НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №1

з предмету «Проектування розподілених систем»

Виконав:

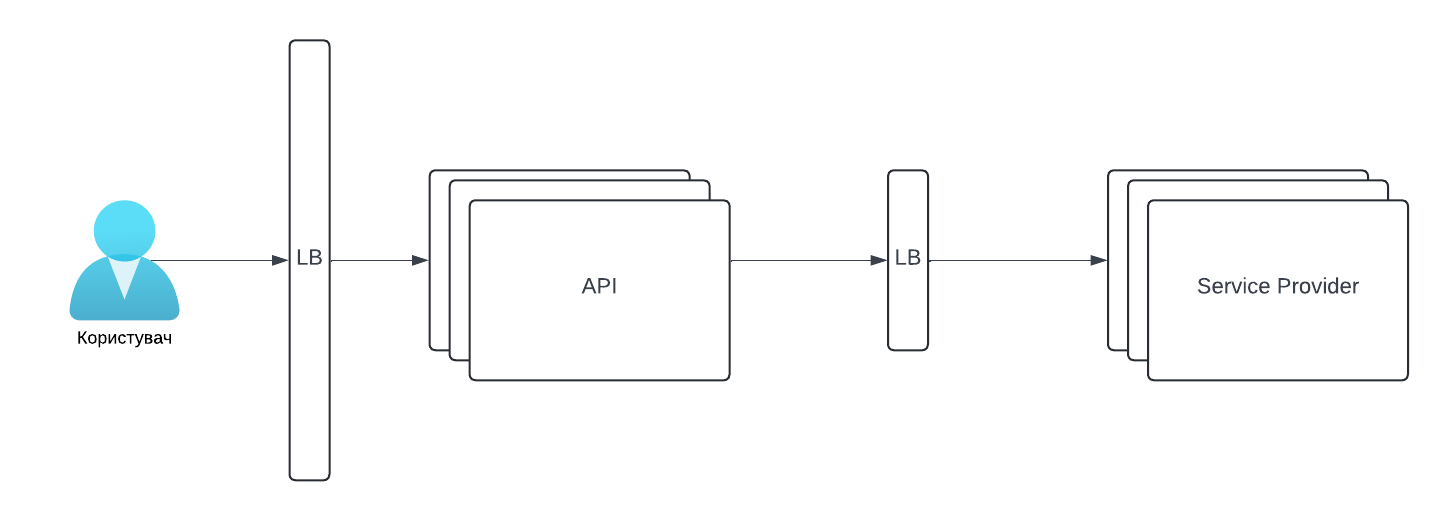
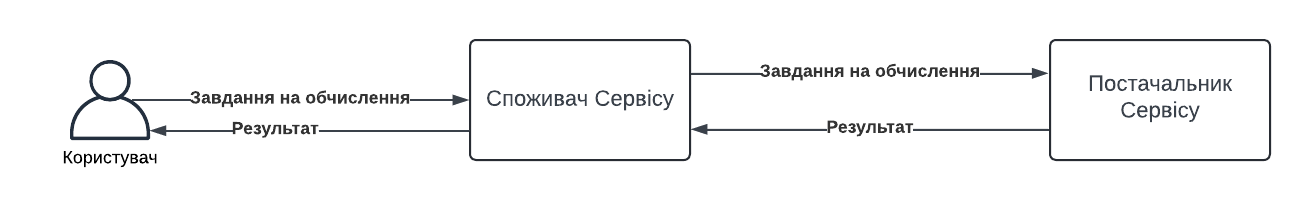
студент групи ІМ-31мн,

Онищук Микола

Київ 2024

**Завдання**

* Реалізувати синхронну комунікація між 2ма сервісами. Cпоживач Cервісу генерує завдання на обчислення і чекає відповіді від Постачальник Сервісу.
* 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
* Розгорнути Load Balancer перед Споживачем Сервісу і/або Постачальником сервісу
* Опціонально: реалізувати протокол gRPC
* Опціонально: авторизація на рівні Споживача Сервісу
* Опціонально: авторизація на рівні Постачальника Сервісу

**Виконання**

Лабораторну роботу було виконано на мові Golang та розгорнуто у Docker. Також було використано Nginx для балансування навантаження. Авторизацію було реалізовано за допомогою jwt-токенів.

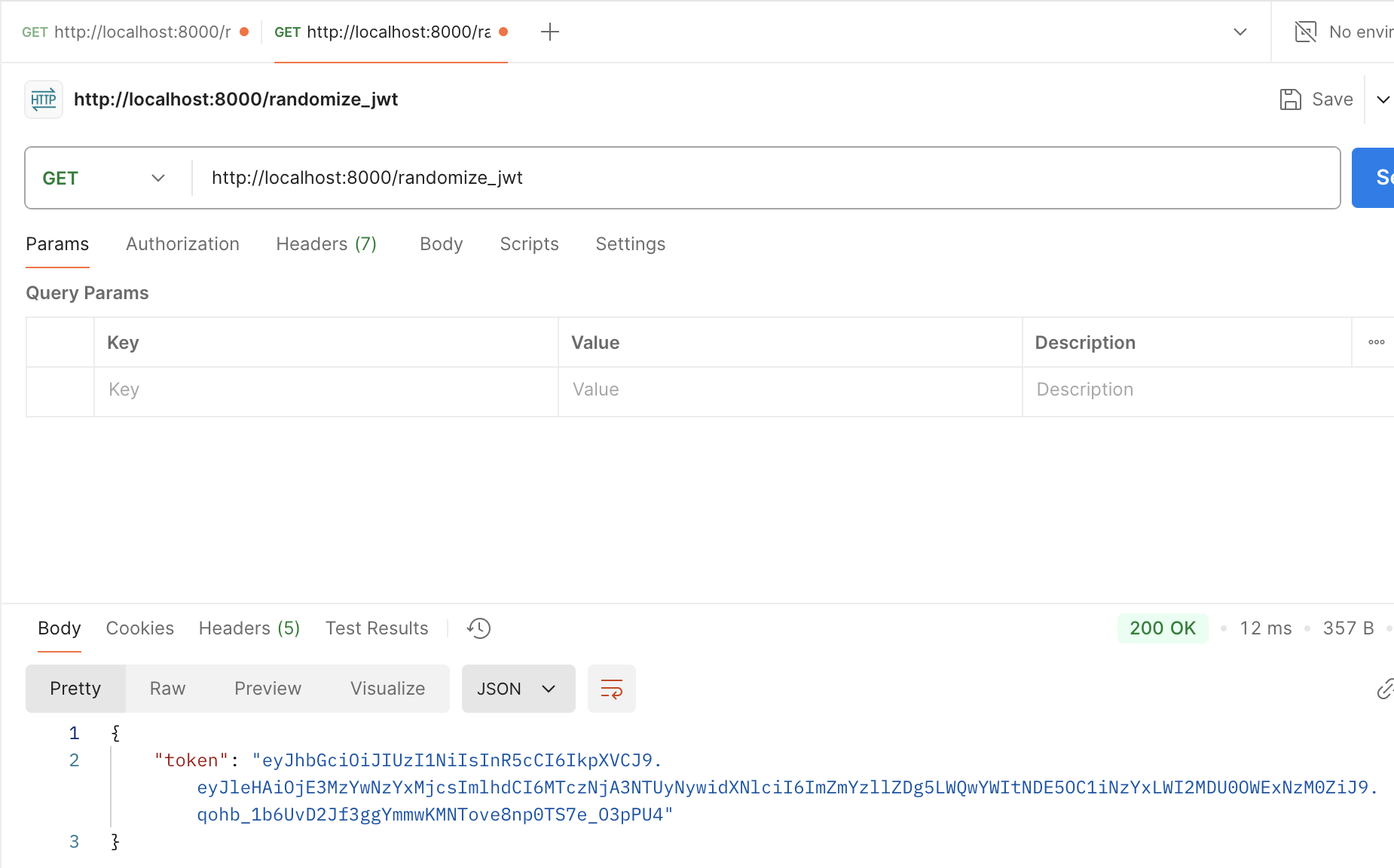
Було виконано всі пункти окрім реалізації протоколу gRPC.

Розроблена система містить балансувальники навантаження для споживача та постачальника, 3 інстанси споживача та 3 інстанси постачальника.

Сервіс споживача має 2 ендпоінти: “/randomize\_jwt” - для генерації випадкового jwt-токена та “/create\_task” - для створення завдання про обчилення та надсилання запиту до постачальника, а сервіс постачальника 1 - “/compute” - для математичних обчислень за даними, надісланими споживачем.

Протестуємо роботу, надсилаючи запити за допомогою “Postman”.

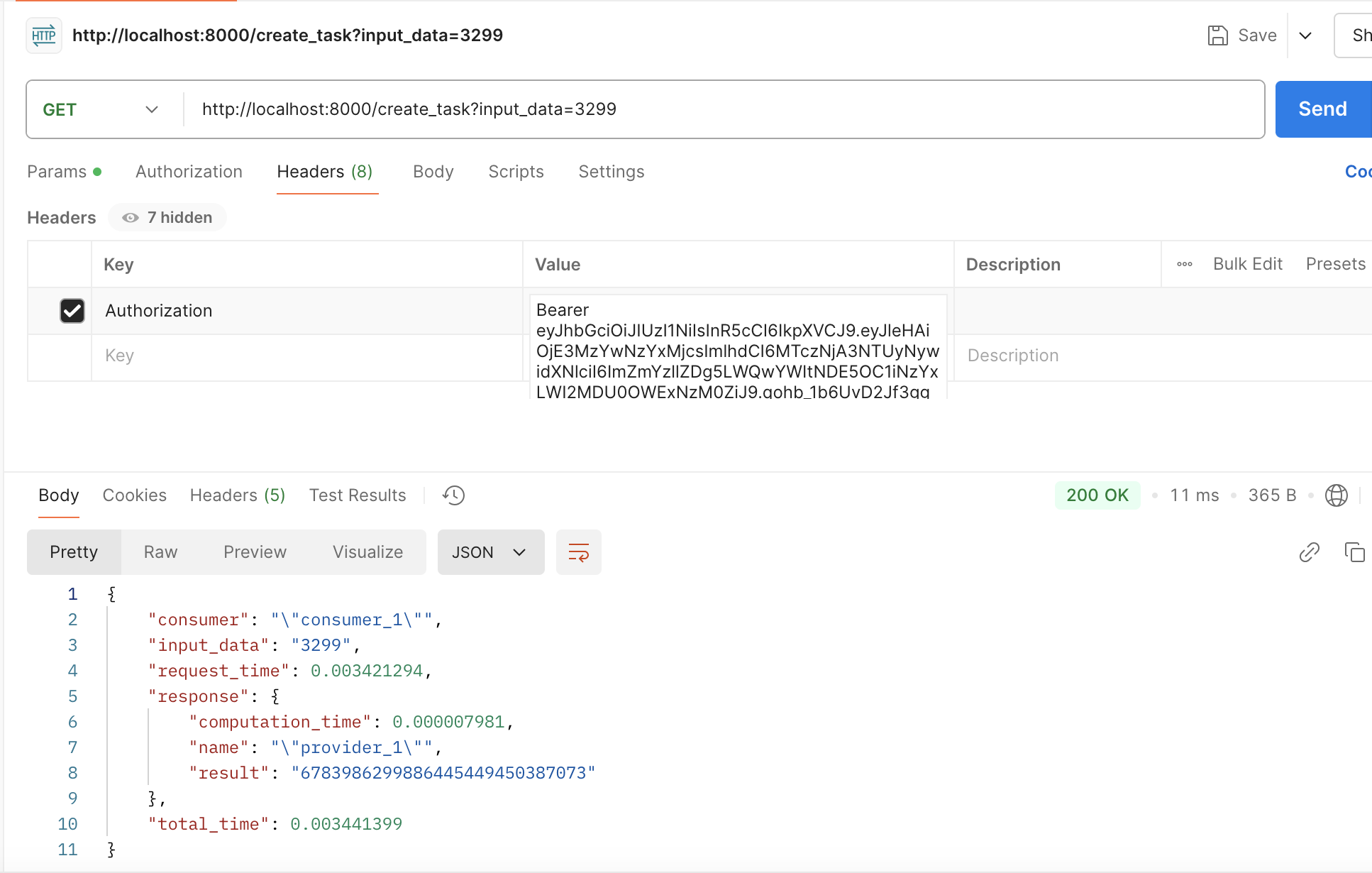
Спочатку надішлемо GET-запит за адресою <http://localhost:80/randomize_jwt> та отримаємо у відповідь згенерований токен (рис. 1).

Рисунок 1 - Генерація токену.

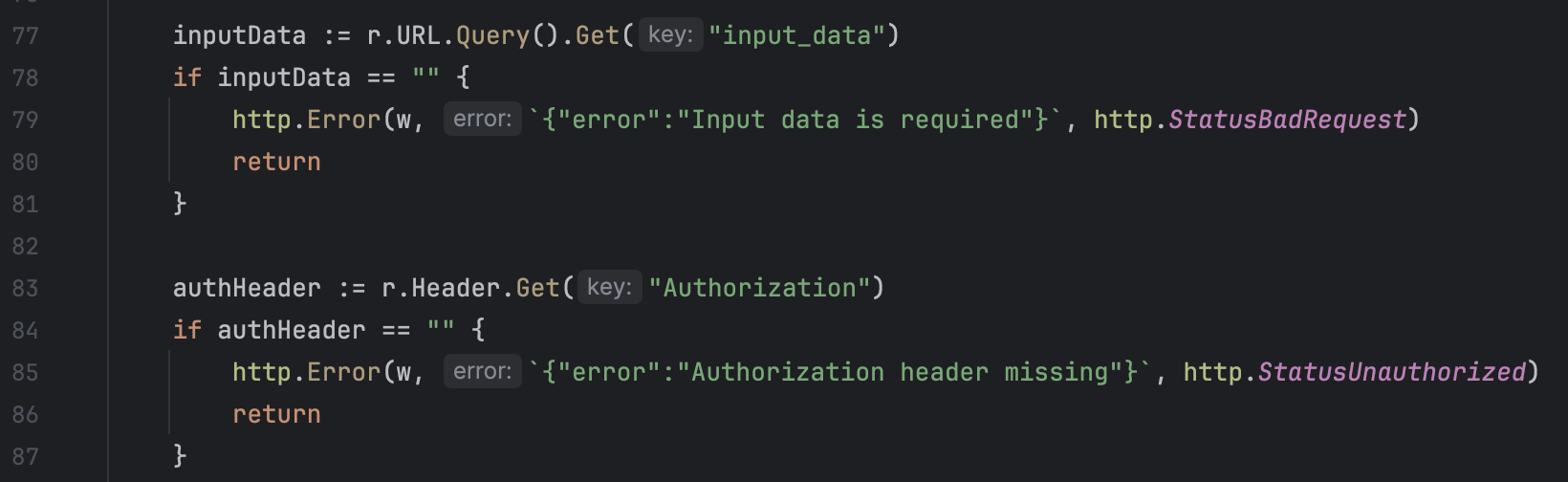
Генерація токену відбувається у наступному блоці коду (рис. 2).

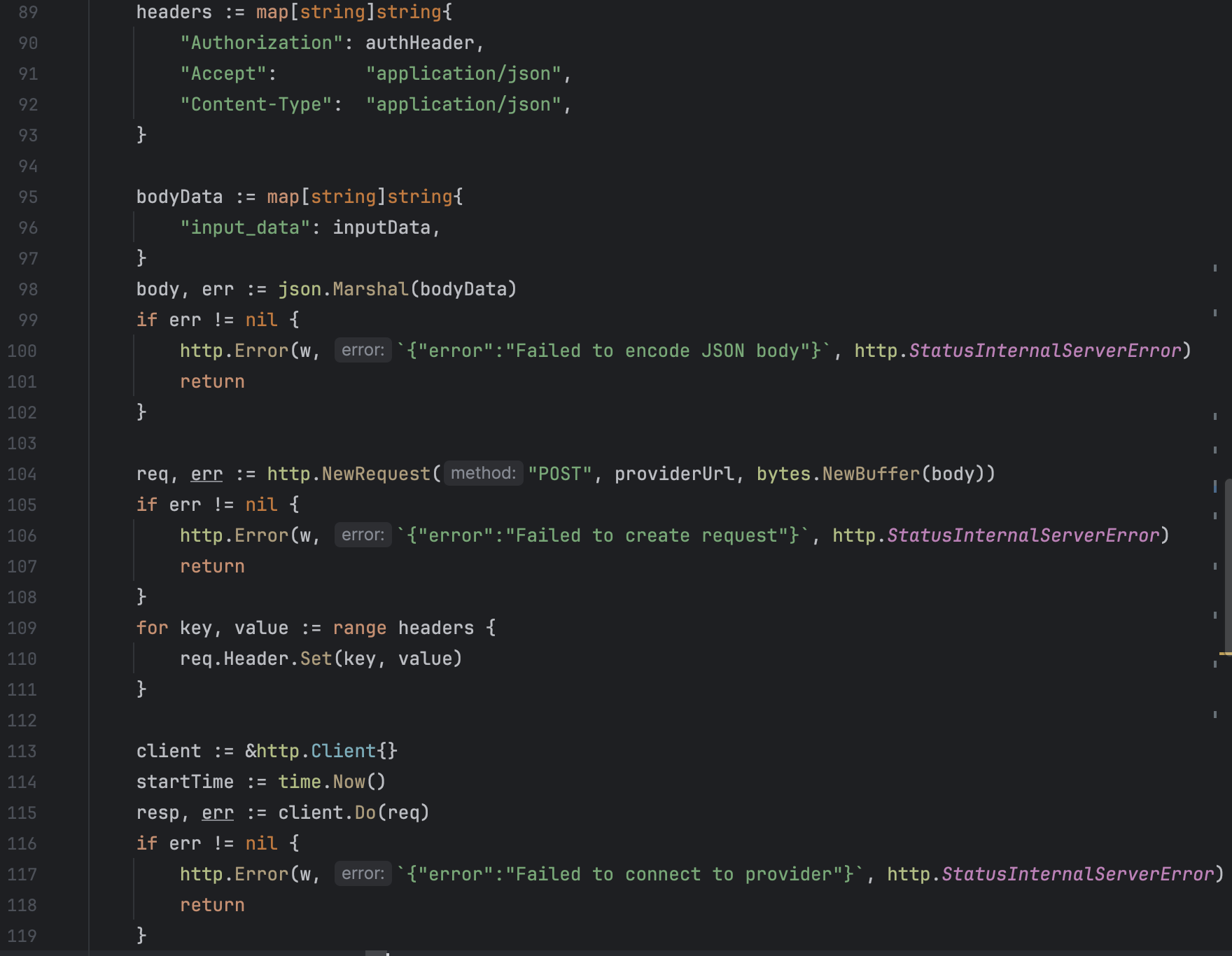
Рисунок 2 - Код генерації токену.

Тепер відправимо GET-запит споживачу за адресою <http://localhost:8000/create_task>. Тут необхідно також додати параметр “input\_data”. Надамо йому значення 3299. Тому шлях виглядатиме наступним чином: <http://localhost:8000/create_task?input_data=3299>. Також до заголовку запиту додамо ключ “Authorization” та значення, що містить префікс “Bearer ” та jwt-ключ згенерований попередньо (рис. 1). У якості відповіді отримаємо імена інстансів споживача та постачальника, що обробили запит, час, що пішов на обробку запиту споживачем та на обчислення постачальником, а також результат, отриманий внаслідок обчислень (рис. 3). Адже споживач у свою чергу відправляє запит постачальнику та повертає нам його відповідь разом з іншими даними.

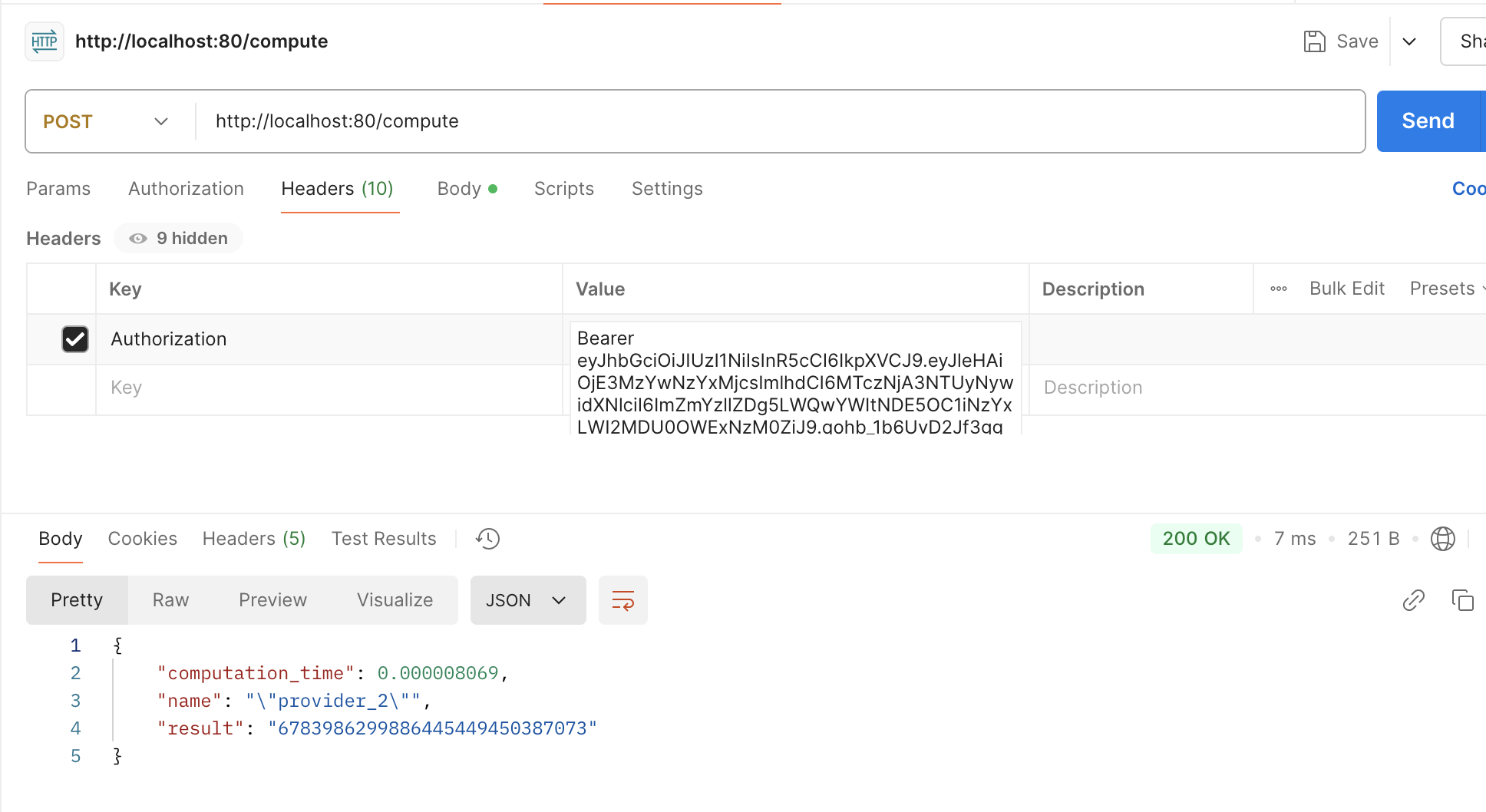
Рисунок 3 - Запит на створення завдання про обчислення.

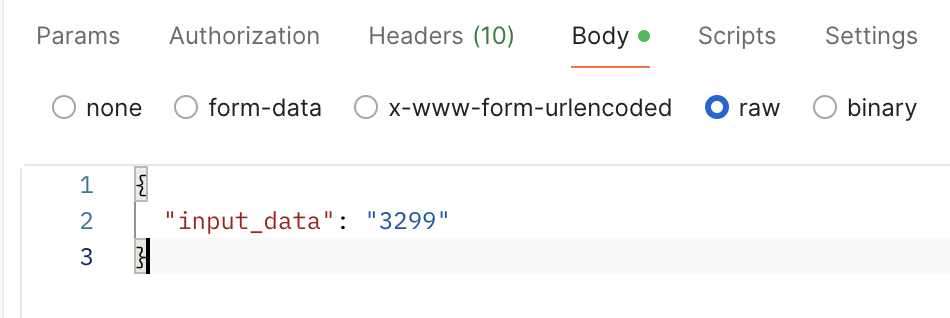
Обробка отриманих даних (рис. 4) та підготовка і відправка запиту постачальнику (рис. 5) відбувається в наступних блоках коду.

Рисунок 4 - Обробка даних з GET-запиту.

Рисунок 5 - Підготовка і відправка запиту постачальнику.

Спробуємо надіслати запит постачальнику напряму. Це POST-запит за адресою <http://localhost:80/compute>. У якості відповіді отримаємо частину відповіді з попереднього запиту, що містилась у полі “response” (рис. 6). Ми так само додаємо до заголовку запиту поле “Authorization”, а також переносимо “input\_data” з параметрів до тіла запиту (рис. 7).

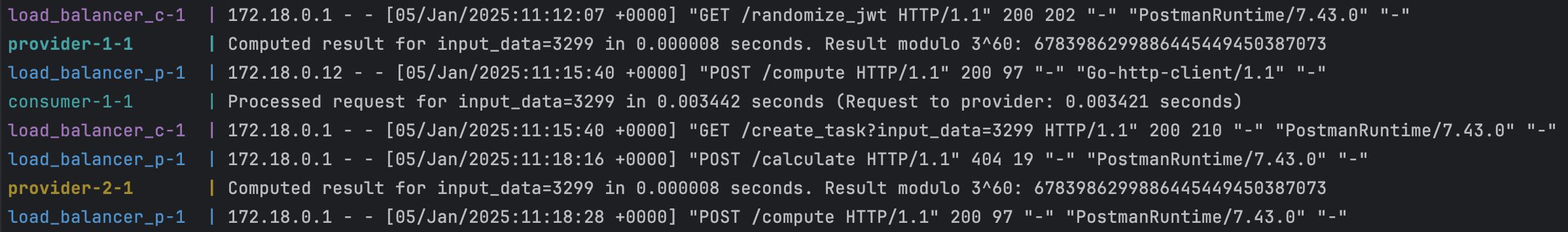
Рисунок 6 - Запит на обчислення постачальнику.

Рисунок 7 - ”input\_data” у тілі запиту.

Обчислення та формування відповіді від постачальника відбувається у наступному блоці коду (рис. 8). В якості обчислень знаходиться сума геометричної прогресії з n=101, де початковим елементом є “input\_data”, а знаменник дорівнює 2. Потім це отримана сума підноситься до модулю для зменшення отриманого числа.

Рисунок 8 - Проведення обчислень та формування відповіді.

Також внаслідок надісланих запитів отримано наступні логи (рис. 8).

Рисунок 8 - Логи сервісів.