НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №4
з дисципліни «Інформаційні системи»
на тему
“ Знайомство з принципом роботи брокера повідомлень”

Виконав:
студент групи КМ-01
Романецький М.С.

Викладач:
Доцент кафедри ПМА
Саяпіна І. О.

Київ — 2023

Зміст

Завдання 1 3
Завдання 2 6
Завдання 3 10
Висновки 14

Завдання 1

1. Змініть код Producer'a, щоб він генерував інший текст повідомлень за Вашим вибором.
2. Змініть назву черги повідомлень як у коді Producer'a, так і в коді Consumer'a та переконайтеся, що вони використовують однакову назву черги.
3. Змініть код Producer'a, щоб генерувати повідомлення з іншим інтервалом.
4. Змініть код Consumer'a, щоб роздрукувати кількість повідомлень, які він отримав, на додаток до вмісту повідомлення.

Код для програми Producer’a:

```
def task_1_producer():
    counter = 1
    QUEUE = 'Romanetskiy'

    while True:
        time_to_sleep = random.randint(500, 5000) / 1000  # 0.5 - 5 секунд
        time.sleep(time_to_sleep)

        connection =
pika.BlockingConnection(pika.ConnectionParameters('localhost'))
        channel = connection.channel()

        channel.queue_declare(queue=QUEUE, durable=False, exclusive=False,
auto_delete=False)

        message = f'Завдання 1 | Повідомлення {counter}'
        body = message.encode('utf-8')

        channel.basic_publish(exchange='', routing_key=QUEUE, body=body)

        print(f'[LOG] Повідомлення N {counter} | (TTS = {time_to_sleep})')
        counter += 1


try:
    task_1_producer()
except KeyboardInterrupt:
    print('[LOG] Зупинено')
```

Код для програми Consumer’а:

```
def task_1_consumer():
    global msg_counter
    msg_counter = 1
    QUEUE = 'Romanetskiy'

    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()

    def callback(ch, method, properties, body):
        global msg_counter
        print(f'Отримано повідомлення N {msg_counter}: {body.decode("utf-8")}')
        msg_counter += 1

    channel.queue_declare(queue=QUEUE, durable=False, exclusive=False,
auto_delete=False)

    channel.basic_consume(queue=QUEUE, on_message_callback=callback, auto_ack=True)

    print('\n[LOG] Початок відстеження черги')
    channel.start_consuming()


try:
    task_1_consumer()
except KeyboardInterrupt:
    print('[LOG] Зупинено')
```

```
[2]  ✓ 46.7s
...  [LOG] Повідомлення N 1 | (TTS = 3.464)
     [LOG] Повідомлення N 2 | (TTS = 4.026)
     [LOG] Повідомлення N 3 | (TTS = 2.172)
     [LOG] Повідомлення N 4 | (TTS = 3.7)
     [LOG] Повідомлення N 5 | (TTS = 4.788)
     [LOG] Повідомлення N 6 | (TTS = 2.451)
     [LOG] Повідомлення N 7 | (TTS = 2.381)
     [LOG] Повідомлення N 8 | (TTS = 3.287)
     [LOG] Повідомлення N 9 | (TTS = 4.264)
     [LOG] Повідомлення N 10 | (TTS = 2.824)
     [LOG] Повідомлення N 11 | (TTS = 4.467)
     [LOG] Повідомлення N 12 | (TTS = 1.723)
     [LOG] Повідомлення N 13 | (TTS = 0.561)
     [LOG] Повідомлення N 14 | (TTS = 2.104)
     [LOG] Повідомлення N 15 | (TTS = 1.761)
     [LOG] Зупинено
```

Рис. 1 – Завдання 1 Producer

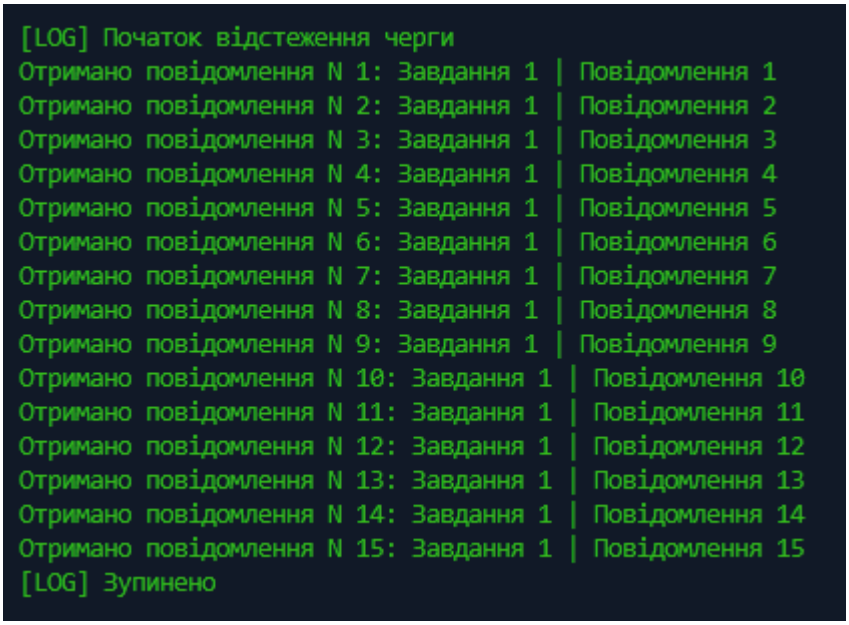

Рис. 2 – Завдання 1 Consumer

Спочатку було запущено consumer, програма вивела повідомлення, що розпочала відстеження черги. Потім запущено producer, програма почала генерувати повідомлення формату:

```
Завдання 1 | Повідомлення {counter}
```

і додавати їх у чергу. У виводі програми producer також можна побачити, що коли повідомлення було надіслано, то в консоль виводиться яке це повідомлення було по списку та інтервал між повідомленнями, оскільки цей інтервал обирається випадково (від 0.5 до 5 секунд) для імітації надсилання реальних повідомлень.

Черга має назву – ‘Romanetskiy’

Завдання 2

Створіть прямий обмін із трьома прив’язаними до нього чергами

1-а черга має отримувати повідомлення з ключем маршрутизації «alert:emergency»

2-а черга має отримувати повідомлення з ключем маршрутизації «alert:urgent»

3-я черга має отримувати повідомлення з ключем маршрутизації «alert:warning»

Додайте нове прив’язування до обмінника, щоб усі повідомлення з ключем маршрутизації «alert» також надсилалися до черги «alert:warning»

Код для програми Producer’a:

```
def task_2_producer():
    counter = 1
    exchange_name = 'direct_alerts'

    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()

    # Створення прямого обміну
    channel.exchange_declare(exchange=exchange_name, exchange_type='direct')

    while True:
        time_to_sleep = 3
        time.sleep(time_to_sleep)

        # Генерування повідомлення з відповідним ключем маршрутизації
        message = f"   \t Завдання 2 | Повідомлення {counter}"
        routing_key = random.choice(['alert:emergency', 'alert:urgent',
'alert:warning', 'alert'])
        body = message.encode('utf-8')

        # Надсилання повідомлення до прямого обміну з вказаним ключем маршрутизації
        channel.basic_publish(exchange=exchange_name, routing_key=routing_key,
body=body)

        print(f'Надіслано повідомлення {counter} | {routing_key}')
        counter += 1


try:
    task_2_producer()
except KeyboardInterrupt:
    print('[LOG] Зупинено')
```

Код для програми Consumer’а:

```
def task_2_consumer():
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()

    exchange_name = 'direct_alerts'

    channel.exchange_declare(exchange=exchange_name, exchange_type='direct')

    def emergency_callback(ch, method, properties, body):
        print(f"Отримано повідомлення з ключем маршрутизації 'alert:emergency':
{body.decode('utf-8')}")

    def urgent_callback(ch, method, properties, body):
        print(f"Отримано повідомлення з ключем маршрутизації 'alert:urgent':
{body.decode('utf-8')}")

    def warning_callback(ch, method, properties, body):
        print(f"Отримано повідомлення з ключем маршрутизації 'alert:warning':
{body.decode('utf-8')}")

    # Черга для 'alert:emergency'
    emergency_queue = 'emergency_queue'
    channel.queue_declare(queue=emergency_queue, durable=False)
    channel.queue_bind(exchange=exchange_name, queue=emergency_queue,
routing_key='alert:emergency')
    channel.basic_consume(queue=emergency_queue,
on_message_callback=emergency_callback, auto_ack=True)

    # Черга для 'alert:urgent'
    urgent_queue = 'urgent_queue'
    channel.queue_declare(queue=urgent_queue, durable=False)
    channel.queue_bind(exchange=exchange_name, queue=urgent_queue,
routing_key='alert:urgent')
    channel.basic_consume(queue=urgent_queue, on_message_callback=urgent_callback,
auto_ack=True)

    # Черга для 'alert:warning'
    warning_queue = 'warning_queue'
    channel.queue_declare(queue=warning_queue, durable=False)
    channel.queue_bind(exchange=exchange_name, queue=warning_queue,
routing_key='alert:warning')
    channel.queue_bind(exchange=exchange_name, queue=warning_queue,
routing_key='alert')
    channel.basic_consume(queue=warning_queue,
on_message_callback=warning_callback, auto_ack=True)

    print('\n[LOG] Початок відстеження черг')
    channel.start_consuming()
```

```python
try:
    task_2_consumer()
except KeyboardInterrupt:
    print('[LOG] Зупинено')
```

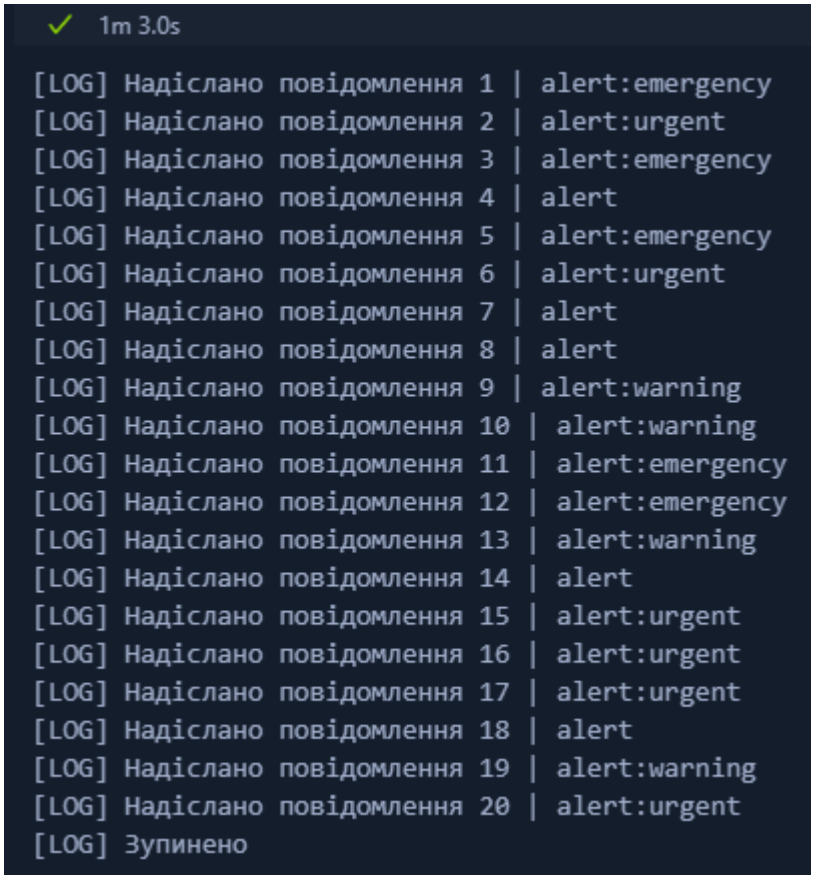

Рис. 3 – Завдання 2 Producer

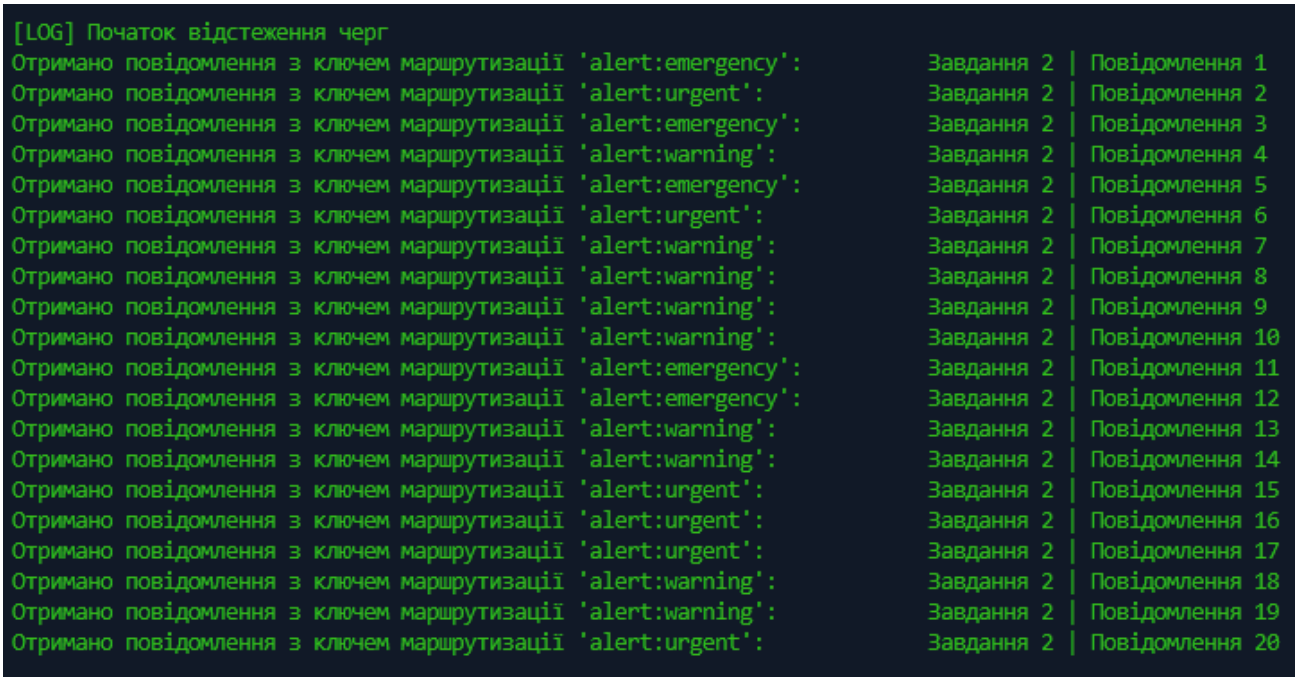

Рис. 4 – Завдання 2 Consumer

Спочатку було запущено consumer, програма вивела повідомлення, що розпочала відстеження черг. Потім запущено producer, програма почала генерувати повідомлення формату:

```
message = f"   \t Завдання 2 | Повідомлення {counter}"
```

і додавати їх у чергу. У виводі програми producer також можна побачити, що коли повідомлення було надіслано, то в консоль виводиться яке це повідомлення було по списку та його ключ маршрутизації.

Черги мають назви – ‘emergency_queue’, ‘urgent_queue’, ‘warning_queue’

Завдання 3

Створіть обмінник для веб-сайту електронної комерції

Ключі маршрутизації мають формат "product.<category>.<action>"

Категорією може бути електроніка, одяг, книги, тощо

Дією може бути create, update чи delete

Майте єдину чергу, яка підписується на обмінник, щоб відстежувати всі зміни (update) продукту.

Код для програми Producer'a:

```
def task_3_producer():
    counter = 1
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()

    channel.exchange_declare(exchange='product_exchange', exchange_type='topic')

    while True:
        time_to_sleep = random.randint(1, 3)
        time.sleep(time_to_sleep)

        category = random.choice(['electronics', 'clothing', 'books'])  # список можливих категорій
        action = random.choice(['create', 'update', 'delete'])  #  список можливих дій

        routing_key = f"product.{category}.{action}"
        message = f'Продукт {action} в категорії {category}'
        body = message.encode('utf-8')

        channel.basic_publish(exchange='product_exchange', routing_key=routing_key, body=body)

        print(f'[LOG] Надіслано повідомлення {counter}\t| {routing_key}   \t| {message}')
        counter += 1


try:
    task_3_producer()
except KeyboardInterrupt:
    print('[LOG] Зупинено')
```

Код для програми Consumer’а:

```
import random
import time

import pika


def task_3_consumer():
    connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
    channel = connection.channel()

    def callback(ch, method, properties, body):
        print(f'Отримано повідомлення: {body.decode("utf-8")}')

    channel.exchange_declare(exchange='product_exchange', exchange_type='topic')
    channel.queue_declare(queue='product_changes')

    # Видалення всіх існуючих зв'язків для queue
    for rk in ['product.*.*', 'product.*.update']:
        channel.queue_unbind(queue='product_changes', exchange='product_exchange',
routing_key=rk)

    # channel.queue_bind(exchange='product_exchange', queue='product_changes',
routing_key='product.*.*')
    channel.queue_bind(exchange='product_exchange', queue='product_changes',
routing_key='product.*.update')

    channel.basic_consume(queue='product_changes', on_message_callback=callback,
auto_ack=True)

    print('\n[LOG] Початок відстеження черги')
    channel.start_consuming()


try:
    task_3_consumer()
except KeyboardInterrupt:
    print('[LOG] Зупинено')
```

Скріншоти програм, коли consumer обробляє лише update

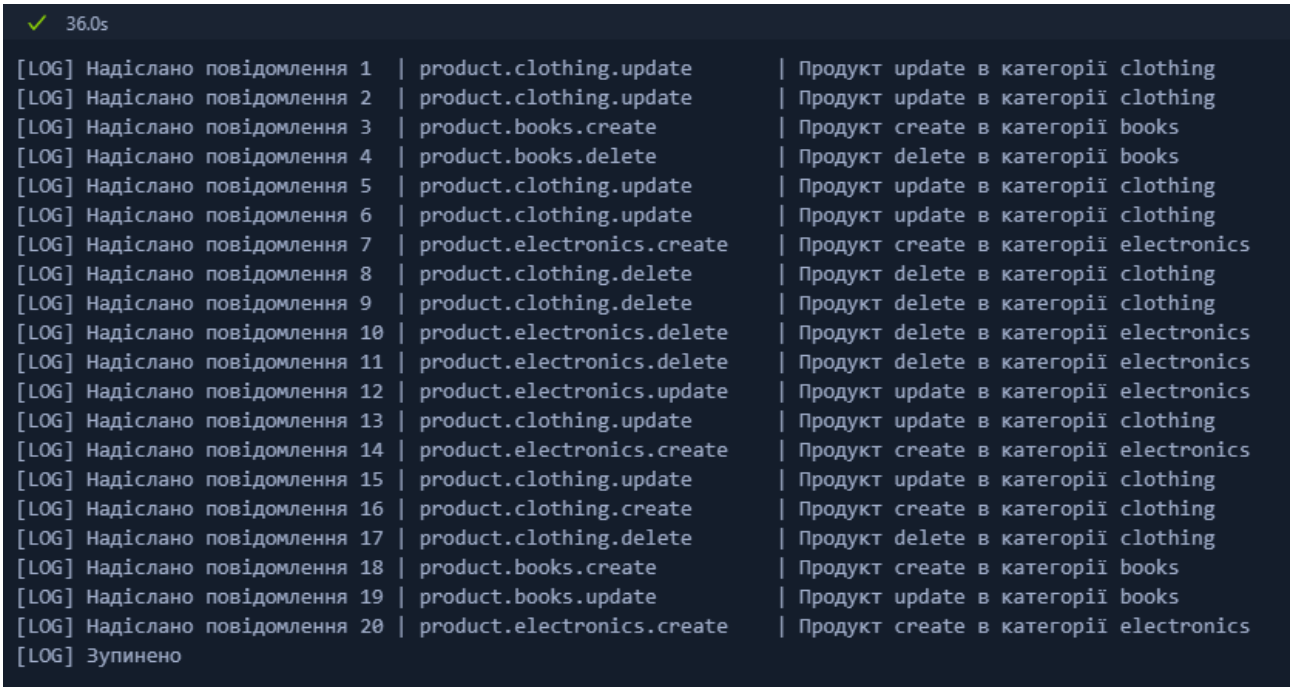

Рис. 5 – Завдання 3 Producer (only update)

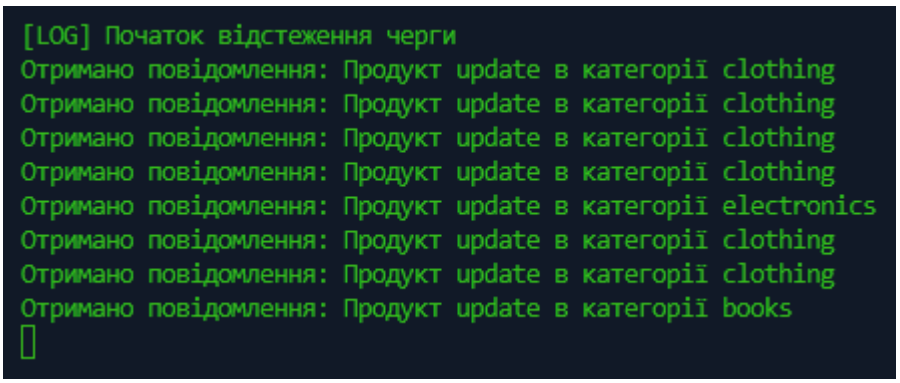

Рис. 6 – Завдання 3 Consumer (only update)

Скріншоти програм, коли consumer обробляє create, update, delete

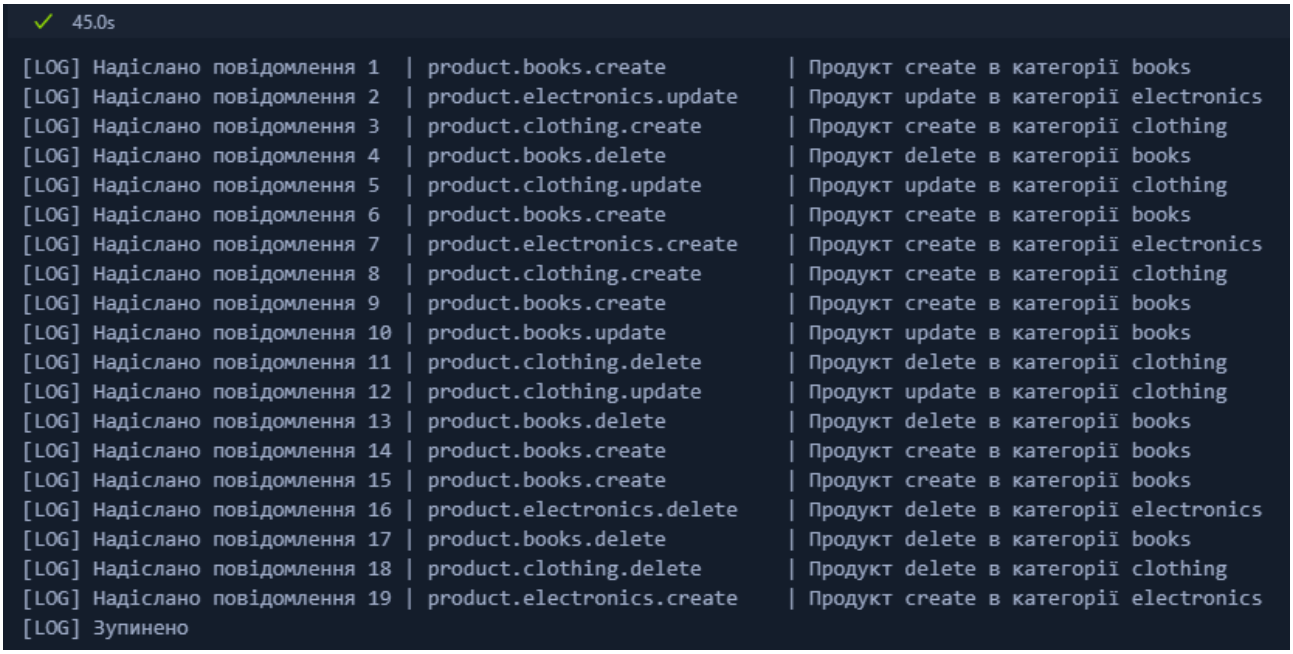

Рис. 7 – Завдання 3 Producer (create, update, delete)

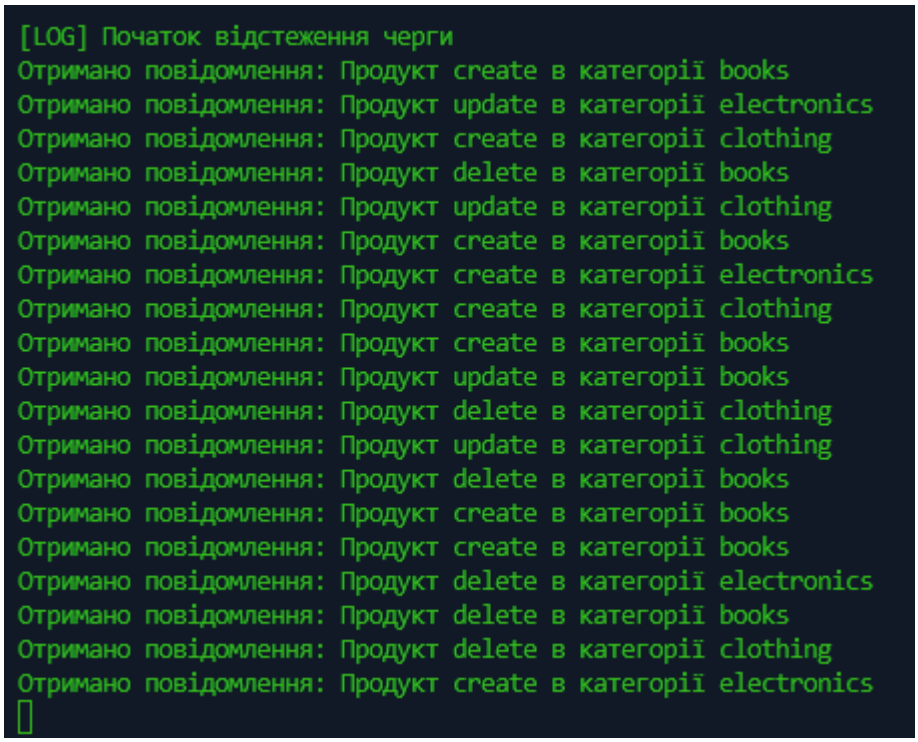

Рис. 8 – Завдання 3 Consumer (create, update, delete)

Висновки

У ході лабораторної роботи ознайомився із базовими принципами роботи брокера повідомлень RabbitMQ та його використання для асинхронного обміну повідомленнями. В процесі були вивчені основні концепції такі, як Producer, Consumer, обмінники та черги.

Завдання дозволили провести практичні вправи із зміною коду, створенням прив’язаних черг, та роботою із ключами маршрутизації. Результатом стала практична робота із реальними сценаріями використання брокера повідомлень для організації асинхронного обміну повідомленнями в розподілених системах.