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 15

|  |  |
| --- | --- |
| Виконала: | Викладач: |
| студентка групи КМ-01 | Саяпіна І. О. |
| Резниченко Є. С. |  |

Київ — 2023

Завдання 1

Код для програми Producer’а:

import pika

import random

import time

import os

import sys

def main():

    counter = 1

    queue = 'Reznichenko'

    while True:

        time\_to\_sleep = random.randint(1, 5)

        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"Повідомлення {counter}"

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

        channel.basic\_publish(exchange='',

                              routing\_key=queue,

                              body=body)

        print(f"Повідомлення надіслано у Default Exchange [N:{counter}]")

        counter += 1

    connection.close()

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        main()

    except KeyboardInterrupt:

        print('Interrupted')

        try:

            sys.exit(0)

        except SystemExit:

            os.\_exit(0)

Код для програми Consumer’а:

import pika

import sys

import os

def callback(ch, method, properties, body):

    global message\_counter

    message\_counter += 1

    print(f"Ми отримали повідомлення: {body.decode('utf-8')}\nЗагальна кількість повідомлень: {message\_counter}\n")

def main():

    global message\_counter

    queue = 'Reznichenko'

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

    channel = connection.channel()

    channel.queue\_declare(queue=queue, durable=False, exclusive=False, auto\_delete=False)

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

    message\_counter = 0

    print("Subscribed to the queue 'Reznichenko'")

    channel.start\_consuming()

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        main()

    except KeyboardInterrupt:

        print('Interrupted')

        try:

            sys.exit(0)

        except SystemExit:

            os.\_exit(0)

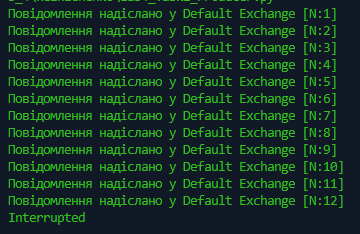


Рис. 1 – Завдання 1 Producer

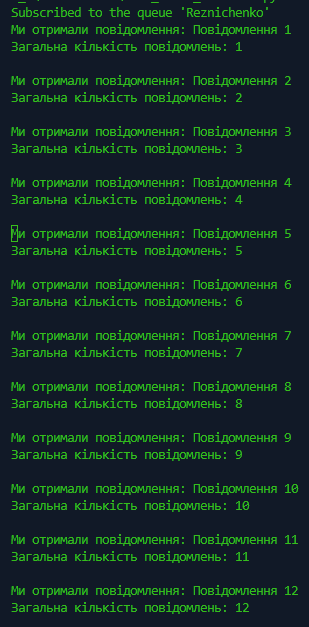


Рис. 2 – Завдання 1 Consumer

Consumer відстежує наявність повідомлень у черзі, а producer заповнює цю чергу повідомленнями. Інтервал обирається випадково від 1 до 5 секунд для імітації надсилання реальних повідомлень.

Завдання 2

Код для програми Producer’а:

import pika

import random

import time

import sys

import os

def main():

    counter = 1

    exchange\_name = 'direct\_exchange'

    routing\_keys = ["report:daily", "report:weekly", "report:monthly"]

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

    channel = connection.channel()

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

    while True:

        time\_to\_sleep = 3

        time.sleep(time\_to\_sleep)

        routing\_key = random.choice(routing\_keys)

        message = f"Повідомлення N {counter}"

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

        channel.basic\_publish(

            exchange=exchange\_name,

            routing\_key=routing\_key,

            body=body

        )

        print(f"Повідомлення надіслано з ключем маршрутизації '{routing\_key}' [N:{counter}]")

        counter += 1

    connection.close()

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        main()

    except KeyboardInterrupt:

        print('Interrupted')

        try:

            sys.exit(0)

        except SystemExit:

            os.\_exit(0)

Код для програми Consumer’а:

import pika

import sys

import os

def container\_callback(queue\_name):

    def callback(ch, method, properties, body):

        print(f"Ми отримали повідомлення: {body.decode('utf-8')} з ключем маршрутизації '{method.routing\_key}' "

              f"з черги '{queue\_name}'")

    return callback

def main():

    exchange\_name = 'direct\_exchange'

    queues\_bindings = {

        "report:daily": "daily\_queue",

        "report:weekly": "weekly\_queue",

        "report:monthly": ["daily\_queue", "weekly\_queue"]

    }

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

    channel = connection.channel()

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

    for routing\_key, queue\_name in queues\_bindings.items():

        if isinstance(queue\_name, list):

            for q in queue\_name:

                channel.queue\_declare(queue=q, durable=False)

                channel.queue\_bind(exchange=exchange\_name, queue=q, routing\_key=routing\_key)

        else:

            channel.queue\_declare(queue=queue\_name, durable=False)

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

    channel.basic\_consume(queue="daily\_queue", on\_message\_callback=container\_callback('daily\_queue'), auto\_ack=True)

    channel.basic\_consume(queue="weekly\_queue", on\_message\_callback=container\_callback('weekly\_queue'), auto\_ack=True)

    print("Started consuming messages...")

    channel.start\_consuming()

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        main()

    except KeyboardInterrupt:

        print('Interrupted')

        try:

            sys.exit(0)

        except SystemExit:

            os.\_exit(0)

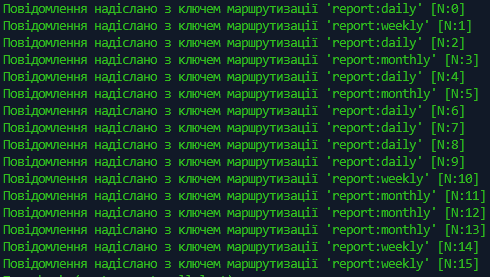


Рис. 3 – Завдання 2 Producer

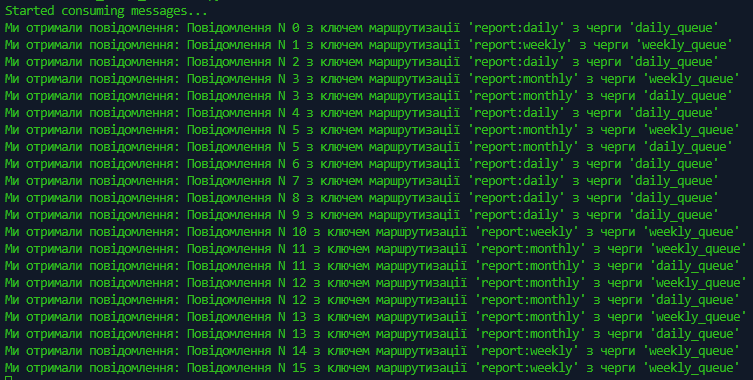


Рис. 4 – Завдання 2 Consumer

Consumer відстежує наявність повідомлень у черзі, а producer заповнює цю чергу повідомленнями. Інтервал обирається випадково від 1 до 5 секунд для імітації надсилання реальних повідомлень.

Створено 2 черги (‘daily’, ‘weekly’). Якщо ключ маршрутизації "report:daily", то повідомлення потрапляє в першу чергу, якщо "report:weekly", то в другу чергу. Але якщо ключ ‘report:monthly’, то повідомлення потрапить у обидві черги.

Завдання 3

Код для програми Producer’а:

import pika

import random

import time

import sys

import os

def main():

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

    channel = connection.channel()

    counter = 1

    channel.exchange\_declare(exchange='sports\_news', exchange\_type='topic')  *# Creating a topic exchange*

    while True:

        time.sleep(1)

        sport\_type = random.choice(['football', 'basketball', 'tennis'])  *# Randomly choosing a sport type*

        news\_category = random.choice(['transfer-news', 'match-report', 'injury-update'])  *# Random news category*

        routing\_key = f'sports.{sport\_type}.{news\_category}'  *# Forming the routing key*

        message = f"[N:{counter}], {sport\_type}, category: {news\_category}"

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

        channel.basic\_publish(exchange='sports\_news',

                              routing\_key=routing\_key,

                              body=body)

        print(f"News sent [{counter}] [Routing Key: {routing\_key}]")

        counter += 1

    connection.close()

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        main()

    except KeyboardInterrupt:

        print('Interrupted')

        try:

            sys.exit(0)

        except SystemExit:

            os.\_exit(0)

Код для програми Consumer’а:

import pika

import sys

import os

def callback(ch, method, properties, body):

    print(f"Отримано: {body.decode('utf-8')}      \t[Routing Key: {method.routing\_key}]")

def main():

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

    channel = connection.channel()

    channel.exchange\_declare(exchange='sports\_news', exchange\_type='topic')  *# Creating a topic exchange*

    result = channel.queue\_declare('', exclusive=True)

    queue\_name = result.method.queue

    channel.queue\_bind(exchange='sports\_news', queue=queue\_name, routing\_key='sports.football.#')  *# Subscribing to football news*

    channel.queue\_bind(exchange='sports\_news', queue=queue\_name, routing\_key='sports.basketball.#')  *# Subscribing to basketball news*

    channel.queue\_bind(exchange='sports\_news', queue=queue\_name, routing\_key='sports.tennis.#')  *# Subscribing to tennis news*

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

    print("Waiting for news...")

    channel.start\_consuming()

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        main()

    except KeyboardInterrupt:

        print('Interrupted')

        try:

            sys.exit(0)

        except SystemExit:

            os.\_exit(0)



Рис. 5 – Завдання 3 Producer



Рис. 6 – Завдання 3 Consumer

Створюємо 3 окремі черги, які прийматимуть різні види спорту

routing\_key='sports.football.#'

routing\_key='sports.basketball.#

routing\_key='sports.tennis.#'

# - означає нуль чи більше слів

Висновки

Під час виконання лабораторної роботи я ознайомилася з основними принципами роботи брокера повідомлень RabbitMQ та його застосуванням для асинхронного обміну повідомленнями. У процесі були досліджені ключові концепції, такі як Producer, Consumer, обмінники та черги.