НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №2

з предмету «Проектування розподілених систем»

Виконав:

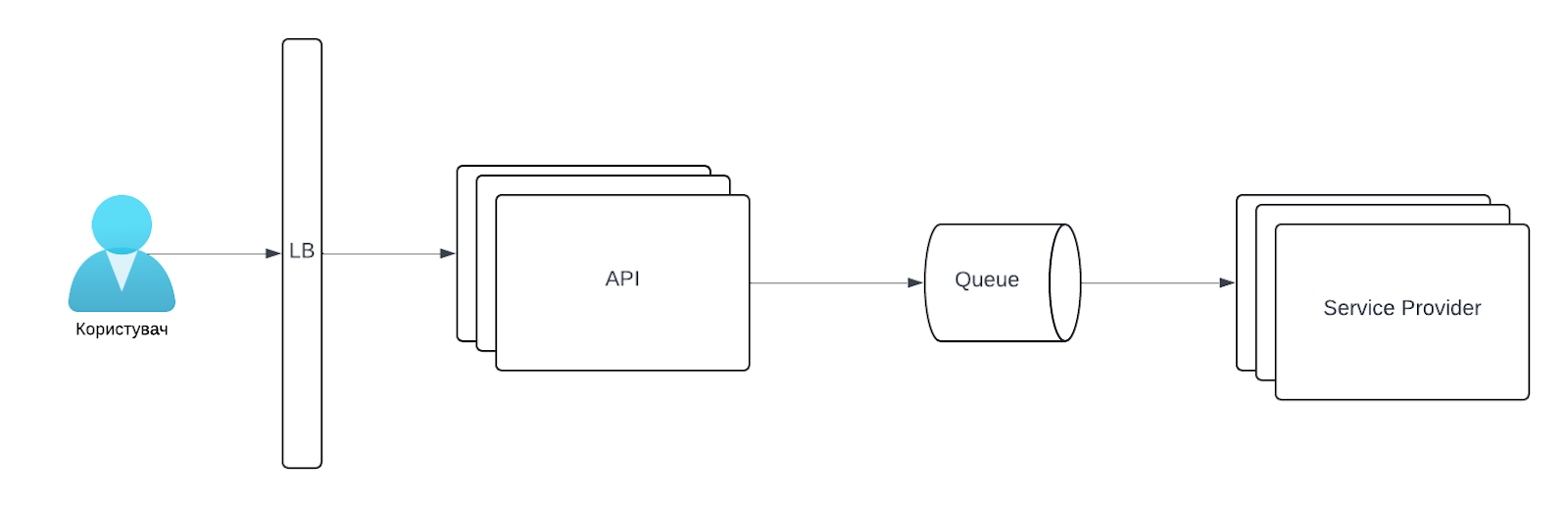
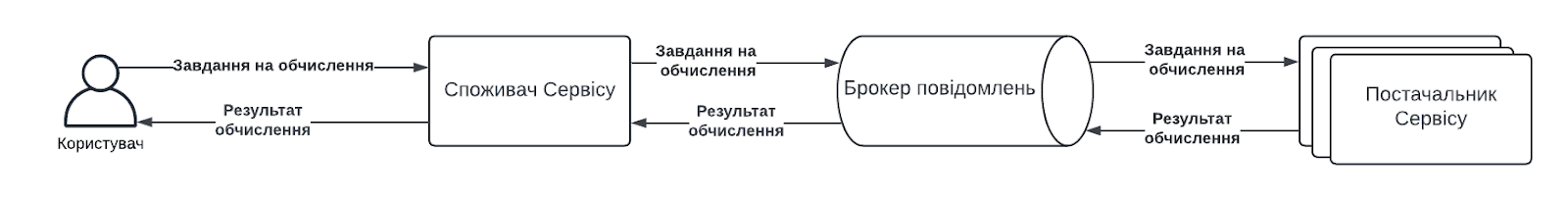
студент групи ІМ-31мн,

Онищук Микола

Київ 2024

**Завдання**

* Реалізувати асинхронну комунікацію між Постачальником Сервісу і Споживачем Сервісу за допомогою Брокера Повідомлень
* Постачальник Сервісу має підраховувати час обчислення і логувати його для подальшого аналізу
* Cпоживач Cервісу має підраховувати час виконання запиту і логувати його для подальшого аналізу
* Реалізувати горизонтальне масштабування засобами Брокера Повідомлень
* Реалізувати чергу с пріоритетами
* Реалізувати request-reply паттерн в асинхронній комунікації
* Порівняти результати синхронної і асинхронної комунікації

**Виконання**

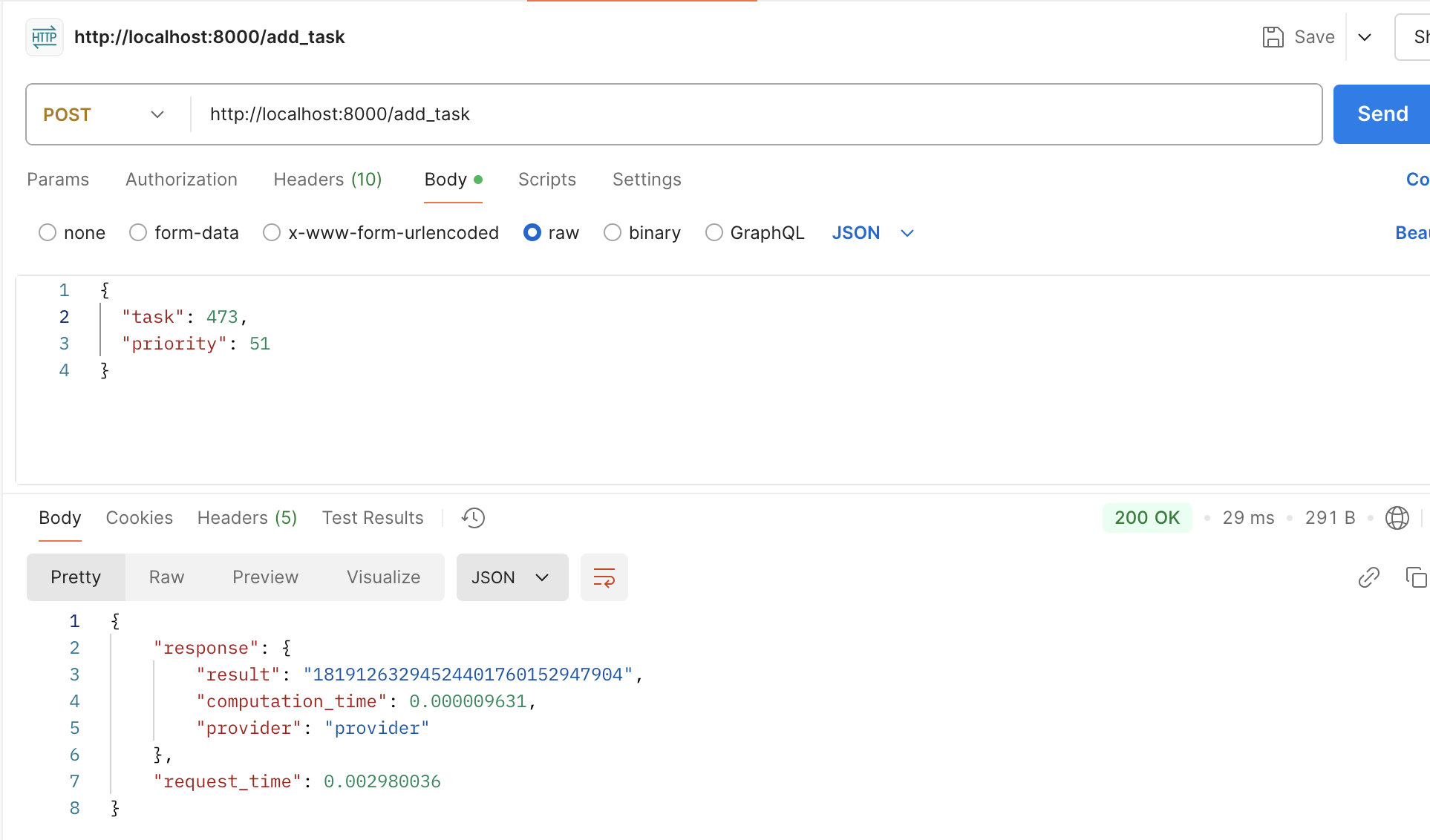
Лабораторну роботу було виконано на мові Golang та розгорнуто у Docker. Також було використано Nginx для балансування навантаження між інстансами споживача та RabbitMQ в якості брокера повідомлень для комунікації між споживачем та постачальником. Також для забезпечення черги з пріоритетами було використано реалізацію, наявну у RabbitMQ.

Розроблена система містить балансувальник навантаження для споживача, 3 інстанси споживача та 3 інстанси постачальника та брокер повідомлень для спілкування між ними.

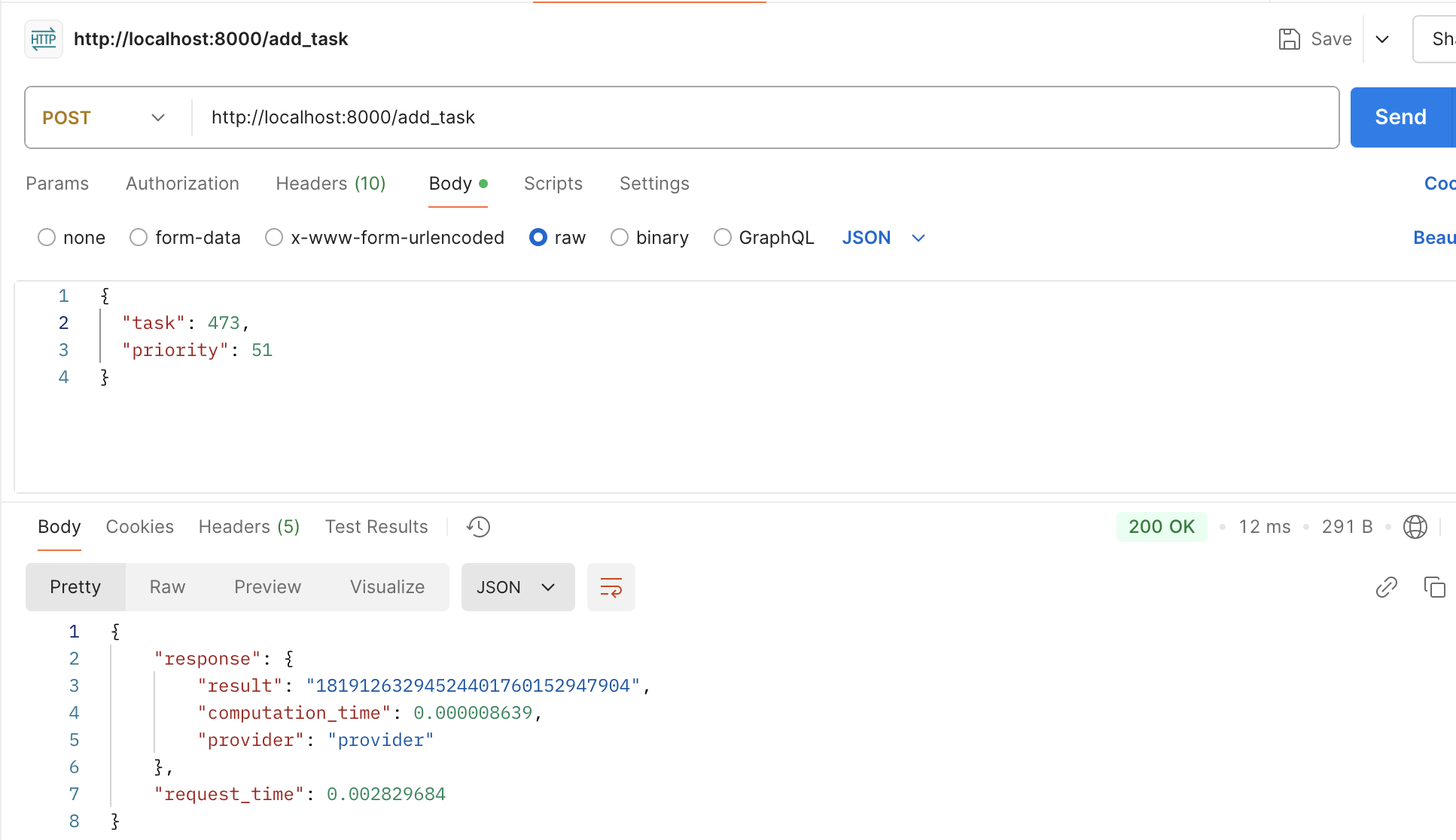
Сервіс споживача має 1 ендпоінт - “/add\_task”, що слугує для прийняття завдання клієнта, після чого відбувається додавання його до черги RabbitMQ, очікування на відповідь від провайдера і повернення результат клієнту у відповіді. Постачальник не має ендпоінтів, адже спілкування між ним та споживачем відбувається через підписку на чергу.

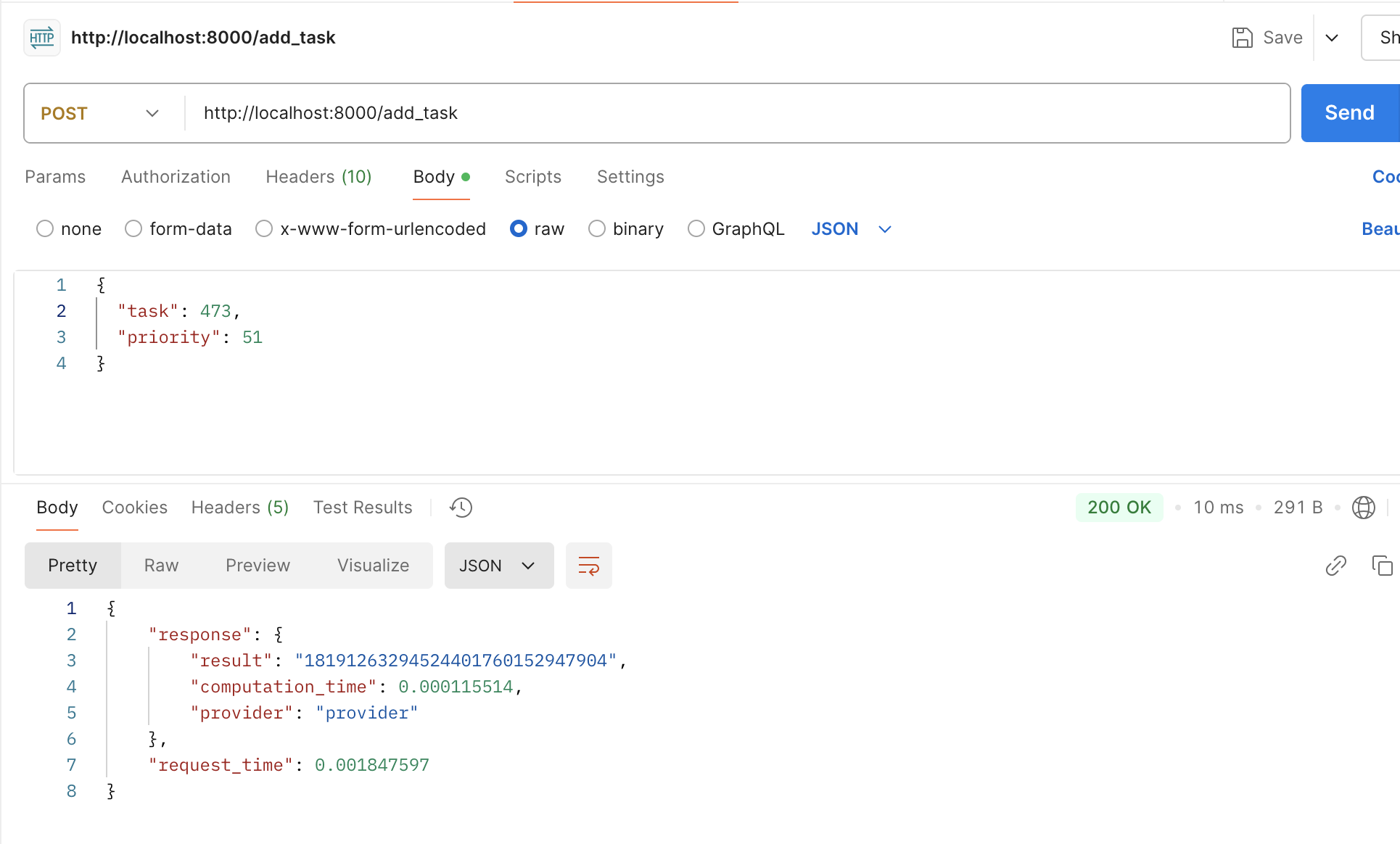
Протестуємо роботу системи, надсилаючи запити за допомогою “Postman”.

Надішлемо POST-запит за адресою <http://localhost:8000/add_task> та отримаємо відповідь майже анологічну відповіді на “/create\_task” у першій лабораторній (рис. 1). Він також повинен містити поля “task” та “priority” у тілі запиту.

Рисунок 1 - Відправка запиту про завдання.

Відправимо аналогічний запит ще два рази (рис. 2-3) для фіксації часу обробки та відповіді.

Рисунок 2 - Відправка другого запиту.

Рисунок 3 - Відправка третього запиту.

Тепер відправимо три GET-запити з аналогічним числом до споживача (рис. 4-6) в системі, виконаної в рамках першої лабораторної, та порівняємо швидкість запитів.

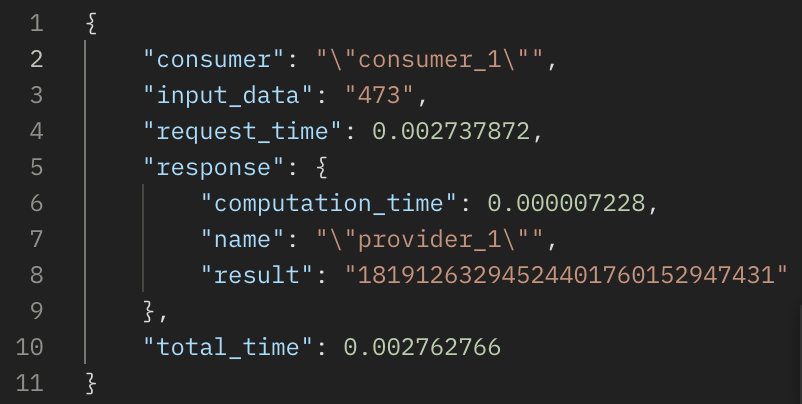
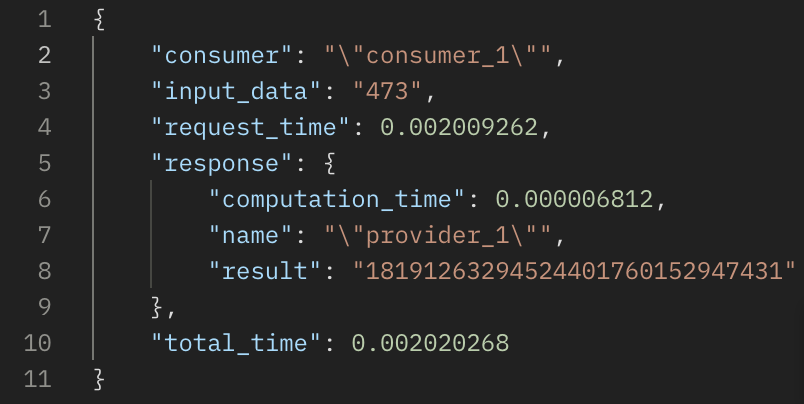
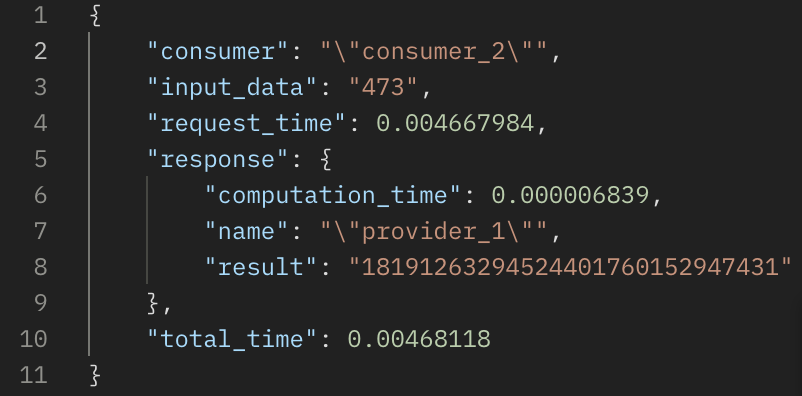


Рисунок 4 - Перший запит до “/create\_task”.

Рисунок 5 - Другий запит до “/create\_task”.

Рисунок 6 - Третій запит до “/create\_task”.

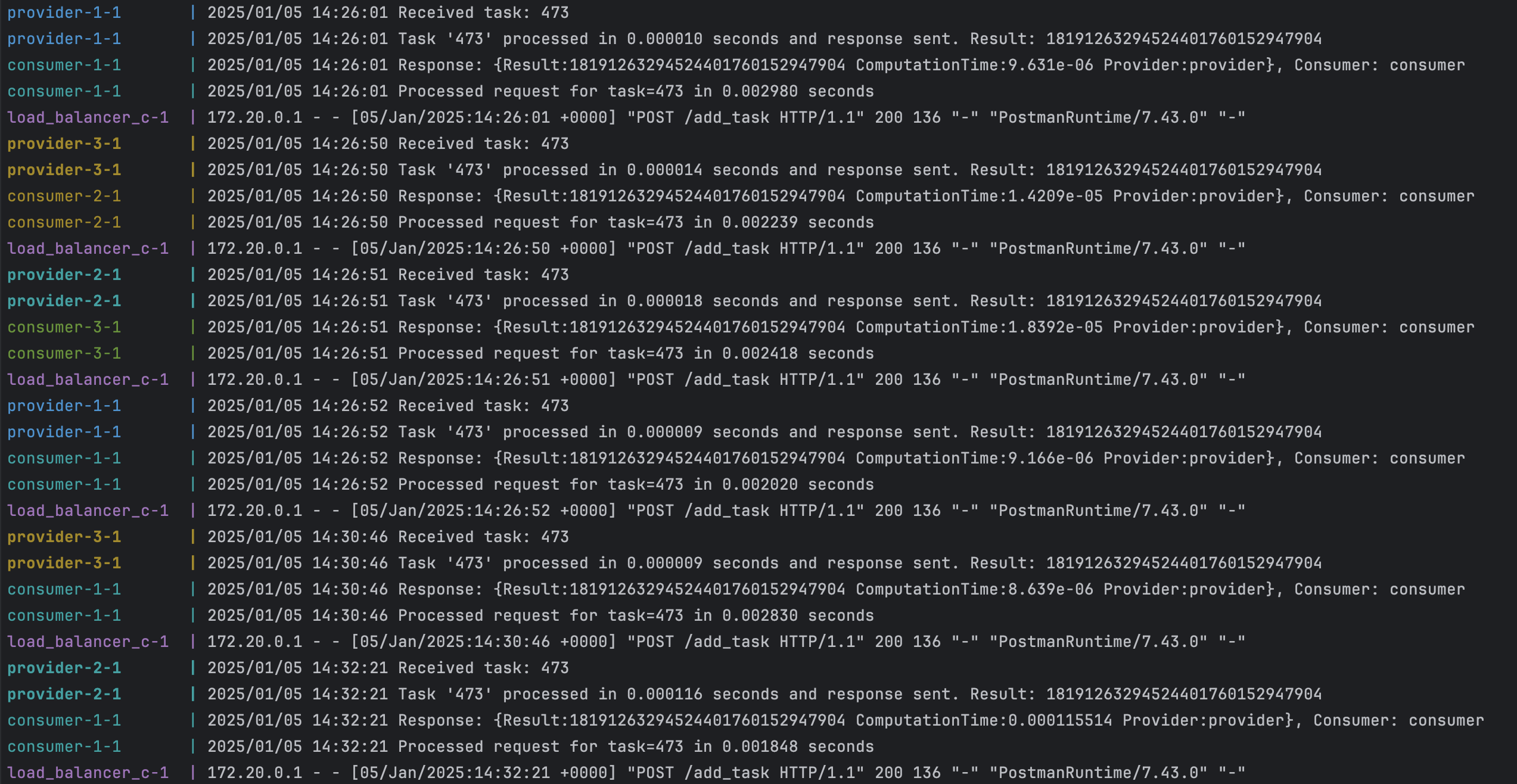
Занесемо отримані дані до таблиці та знайдемо середні арифметичні (табл. 1).

Таблиця 1 – Час виконання запитів під час використання синхронної та асинхронної комунікації між споживачем та постачальником.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Час запиту 1 | Час запиту 2 | Час запиту 3 | Середнє арифметичне |
| Синхронна комунікація | 0.002762766 | 0.002020268 | 0.004667984 | 0,0031503393 |
| Асинхронна комунікація | 0.002980036 | 0.002829684 | 0.001847597 | 0,002552439 |

Отже, середнє арифметичне часу виконання запитів за використання асинхронної комунікації менше, ніж за використання синхронної комунікації, хоч і можна побачити, що при деяких запитах час виконання під час синхронної комунікації був навіть меншим. Ця різниця мала б зрости на користь асинхронної комунікації в разі великого навантаження на систему та великої кількості майже одночасних запитів.

Внаслідок надісланих запитів було отримано наступні логи (рис. 7).

Рисунок 7 - Логи при виконанні запитів.