості прибуття мережі (λ) та швидкості завдань, що надходять із Σ_C назад до Σ_m.

СМО (системи масового обслуговування) – це моделі систем, в які в випадкові моменти часу ззовні або зсередини надходять заявки. СМО з очікуваннями (чергою) – це СМО, в якої заявка, що прийшла в момент, коли всі канали зайняті, не йде, а стає в чергу на обслуговування. Приклад СМО з необмеженою чергою зображений на рисунку 3.2.

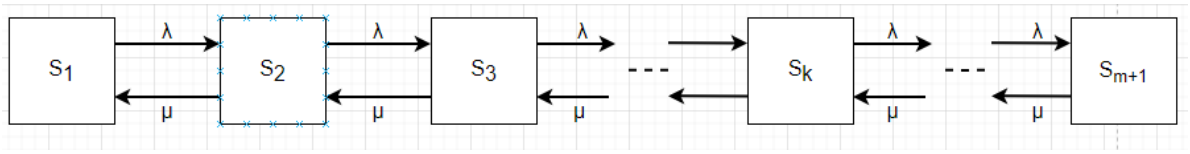

Рисунок 3.2 – СМО з необмеженою чергою

В СМО з чергою надходить потік заявок з інтенсивністю λ; інтенсивність обслуговування μ (в середньому безперервно зайнятий канал буде видавати p = λ/μ обслужених заявок в одиницю (часу). Оскільки обмеження на довжину черги відсутня, то будь-яка заявка може бути обслужена, тому Робс = 1.

Відносна пропускна здатність має наступний вигляд:

$$Q = P_{обс} = 1 => P_{від} = 0 , \quad (3.1)$$

де $P_{обс}$ – імовірність обслуговування заявки;

$P_{від}$ – імовірність відмови заявки.

Абсолютна пропускна здатність:

$$A = \lambda , \quad (3.2)$$

де λ – інтенсивність потоку заявок.

Середнє число заявок в черзі обчислюється наступним виразом:

$$L_{ч} = \frac{p^2}{1-p} , \quad (3.3)$$

де p – приведена інтенсивність потоку заявок.

Середнє число заявок в системі обчислюється наступним виразом:

$$L_{сист} = L_{ч} + p , \quad (3.4)$$

де Lч – середнє число заявок в черзі;

p – приведена інтенсивність потоку заявок.

Середній час очікування обслуговування в черзі обчислюється наступним виразом:

$$T_{ч} = \frac{L_{ч}}{\lambda} , \quad (3.5)$$

де $L_{ч}$ – середнє число заявок в черзі;

λ – інтенсивність потоку заявок.

Формула середнього часу перебування заявки в системі:

$$T_{\text{сист}} = \frac{L_{\text{сист}}}{\lambda}, \tag{3.6}$$

де $L_{\text{сист}}$ – середнє число заявок в системі;

λ – інтенсивність потоку заявок.

У теорії масового обслуговування, мережа Джексона – це клас мережі масового обслуговування, де розподіл рівноваги особливо просто обчислити, оскільки мережа має рішення продуктової форми. У відкритій мережі Джексона з m M / M / 1 черг, де коефіцієнт використання ρі менше 1 у кожній черзі, розподіл імовірності стаціонарного стану мережі може бути виражено як добуток імовірностей стану окремих черг π(k1,k2,…kN) = π1 (k1) * π2(k2) * … * πn(kn)[18].

Розглядаючи модель як мережу Джексона, час відгуку черги задається наступним виразом[19]:

$$\mathrm{T} = \frac{\mathrm{F}}{\mathrm{C}} + \frac{I}{1 - \lambda\,\mathrm{I}} + \frac{F}{\mathrm{J} - \lambda\,\mathrm{F}} + \frac{F(B + MY)}{\mathrm{BM} - \lambda\,\mathrm{F(B + MY)}}, \tag{3.7}$$

де F – середній розмір файлу;

C – пропускна здатність мережі клієнта;

I – час ініціалізації;

λ – швидкість прибуття в мережу;

J – пропускна здатність серверної мережі;

B – розмір буфера;

M – динамічна швидкість сервера;

Y – статичний час сервера.

Для проведення тестового розрахунку середнього часу відгуку, потрібно підібрати параметри обчислення для формули 3.7. Середній розмір файлу – 218750 байт; розмір буфера – 104166 байт; пропускна здатність мережі клієнта – 30 мегабіт/с. (3750000 байт/с.); пропускна здатність серверної мережі 63 мегабіта/с. (7875000 байт/с.); швидкість прибуття в мережу – концептуально

часто легше перевести будь-яке посилання на λ у відповідне значення "звернень за день"; просто помножимо λ на 60 * 60 * 24 = 86 400; час ініціалізації – 0.005 с; динамічна швидкість сервера – 55634201 байт/с; статичний час сервера – 0.000014 с. На рисунку 3.3. зображено відношення кількості звернень до середнього часу відгуку.

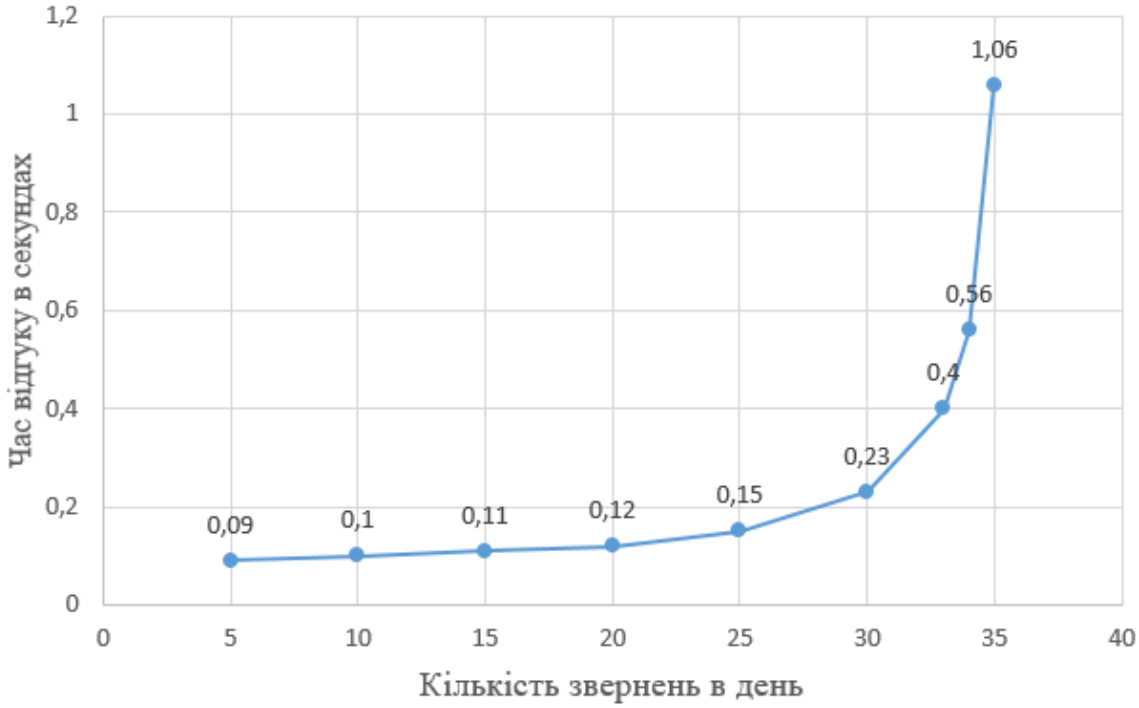

Рисунок 3.3 – Типова крива часу відгуку

Для цього веб-сервера 3024000 (λ = 35) звернень в день – це теоретична верхня межа кількості звернень за день які можна обслуговувати.

Поліпшити кількість звернень в день можна за допомогою збільшення швидкості інтернету веб-сервера. Для прикладу збільшимо швидкість веб-сервера вдвічі, до 126 мегабіт/сек, та проведемо розрахунок середнього часу відгуку і знайдемо верхню межу звернень в день.

Результат експерименту зображений на рисунку 3.4.

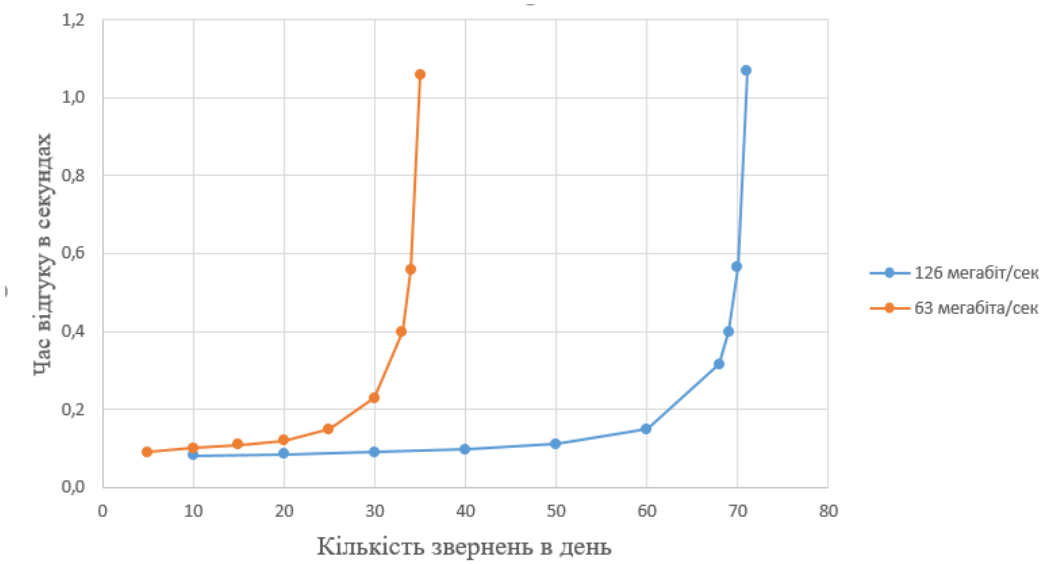

Рисунок 3.4 – Поліпшення ємності веб-додатка

Як видно із рисунка 3.4 при збільшенні швидкості веб-сервера була покращена ємність веб-додатка з 3024000 ($\lambda = 35$) звернень в день до 6134400 ($\lambda = 71$). Також був покращений час відгуку на меншій кількості запитів в день. При значенні $\lambda = 33$ час відгуку був покращений на 125% з 0.4 до 0.092 секунд.

3.2 Клієнт-серверна модель

Клієнт-серверна модель – це розподілена структура додатків, яка розділяє завдання або робочі навантаження між провайдерами ресурсу чи послуги, які називаються серверами, та запитувачами послуг, які називаються клієнтами. Часто клієнти та сервери спілкуються через комп'ютерну мережу на окремому обладнанні, але і клієнт, і сервер можуть перебувати в одній системі. Хост сервера запускає одну або кілька серверних програм, які діляться своїми ресурсами з клієнтами. Клієнт не ділиться жодним із своїх ресурсів, але він запитує вміст або послугу від сервера. Отже, клієнти ініціюють сеанси спілкування із серверами, які очікують на вхідні запити. Клієнт-серверна модель зображена на рисунку 3.5.

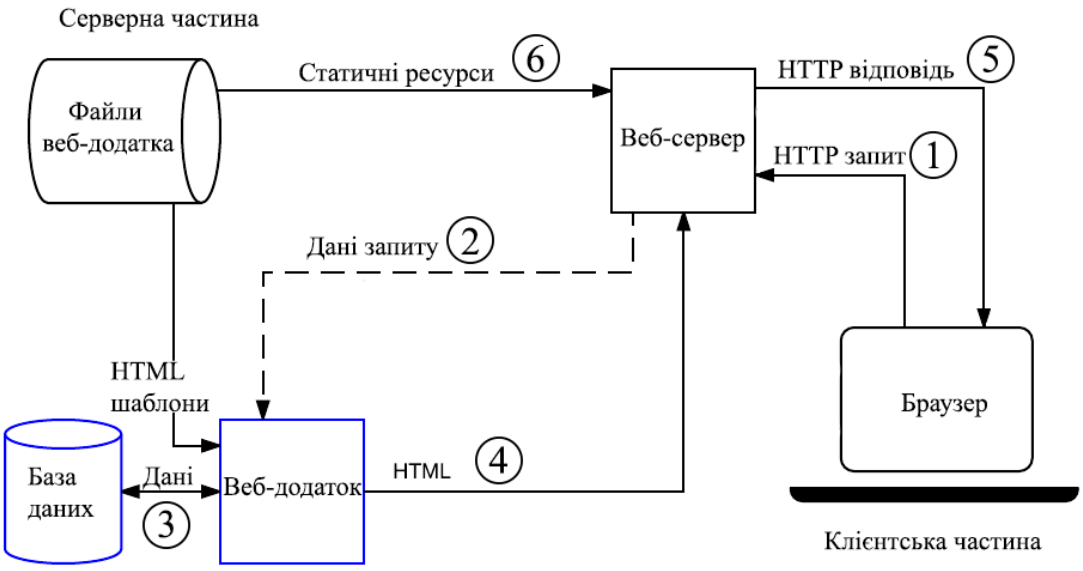

Рисунок 3.5 – Клієнт-серверна модель

Як видно із рисунка 3.5 на стороні клієнта ми маємо комп’ютер який використовує веб-браузер. Зв'язок з веб-сервером виконується за допомогою HTTP запитів та відповідей. Сервер включає такі компоненти як: база даних, веб-додаток та його файли. Далі буде наведений приклад роботи клієнт-серверної моделі.

Першою дією буде подання HTTP запиту до веб-серверу. Другою дією будуть виконуватися запити на динамічні ресурси які пересилаються до коду на стороні сервера. Дані запиту передають таку інформацію як: URL кодування, GET та POST дані. Третім етапом буде зв'язок веб-додатка з базою даних, для зчитування або запису даних в базу даних та отримання HTML шаблонів які потрібні для відображення інформації на сайті. Зв’язок веб-додатка з базою даних виконується за допомогою додатка XAMPP. База даних – це структура даних, яка зберігає організовану інформацію. Четвертим та п’ятим етапом буде надсилання та зчитування відповіді що містить сформований HTML. Файли веб-додатка містять статичні ресурси зображені під номером 6, ці ресурси фіксовані та відображають однаковий вміст для кожного користувача, як правило, написаний виключно у форматі HTML.

Характеристика клієнт-сервер описує взаємозв'язок взаємодіючих програм у додатку. Серверний компонент надає функцію або послугу одному або багатьом клієнтам, які ініціюють запити на такі послуги. Сервери класифікуються за послугами, які вони надають. Наприклад, веб-сервер обслуговує веб-сторінки, а файловий – комп'ютерні файли.

Браузери спілкуються з веб-серверами за допомогою протоколу передачі HyperText (HTTP). Коли ми натискаємо посилання на веб-сторінці, подаємо форму або виконуємо пошук, браузер надсилає на сервер запит HTTP[6].

Цей запит включає: URL-адресу, що ідентифікує цільовий сервер і ресурс (наприклад, файл HTML, конкретна точка даних на сервері або інструмент для запуску); метод, який визначає необхідні дії (наприклад, для отримання файлу або для збереження або оновлення деяких даних).

Веб-сервери чекають повідомлень клієнтських запитів, обробляють їх, коли вони надходять, і відповідають у веб-браузері повідомленням HTTP Response. Відповідь містить код стану відповіді HTTP, який вказує, чи вдався запит (наприклад, "200 OK" для успіху, "404 Not Found ", якщо ресурс не вдається знайти, "403 Forbidden ", якщо користувач не має права бачити ресурс тощо). Тіло успішної відповіді на запит GET міститиме запитаний ресурс. Коли повертається HTML-сторінка, вона відображається веб-браузером.

Статичний та динамічний вміст веб-додатка. Грубо кажучи, сервер може подавати як статичний, так і динамічний вміст. Термін динамічний означає, що сервер обробляє вміст або навіть генерує його на льоту з бази даних. Такий підхід забезпечує більшу гнучкість, але технічний стек є більш складним, що робить значно складнішим створення веб-сайту.

Візьмемо, наприклад, веб-додаток “addressbook”. На веб-сервері, який його розміщує, є сервер додатків, який бере вміст користувачів з бази даних, форматує, розміщує в деяких шаблонах HTML і надсилає результати для перегляду. Приклад динамічного вмісту веб-додатку зображено на рисунку 3.6.

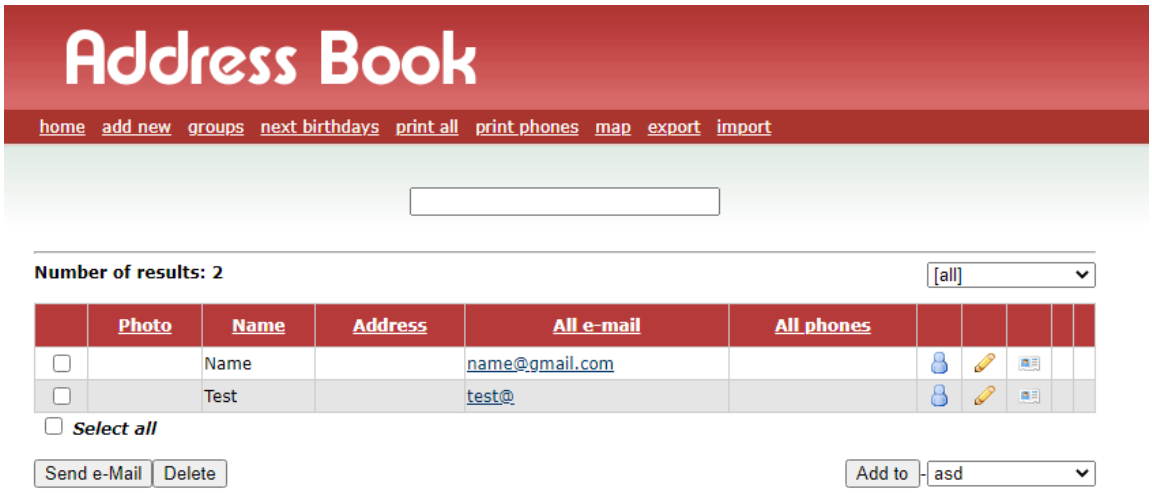

Рисунок 3.6 – Динамічний вміст веб-додатку

Поширене та найкраще програмне забезпечення веб-серверів на ринку:

- сервер Apache HTTP. Розроблений Apache Software Foundation, це безкоштовний веб-сервер із відкритим кодом для Windows, Mac OS X, Unix, Linux, Solaris та інших операційних систем;
- інтернет-інформаційні служби Microsoft (IIS). Розроблений Microsoft для платформ Microsoft. Він не є відкритим, але широко використовується;
- nginx – популярний веб-сервер з відкритим кодом для адміністраторів завдяки легкому використанню ресурсів та масштабованості. Він може обробляти багато одночасних сесій завдяки архітектурі, керованій подіями;
- lighttpd. Безкоштовний веб-сервер, який постачається з операційною системою FreeBSD. Він розглядається як швидкий та безпечний, при цьому споживаючи менше енергії центрального процесора.

3.3 Програмне забезпечення сервера

Для запуску веб-додатка буде використовуватися XAMPP. XAMPP – це безкоштовний пакет рішень для веб-серверів з відкритим кодом, розроблений Apache Friends, що складається в основному з HTTP-сервера Apache, бази даних MySQL та інтерпретатори сценаріїв, написаних мовами програмування

PHP та Perl. Оскільки більшість фактичних розгортань веб-серверів використовують ті самі компоненти, що і XAMPP, це робить можливим перехід від локального тестового сервера до реального. Простота розгортання XAMPP означає, що розробник може швидко та просто встановити стек WAMP або LAMP в операційній системі, з перевагою, ніж загальні додаткові програми, такі як WordPress та Joomla! також можна встановити з такою ж легкістю за допомогою Bitnami.

MySQL – це система управління реляційними базами даних із відкритим кодом (СУБД). Реляційна база даних організовує дані в одну або кілька таблиць даних, в яких типи даних можуть бути пов’язані між собою; ці відносини допомагають структурувати дані. SQL – це мова, яку програмісти використовують для створення, модифікації та вилучення даних з реляційної бази даних, а також контролю доступу користувачів до бази даних. На додаток до реляційних баз даних та SQL, СУБД, як MySQL, працює з операційною системою для реалізації реляційної бази даних у системі зберігання даних комп'ютера, управляє користувачами.

4 РОЗРОБКА СЦЕНАРІЇВ ТЕСТУВАННЯ

4.1 Додаток Apache Jmeter

Додаток Apache JMeter – це програмне забезпечення з відкритим кодом, чиста програма на Java, призначена для завантаження тестової функціональної поведінки та вимірювання продуктивності. Спочатку віно було розроблено для тестування веб-програм, але з тих пір розширилось до інших тестових функцій.

Apache JMeter може використовуватися для тестування продуктивності як на статичних, так і на динамічних ресурсах, веб-динамічних додатках[3].

Він може бути використаний для імітації великого навантаження на сервер, групу серверів, мережі або об'єкта, щоб перевірити його міцність або проаналізувати загальну продуктивність при різних типах навантаження.

Особливості Apache JMeter включають можливість завантаження та перевірки продуктивності багатьох різних типів програм / серверів / протоколів:

- інтернет – HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET);
- веб-сервіси SOAP / REST;
- FTP;
- власні команди або сценарії оболонки;
- TCP;

Повнофункціональна тестова IDE, яка дозволяє швидко записувати план тесту (із браузерів або власних програм), створювати та налагоджувати.

Режим CLI (режим командного рядка (раніше називався Non GUI) / безголовий режим) для завантаження тесту з будь-якої сумісної з Java ОС (Linux, Windows, Mac OSX). Повний і готовий до презентації динамічний звіт HTML. Простота кореляції завдяки можливості витягувати дані з найпопулярніших форматів відповідей, HTML, JSON, XML або будь-якого

текстового формату. Повна богатопотокова структура дозволяє одночасну вибірку для багатьох потоків та одночасну вибірку різних функцій окремими групами потоків.

4.2 Інструкція налаштування програмного забезпечення

4.2.1 Встановлення Jmeter

Оскільки JMeter – це додаток розроблений на Java, для нього потрібен повністю сумісний JVM 6 або вище. Потрібно завантажити та встановити останню версію Java SE Development Kit.

Після завершення встановлення ми можемо скористатися наведеною нижче процедурою, щоб перевірити, чи успішно встановлено Java JDK в нашій системі.

Після завершення встановлення ми можемо скористатися наведеною нижче процедурою, щоб перевірити, чи успішно встановлено Java JDK в нашій системі:

- у Windows / Linux перейти до Terminal (windows+R та ввести команду “cmd”);
- ввести команду “java –version”.

Якщо після введення команди нічого не відображається, потрібно переінсталювати середовище виконання Java SE. Встановлена Java JDK зображена на рисунку 4.1.

```
C:\Users\Administrator>java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

Рисунок 4.1 – Встановлена Java JDK

Далі потрібно вибрати файл Binaries (zip або tgz) для завантаження, який знаходиться на офіційному сайті. Після цього потрібно розпакувати архів. Для запуска програми відкриємо файл “jmeter.bat” який знаходиться в папці bin.

Наступним етапом буде встановлення змінної середовища JAVA_HOME для вказівки на розташування базового каталогу, де Java встановлена на комп'ютері. Потрібно встановити змінну середовища JAVA_HOME на “C:\ Program Files\Java\jdk1.8.0_211”, та змінну “Path” на “C:\Program Files\Java\jdk1.8.0_211” як зображено на рисунку 4.2.

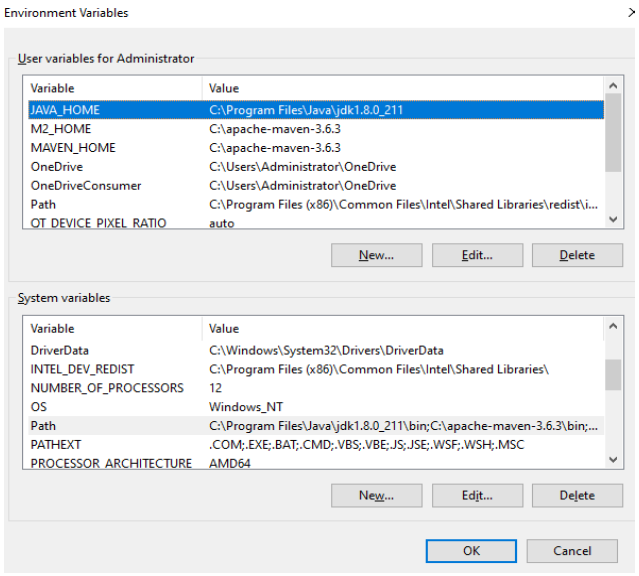

Рисунок 4.2 – Змінна середовища JAVA_HOME

4.2.2 Інструкція налаштування XAMPP та веб-додатку

Як було сказано раніше XAMPP – це збірка веб-сервера, що містить Apache, MySQL, інтерпретатор скриптів PHP, мову програмування Perl і велику кількість додаткових бібліотек, що дозволяють запустити повноцінний веб-сервер. За допомогою цього додатку ми будемо запускати сайт “Address book”, на якому будуть проводитися наші тести. Після встановлення додатку відкриваємо головний інтерфейс та запускаємо два модуля: Apache; MySQL. Успішне виконання запуску модулів зображено на рисунку 4.3.

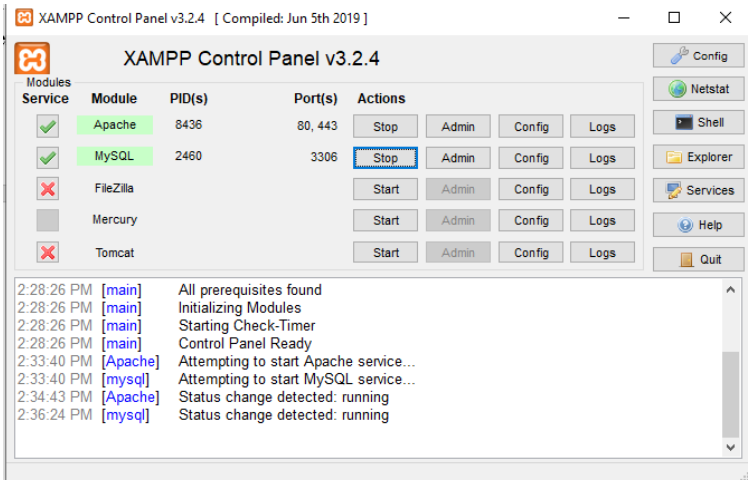

Рисунок 4.3 – Успішне увімкнення модулів

Якщо модулі не запускаються, тоді потрібно відкрити сервіси windows та увімкнути наступні сервіси: Apache; MySQL, які знаходяться в меню “служби” windows. Далі для перевірки успішної установки відкриємо браузер та зайдемо в головний інтерфейс “phpMyAdmin”. Далі перед нами повинно з’явитися наступне вікно з інтерфейсом який зображений на рисунку 4.4. Цей інтерфейс буде використовуватися для подальшого управління базою даних.

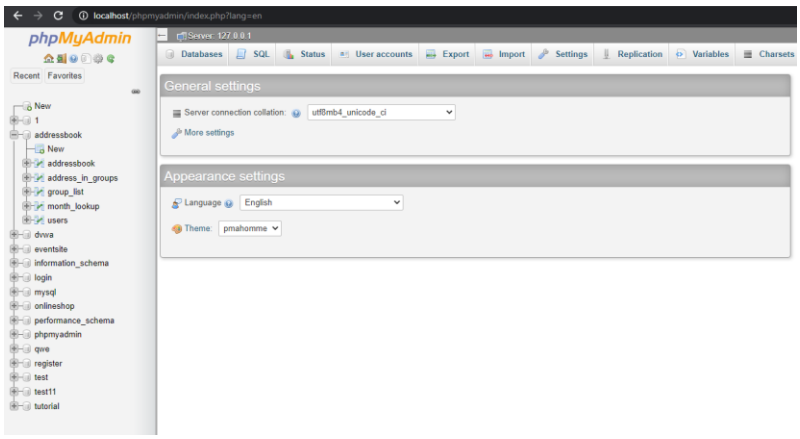

Рисунок 4.4 – Головний інтерфейс управління БД

В вікні phpmyadmin створюємо нову базу даних під назвою addressbook. Після цього переходимо в папку xampp та переміщуємо сайт “addressbook” в

папку htdocs. В наступному етапі потрібно змінити назву бази даних в файлі який знаходиться в наступній директорії: “htdocs\addressbook \config\cfg.db.php”. В рядку $dbname змінюємо назву на addressbook. Далі в інтерфейсі бази даних натискаємо кнопку “імпорт” та вибираємо файл “addressbook.sql” який знаходиться в папці addressbook. Налаштування веб-додатку виконано успішно, тепер відкриваємо браузер та переходимо на нашу локальну адресу http://localhost/addressbook/. В полях логін та пароль вводимо “admin” та “secret”.

4.2.3 Ознайомлення з функціоналом програми Jmeter

JMeter імітує групу користувачів, що відправляють запити на цільовий сервер, і повертає статистичні дані, які показують ефективність та функціональність цільового сервера або програми через таблиці, графіки[3]. На рисунку 3.5. зображений головний екран додатку. На панелі інструментів знаходяться кнопки керування файлом проекту, такі як: зберегти, запустити, та очистити пам'ять та ін.. В об’єкті “Test Plan” знаходяться майже всі налаштування та функції додатку Jmeter. В правому верхньому кутку зображений час з початку виконання проекту та кількість активних користувачів.

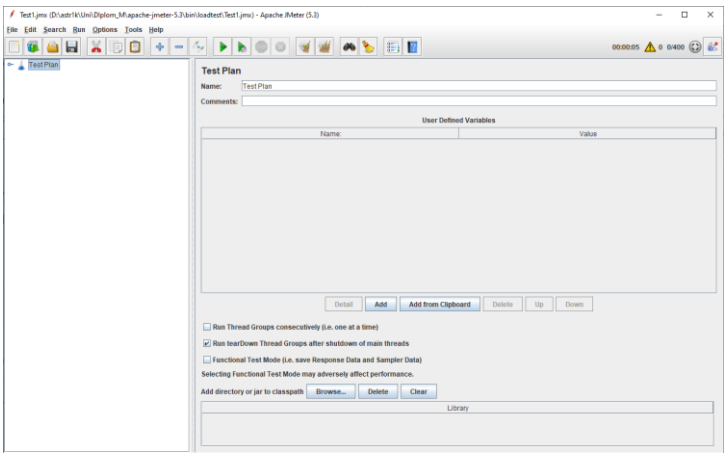

Рисунок 4.5 – Головний екран Jmeter

4.3 Структура платформи тестування

Структура платформи тестування складається з серверної та клієнтської системи. До клієнтської частини відноситься Jmeter та ServerAgent (клієнтська частина), для серверної в свою чергу відноситься XAMPP, ServerAgent (серверна частина), та веб-додаток. Структура платформи для тестування зображена на рисунку 4.6.

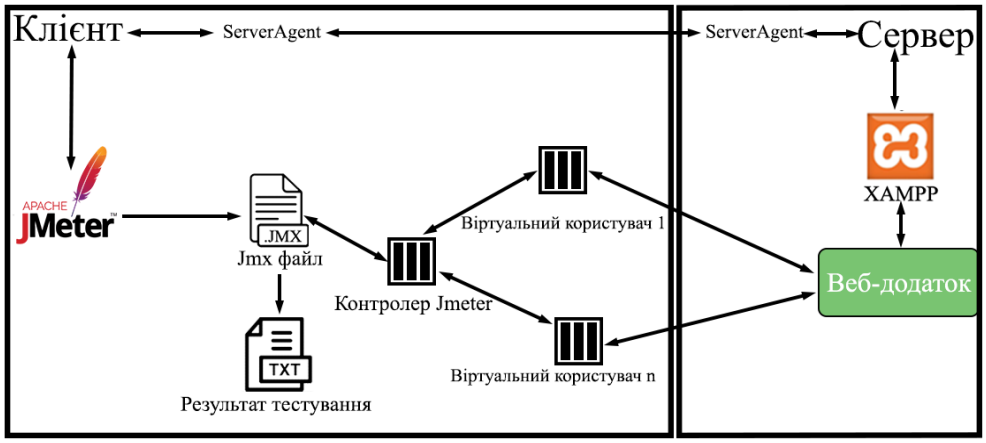

Рисунок 4.6 – Структура платформи для тестування

Як видно з рисунка 4.6, за допомогою Jmeter створюється сценарій тестування який зберігається в jmx файл та має контролер тестування. Цей контролер тестування в свою чергу має різні параметри. Для різних видів тестування використовуються різні компоненти контролера. Jmeter симулює поведінку реальних користувачів до веб-додатку. Результати тестування зберігаються в csv форматі, доступ до якого відбувається за допомогою програми MS Excel.

XAMPP виконує роль програмного забезпечення веб-сервера, за допомогою якого здійснюється хостинг веб-додатка. XAMPP поєднує в собі HTTP сервер Apache, базу даних (MySQL), та мову програмування PHP. Поєднання цих компонентів дає змогу запустити веб-додаток без встановлення додаткового програмного забезпечення. Зв'язок Jmeter з веб-

сервером відбувається завдяки HTTP запитам. HTTP працює як протокол відповіді на запит між клієнтом та сервером. Jmeter надсилає на сервер запит HTTP, тоді сервер повертає клієнту відповідь. Відповідь містить інформацію про статус запиту, а також може містити запитуваний вміст.

ServerAgent поєднує серверне та клієнтське апаратне забезпечення, що дає змогу здійснювати віддалений моніторинг ресурсів веб-сервера з клієнтського комп'ютера під час тестування веб-додатка. Головна структура JMX файлу складається з наступних компонентів:

- test plan: головний елемент який описує ряд кроків, які виконуватиме JMeter під час запуску тестування;
- thread group: це найпростіша група потоків, яка існує. Вона має деякі стандартні налаштування, і її можна адаптувати під більшість сценаріїв тестування навантаження;
- sampler: дозволяють JMeter надсилати різні типи запитів на сервер;
- listener: прослуховувач – це компонент, який показує результати виконання тестування.

Test Script Recorder – це один із компонентів, який дозволяє записувати модель поведінки сценарію за допомогою веб-браузера. Так як наш веб-сервер знаходиться на одному і тому ж комп'ютері на якому виконуються тести, ми будемо використовувати проксі сервер та localhost для доступу до веб-додатку.

Проксі-сервер – це посередник, який знаходиться між мережею та зовнішнім Інтернетом. Коли ми переходите на веб-адресу у своєму браузері, запит надходить на проксі-сервер, який завантажує сторінку, а потім відправляє її на комп'ютер. Для доступу до веб-додатку через проксі використовується порт 8080 tcp.

4.4 Тестування продуктивності

Для виконання тесту продуктивності будемо використовувати графічний інтерфейс Jmeter. Тест буде виконуватися на локальному сайті

Address Book. Мета виконання даного тесту є перевірка регулярного використання веб-додатку за допомогою симуляції різних http запитів від невеликої кількості користувачів.

Першим кроком, який потрібно робити з кожним планом тестування JMeter, є додавання елемента Thread Group. Група потоків повідомляє JMeter кількість користувачів, яких потрібно імітувати, як часто користувачі повинні надсилати запити та скільки запитів вони повинні відправляти. Додаємо функцію “Thread Group” яка знаходиться в “TestPlan – Add – Threads – Thread Group”. Інтерфейс елемента Thread Group та його параметри зображені на рисунку 4.7.

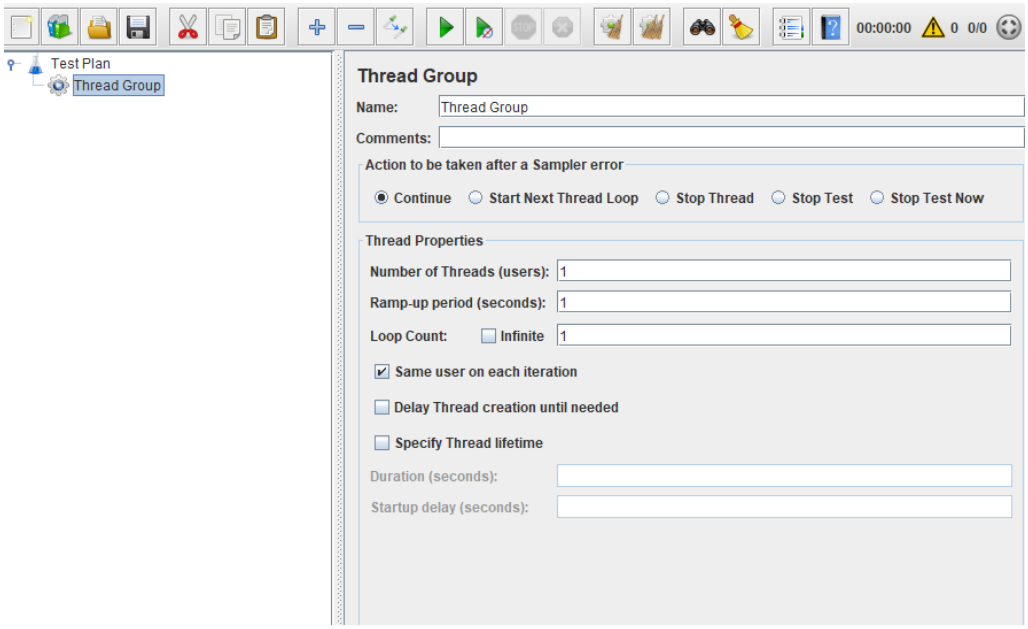

Рисунок 4.7 – Головний екран Jmeter

Елементи групи потоків є початковими пунктами плану тестування. Як випливає з назви, елементи групи потоків контролюють кількість потоків, які JMeter використовуватиме під час тесту.

Додамо елемент HTTP(S) Test Script Recorder, який знаходиться в “TestPlan – Add – Non-Test Elements”. За допомогою цього елементу ми будемо записувати наші дії які виконуються в браузері.

Це допоможе виконати тест продуктивності з різними http запитами, що є більш реалістичним сценарієм використання веб-додатку. Інтерфейс цього елементу зображений на рисунку 4.8.

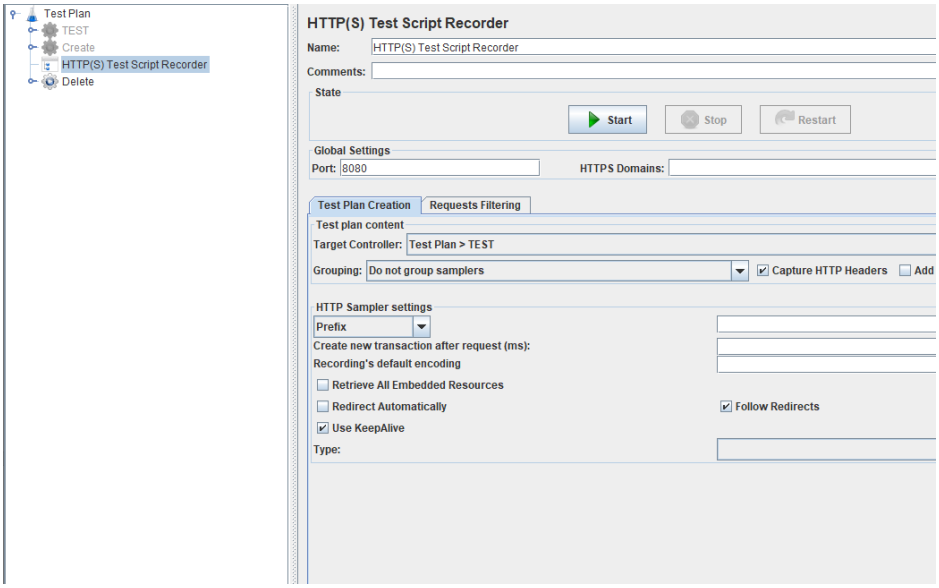

Рисунок 4.8 – Інтерфейс елементу HTTP(S) Test Script Recorder

Наступним елементом будуть прослуховувачі. Більшість прослуховувачів виконують кілька ролей, крім того, щоб прослуховувати результати тесту. Вони також надають засоби для перегляду, збереження та читання збережених результатів тесту.

Для тестування продуктивності потрібно додати такі прослуховувачі як: “Graph Results”, “Summary Report”, “Generate Summary Results”, “jp@gc – Response Times Over Time”. В прослуховувачі Summary Report створюємо файл в який буде записуватися вся інформація. Також на час тесту потрібно вимкнути інші прослуховувачі, тому що вони споживають багато ресурсів.

Відкриваємо елемент Test Script Recorder. В полі Port вписуємо: 8080. Елемент Target Controller змінюємо на: “Test Plan > Thread Group”. В браузері Firefox відкриваємо налаштування, переходимо в налаштування мережі та вмикаємо ручну конфігурацію проксі. Проксі по HTTP вписуємо localhost, а в порт 8080. Ця інформація зображена на рисунку 4.9.

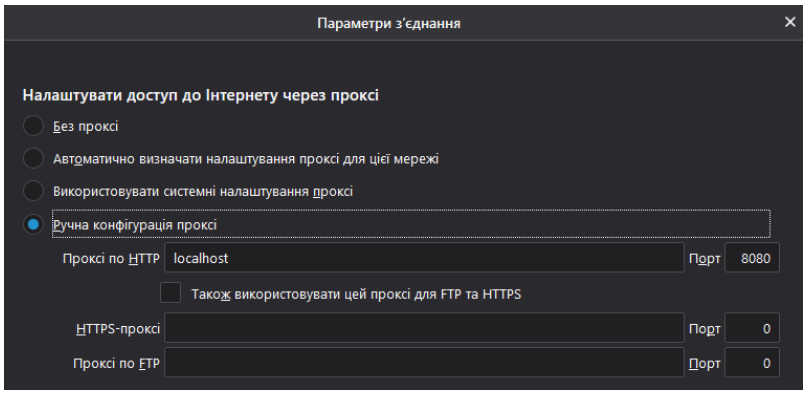

Рисунок 4.9 – Інтерфейс налаштування проксі серверу

Для того щоб firefox реагував на localhost, в адресному рядку пишемо “about:config”, далі в пошуку вписуємо “network.proxy.allow_hijacking_ localhost” та змінюємо false на true. Після виконання цих дій ми приступаємо до запису відповідних запитів. В Test Script Recorder натискаємо “Start” в браузері в налаштуваннях переходимо в розділ “сертифікати” та імпортуємо сертифікат “ApacheJMeter-TemporaryRootCA.crt” який розташований в папці jmeter/bin/. Переходимо по адресі “http://localhost/addressbook/”, вписуємо логін та пароль, переходимо в розділ “groups” та натискаємо “new groups”, заповнюємо інформацію та натискаємо Enter information та повертаємось в розділ “groups”. Після виконання цих дій в Jmeter натискаємо Stop. Під елементом Thread Group з'явилися HTTP запити які зображені на рисунку 4.10. Для того щоб не записувати статичні елементи потрібно додати фільтр: script recorder – request filtering–exclude–“.*\.(jpg|gif|png)”.

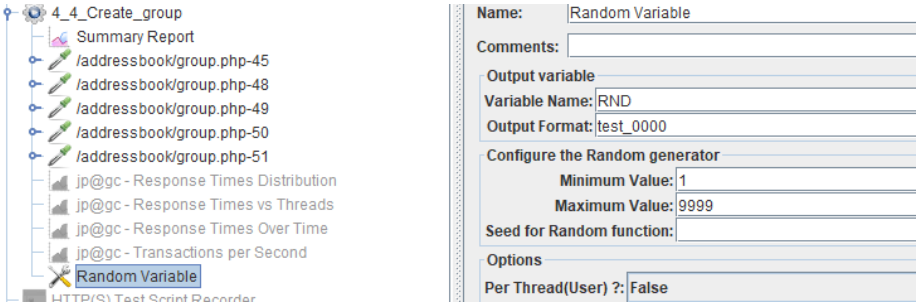

Рисунок 4.10 – HTTP запити

В параметрі “/group.php-50” виставляємо наступні значення для рядків Name та Value: group_name: name_${RND}; group_parent_id: 0; group_header: header_${RND}; group_footer: footer_${RND}; submit: Enter information. Далі додаємо елемент “Random Variable” та вписуємо налаштування. В полі Variable Name пишемо RND; Output Format test_0000; Min та Max: 1 та 9999. Перед тестуванням виконаємо налаштування “Thread Group”. Виставляємо такі параметри як:

- кількість потоків: 50 користувачів;
- період нарощування потужності: 45 секунд;
- тривалість виконання: 60 секунд;

Після проведення тесту перевіряємо результати які знаходяться в графіках. Результат прослуховувача Response Times Over Time зображений на рисунку 4.11.

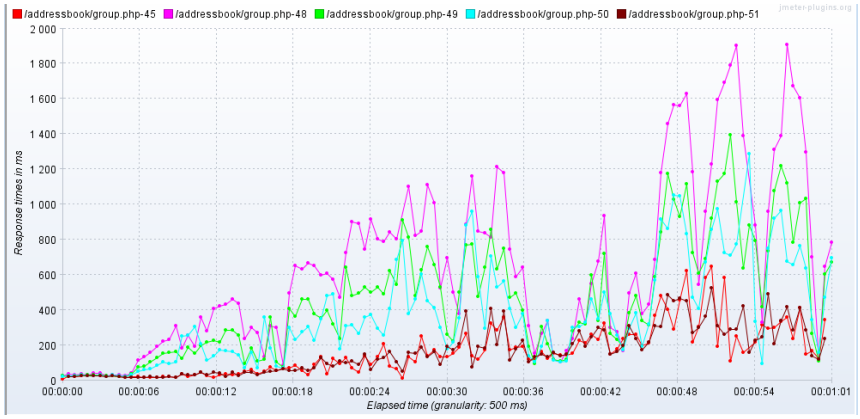

Рисунок 4.11 – Результат прослуховувача Response Times Over Time

З цього рисунку ми бачимо що найбільший час відгуку був на сторінці логіну до сайту, яка зображена фіолетовим кольором. Але навіть під кінець тесту ми маємо результати часу відгуку, які не перевищують 2 секунди.

Як видно із рисунка 4.12. більшість запитів знаходяться в діапазоні від 1 до 400мс. що є дуже хорошим результатом для такого тесту.

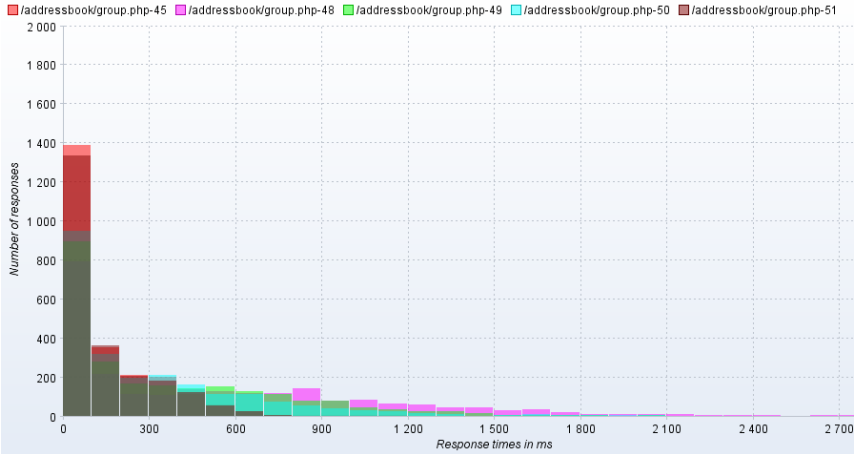

Рисунок 4.12 – Розподіл часу відгуку

Так як ми проводили тестування функціоналу веб-додатку, ми маємо результат виконання запитів – це створені за допомогою jmeter групи під назвою “TEST” на сайті Address Book. Результат виконання зображений на рисунку 4.13.

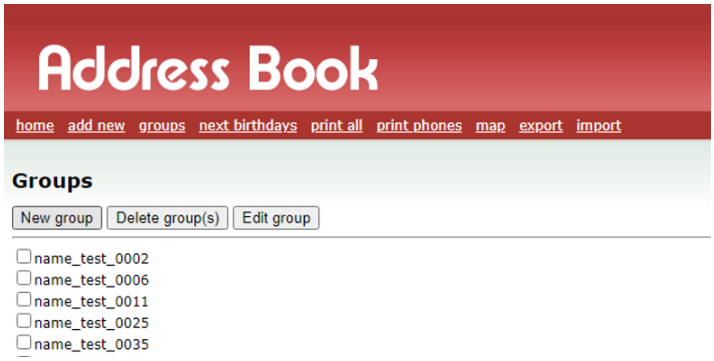

Рисунок 4.13 – Результат виконання запитів.

4.5 Тестування видалення груп

Так як в попередньому тесті ми створили велику кількість груп за допомогою використання POST запитів, то в цьому тесті ми виконаємо налаштування автоматичного видалення всіх груп та протестуємо продуктивність веб-додатку при таких умовах. Для цього тесту знову будемо

використовувати HTTP(S) Test Script Recorder для запису процесу видалення елементів груп. Після запису в jmeter повинні з’явитися запити. З усіх запитів ми залишаємо лише два: логін на сторінку та видалення групи. Ці запити зображені на рисунку 4.14.

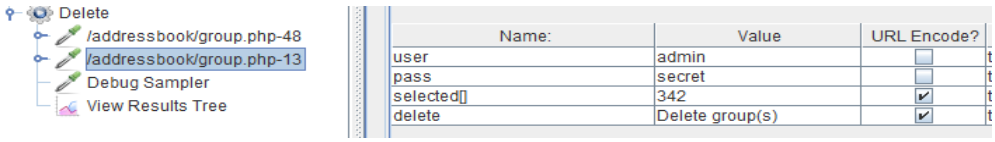

Рисунок 4.14 – Результат запису

Так як ми видалили лише одну групу, то в полі “selected” ми маємо одне значення “342”, яке відповідає значенню видаленого елементу. Тому для видалення всіх елементів потрібно використовувати постпроцесор під назвою “ XPath Extractor”. Елементи зображені на рисунку 3.15.

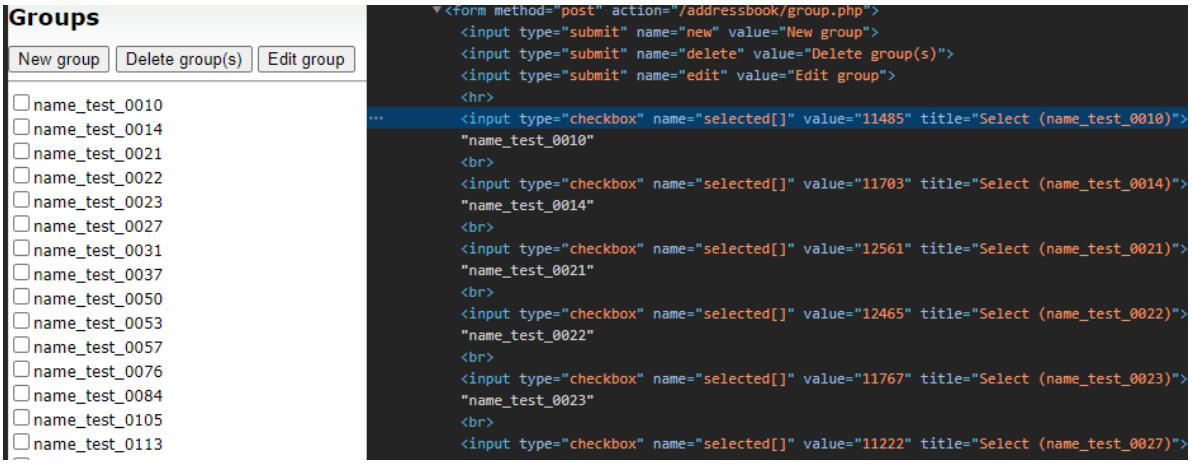

Рисунок 4.15 – Елементи груп

Далі нам потрібно отримати xpath запит який буде знаходити значення кожної групи та передавати до елементу selected[] який зображений на рисунку 3.15. В браузері переходимо до сторінки з групами, натискаємо f12 та знаходимо потрібний елемент з групами. Для вибору значення груп був

написаний наступний запит:“//input[@name='selected[]']//@value”, який шукає атрибут з іменем “selected[]” та отримує id елемента з поля “value”.

Після виконання цих дій ми переходимо до налаштування XPath Extractor, та вписуємо наступні параметри:

- name of created variable: ID;
- xpath query: //input[@name='selected[]']//@value;
- match No.: -1;
- default Value: Error.

XPath Extractor розміщується під першим http запитом для того щоб він встиг записати дані в змінну та передати їх до другого запиту. В другому запиті в полі “selected[]” в значенні “Value” вписуємо змінну “${ID}”. Запускаємо тест та перевіряємо успішність виконання запиту за допомогою View Results Tree. Як видно з рисунку 4.16. в полі нашого запиту підставилось значення id:12109. Також ми бачимо повідомлення про успішність видалення групи.

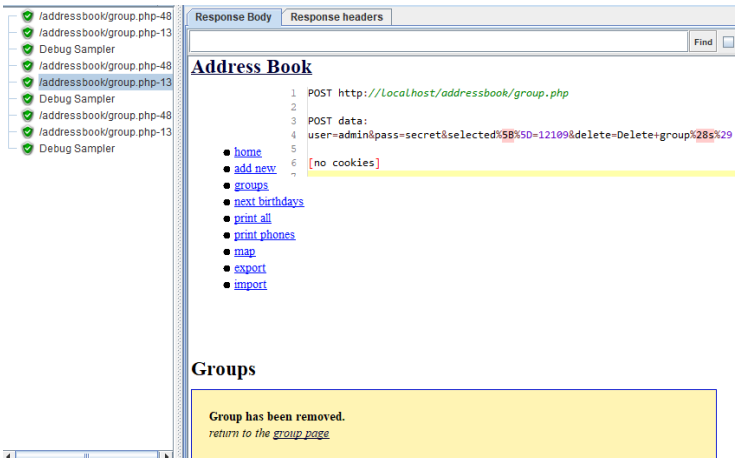

Рисунок 3.16 – Перевірка виконання запиту

Перед повним тестуванням виконаємо налаштування “Thread Group”. Виставляємо наступні параметри: 175 користувачів, нарощування потужності 50 секунд та тривалість тестування в 60 секунд. Після успішного виконання тесту за допомогою Debug Sampler ми отримуємо список всіх id груп які були видалені. Ця інформація зображена на рисунку 4.17.

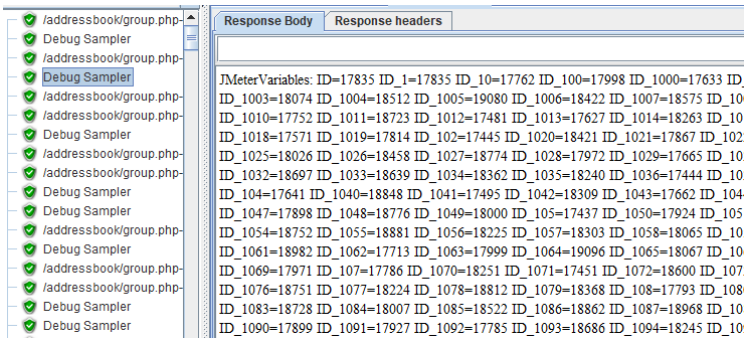

Рисунок 4.17 – ID видалених груп

На наступному рисунку 4.18. зображений час відгуку веб-додатку впродовж всього тесту. Ми бачимо що під кінець тесту час відгуку досягав відмітки в 1200-2400мс, а середній час був 303мс. Ці результати цілком відповідають нашим вимогам.

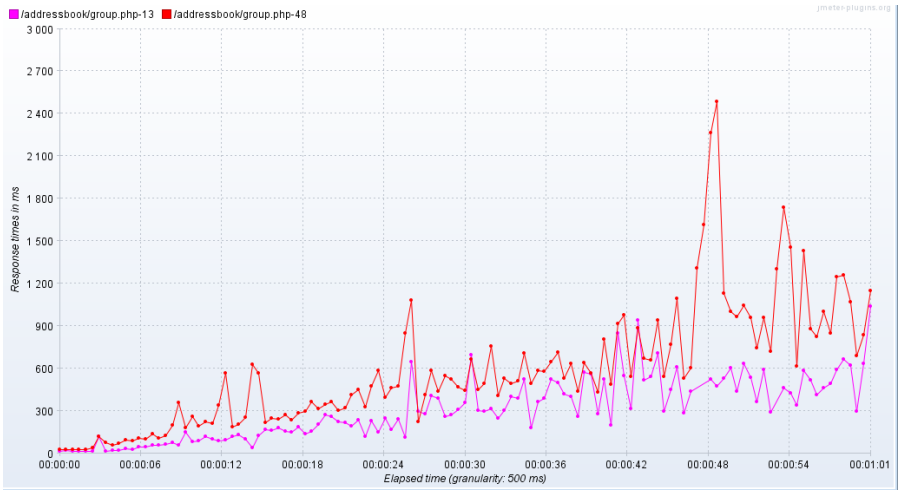

Рисунок 4.18 – Час відгуку

В даному тесті було виконано видалення великого списку груп, які були створені в попередньому тесті. Це було виконано за допомогою елементу XPath Extractor, який дозволяє виконувати запити для пошуку елементів в html, та зберігати дані в змінну, яка в майбутньому буде використана для видалення елементів.

4.6 Визначення точки насичення

Мета даного тесту – реалізувати безперервно зростаюче навантаження і визначити точку насичення (для нашої конкретної програмно-апаратної конфігурації), тобто максимальну кількість паралельно активних потоків, при перевищенні якого починаються відмови в обслуговуванні.

Для цього тестування будемо використовувати веб-додаток та елементи із минулого тесту. Перед тестуванням виконаємо налаштування “Thread Group”. Виставляємо такі параметри як: 807 користувачів, період нарощування потужності 88 секунд та тривалість виконання 88 секунд. Для того щоб вчасно побачити точку насичення, та виконати аналіз даних, добавимо наступні прослуховувачі: Graph Result; Summary Report. Після запуску тесту потрібно дивитися на кількість активних користувачів, та на поля “Devitaion” і “Summary Report”. Коли середньоквадратичне відхилення почне різко зростати, та в полі Error почнуть з’являтися помилки, то це означає що ми знайшли точку насичення для нашої конфігурації.

Як видно з рисунку 4.19 час відгуку почав різко зростати коли кількість користувачів перевищила відмітку в 500. Саме в цей час середньоквадратичне відхилення почало зростати одночасно з кількістю користувачів.

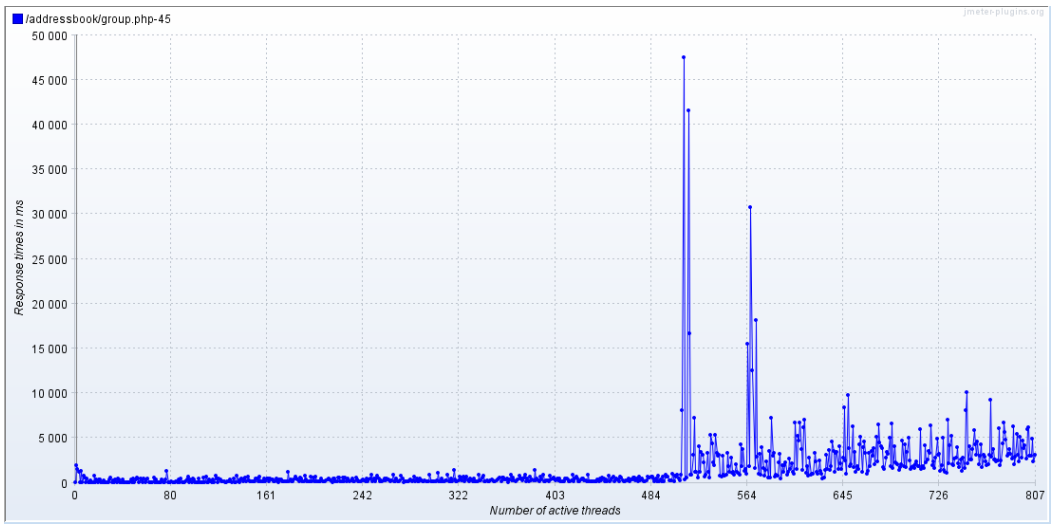

Рисунок 4.19 – Співвідношення часу відгуку та кількісті користувачів

Після перевищення відмітки в 500 користувачів(точки насичення), ми почали отримувати коди помилок. Це означає що сервер не встигає обробляти всі запити. На рисунку 4.20 зображений графік успішних відповідей та помилок на запити. Відсоток помилок становить 2.4% від всіх запитів.

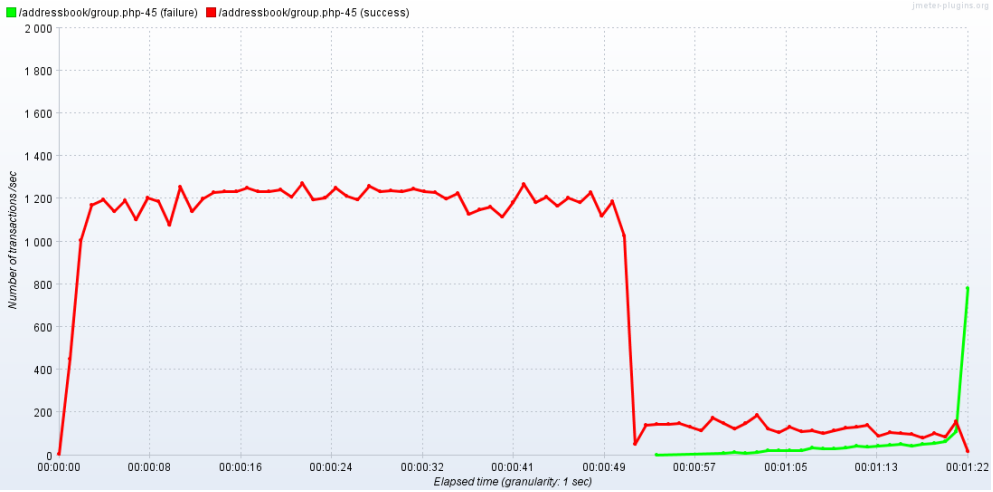

Рисунок 4.20 – Успішні відповіді та помилки

За допомогою даного тестування ми визначили точку насичення в 500-550 активних користувачів. Ця інформація дає нам можливість підготуватися до таких навантажень та уникати поганої роботи сайту.

4.7 Тестування динамічного навантаження та моніторинг ресурсів сервера

Мета даного тестування полягає в динамічному навантаженні та перевірці роботи веб-додатку після пікового навантаження. Також в цьому тесті ми будемо використовувати PerfMon Server Agent для моніторингу ресурсів нашого серверу. Для початку потрібно скачати ServerAgent та розпакувати його на нашому сервері. Після цього запускаємо файл під назвою startAgent.bat. Якщо все правильно зроблено то ми отримаємо інформацію яка зображена на рисунку 4.21.

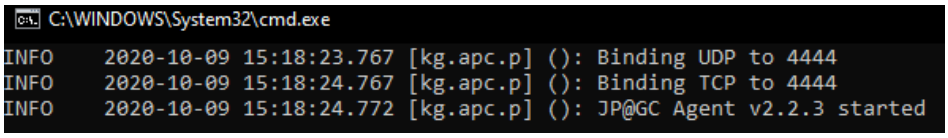

Рисунок 4.21 – Робочий ServerAgent

Далі на клієнті відкриваємо командний рядок та підключаємося до нашого серверу за допомогою команди “telnet ip серверу порт”, наприклад “telnet 192.168.1.1 4444”. Після цього перевіряємо підключення до серверу. Для цього пишемо команду “test”, натискаємо enter, після цього повинен з’явитися напис “YEP”. Після того як налаштування серверу зроблено успішно, відкриваємо Jmeter, переходимо до вкладки “options – plugins manager” шукаємо та додаємо плагін “PerfMon (Servers Performance Monitoring)”. З попереднього тесту нам відомо що пік потужності тестового веб-додатку від 500 до 550 одночасно активних користувачів. Для динамічного тесту використовується елемент: Ultimate Thread Group, який дає повний контроль над динамічною кількістю користувачів. Налаштування будуть спиратися на нашу точку насичення в 500 користувачів. На рисунку 4.22 зображені налаштування та графік активних користувачів в період часу.

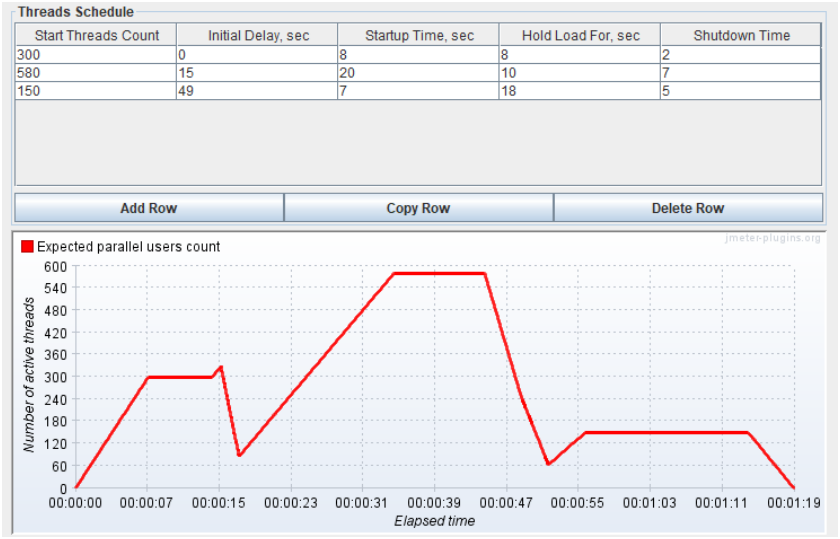

Рисунок 4.22 – Налаштування користувачів

Додаємо плагін PerfMon, “Add – Listeners – jp@gc – PerfMon Metrics Collector”. В вікні інтерфейсу як зображено на рисунку 4.23 в розділі “host/ip” пишемо ip нашого серверу, в даному випадку localhost, в розділі “port” пишемо 4444, та в “metric to collect” вибираємо “CPU, Memory, Disks I/O”

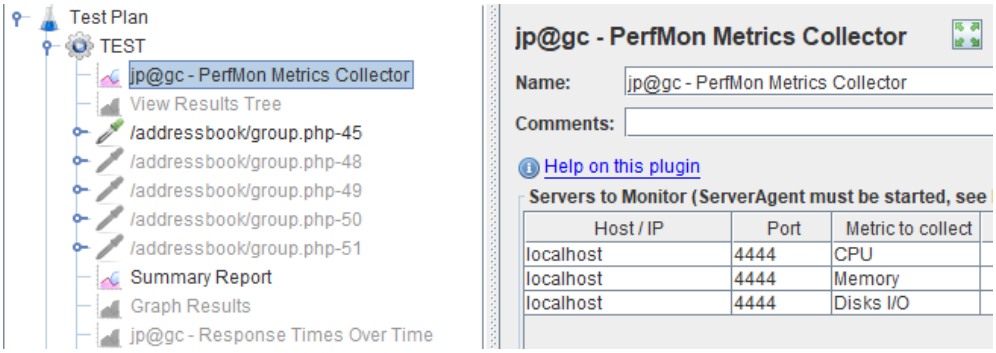

Рисунок 4.23 – Налаштування плагіну PerfMon

Для наступного графіку нам потрібен плагін jp@gc – Response Codes per Second. Як і в попередньому тесті ми починаємо отримувати помилкові запити на п’ятистах активних користувачах, це зображено на рисунку 4.24 на 31 секунді. Але після зменшення навантаження на 55 секунді до 160 користувачів ми продовжуємо отримувати помилкові запити.

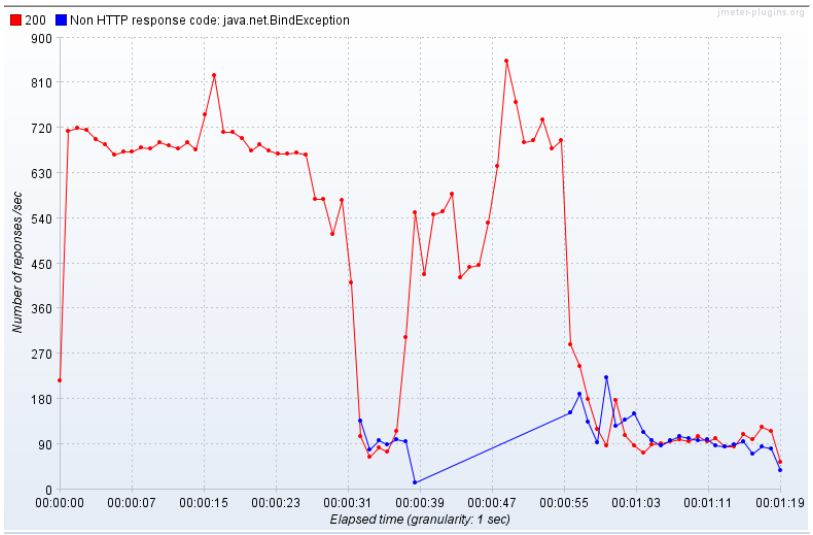

Рисунок 4.24 – Співвідношення помилкових запитів

В таблиці 4.1. наведені основні метрики які були отримані в тесті. Такі як: кількість запитів, середній та максимальний час відгуку, процент помилкових запросів, пропускна здатність та інші.

Таблиця 4.1 – Основні метрики Get запиту

Тип запиту	Поняття				
	Кількість запитів	Середній час відгуку, с	Максимальний час відгуку, с	Помилкові запити, %	Кількість запитів в секунду
Get	38709	0.37	6.452	8.42	490

Як видно із таблиці 4.1 ми із 38709 запитів 8.42% було помилкових, що говорить про те що система не може витримати навантаження в 490 кб/с.

Також основним елементом тестування є час відгуку. На рисунку 4.25. зображено розподілення кількості відповідей серверу на час відгуку. Ми маємо 82.7% відповідей з середнім часом відгуку в 300мс. та 17.3% відповідей з часом відгуку від 500мс до 6400мс. Час відгуку до 2 секунд є нормальним результатом.

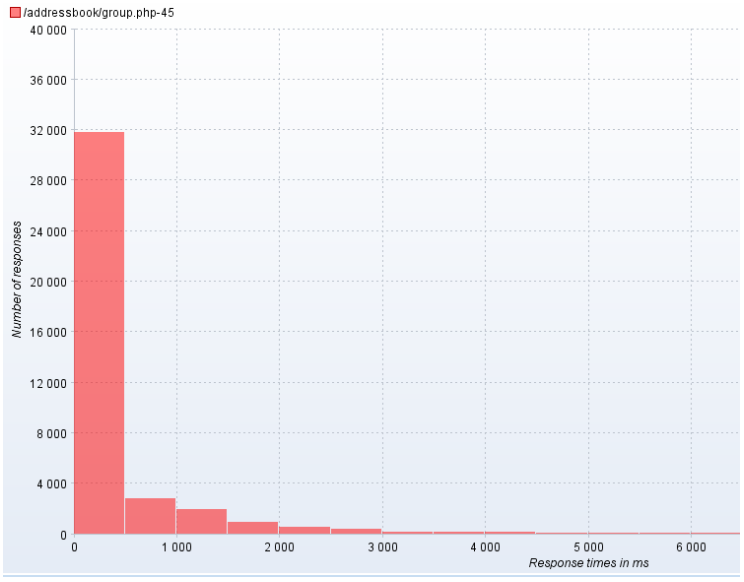

Рисунок 4.25 – Час відгуку відповідей

Як видно з рисунку 4.26 моніторинг показує стан трьох компонентів серверу: завантаження процесору, оперативної пам’яті та запис на диск.

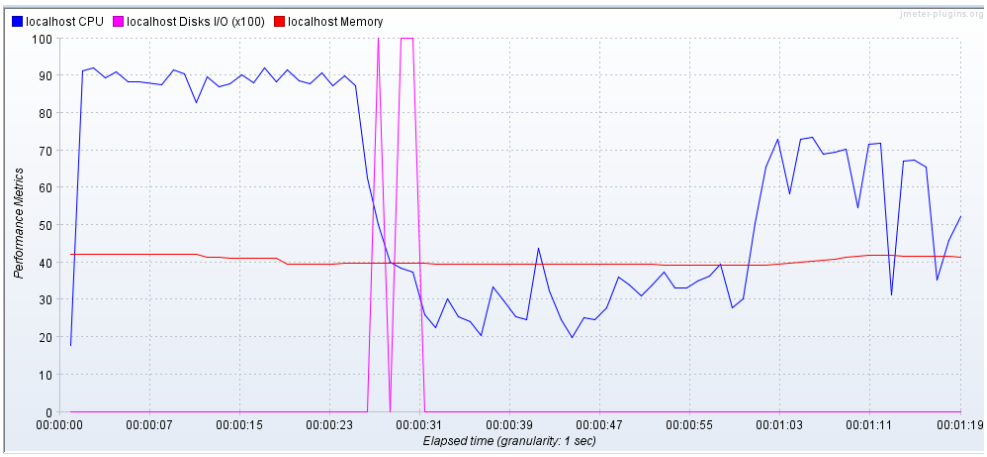

Рисунок 4.26 – Моніторинг ресурсів серверу

Після проведення тесту динамічного навантаження можна зробити такий висновок: веб-додаток може витримати до 500 активних користувачів, після цієї відмітки ми починаємо отримувати помилкові запити. Із таблиці 4.1 видно що із 38709 запитів 8.42% були помилкові, тобто це 3259 запитів. Також ми дізналися що наш веб-додаток все одно відповідає помилковими запитами навіть після зниження навантаження до 160 активних користувачів. Протягом тесту ми збирали метрики серверу. Навантаження процесору було від 20% до 90%, завантаження пам’яті 40%, а диск від 0.1% до 1%.

4.8 Тестування методів HTTP на JSON

JSON (англ. JavaScript Object Notation) – текстовий формат обміну даними, заснований на JavaScript. Для використання javaScript та JSON потрібно інсталювати Node.js та NPM з офіційного сайту. Node.js – це середовище виконання, яке включає все необхідне для запуску програми, написаної на JavaScript. Він використовується для запуску сценаріїв на сервері для візуалізації вмісту перед його доставкою до веб-браузера.

Протокол передачі гіпертексту (HTTP). HTTP – це протокол, що використовується для передачі даних через Інтернет. Він є частиною набору протоколів Інтернету та визначає команди та служби, що використовуються для передачі даних веб-сторінки.

HTTP також визначає такі команди, як GET і POST, які використовуються для обробки поданих форм на веб-сайтах. Команда CONNECT використовується для забезпечення безпечного з'єднання, яке шифрується за допомогою SSL. Зашифровані з'єднання HTTP здійснюються через HTTPS, розширення HTTP, призначене для безпечної передачі даних.

Метод GET вимагає представлення зазначеного ресурсу. Запити за допомогою GET повинні лише отримувати дані, а не змінювати їх.

Метод POST використовується для подання даних до вказаного ресурсу, часто спричиняючи зміну стану на сервері.

NPM розшифровується як Node Package Manager, який є додатком та сховищем для розробки та спільного використання коду JavaScript.

Для перевірки інсталяції в командному рядку вводимо наступні команди: “node -v”, “npm -v”. Якщо після введення цих команд в командному рядку появляться версії програм, то все правильно встановлено.

Для перевірки JSON була розроблений міні веб-додаток для тестування.

Для розміщення json файлів використовувалася проста "база даних" json-server, яка використовує файл JSON для Node.JS.

Інсталяція цього сервера виконується через командний рядок, за допомогою команди: “npm install -g json-server”. Далі переходимо в папку в яку було встановлено npm“:C:\Users\Administrator \AppData \Roaming\npm \node_module- s\json-server\” та створюємо папку с будь-якою назвою, наприклад test_server.

Після цього створюємо файл з назвою db.json та пишемо структуру даних на json яка зображена в прикладі 4.1.

```
{ "employees":[ {
            "id": 1,
            "first_name": "Alex",
            "last_name": "Palmer",
            "email": "alex@gmail.com",
            "phone": "124421421"},   {
            "id": 2,
            "first_name": "Frank",
            "last_name": "Smith",
            "email": "Frank@gmail.com",
            "phone": "5215123421" },{"id": 3,
            "first_name": "Ann",
            "last_name": "Eschweiler",
            "email": "Ann@gmail.com",
            "phone": "5214241241"} ]}
```

Приклад 4.1 – Додання інформації до файлу (файл db.json)

Після цього в командному рядку шукаємо шлях до створеної папки test_server та вписуємо наступну команду яка виконає запуск сервера: “json-server --watch db.json“. Після правильного налаштування відкриваємо браузер та переходимо на локальну адресу: “http://localhost:3000/employees/”. Для швидкої генерації інформації будемо використовувати Faker.js. Для інсталювання faker відкриваємо cmd.exe та вказуємо шлях до нашого серверу. Далі пишемо команду “npm init” та натискаємо enter доки не з’явиться напис “Is this ok?”, вписуємо букву “y” та натискаємо кнопку далі. Після цього вписуємо команду “npm install faker”. В папці з сервером створюємо файл та називаємо його “employees.js”. Далі напишемо javascript код який наведений в прикладі 4.2, який створює 50 співробітників з наступними полями: id, first_name, last_name, email, phone.

```
var faker = require('faker') let fs = require('fs');
function generateEmployees(){ var employees = []
    for (var id = 1; id < 51; id++){
        var firstName = faker.name.firstName();
        var lastName = faker.name.lastName();
        var email = faker.internet.email();
        var phone = faker.phone.phoneNumber();
        employees.push({ "id": id, "first_name": firstName,
            "last_name": lastName, "email": email,
            "phone": phone})} return {"employees": employees}}
module.exports = generateEmployees let dataObj = generateEmployees();
fs.writeFileSync('db.json', JSON.stringify(dataObj, null, '\t'));
```

Приклад 4.2 – Випадкова генерація даних (файл employees.js)

На виході ми отримаємо файл який виконує роль бази даних, вміст цього файлу зображений на рисунку 4.27.

```
{
    "id": 14,
    "first_name": "Carey",
    "last_name": "Jast",
    "email": "Ashley_Simonis12@yahoo.com",
    "phone": "443.517.6178"
},
{
    "id": 15,
    "first_name": "Dolly",
    "last_name": "Reilly",
    "email": "Daphne_Ritchie@gmail.com",
    "phone": "1-855-624-7051"
},
{
    "id": 16,
    "first_name": "Laura",
    "last_name": "Langosh",
    "email": "Octavia.Will45@yahoo.com",
    "phone": "1-717-457-3048"
},
```


Рисунок 4.27 – Вміст файлу серверу

Після цього нам потрібно запустити наш сервер заново, щоб оновити інформацію в фалі. В командному рядку шукаємо шлях до створеної папки test_server та вписуємо наступну команду яка виконає запуск сервера: “json-server employees.js“. Результат правильного запуску зображений на рисунку 4.28.

Рисунок 4.28 – Запуск серверу

Для роботи з JSON потрібно додати наступні елементи в Jmeter: Thread Group; sampler(“http request”, “debug sampler”); post-processors(“json exctractor”); listeners(“view results tree”). Елемент thread group залишаємо з стандартними налаштуваннями.

Http Request:

- server Name or ip: localhost;
- port: 3000;
- HTTP Request: GET;
- path: /employees.

GET використовується для запиту даних із зазначеного ресурсу. Запити GET мають деякі властивості: можна кешувати, залишаються в історії браузера, можна додати в закладки. Для першої задачі виведемо перші 20 id співробітників та інформацію про них. Для цього в елементі json extractor зробимо наступні налаштування:

- names of created varibales: ID;
- JSON Path expression: $..[?(@.id <=20)].["first_name","last_name", "email","id"];
- match No.: -1;
- default Values: Value.

Після цього зберігаємо та запускаємо проект. В полі View Results Tree, знаходимо елемент “Debug Sampler” та відкриваємо елемент Response data. Результат нашого запиту зображений на рисунку 4.29.

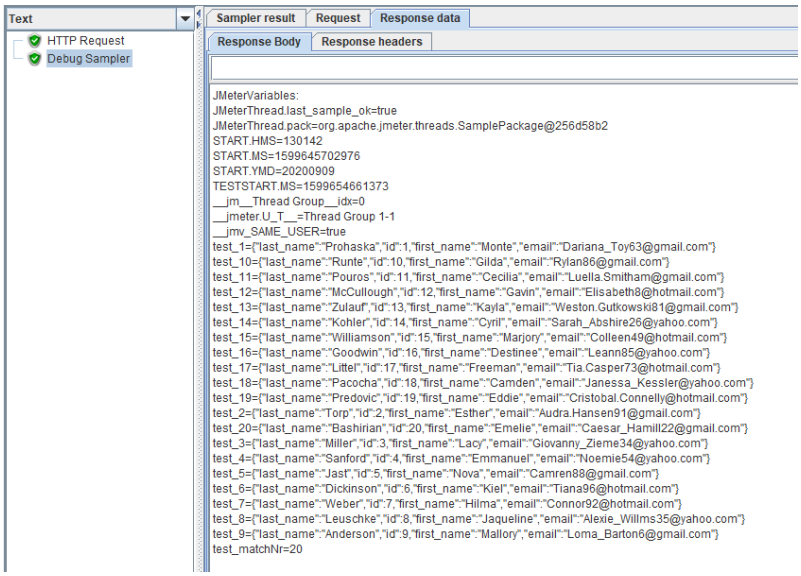

Рисунок 4.29 – Результат вибірки по ID

Виконаємо ще декілька Get запитів, таких як: знайти співробітників в яких телефонні номери починаються на цифру 5, та знайти співробітників в яких ім’я починається на літеру “K”.

Для цього додаємо два різних JSON Extractor, та виставляємо наступні параметри для першого файлу:

- names of created varibales: First_name;
- JSON Path expression: $..[?(@.first_name =~ /K.*/i)];
- match No.: -1;
- default Values: Value;
- параметри для другого файлу:
- names of created varibales: Phone_Number;
- JSON Path expression: $..[?(@.phone =~ /5.*/i)];
- match No.: -1;
- default Values: Value.

Результат виконання запиту зображений на рисунку 4.30.

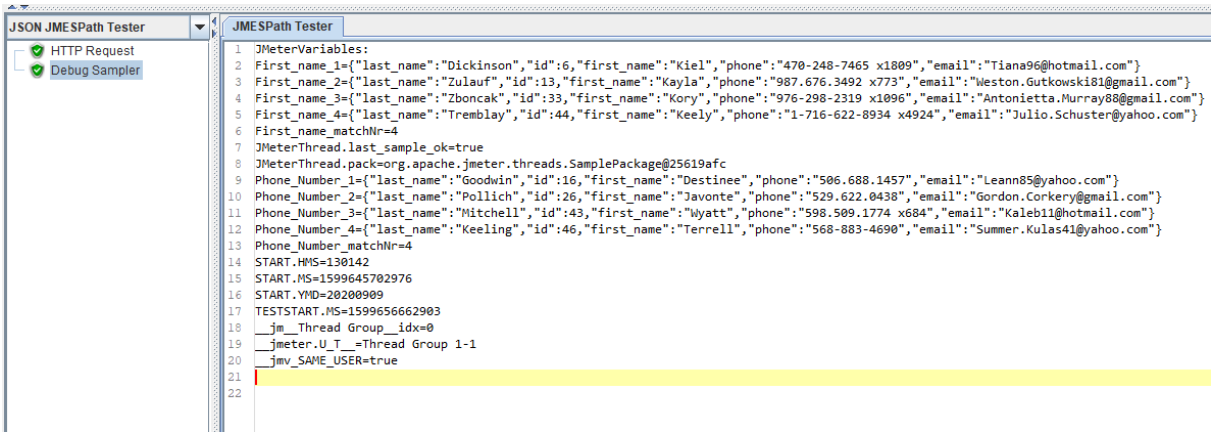

Рисунок 4.30 – Результат вибірки по номеру телефона та літері.

Також перевіримо http запити в цьому веб-додатку під невеликим навантаженням. Перед тестуванням виконаємо налаштування “Thread Group”. Виставляємо такі параметри як: 450 користувачів, період нарощування потужності 30 секунд та тривалість виконання 50 секунд.

Також будемо використовувати get та post запити до серверу. Після проведення тесту ми маємо наступний результат який наведено в таблиці 4.2.

Таблиця 4.2 – Результат тестування Get та Post запитів

Тип запиту	Поняття				
	Кількість запитів	Середній час відгуку, мс	Максимальний час відгуку, мс	Помилкові запити, %	Кількість запитів в секунду
Get	3243	2933	6437	0	62
Post	3243	2103	4993	0	62
Разом	6486	2534	6437	0	124

Ці тести проводилися для того щоб перевірити правильність та цілісність інформації на нашому сервері. Як видно з тестів, всі наші запити були виконані та отримані результати відповідають нашим запитам. Так як кількість помилкових запитів була 0 відсотків, то вся інформація була зчитана та записана до бази даних. На рисунку 4.31 зображене співвідношення кількості потоків до часу відгуку для get та post запитів.

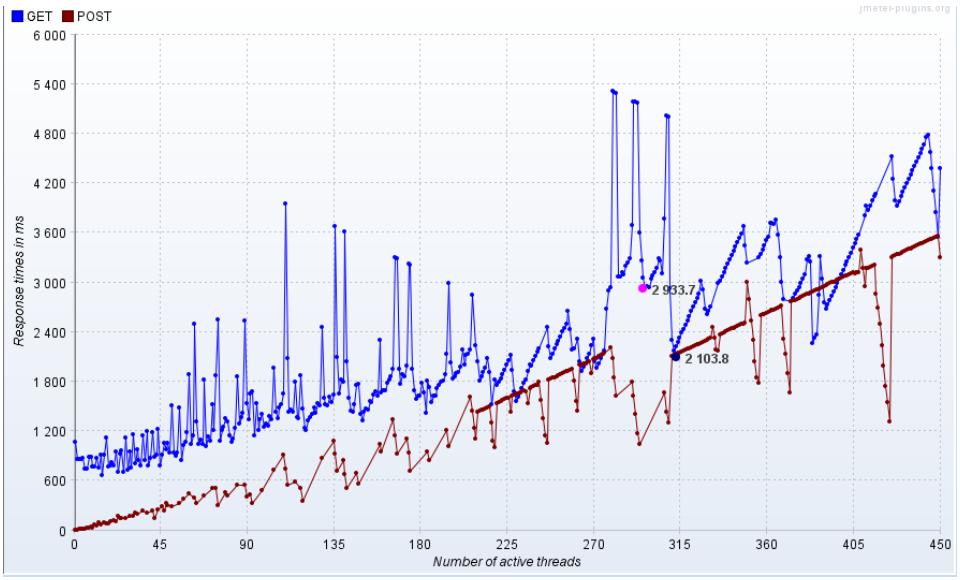

Рисунок 4.31 – Співвідношення кількості потоків до часу відгуку

З графіка видно що з ростом кількості активних користувачів зростає час відгуку. На 450 активних користувачів час відгуку становить 3.6 секунди що є відчутною затримкою.

4.9 Тест на витривалість

Тест на витривалість буде проводитися для визначення стабільності веб-додатку протягом тривалого часу[1]. Для перевірки стану серверу (наприклад, навантаження процесора, навантаження диска, використання пам’яті або пропускна здатність мережі) будемо використовувати додаток ServerAgent та плагін PerfMon.

Виконаємо налаштування “Thread Group”. Виставляємо такі параметри як: 300 користувачів, період нарощування потужності 90 секунд та тривалість виконання тестування 400 секунд.

На рисунку 4.32 зображений час відгуку сторінки логіну та експорту. Ми бачимо що логін до сайту вимагає більше часу у відмінності від переходу до сторінки експорту.

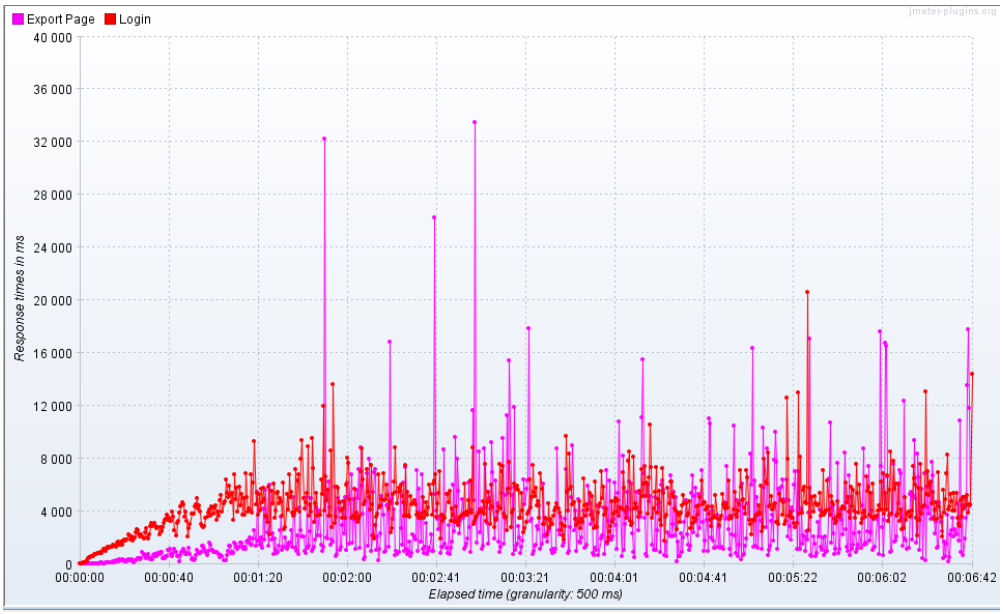

Рисунок 4.32 – Розподіл часу відгуку

На рисунку 4.33 зображений стан серверу під час проведення тесту на тривалість. Ми можемо побачити що вузьким місцем в нашому сервері був процесор, який перебував в 100% навантаженні протягом всього тесту.

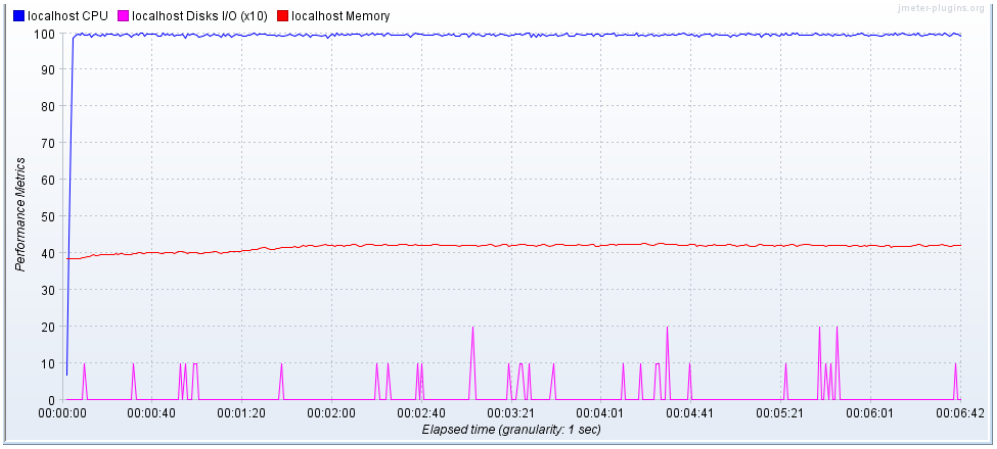

Рисунок 4.33 – Стан сервера

На рисунку 4.34 зображено розподілення часу відгуку. Як видно 90% значень сторінки логіна мають час відгука від 5мс до 4000мс, та сторінка експорту має значення від 5мс до 2000мс. Також 10% значень мають час відгука від 2000мс до 36000мс.

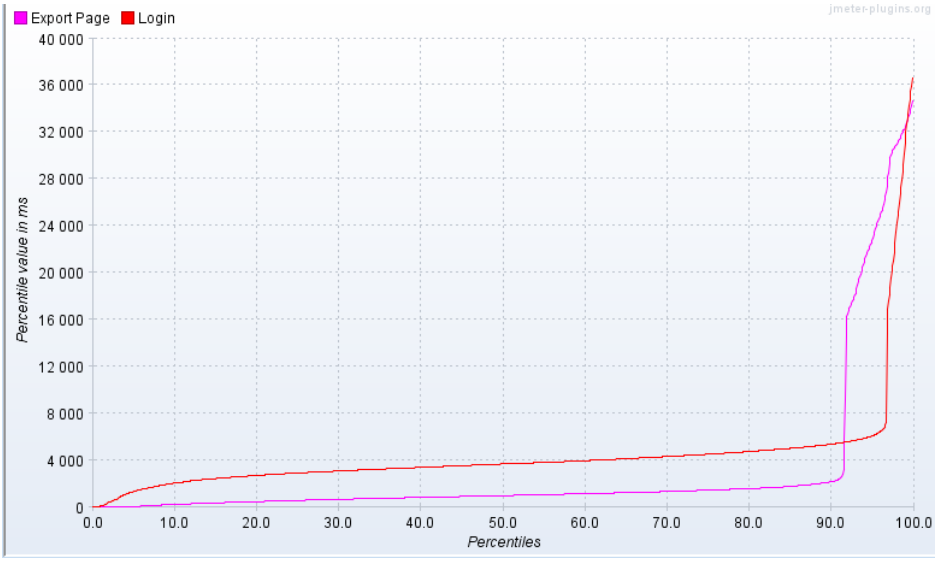

Рисунок 4.34 – Перцентиль часу відгуку

Тест на витривалість проводився, щоб виявити, як система поводиться при тривалому використанні. За допомогою цього тесту ми дізналися що наша система може витримати 250 одночасних користувачів які виконують процедуру логіну до веб-додатку та перехід до вкладки експорту. Тест виконувався впродовж 400 секунд, і ми бачимо що час відгуку сторінки логіна був за невеликими межами комфортного значення в 2 секунди. Вузьким місцем в нашому сервері був процесор, який був навантажений на 100%.

4.10 Додаткові види тестування

SOAP (Simple Object Access Protocol) – це специфікація протоколу обміну повідомленнями для обміну структурованою інформацією при реалізації веб-служб у комп’ютерних мережах. Його мета – забезпечити розширюваність, нейтральність, багатослівність та незалежність. Він використовує набір інформаційних даних XML для свого формату повідомлень і покладається на протоколи прикладного рівня, найчастіше протокол передачі гіпертексту HTTP.

Протокол передачі файлів (FTP) – це стандартний мережевий протокол, який використовується для передачі комп'ютерних файлів між клієнтом і сервером в комп'ютерній мережі. FTP побудований на архітектурі моделі клієнт-сервер, використовуючи окреме керування та з'єднання даних між клієнтом та сервером. Цей вид тестування дозволяє перевірити завантаження файлів з серверу та на сервер. В Jmeter цей тест виконується за допомогою елемента FTP request.

У послугах протоколу TCP клієнти надсилають текстові або двійкові повідомлення до служби та отримують відповіді від неї. Елементи TCP Sampler та TCP Sampler Config Apache JMeter перевіряють навантаження цих типів послуг. У JMeter є три реалізації TCP: одна для обміну текстовими даними та дві для обміну двійковими даними із перевіреною службою. Кожна з реалізацій забезпечується певним класом, який є «шаблоном» або групою

атрибутів об’єкта. В Jmeter цей тест виконується за допомогою елемента TCP sampler.

Java Database Connectivity дозволяє тестувати базу даних. Під час тестування API, веб-сервісу чи інших системних частин, можливо, доведеться записати або отримати дані з бази даних. Мета цієї взаємодії – перевірити правильність запису конкретних даних у БД або підготувати тестові дані до тестів, додавши конкретні записи до бази даних. В Jmeter цей тест виконується за допомогою елемента JDBC Request.

4.11 Аналіз результатів тестування

В результаті проведення тестів продуктивності ми визначили можливості нашого веб-додатка. Це дозволить нам грамотно скористатися наявними можливостями сайту, що забезпечить стабільну і швидку роботу сайту, так як ми знаємо з якою кількістю користувачів наш сайт не втрачає продуктивність, у вигляді часу відповіді сервера.

В результаті виконання об'ємного тесту відбувалося додавання нових даних на сервер з навантаженням в 50 одночасних користувачів. Після виконання тесту всі дані були додані в базу даних і час відповіді не перебільшувала 2 секунди.

В тестуванні на визначення точки насичення був визначений ліміт в 500 одночасних користувачів. Після проходження цього числа починають з'являтися помилкові запити.

В тестуванні на витривалість було визначено що на тривалому часовому відрізку та при кількості в 300 користувачів, навантаження на процесор серверу був від 90 до 100%. Це говорить про те що вузьким місцем в нашому веб-додатку є процесор. Заміна процесора на кращий, приведе до поліпшення масштабування і продуктивності сайту.

ВИСНОВКИ

В атестаційній роботі були розроблені сценарії автоматизації тестування продуктивності веб-додатків.

Тестування та оцінка продуктивності виконувалася в програмі Jmeter. В якості веб-додатку використовувався сайт addressbook, який був розміщений за допомогою програми XAMPP.

В атестаційній роботі були розроблені автоматизовані сценарії для методів виміру продуктивності веб-додатку. До таких методів входять: функціональне тестування, визначення точки насичення, тестування динамічного навантаження, тест на витривалість.

Результатом виконання тестування було знаходження таких метрик як: середній час відповіді, максимальний час відповіді, частота помилок, та стан серверу впродовж тестування.

Розроблені сценарії тестування продуктивності можна використовувати як в навчальних цілях так і для реальних завдань.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Performance Testing Guidance for Web Applications /– M. Microsoft Press;, 2007. – 288 с.

2. The Art of Application Performance Testing / Ian Molyneaux – M. : O'Reilly Media. – 2009. – №1. – 158 с.

3. Apache JMeter [Еклектронний ресурс] / Jmeter Apache – режим доступу: www/ URL:https://jmeter.apache.org/ − 11.10.2020 р. – назв. з екрану.

4. Octo Perf [Еклектронний ресурс] / OctoPerf – режим доступу: www/ URL:https://octoperf.com/blog/2015/06/17/response-time-e-commerce/ − 17.06.2015 р. – назв. з екрану.

5. Software testing [Еклектронний ресурс] / Wikipedia – режим доступу: www/ URL:https://en.wikipedia.org/wiki/Software_testing/ − 15.09.2020 р. – назв. з екрану.

6. Web server [Еклектронний ресурс] / Wikipedia – режим доступу: www/ URL: https://en.wikipedia.org/wiki/Web_server/ − 03.11.2020 р. – назв. з екрану.

7. The way of the web tester / Jonathan Rasmusson – M. Pragmatic Bookshelf;, 2016. – 258 с.

8. Web Load Testing for Dummies / Scott Barber – M. : John Wiley & Sons, Inc. – 2011. – 44 с.

9. Pro Apache JMeter Web Application Performance Testing / Sai Matam, Jagdeep Jain – M. :Apress – 2017. – №1. – 358 с.

10. Web Performance Testing: Methodologies, Tools and Challenges / Vinayak Hegde, Pallavi M. S. / International Journal of Scientific Engineering and Research Vol. 2 No. 1, available at : https://www.ijser.in/v2i1.php

11. Analysis of Performance Testing on Web Applications / Ms. S. Sharmila, Dr. E. Ramadevi / International Journal of Advanced Research in Computer and Communication Engineering Vol. 3 No. 3, available at : http://ijarcet.org/?p=1770

12. Integrated Approach to Web Performance Testing: A practitioner’s Guide / B.M. Subraya – M. Infosys Technologies Limited;, 2006. – 368 с.

13. Why Web Performance Matters: Is Your Site Driving Customers Away? / Gomez / The Web Performance Devision of Compuware, available at: https://www.axity.com/wp-content/uploads/2020/08/201110_why_web_performance_ matters.pdf

14. Bing and Google Agree: Slow Pages Lose Users [Еклектронний ресурс] / Radar – режим доступу: www/ URL: http://radar.oreilly.com/2009/06/bing-and-google-agree-slow-pag.html − 15.11.2020 р. – назв. з екрану.

15. Response time distribution and outlier analysis [Еклектронний ресурс] / dynatrace – режим доступу: www/URL:https://www.dynatrace.com/support/help/how-to-use-dynatrace /transactions-andservices/analysis /response-time-distribution-and-outlier -analysis/ − 17.11.2020 р. – назв. з екрану.

16. A Study of Factors Affecting Websites Page Loading Speed for Efficient Web Performance / Jatinder Manhas/ International Journal of Computer Sciences and Engineering Vol. 1 No. 3, available at: https://www.researchgate.net /publication/274070742_A_Study_of_Factors_Affecting_Websites_Page_Loading _Speed_for_Efficient_Web_Performance

17. Ponomarenko O., Gorbachov V., Abdulrahman Kataeba Batiaa., Kotkova O. The Software Platform for Evaluation of Effectiveness of Network Systems Analysis Technologies Proceedings of IEEE East-West Design & Test Symposium (EWDTS), Batumi, Georgia, 2019, pp. 513-516.

18. Jackson_network [Еклектронний ресурс] / Wikipedia – режим доступу: www/ URL: https://en.wikipedia.org/wiki/Jackson_network/ − 27.11.2020 р. – назв. з екрану.

19. Louis P. Slothouber. A Model of Web Server Performance: дис. … доктор філософії / Louis P. Slothouber. – M., 1996. – 15 с.