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”

Варіант 7

|  |  |
| --- | --- |
| Виконав: | Викладач: |
| студент групи КМ-01 | Доцент кафедри ПМА |
| Іваник Ю. П. | Саяпіна І. О. |

Київ — 2023

***Зміст***

[***Завдання 1*** 3](#_Toc153142753)

[***Завдання 2*** 6](#_Toc153142754)

[***Завдання 3*** 10](#_Toc153142755)

[***Висновки*** 13](#_Toc153142756)

# ***Завдання 1***

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

Код для програми Producer’а:

def task\_1\_producer():

    counter = 1

    QUEUE = 'Ivanyk'

    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}')

        counter += 1

try:

    task\_1\_producer()

except KeyboardInterrupt:

    print('[LOG] Зупинено')

Код для програми Consumer’а:

import random

import time

import pika

def task\_1\_consumer():

    global msg\_counter

    msg\_counter = 1

    QUEUE = 'Ivanyk'

    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')

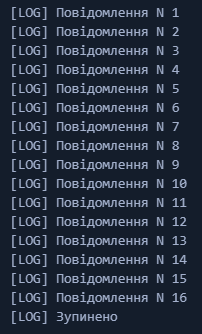


Рис. 1 – Завдання 1 Producer

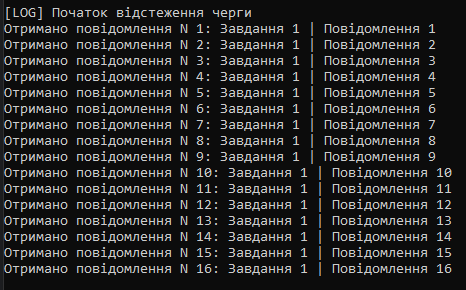


Рис. 2 – Завдання 1 Consumer

Спочатку було запущено consumer, програма повідомила, що почала відстежувати чергу. Далі запущено producer, програма розпочала генерування повідомлень і їх додавання до черги.

# ***Завдання 2***

Створіть прямий обмін із трьома прив’язаними до нього чергами.

Перша черга повинна отримувати повідомлення з ключем маршрутизації "production", друга черга повинна отримувати повідомлення з ключем маршрутизації "testing", а третя черга повинна отримувати повідомлення з ключем маршрутизації "development". Додайте нову прив’язку до обмінника, щоб усі повідомлення з ключем маршрутизації «debug» також надсилалися до черги «development».

Код для програми Producer’а:

def task\_2\_producer():

    counter = 1

    exchange\_name = 'direct\_routing'

    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(['production', 'testing', 'development', 'debug'])

        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'[LOG] Надіслано повідомлення {counter} | {routing\_key}')

        counter += 1

try:

    task\_2\_producer()

except KeyboardInterrupt:

    print('[LOG] Зупинено')

Код для програми Consumer’а:

import random

import time

import pika

def task\_2\_consumer():

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

    channel = connection.channel()

    exchange\_name = 'direct\_routing'

    channel.exchange\_declare(exchange=exchange\_name, exchange\_type='direct')

    def production\_callback(ch, method, properties, body):

        print(f"Отримано повідомлення з ключем маршрутизації 'production': {body.decode('utf-8')}")

    def testing\_callback(ch, method, properties, body):

        print(f"Отримано повідомлення з ключем маршрутизації 'testing': {body.decode('utf-8')}")

    def development\_callback(ch, method, properties, body):

        print(f"Отримано повідомлення з ключем маршрутизації 'development': {body.decode('utf-8')}")

    # Черга для 'production'

    production\_queue = 'production\_queue'

    channel.queue\_declare(queue=production\_queue, durable=False)

    channel.queue\_bind(exchange=exchange\_name, queue=production\_queue, routing\_key='production')

    channel.basic\_consume(queue=production\_queue, on\_message\_callback=production\_callback, auto\_ack=True)

    # Черга для 'testing'

    testing\_queue = 'testing\_queue'

    channel.queue\_declare(queue=testing\_queue, durable=False)

    channel.queue\_bind(exchange=exchange\_name, queue=testing\_queue, routing\_key='testing')

    channel.basic\_consume(queue=testing\_queue, on\_message\_callback=testing\_callback, auto\_ack=True)

    # Черга для 'development'

    development\_queue = 'development\_queue'

    channel.queue\_declare(queue=development\_queue, durable=False)

    channel.queue\_bind(exchange=exchange\_name, queue=development\_queue, routing\_key='development')

    channel.queue\_bind(exchange=exchange\_name, queue=development\_queue, routing\_key='debug')

    channel.basic\_consume(queue=development\_queue, on\_message\_callback=development\_callback, auto\_ack=True)

    print('\n[LOG] Початок відстеження черг')

    channel.start\_consuming()

try:

    task\_2\_consumer()

except KeyboardInterrupt:

    print('[LOG] Зупинено')

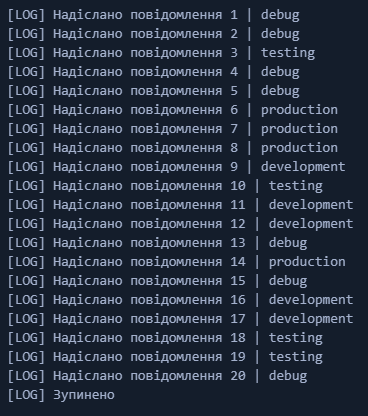


Рис. 3 – Завдання 2 Producer

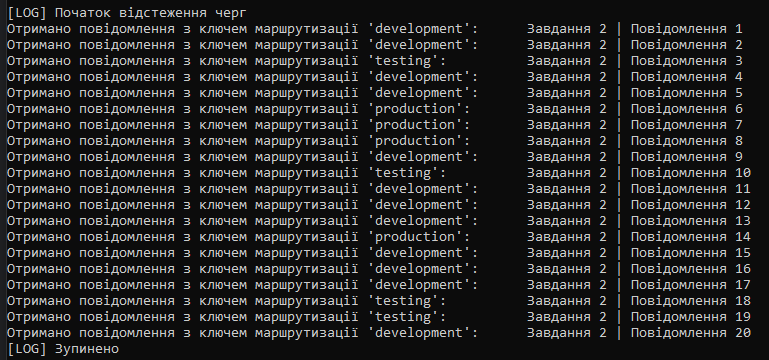


Рис. 4 – Завдання 2 Consumer

Спочатку був запущений споживач (consumer), і програма вивела повідомлення про те, що вона почала відстежування черги. Після цього був запущений виробник (producer), який розпочав генерувати повідомлення і додавати їх до черги. У виводі програми виробника також можна помітити, що після надсилання повідомлення в консоль виводиться інформація про це повідомлення у вигляді списку та його ключа маршрутизації.

# ***Завдання 3***

Створіть тематичний обмін із чергою для кожної категорії новин

(наприклад, sports, politics, entertainment тощо). Прив’яжіть кожну чергу до

обмінника за допомогою ключа маршрутизації "news.{category}".

Код для програми Producer’а:

def task\_3\_producer():

    counter = 1

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

    channel = connection.channel()

    channel.exchange\_declare(exchange='news\_exchange', exchange\_type='topic')

    while True:

        time\_to\_sleep = random.randint(1, 3)

        time.sleep(time\_to\_sleep)

        category = random.choice(['sports', 'politics', 'entertainment'])  # список можливих категорій

        routing\_key = f"news.{category}"

        message = f'Новина в категорії {category}'

        body = message.encode('utf-8')

        channel.basic\_publish(exchange='news\_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='news\_exchange', exchange\_type='topic')

    channel.queue\_declare(queue='news')

    # Видалення всіх існуючих зв'язків для queue

    for rk in ['news.\*']:

        channel.queue\_unbind(queue='news', exchange='news\_exchange', routing\_key=rk)

    channel.queue\_bind(exchange='news\_exchange', queue='news', routing\_key='news.\*')

    channel.basic\_consume(queue='news', 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')



Рис. 5 – Завдання 3 Producer

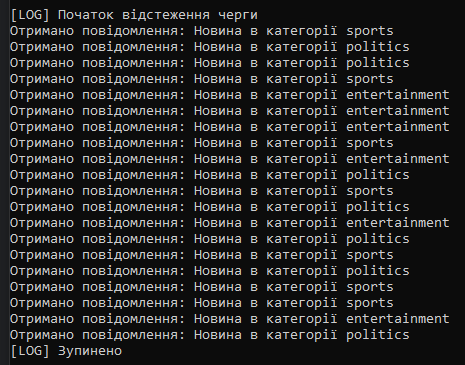


Рис. 6 – Завдання 3 Consumer

# ***Висновки***

Під час виконання лабораторної роботи я отримав знання про основні принципи функціонування брокера повідомлень RabbitMQ та його застосування для асинхронного обміну повідомленнями. Мною були освоєні ключові концепції, такі як Producer, Consumer, обмінники та черги.

В процесі виконання завдань я створював прив'язані черги та працював із ключами маршрутизації. Це дозволило мені здійснити практичні вправи, що стосуються реальних сценаріїв використання брокера повідомлень для організації асинхронного обміну повідомленнями у розподілених системах. Отримані в ході цієї роботи навички та досвід використання RabbitMQ виявляться корисними при подальшій роботі з розподіленими системами та асинхронною комунікацією.