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: | Викладач: |
| студент групи КМ-01 | Доцент кафедри ПМА |
| Романецький М.С. | Саяпіна І. О. |

Київ — 2023

***Зміст***

[***Завдання 1*** 3](#_Toc151386222)

[***Завдання 2*** 6](#_Toc151386223)

[***Завдання 3*** 10](#_Toc151386224)

[***Висновки*** 14](#_Toc151386225)

# ***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 = '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')

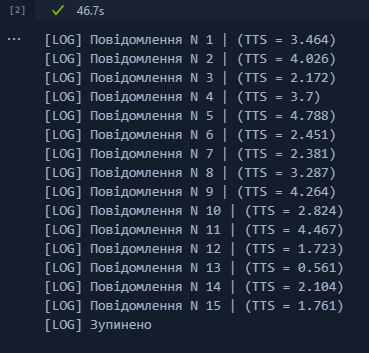


Рис. 1 – Завдання 1 Producer

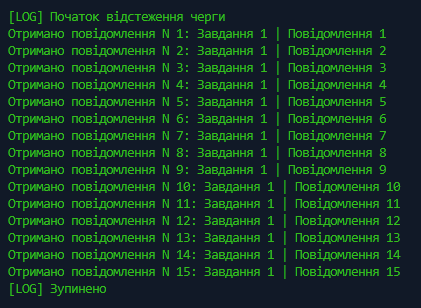


Рис. 2 – Завдання 1 Consumer

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’а:

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()

try:

    task\_2\_consumer()

except KeyboardInterrupt:

    print('[LOG] Зупинено')

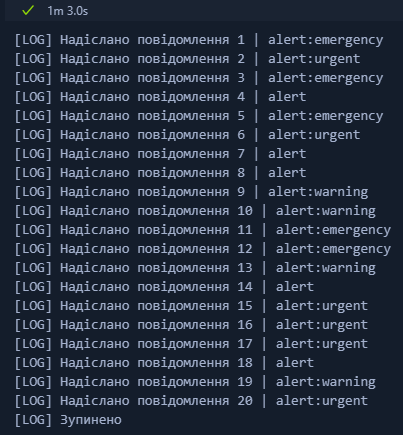


Рис. 3 – Завдання 2 Producer

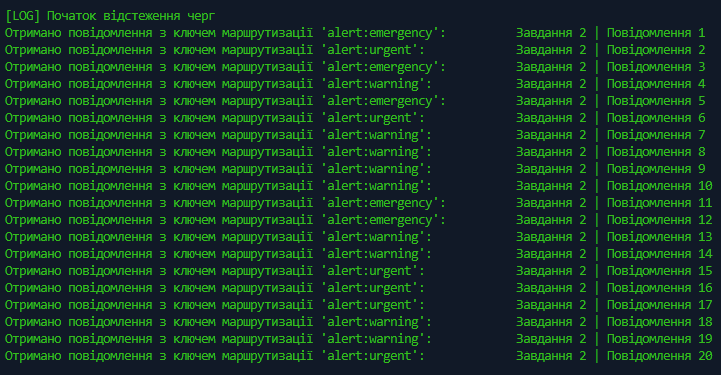


Рис. 4 – Завдання 2 Consumer

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’а:

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

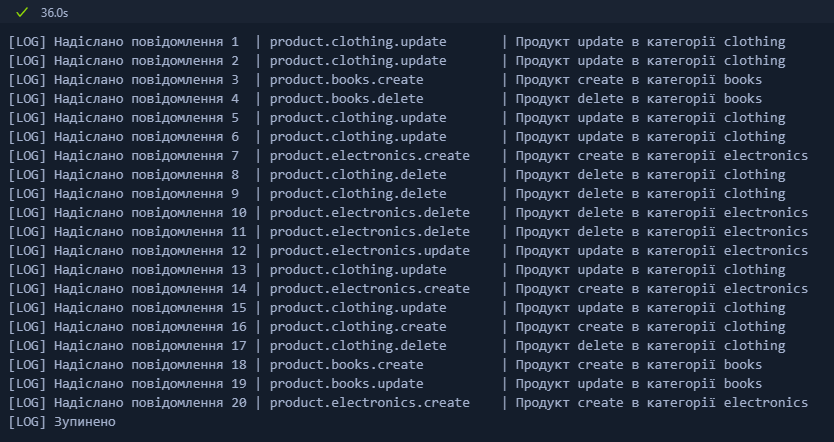


Рис. 5 – Завдання 3 Producer (only update)

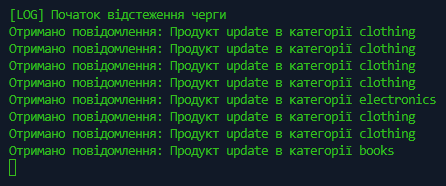


Рис. 6 – Завдання 3 Consumer (only update)

Скріншоти програм, коли consumer обробляє create, update, delete

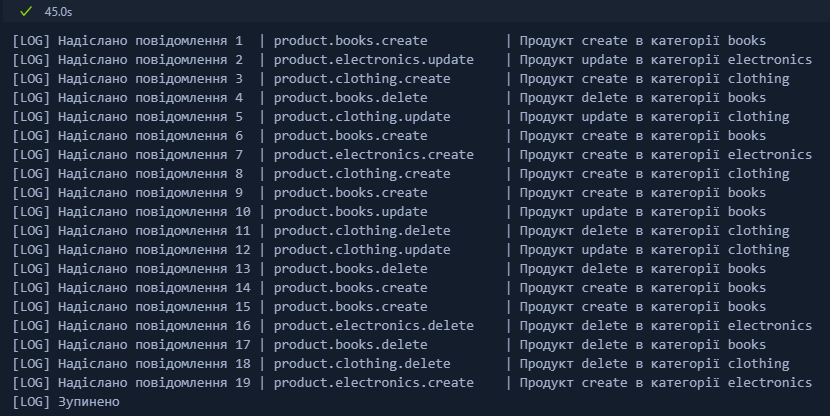


Рис. 7 – Завдання 3 Producer (create, update, delete)

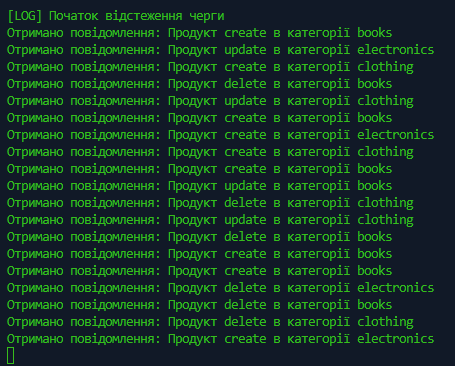


Рис. 8 – Завдання 3 Consumer (create, update, delete)

# ***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.