Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

РЕФЕРАТ

з дисципліни «Якість програмного забезпечення та тестування»

на тему: «Навантажувальне тестування»

Виконав:

ст. групи КНТ-147сп Д.С. Куделя

Прийняв:

к.т.н., професор Г.В. Табунщик

2019

Зміст

[Вступ 3](#_Toc344364828)

[1 Огляд навантажувального тестування 4](#_Toc344364829)

[1.1 Навантажувалье тестування програмних додатків 4](#_Toc344364829)

[1.2 Основні принципи навантажувального тестування 5](#_Toc344364829)

[1.3 Необхідний набір інструментів для тестування продуктивності 7](#_Toc344364829)

[1.4 Основні метрики продуктивності 9](#_Toc344364829)

[2 Тестування за допомогою Visual Studio Team Services 12](#_Toc344364829)

[Висновки 17](#_Toc344364829)

[Список літератури 18](#_Toc344364829)

Вступ

Тестування програмного забезпечення − процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином[1].

Тестування продуктивності в програмній інженерії – це тестування, яке проводиться з ціллю визначення, як швидко працює програма або її частина під деяким навантаженням[2].

В тестуванні продуктивності розрізняють наступні напрямки:

* навантажувальне;
* стресове;
* тестування стабільності;
* тестування на відмову та відновлення;
* об'ємне тестування;
* шип тестування.

Навантажувальне тестування − підвид тестування продуктивності, збір показників і визначення продуктивності та часу відгуку програмно-технічної системи або пристрою у відповідь на зовнішній запит з метою встановлення відповідності вимогам, що висуваються до даної системи або пристрою. Воно зазвичай проводиться для того, щоб оцінити поведінку програми із заданим очікуваним навантаженням. Цим навантаженням може бути, наприклад, кількість користувачів, які будуть одночасно працювати з програмою. Такий вид тестування дозволяє отримати час відгуку всіх найважливіших бізнес-транзакцій.

Для того, щоб дослідити час відгуку зовнішніх систем на високих або пікових навантаженнях, проводиться стрес-тестування, при якому створюване навантаження на систему перевищує звичайні сценарії її використання. Не існує чітких кордонів між навантажувальним та стресс-тестуванням, але не варто змішуваті ці поняття, оскільки ці види тестування відповідають на різні бізнес-питання і використовують різну методологію.

1 Огляд навантажувального тестування

* 1. Навантажувалье тестування програмних додатків

Термін «навантажувальне тестування» може бути використаний в різних значеннях в професійному середовищі тестування ПО. Взагалі, він означає практику моделювання очікуваного використання програми шляхом емуляції роботи декількох користувачів одночасно. Таким чином, таке тестування найбільш підходить для багатокористувацьких систем, найчастіше з використанням архітектури клієнт-сервер (наприклад, веб-серверів). Однак інші типи програмних систем також можуть бути протестовані аналогічним чином. Наприклад, текстовий або графічний редактор можна змусити прочитати дуже великий документ; а фінансовий пакет − створити звіт на основі даних за кілька років. Найбільш адекватно розроблений тест на навантаження дає більш точні результати.

Основна мета навантажувального тестування − створити специфічне для системи навантаження (наприклад, віртуальні користувачі) і, як правило, за допомогою однакового програмного та апаратного забезпечення спостреігати за показниками продуктивності системи[3].

Наприклад, веб-сервіс з функціональністю кошика призначений для 100 одночасних користувачів, які дотримуються певного конкретного сценарію (визначені дії у визначених пропорціях):

* 30 користувачів переглядають продукт та виходять із системи;
* 30 користувачів додають товар у кошик, оформлюють замовлення та виходять із системи;
* 30 користувачів використовують функцію повернення товару та виходять із системи;
* 30 користувачів увійшли в систему та не мають жодної активності.

У цьому випадку тестування навантаження має імітувати описаний вище типовий сценарій веб-служб, щоб гарантувати, що система готова до введення в експлуатацію. У цьому випадку продуктивність системи в цілому або кожного вузла системи, зокрема, може бути знята для аналізу.

В ідеалі критеріями ефективності тестування продуктивності є вимоги до продуктивності системи, які формулюються та документуються на етапі розробки функціональних вимог до системи перед програмуванням основних архітектурних рішень. Однак часто трапляється так, що такі вимоги не були чітко сформульовані або взагалі не були сформульовані. У цьому випадку першим випробуванням навантаження буде дослідницьке навантаження та базуватиметься на обґрунтованих припущеннях щодо очікуваного навантаження та споживання апаратних ресурсів.

Один з найкращих підходів до використання тестування навантаження для вимірювання продуктивності системи − тестування на ранньому етапі. Навантажувальне тестування на перших етапах готовності архітектурного рішення до визначення його стійкості називається тестуванням «підтвердження концепції».

* 1. Основні принципи навантажувального тестування

Нижче наведено деякі експериментальні факти, загальні принципи, які використовуються при тестуванні продуктивності в цілому та застосовні до будь-якого типу тестування працездатності (зокрема, тестування навантаження).

1. Унікальність запитів − навіть сформувавши реалістичний сценарій роботи з системою на основі статистики її використання, необхідно розуміти, що з цього сценарію завжди будуть винятки. У випадку прикладу, наведеного вище, це може бути користувач, який отримує доступ до унікальних сторінок веб-служб, відмінних від усіх інших.
2. Час реакції системи − у загальному випадку час реакції системи підпорядковується нормальній функції розподілу. Зокрема, це означає, що, провівши достатню кількість вимірювань, можна визначити ймовірність того, що відповідь системи на запит впаде через той чи інший часовий інтервал.
3. Залежність часу відгуку системи від ступеня розповсюдження цієї системи − різниця нормального розподілу часу відповіді системи на запит пропорційна відношенню кількості системних вузлів при паралельній обробці таких запитів та кількості запитів на вузол. Тобто на діапазон значень часу відповіді системи впливає одночасно кількість запитів на системний вузол та кількість вузлів, кожен з яких додає певну випадкову затримку в обробці запитів.
4. Діапазон часу реакції системи − з тверджень 1, 2 і 3 ми також можемо зробити висновок, що при достатньо великій кількості вимірювань часу обробки запитів у будь-якій системі завжди будуть запити, час обробки яких перевищує максимум, зазначений у вимогах; крім того, чим довше загальний час експерименту, тим вище будуть нові максимуми. Цей факт необхідно враховувати при формуванні вимог до продуктивності системи, а також під час регулярних тестувань навантаження.
5. Точність завантаження профілю − необхідна точність відтворення профілів навантаження тим дорожче, чим більше компонентів містить система. Часто неможливо врахувати всі аспекти профілю навантаження для складних систем, оскільки чим складніша система, тим більше часу буде витрачено на проектування, програмування та підтримання для неї адекватного профілю навантаження, що не завжди є необхідністю. Оптимальним підходом у цьому випадку є балансування між витратами на розробку тесту та охоплення функціональності системи, в результаті чого є припущення про вплив на загальну продуктивність тієї чи іншої частини тестованої системи.
   1. Необхідний набір інструментів для тестування продуктивності

Слід зазначити, що для більшості типів тестування продуктивності використовуються ті самі інструменти, які можуть виконувати типові завдання.

Існує поширена помилкова думка, що інструменти для стрес-тестування системи є тими ж інструментами запису та відтворення, що й інструменти для автоматичного тестування регресії. Інструменти для тестування навантаження працюють на рівні протоколу, тоді як інструменти для автоматизації тестування регресії працюють на рівні об’єктів графічного інтерфейсу користувача.

Наприклад, існує стандартний Інтернет-браузер, який виконує функцію натискання на вказане посилання при натисканні кнопки. У цьому випадку для автоматизації регресійного тестування необхідно написати скрипт, який передає клавішу миші та кнопки на браузер, при цьому для створення скрипту для стрес-тестування необхідно записати передачу гіперпосилання з браузера для кількох користувачів, включаючи унікальне ім’я користувача та пароль.

Існують різні інструменти для виявлення та дослідження проблем у різних вузлах системи. Усі вузли системи можна класифікувати наступним чином:

* додаток;
* база даних;
* мережа;
* обробка на стороні клієнта;
* балансування навантаження.

Слід також зазначити появу мережевих додатків Business-to-business (B2B), що використовують угоду про рівень обслуговування (або SLA, Угоду про рівень обслуговування). Зростаюча популярність B2B-прикладних програм призвела до того, що все більше додатків переходять на сервісно-орієнтовану архітектуру, в якій обмін інформацією обробляється без участі веб-браузерів. Прикладом такої взаємодії є бюро туристичних послуг, яке запитує інформацію про конкретний рейс між Ужгородом та Дніпром, тоді як авіакомпанія повинна надати відповідь протягом 5 секунд. Часто за порушення угоди про домовленість за угодою може призвести до великого штрафу.

Деякі засоби стрес-тестування представлені нижче:

* OpenSTA (від Оpen System Testing Architecture) − безкоштовне програмне забезпечення для навантажувального/стрес-тестування, ліцензоване GNU GPL. Використовує розподілену архітектуру додатків на основі CORBA. Доступна версія для Windows, хоча існують проблеми сумісності з Windows Vista. Підтримка була припинена в 2007 році.
* IBM Rational Performance Tester (від IBM) − засноване на середовищі розробки Eclipse ПЗ, що дозволяє створювати навантаження великих обсягів і вимірювати час відгуку для додатків з клієнт-серверною архітектурою. Потребує ліцензування.
* JMeter (від Apache Jakarta Project) − заснований на Java багатоплатформовий інструментарій, що дозволяє виробляти навантажувальні тести з використанням JDBC / FTP / LDAP / SOAP / JMS / POP3 / HTTP / TCP з'єднань. Дає можливість створювати велику кількість запитів з різних комп'ютерів і контролювати процес з одного з них.
* HP LoadRunner (від HP) − інструмент для навантажувального тестування, спочатку розроблений для емуляції роботи великої кількості паралельно працюючих користувачів. Також може бути використаний для unit- або інтеграційного тестування.
* LoadComplete (від SmartBear) − пропріетарний продукт, що дозволяє проводити тестування навантаження веб-додатків.
* Visual Studio Team System (від Microsoft) − інструмент для тестування продуктивності включаючи load / unit testing.
* WebLOAD (від RadView) − інструмент для навантажувального тестування для веб- і мобільних додатків, включаючи веб-панель для аналізу тесту на ефективність Застосовується для великомасштабних навантажень, які можуть генеруватися також із хмари. Ліцензований.
  1. Основні метрики продуктивності

Одним із результатів, отриманих під час тестування навантаження та використаних у майбутньому для аналізу, є показники ефективності застосування. Основні з них обговорюються нижче.

1. Споживання ресурсів процесора,% − показник, який показує, скільки часу з заданого інтервалу витратив процесор на обчислення для вибраного процесу. У сучасних системах важливим фактором є здатність процесу працювати в декількох потоках, щоб процесор міг паралельно виконувати обчислення. Аналіз історії споживання ресурсів процесора може пояснити вплив на загальну продуктивність системи оброблюваних потоків даних, конфігурацій програми та операційної системи, багатопотокових та інших факторів.

2. Витрата оперативної пам’яті, МБ − метричний показник обсягу пам'яті, яку використовує програма. Використовувана пам'ять поділяється на кілька категорій:

1. віртуальна − кількість віртуального адресного простору, який використовує процес. Цей обсяг передбачає як використання відповідного місця на диску, так і оперативної пам’яті. Віртуальна система пам'яті забезпечує те, що потоки в одному процесі не отримують доступу до пам'яті, що належить іншому процесу.
2. приватна − кількість адресного простору, зайнятого процесом і не поділяється з іншими процесами.
3. робочий набір − набір сторінок пам'яті, нещодавно використовуваних процесом. У випадку, коли вільної пам’яті достатньо, сторінки залишаються в наборі, навіть якщо вони не використовуються. У випадку, коли вільної пам’яті мало, використовувані сторінки переміщуються з оперативної пам’яті на жорсткий диск (або інший диск, наприклад флеш-пам’ять), звільняючи оперативну пам’ять для завантаження інших активних сторінок пам’яті.
4. спільна − обсяг фізичної пам'яті, який використовується процесом, який може бути спільним з іншими процесами. Хоча пам'ять, що виділяється на процес, повинна бути ізольованою, але процеси іноді потребують можливості обміну інформацією. Спільна пам'ять - це найшвидший спосіб міжпроцесорного спілкування.

Коли програма працює, пам'ять заповнюється посиланнями на об'єкти, які, якщо вони не використовуються, можуть бути очищені спеціальним автоматичним процесом, який називається сміттєзбірником. Час, витрачений процесором на очищення пам’яті таким чином, може бути значним, якщо процес зайняв усю наявну пам’ять (у Java - так званий «постійний повний GC») або коли на процес було виділено велику кількість пам'яті, потрібно прибирати. На час, необхідний для очищення пам'яті, доступ до сторінок виділеної пам'яті може бути заблокований, що може вплинути на остаточний час обробки даних цим процесом.

3. Споживання мережевих ресурсів. Цей показник не пов'язаний безпосередньо з продуктивністю програми, але його показники можуть вказувати на обмеження загальної продуктивності системи. Наприклад, серверна програма, обробляючи запит користувача, повертає йому відеопотік за допомогою мережевого каналу на 2 мегабіти. У вимозі зазначено, що сервер повинен обробляти 5 запитів користувачів одночасно. Тестування навантаження показало, що сервер може ефективно надавати дані лише 4 користувачам одночасно, оскільки мультимедійний потік має бітрейт в 500 кілобітів. Очевидно, що забезпечити цей потік 5 користувачам одночасно неможливо через перевищення пропускної здатності мережевого каналу, а це означає, що система не відповідає заданим вимогам до продуктивності, хоча в той же час споживання процесорних ресурсів і ресурсів пам'яті. може бути низьким.

4. Робота з підсистемою диска (час очікування вводу / виводу) − робота з дисковою підсистемою може істотно вплинути на продуктивність системи, тому збір статистики щодо роботи з диском може допомогти виявити вузькі місця в цій області. Велика кількість читання або запису може змусити процесор працювати в режимі очікування, очікуючи на обробку даних з диска, і, як результат, збільшити споживання процесора і збільшити час відгуку.

5. Запит часу на виконання, мс. час виконання програми залишається одним з найважливіших показників продуктивності системи чи програми. Цей час може бути виміряний на стороні сервера, як показник часу, який потрібен стороні сервера для обробки запиту; і про клієнта як показник загального часу, необхідного для серіалізації / деріаріалізації, пересилання та обробки запиту. Слід зазначити, що не кожна програма тестування продуктивності може вимірювати обидва рази.

2 Тестування за допомогою Visual Studio Team Services

Після створення безкоштовного облікового запису в службі Visual Studio Team Services потрібно створити новий проект із зазначенням імені та системи контролю версій, яку ви будете. Після цього нам потрібно прив’язати створений обліковий запис до Visual Studio. Для цього нам потрібно в Team Explorer у Visual Studio ввести параметри з'єднання для нашого проекту, як зображено на рисунку 2.1[4].

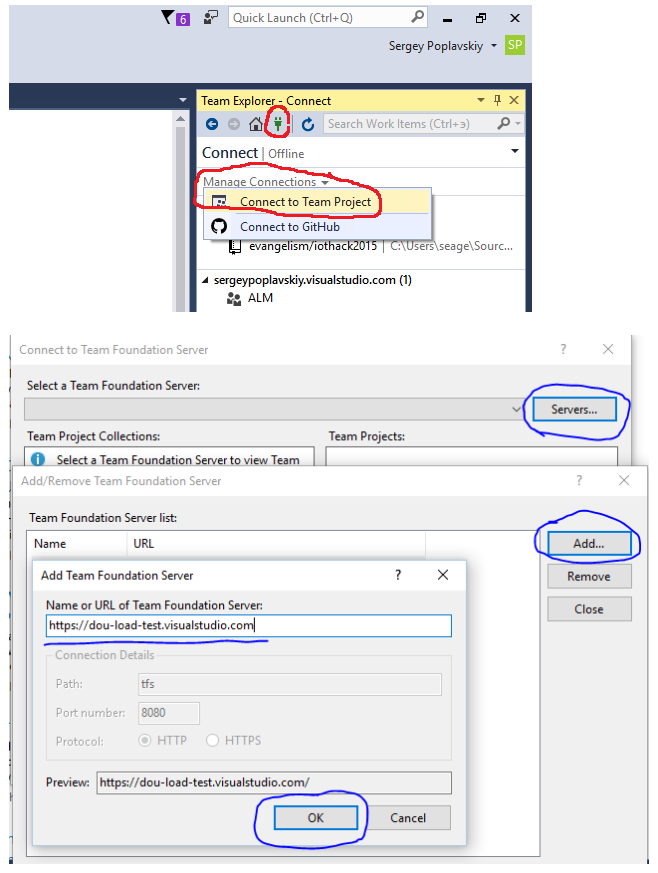


Рисунок 2.1 – З’єднання свтореного облікового запису з Visual Studio

Після з'єднання у вікні «Connect to Team Foundation Server» відобразиться список проектів. У нас він один, його треба обрати.

Далі потрібно створити новий проект в Visual Studio. У розділі «Test» вибираємо «Web performance and Load Test Project», задаємо нове ім'я проекту. Після натискання на кнопку «ОК» у нас з'явиться новий проект веб тесту. Власне, зараз потрібно записати цей веб тест. Для цього в нашому веб тесті потрібно натиснути кнопку «Add recording», після чого відкриється веб браузер. Все, що буде в ньому виконуватись – буде записано. Проведемо тестування на прикладі сайту «dou.ua».

Далі повертаємося в Visual Studio і тиснемо «Stop recording». Якщо після натискання «Stop recording» нічого не змінилося (в веб тесті не з'явився записаний url з набором атрибутів), потрібно включити аддон в веб браузері «Microsoft Web Test Recorder Helper». У підсумку тест повинен виглядати приблизно як на рисунку 2.2.

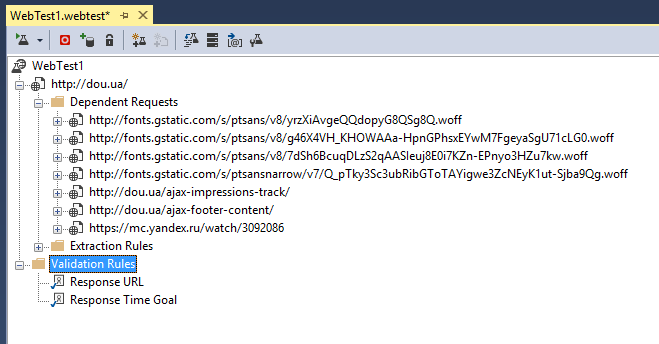


Рисунок 2.2 – Приклад записаного тесту

Веб-тест записано, тепер його потрібно запустити з під потрібної кількості віртуальних користувачів. Для цього в наш проект Visual Studio додаємо Load Test. Потрібно натиснути на проекті нашого веб тесту в   
Solution Explorer -> Add -> Load Test. Відкриється wizard, який потрібно пройти. На першому кроці потрібно обрати «Cloud-based Load Test with Visual Studio Team Services», далі оберемо локацію дата центру, звідки буде генеруватися навантаження, встановлюємо тривалість тесту. Після доходимо до «Load Pattern», обираємо «Step Load». Сенс в тому, що є стабільне навантаження. Це означає, що навантаження буде генерувати від заданого кількості користувачів протягом тесту.

Ми ж вибираємо більш цікавий сценарій, в межах якого кількість одночасних користувачів буде поступово зростати за заданим шаблоном до 1000 користувачів, як зображено на рисунку 2.3.

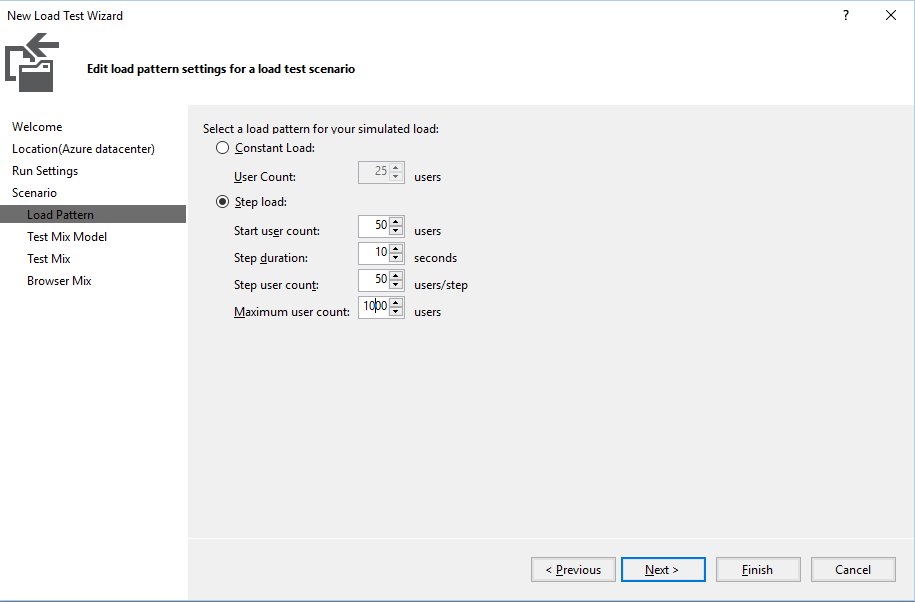


Рисунок 2.3 – Шаблон зростання кількості користувачів

Далі в Test Mix додаємо наш веб тест і після цього можемо конфігурувати мікс браузерів, після чого тиснемо Finish. Запускаємо навантажувальний тест.

Пройде якийсь час, і ми почнемо бачити графіки залежності часу відповіді від кількості одночасних користувачів, як зображено на рисунку 2.4.

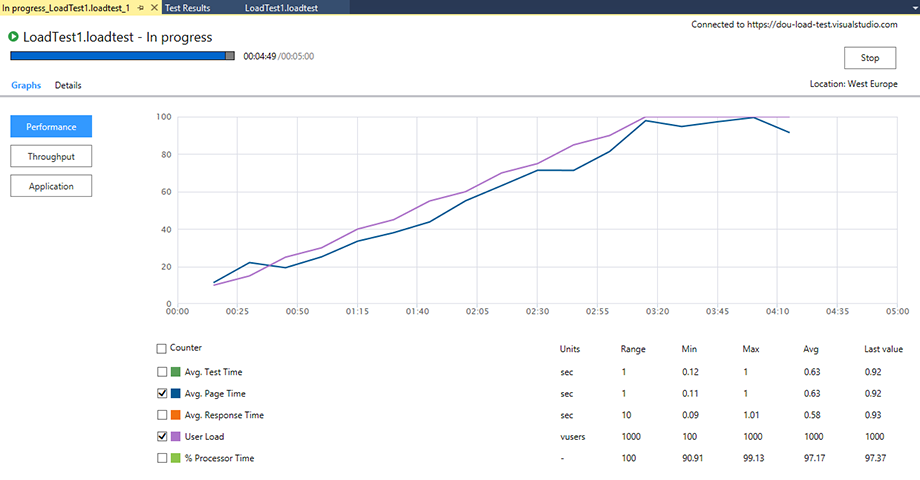


Рисунок 2.4 – Графік залежності часу відповіді від кількості користувачів

По закінченню тесту можна завантажити в Visual Studio детальний звіт і подивитися, що відбувається. В даному конкретному випадку можна побачити, що майже відразу почали з’являтися помилки. Статистика по всім помилкам тесту зображена на рисунку 2.5. Ми можемо проаналізувати кожен запис окремо, як зображено на рисунку 2.6.

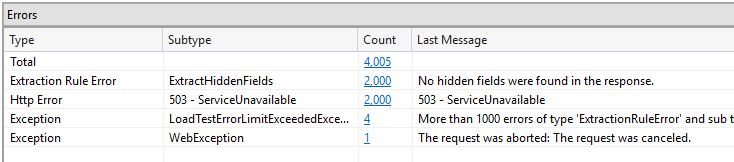


Рисунок 2.5 – Статистика по всім помилкам

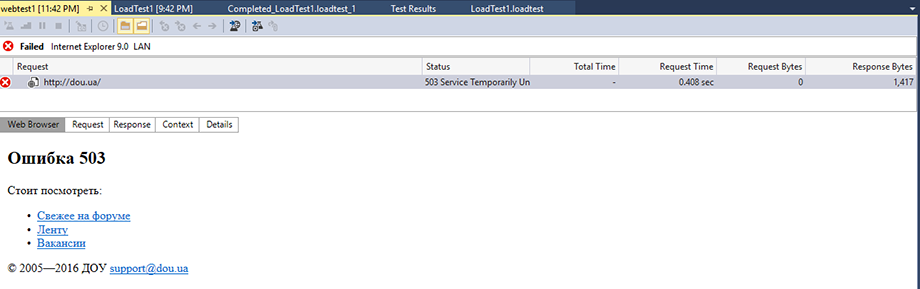


Рисунок 2.6 – Перегляд запису помилки

Детальні звіти можна вивантажити в Excel і проаналізувати там. А якщо є попередні вимірювання, то Visual Studio вміє аналізувати, наскільки змінилася продуктивність з минулого разу.

Висновки

В рефераті було розглянуто поняття тестування, тестування продуктивності та поняття навантажувального тестування.

Було описано основні питання в області навантажувального тестування програмних додатків, основні принципи навантажувального тестування, необхідний інструментарій для навантажувального тестування, і основні метрики навантажувального тестування.

Також було надано приклад проведення навантажувального тестування веб-сторінки за допомогою сервісу Visual Studio Team Services.

СПИСОК ЛІТЕРАТУРИ

1. Wikipedia: Тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Тестування>\_програмного  
   \_забезпечення;
2. Wikipedia: Тестування продуктивності [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ Тестування\_продуктивності;
3. Wikipedia: Нагрузочное тестирование [Електронний ресурс]. – Режим доступу: https:// ru.wikipedia.org/wiki/Нагрузочное\_тестирование;
4. Нагрузочное тестирование, используя облачный сервис [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/load-testing/.