Тест-план для JMeter: QA Pro REST App

Інсталяція:

https://hub.docker.com/r/oleksandrgolubishko/qa_pro_rest_app
docker run -p 3001:3001 oleksandrgolubishko/qa_pro_rest_app
http://localhost:3001/characters

Ціль тестування

Ціллю даного тестування є перевірка продуктивності та стабільності роботи QA Pro REST App додатку при різних сценаріях навантаження.

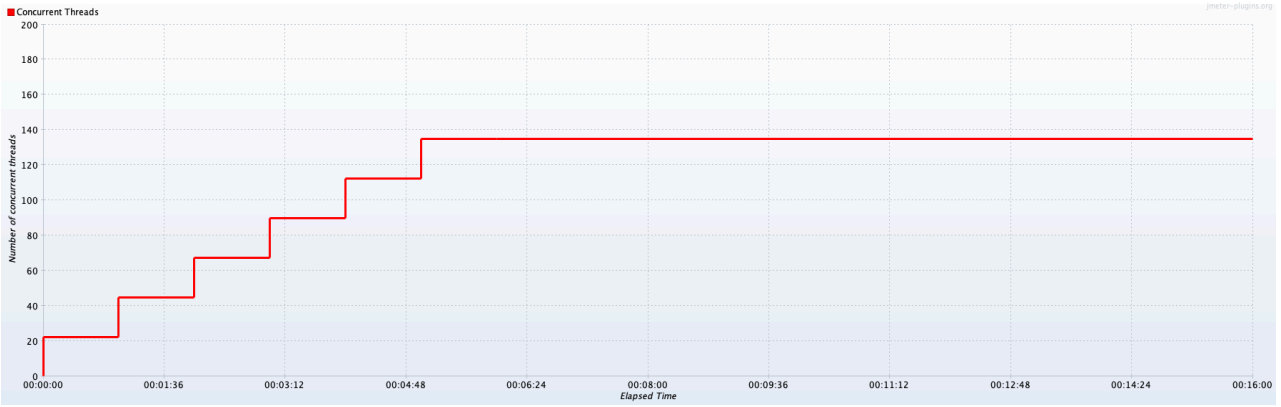
Стратегія тестування:

Група	Поведінка	Методи
Група 1	Читачі	GET, GET{id}
Група 2	Читачі + створювачі	GET, POST
Група 3	Повний CRUD	GET, POST, PUT, DELETE

Профілі навантаження:

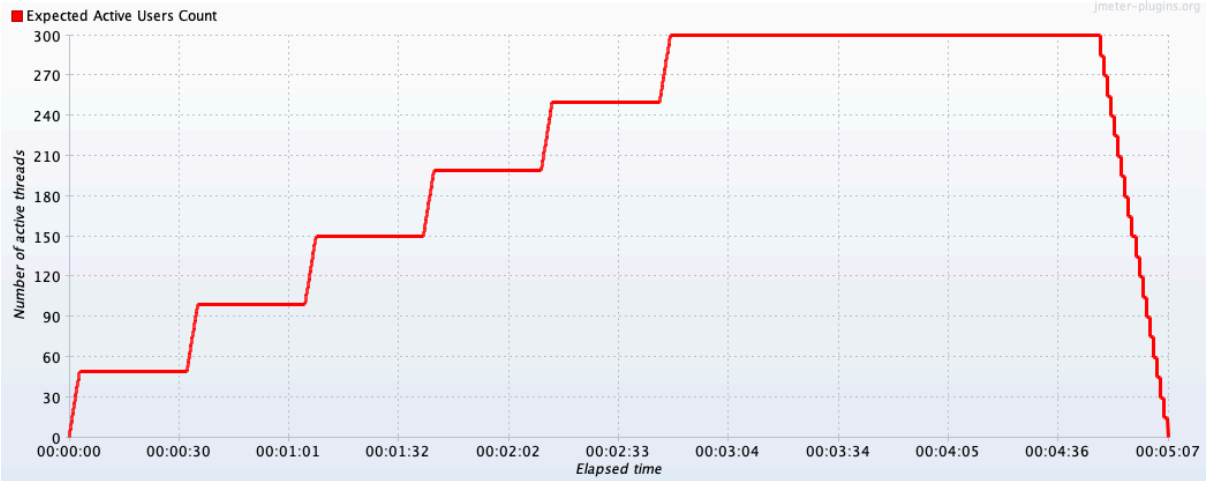
1. Load test

Параметр	**Значення**
Thread Group type	bzm - Concurrency Thread Group
Target Concurrency	135 користувачів
Ramp Up Time (min)	6 хв
Ramp-Up Steps Count	6 хв
Hold Ttarget Rate Time (min)	10 хв
Strategy	45/45/45 користувачів для груп 1/2/3



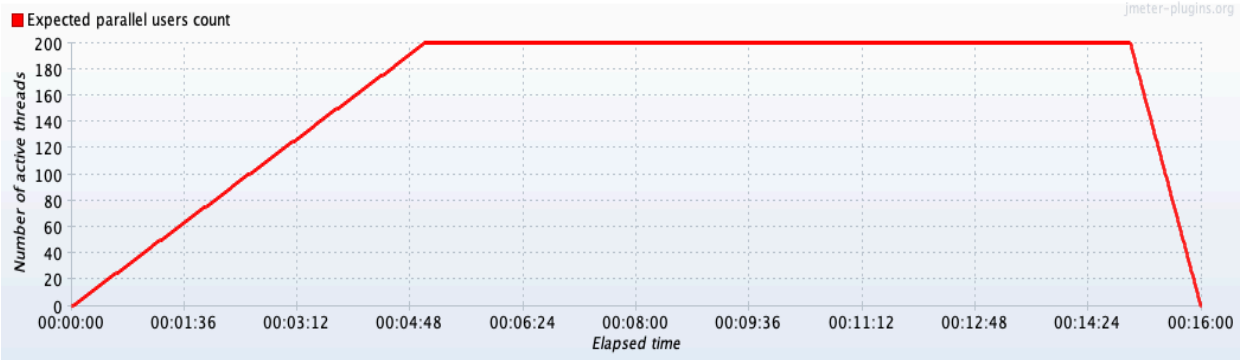
2. Stress test

Параметр	Значення
Thread Group type	jp@gc - Stepping Thread Group
This group will start	300 користувачів
First, wait for	0 seconds
Then start	50 threads
Next, add	50 threads every 30 seconds
using ramp-up	3 seconds
Hold load for	120 seconds
Finally, stop	15 threads every 1 second
Strategy	100 / 100 / 100 користувачів для Груп 1 / 2 / 3



3. Peak test

Параметр	Значення
Thread Group type	jp@gc - Ultimate Thread Group
Start Threads Count	200 користувачів
Initial Delay	0 секунд
Startup Time	300 секунд (5 хвилин)
Hold Load For	600 секунд (10 хвилин)
Shutdown Time	60 секунд
Стратегія груп	70 / 70 / 60 користувачів для Груп 1 / 2 / 3



Результати тестування:

Load test

1. Active Threads Over Time

Перевірка стабільності додатку при навантаженні до 135 одночасних користувачів, показало:

- Навантаження зростає поетапно - кожні ~60–70 секунд.

- Після 6 хвилин система досягає піку - 135 потоків - і успішно утримує це навантаження 10 хвилин.

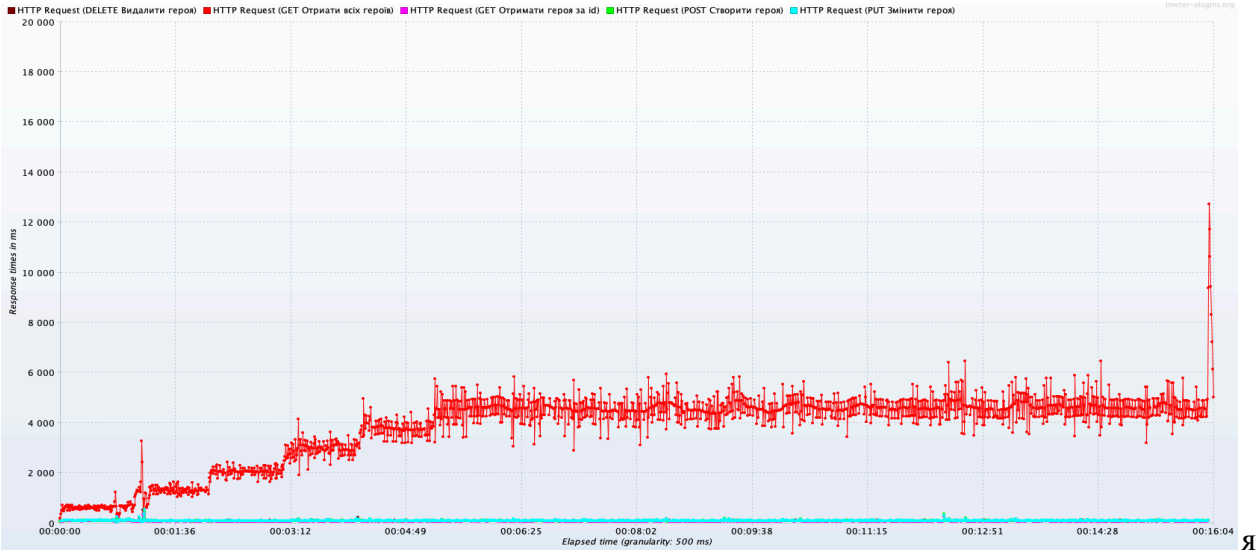
- Плавне завершення після плато - без обривів чи зависань.

2. Response Times Over Time

Оцінка, того як змінювався загальний час відповіді (response time) на кожен тип запиту під час Load Test на 135 concurrent users, показало:

- Запити GET by ID, POST, PUT, DELETE протягом усього тесту стабильні (100-200 мс), не показують затримок.

- Запит GET Отримати всіх героїв з часом чим більше ставало користувачів - тим повільніше сервер обробляв цей запит. Ближче до завершення тесту: Час відповіді зростав до 4000–5000 мс; Пікові стрибки досягали 13 000+ мс**.** Це ознака деградації продуктивності на цьому endpoint'і, не критичний збій, але сигнал до оптимізації.



Я

3. HTML Dashboard (Statistics)

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**89740**	**0**	**0.00%**	**1221.10**	**17**	**14599**	**104.00**	**12909.00**	**13621.00**	**14167.93**	**93.03**	**132731.02**	**16.40**
HTTP Request (DELETE Видалити героя)	8957	0	0.00%	103.94	26	576	102.00	118.00	131.00	154.42	9.34	2.67	1.70
HTTP Request (GET Отриати всіх героїв)	27016	0	0.00%	3830.90	56	14599	114.00	13681.00	13895.00	14336.00	28.01	132713.52	4.32
HTTP Request (GET Отримати героя за id)	26888	0	0.00%	86.82	17	790	75.00	110.00	136.00	187.00	28.00	7.33	4.35
HTTP Request (POST Створити героя)	17921	0	0.00%	104.69	27	718	103.00	118.00	129.00	156.00	18.67	5.09	4.03
HTTP Request (PUT Змінити героя)	8958	0	0.00%	105.43	26	756	103.00	120.00	132.00	156.00	9.34	2.50	2.07

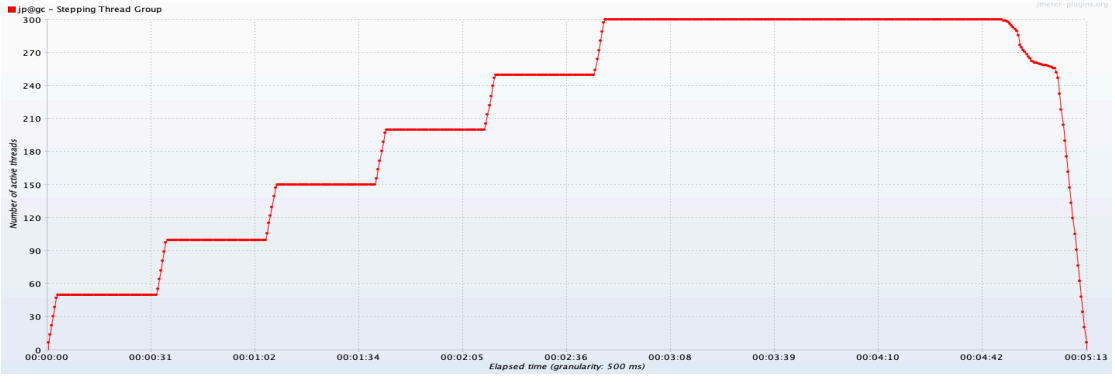
З таблиці видно, що помилок немає, система в цілому стабільна при 135 одночасних користувачах, але 95-й та 99-й перцентиль - критично високі (13–14 сек) вузьке місце - GET /characters (без пагінації/оптимізації). Інші запити в межах SLA (≤ 200 мс) - тобто не страждають від загального навантаження.

Stress test

1. Active Threads Over Time

Перевірка, як система поводиться при поступовому збільшенні навантаження до 300 одночасних користувачів, що значно перевищує очікуване робоче навантаження, показало:

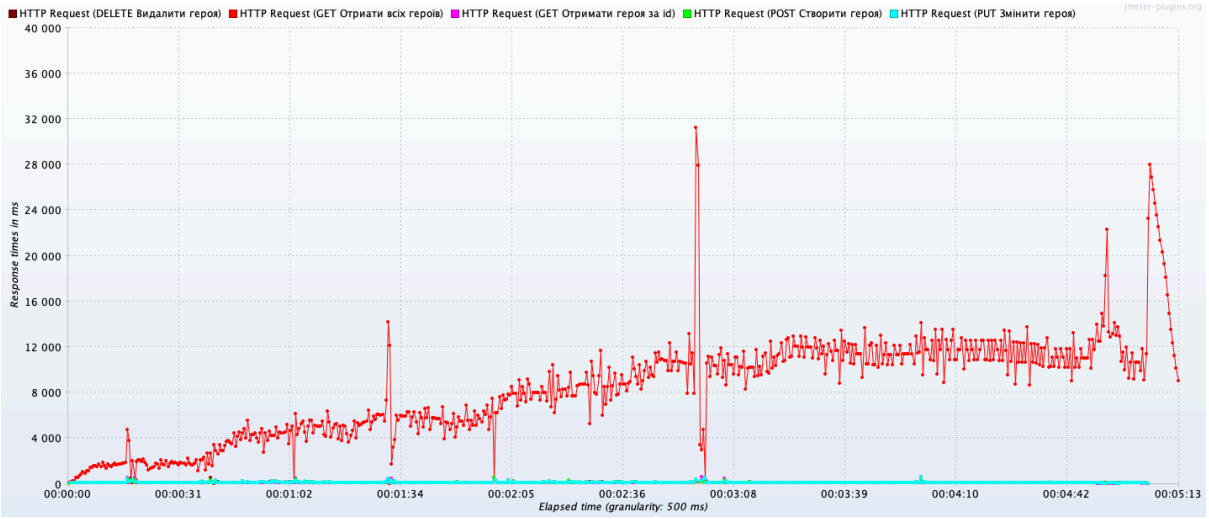
- Навантаження зростало поетапно: кожні 30 секунд - +50 користувачів.
- Пік у 300 потоків був досягнутий і утримувався протягом 2 хвилин.
- Потім відбулося контрольоване завершення - без обривів чи зависань.



2. Response Times Over Time

Оцінка, того як змінювався загальний час відповіді (response time) на кожен тип запиту під час Load Test на 300 concurrent users, показало:

- Час відповіді для GET all зростав стабільно з кожним кроком навантаження.
- Присутні потужні піки latency — до 28 000–36 000 мс на піку.
- Інші запити зберігали стабільний час ≤ 200 мс.



3. HTML Dashboard (Statistics)

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	24707	0	0.00%	2702.56	9	36067	113.00	15833.00	30963.00	35009.99	78.81	124429.13	13.88
HTTP Request (DELETE Видалити героя)	2440	0	0.00%	119.91	35	663	113.00	146.00	165.00	309.34	8.03	2.30	1.46
HTTP Request (GET Отриати всіх героїв)	7622	0	0.00%	8508.92	46	36067	135.00	32866.80	34504.95	35496.31	24.31	124414.46	3.75
HTTP Request (GET Отримати героя за id)	7324	0	0.00%	101.34	13	893	85.00	139.50	192.00	363.75	24.04	6.29	3.73
HTTP Request (POST Створити героя)	4881	0	0.00%	120.17	10	694	114.00	144.00	163.00	321.36	16.03	4.37	3.46
HTTP Request (PUT Змінити героя)	2440	0	0.00%	121.19	9	669	114.00	145.00	164.00	360.77	8.02	2.15	1.78

З таблиці видно, що всі запити були успішно оброблені - помилок немає, що свідчить про загальну стабільність системи при навантаженні до 300 одночасних користувачів.

Проте 95-й та 99-й перцентилі для запиту GET Отримати всіх героїв є критично високими - до 32–35 секунд, що свідчить про серйозне вузьке місце на цьому endpoint'і.

Це запит без пагінації, який повертає велику кількість даних, тому його слід оптимізувати.

Водночас усі інші запити (GET by ID, POST, PUT, DELETE) залишаються в межах SLA - з середнім часом відповіді близько 100–120 мс, навіть при максимальному навантаженні. Це підтверджує, що загальне навантаження не впливає критично на основну функціональність системи.

Peak test

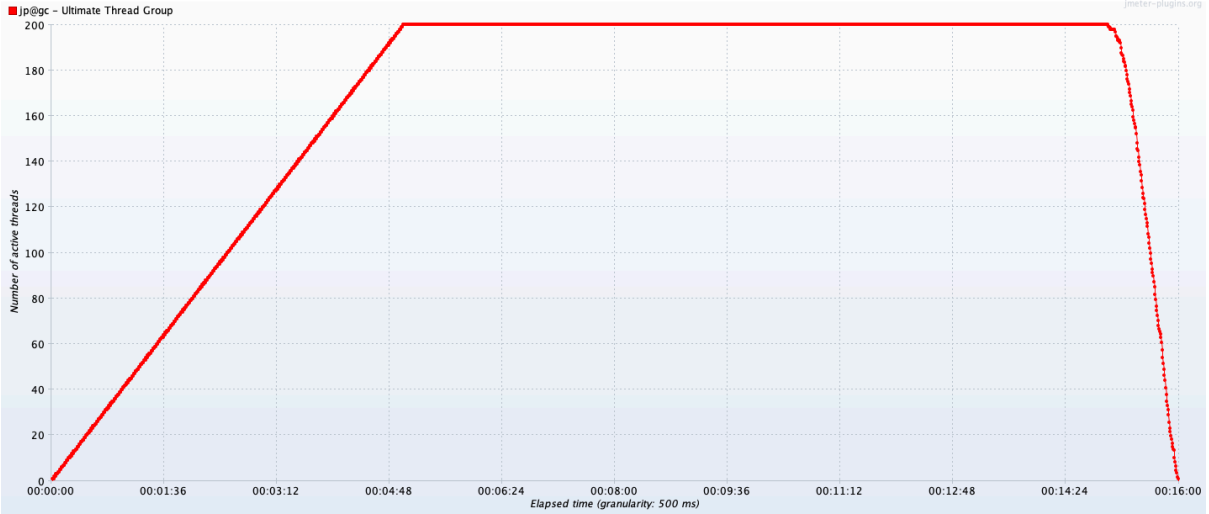
1. Active Threads Over Time

Перевірика, чи зможе система стабільно працювати при максимально допустимому навантаженні — 200 одночасних користувачів — протягом тривалого часу, і виявити, чи

з’являється деградація або нестабільність на піку.

Характеристика:

- Користувачі додавались поступово протягом 5 хвилин.
- Після чого система утримувала 200 потоків протягом 10 хвилин.
- Завершення пройшло плавно, без обривів.

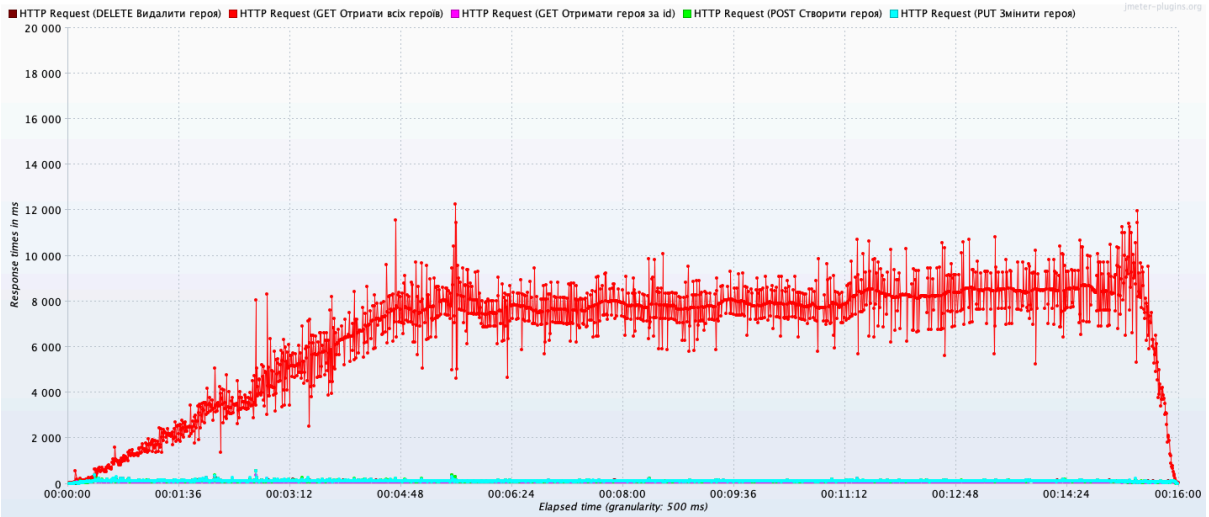


2. Response Times Over Time

Запит GET/characters (червона лінія) показав стабільне зростання latency, починаючи з ~3-4 секунди до ~7000+ мс.

У пікові моменти час відповіді зростав до 25–26 сек, хоча система не “падала”.

Інші запити залишались стабільними - < 200 мс.



3. HTML Dashboard (Statistics)

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**74309**	**0**	**0.00%**	**2111.71**	**0**	**26010**	**124.00**	**10207.60**	**24986.95**	**25450.99**	**77.52**	**124293.07**	**13.67**
HTTP Request (DELETE Видалити героя)	7401	0	0.00%	121.02	2	560	119.00	147.00	161.00	225.96	7.72	2.21	1.40
HTTP Request (GET Отриати всіх героїв)	22437	0	0.00%	6733.95	32	26010	136.00	23980.90	25032.00	25452.99	23.41	124278.51	3.61
HTTP Request (GET Отримати героя за id)	22249	0	0.00%	99.82	0	718	87.00	133.00	177.00	238.00	23.22	6.08	3.61
HTTP Request (POST Створити героя)	14819	0	0.00%	121.71	0	603	120.00	145.00	160.00	228.60	15.47	4.21	3.34
HTTP Request (PUT Змінити героя)	7403	0	0.00%	122.86	2	572	120.00	149.00	163.00	241.00	7.73	2.07	1.71

З таблиці видно, що всі запити були успішно оброблені — помилок не виявлено, що свідчить про загальну стабільність системи при навантаженні до 200 одночасних користувачів.

Разом з тим, 95-й та 99-й перцентилі для запиту GET Отримати всіх героїв залишаються критично високими — до 25 секунд, що знову підтверджує наявність вузького місця на цьому endpoint'і.

Водночас усі інші запити (GET by ID, POST, PUT, DELETE) демонструють стабільну продуктивність, з середнім часом відповіді в межах 100–120 мс, навіть на піку навантаження. Це означає, що загальне навантаження не створює критичного впливу на основну функціональність системи.

Висновок по результатах тестування та пропозиції

В результаті проведених навантажувальних, стресових та пікових тестів було оцінено продуктивність системи в різних умовах навантаження. Тести дозволили виявити як нормальну поведінку системи при типовому навантаженні, так і реакцію на граничні значення, які перевищують очікувані робочі межі.

Аналіз результатів показав:

- Система демонструє стабільну роботу при середньому навантаженні (Load Test), з прийнятним часом відгуку та відсутністю критичних збоїв.

- Під час Stress Test було зафіксовано зниження продуктивності після досягнення системних лімітів (на рівні ОС), що очікувано для робочого середовища без оптимізації під високі навантаження.

- У Peak Test система показала здатність короткочасно витримувати різке збільшення навантаження, однак з незначними затримками в обробці запитів.

Виявлені вузькі місця:

Обмеження на рівні ОС (ulimit) могли впливати на кількість одночасно відкритих з'єднань та оброблюваних потоків, що потенційно створювало штучні ліміти при високому навантаженні.

У деяких сценаріях спостерігалося поступове зростання латентності при одночасному виконанні великої кількості запитів, що свідчить про обмеженість ресурсів.

Тестове середовище (локальна машина) могло впливати на точність результатів, особливо при досягненні навантаження 200–300 користувачів.

Рекомендація:

Для отримання більш точних та реалістичних результатів рекомендується проводити навантажувальне тестування на продуктивніших технічних середовищах з попередньо налаштованими системними лімітами.

Найбільш критичним вузьким місцем у системі виявився запит GET /characters, який повертає повний список об'єктів без будь-якої пагінації або фільтрації. Саме цей endpoint демонстрував найгіршу продуктивність під час всіх типів тестування (Load, Stress, Peak): середній час відповіді перевищував 6–8 секунд, а перцентилі 95% і 99% доходили до 25–30 секунд, що є неприйнятним для реального користувача. Інші запити залишались стабільними, що підтверджує — вузьке місце зосереджене саме на GET /characters.

Цей запит потребує обов'язкової оптимізації, зокрема — впровадження пагінації, кешування або обмеження обсягу переданих даних.

Для подальшого вдосконалення сценаріїв навантаження доцільно створити окрему документацію, яка базується на спостереженнях за реальним трафіком сайту. Таке джерело даних допоможе визначити: середню кількість одночасних користувачів, години пікової активності, типові дії, які виконує більшість відвідувачів.

Це, у свою чергу, дасть змогу будувати сценарії тестування, що максимально відповідають реальному використанню системи, та дозволить планувати навантаження більш обґрунтовано та ефективно.