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

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

Виконала:
студент групи КМ-01
Боженко А. О.

Керівник:
Саяпіна . І. О.

Київ – 2023

Зміст

МЕТА

Дослідити організацію асинхронного режиму обміну повідомлень на основі роботи з брокером повідомлень RabbitMQ.

Основна частина

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.

   6. За результатами роботи додайте до звіту код Producer`a, Consumer`a та скріни 10-15 повідомлень, що вони виводять.

Виконання:

```
PS D:\inform_systems\лабки\lab_4\defaultExchange> python publisher.py
Message is sent to Default Exchange [N:1]
Message is sent to Default Exchange [N:2]
Message is sent to Default Exchange [N:3]
Message is sent to Default Exchange [N:4]
Message is sent to Default Exchange [N:5]
Message is sent to Default Exchange [N:6]
Message is sent to Default Exchange [N:7]
Message is sent to Default Exchange [N:8]
Message is sent to Default Exchange [N:9]
Message is sent to Default Exchange [N:10]
Message is sent to Default Exchange [N:11]
Message is sent to Default Exchange [N:12]
Message is sent to Default Exchange [N:13]
Message is sent to Default Exchange [N:14]
Message is sent to Default Exchange [N:15]
Message is sent to Default Exchange [N:16]
Message is sent to Default Exchange [N:17]
Message is sent to Default Exchange [N:18]
Interrupted
PS D:\inform_systems\лабки\lab_4\defaultExchange>
```

Мал 1.1 – скриншот тестування роботи Publisher’a

```
PS D:\inform_systems\лабки\lab_4\defaultExchange> python consumer.py
Subscribed to the queue 'hugs_dispatcher'
Received message: Number of people needing hugs since the app run: 0, total number of messages: 1
Received message: Number of people needing hugs since the app run: 1, total number of messages: 2
Received message: Number of people needing hugs since the app run: 2, total number of messages: 3
Received message: Number of people needing hugs since the app run: 3, total number of messages: 4
Received message: Number of people needing hugs since the app run: 4, total number of messages: 5
Received message: Number of people needing hugs since the app run: 5, total number of messages: 6
Received message: Number of people needing hugs since the app run: 6, total number of messages: 7
Received message: Number of people needing hugs since the app run: 7, total number of messages: 8
Received message: Number of people needing hugs since the app run: 8, total number of messages: 9
Received message: Number of people needing hugs since the app run: 9, total number of messages: 10
Received message: Number of people needing hugs since the app run: 10, total number of messages: 11
Received message: Number of people needing hugs since the app run: 11, total number of messages: 12
Received message: Number of people needing hugs since the app run: 12, total number of messages: 13
Received message: Number of people needing hugs since the app run: 13, total number of messages: 14
Received message: Number of people needing hugs since the app run: 14, total number of messages: 15
Received message: Number of people needing hugs since the app run: 15, total number of messages: 16
Received message: Number of people needing hugs since the app run: 16, total number of messages: 17
Received message: Number of people needing hugs since the app run: 17, total number of messages: 18
Interrupted
PS D:\inform_systems\лабки\lab_4\defaultExchange>
```

Мал 1.2 – скриншот тестування роботи Consumer’a

Зміни:

1. Повідомлення Publisher’a змінено з

```
“Message from publisher N: [counter]”
```

На

```
“Number of people needing hugs since the app run: [counter]”
```

2. Назва черги перйменована з «dev-queue» на «hugs_dispatcher»
3. Інтервал затримки генерування повідомлення змінено з [1, 3] до [2, 5] секунд
4. Додано до Consumer’a змінну counter, що у функції callback збільшується й демонструємо в стандартному потоці (command line), при отриманні повідомлення від Publisher’a

Завдання 2:

1. Відповідно до варіанту створити необхідні обмінники (exchange), черги (queue) та зв'язки (binding).
   2. Навести код Consumer'ів та Publisher'а, який дозволяє перевірити правильність налаштувань.
   3. Навести скрін з 15-20 повідомленнями, відправленими Publisher`ом та відповідних отриманих повідомлень Consumer'ів.

Варіант 3:

Створіть прямий обмін із трьома прив’язаними до нього чергами.
Перша черга повинна отримувати повідомлення з ключем маршрутизації "high-priority", друга черга повинна отримувати повідомлення з ключем

маршрутизації "medium-priority", третя черга повинна отримувати повідомлення з ключем маршрутизації "low-priority".

Виконання:

```
PS D:\inform_systems\лабки\lab_4\directExchange> python publisher.py
        [high-priority] message is sent to direct exchange: [N:1]
        [low-priority] message is sent to direct exchange: [N:1]
        [medium-priority] message is sent to direct exchange: [N:1]
        [high-priority] message is sent to direct exchange: [N:2]
        [low-priority] message is sent to direct exchange: [N:2]
        [low-priority] message is sent to direct exchange: [N:3]
        [high-priority] message is sent to direct exchange: [N:3]
        [medium-priority] message is sent to direct exchange: [N:2]
        [low-priority] message is sent to direct exchange: [N:4]
        [high-priority] message is sent to direct exchange: [N:4]
        [low-priority] message is sent to direct exchange: [N:5]
        [high-priority] message is sent to direct exchange: [N:5]
        [medium-priority] message is sent to direct exchange: [N:3]
        [high-priority] message is sent to direct exchange: [N:6]
        [low-priority] message is sent to direct exchange: [N:6]
        [low-priority] message is sent to direct exchange: [N:7]
        [high-priority] message is sent to direct exchange: [N:7]
        [medium-priority] message is sent to direct exchange: [N:4]
        [low-priority] message is sent to direct exchange: [N:8]
        [high-priority] message is sent to direct exchange: [N:8]
        [high-priority] message is sent to direct exchange: [N:9]
        [low-priority] message is sent to direct exchange: [N:9]
        [medium-priority] message is sent to direct exchange: [N:5]
        [high-priority] message is sent to direct exchange: [N:10]
        [low-priority] message is sent to direct exchange: [N:10]
        [high-priority] message is sent to direct exchange: [N:11]
        [low-priority] message is sent to direct exchange: [N:11]
        [medium-priority] message is sent to direct exchange: [N:6]
Interrupted
```

Мал. 2.1 – скриншот тестування Publisher’a

```
PS D:\inform_systems\лабки\lab_4\directExchange> python low_priority_consumer.py
Waiting for [low-priority] messages...
Received message: [Number of [low-priority] messages: 1]
Received message: [Number of [low-priority] messages: 2]
Received message: [Number of [low-priority] messages: 3]
Received message: [Number of [low-priority] messages: 4]
Received message: [Number of [low-priority] messages: 5]
Received message: [Number of [low-priority] messages: 6]
Received message: [Number of [low-priority] messages: 7]
Received message: [Number of [low-priority] messages: 8]
Received message: [Number of [low-priority] messages: 9]
Received message: [Number of [low-priority] messages: 10]
Received message: [Number of [low-priority] messages: 11]
```

Мал. 2.2 – скриншот тестування Consumer’а черги low-priority

```
PS D:\inform_systems\лабки\lab_4\directExchange> python medium_priority_consumer.py
Waiting for [medium-priority] messages...
Received message: [Number of [medium-priority] messages: 1]
Received message: [Number of [medium-priority] messages: 2]
Received message: [Number of [medium-priority] messages: 3]
Received message: [Number of [medium-priority] messages: 4]
Received message: [Number of [medium-priority] messages: 5]
Received message: [Number of [medium-priority] messages: 6]
```

Мал. 2.3 – скриншот тестування Consumer’а черги medium-priority

```
PS D:\inform_systems\лабки\lab_4\directExchange> python high_priority_consumer.py
Waiting for [high-priority] messages...
Received message: [Number of [high-priority] messages: 1]
Received message: [Number of [high-priority] messages: 2]
Received message: [Number of [high-priority] messages: 3]
Received message: [Number of [high-priority] messages: 4]
Received message: [Number of [high-priority] messages: 5]
Received message: [Number of [high-priority] messages: 6]
Received message: [Number of [high-priority] messages: 7]
Received message: [Number of [high-priority] messages: 8]
Received message: [Number of [high-priority] messages: 9]
Received message: [Number of [high-priority] messages: 10]
Received message: [Number of [high-priority] messages: 11]
```

Мал. 2.4 – скриншот тестування Consumer’а черги high-priority

Завдання 3:

1. Відповідно до варіанту створити необхідні обмінники (exchange), черги (queue) та зв'язки (binding).
   2. Навести код Consumer'ів та Publisher'а, який дозволяє перевірити правильність налаштувань.
   3. Навести скрін з 15-20 повідомленнями, відправленими Publisher`ом та відповідних отриманих повідомлень Consumer'ів.

Варіант 3:

Створіть обмінник для програми обміну повідомленнями, де ключі маршрутизації мають формат «message.<recipient_id>.<message_type>», де recipient_id — це унікальний ідентифікатор користувача, який є одержувачем повідомлення, а message_type може бути текстом , зображення чи відео. Створіть окремі черги, які підписуються на обмінник для кожного одержувача, щоб отримувати відповідні повідомлення.

Виконання:

Нехай є 3 Consumer’и:

user_1

user_2

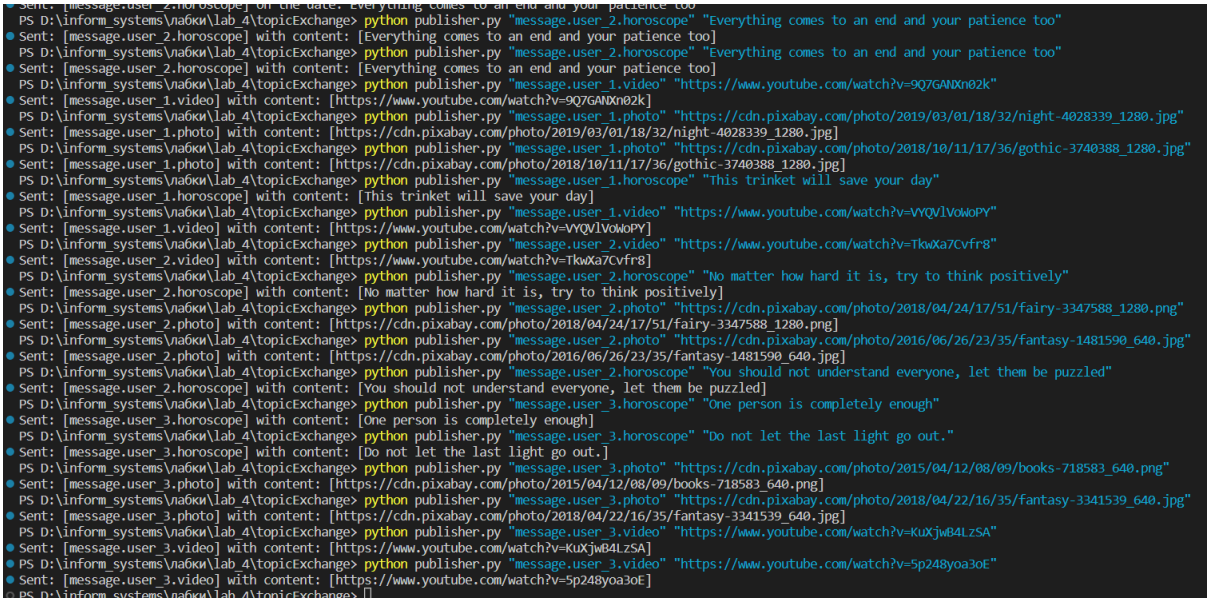
user_3

Є 3 типи повідомлень:

horoscope – текстове

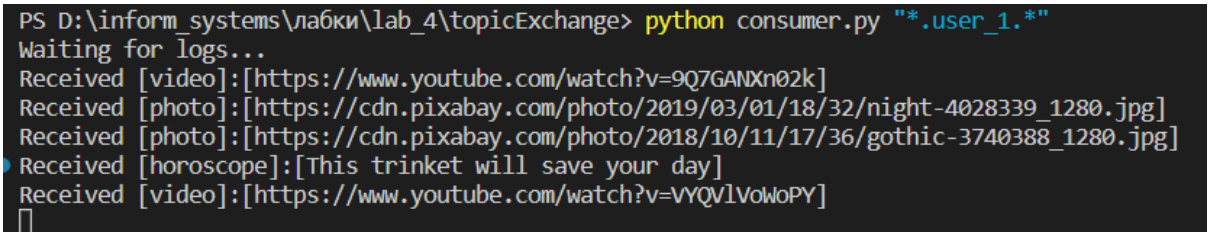
photo – гіперпосилання на фото

Video - uіперпосилання на відео



Мал. 3.1 – скриншот тестування Publisher’а,

надсилання різним consumer’ам повідомлень



Мал. 3.2 – скриншот тестування Consmer’а, що підписаний на чергу user_1

```
PS D:\inform_systems\лабки\lab_4\topicExchange> python consumer.py "*.user_2.*"
Waiting for logs...
Received [horoscope]:[Everything comes to an end and your patience too]
Received [video]:[https://www.youtube.com/watch?v=TkwXa7Cvfr8]
Received [horoscope]:[No matter how hard it is, try to think positively]
Received [photo]:[https://cdn.pixabay.com/photo/2018/04/24/17/51/fairy-3347588_1280.png]
Received [photo]:[https://cdn.pixabay.com/photo/2016/06/26/23/35/fantasy-1481590_640.jpg]
Received [horoscope]:[You should not understand everyone, let them be puzzled]
```

Мал. 3.2 – скриншот тестування Consmer'а, що підписаний на чергу user_2

```
PS D:\inform_systems\лабки\lab_4\topicExchange> python consumer.py "*.user_3.*"
Waiting for logs...
Received [horoscope]:[One person is completely enough]
Received [horoscope]:[Do not let the last light go out.]
Received [photo]:[https://cdn.pixabay.com/photo/2015/04/12/08/09/books-718583_640.png]
Received [photo]:[https://cdn.pixabay.com/photo/2018/04/22/16/35/fantasy-3341539_640.jpg]
Received [video]:[https://www.youtube.com/watch?v=KuXjwB4LzSA]
Received [video]:[https://www.youtube.com/watch?v=5p248yoa3oE]
```

Мал. 3.2 – скриншот тестування Consmer'а, що підписаний на чергу user_3

Висновки

Реалізовано 3 типи exchange подій у брокері повідомлень RabbitMQ: default (1 завдання), direct (2 завдання), topic (3 завдання). Скрипти publisher’ів та Consumer’ів реалізовано мовою python, із застосуванням бібліотеки-клієнта pika. Програми елементарні, у формі консольних застосунків.

Додаток А. Скрипти програм

Завдання 1.Publisher

```
import pika
from random import randint
from time import sleep
import os
import sys

IP_ADDRESS = "localhost"
QUEUE = "hugs_dispatcher"
MIN_TIME_TO_SLEEP = 2
MAX_TIME_TO_SLEEP = 5

def main():
    counter = 0
    while (True):
        time_to_sleep = randint(MIN_TIME_TO_SLEEP, MAX_TIME_TO_SLEEP)
        sleep(time_to_sleep)

        # set up connection to broker
        connection = pika.BlockingConnection(pika.ConnectionParameters(IP_ADDRESS))
        # set up channel
        channel = connection.channel()
        # set up queue
        channel.queue_declare(queue=QUEUE,
                              durable=False,
                              exclusive=False,
                              auto_delete=False)

        message = f"Number of people needing hugs since the app run: {counter}"
        counter += 1
        body = bytes(message, encoding="utf-8")

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

        print(f"Message is sent to Default Exchange [N:{counter}]")


if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemError:
            os._exit(0)
```

Завдання 1.Consumer

```
import pika
import sys
import os

IP_ADDRESS = "localhost"
QUEUE = "hugs_dispatcher"
MIN_TIME_TO_SLEEP = 2
MAX_TIME_TO_SLEEP = 5

def main():
    global counter

    # set up connection to broker
    connection = pika.BlockingConnection(pika.ConnectionParameters(IP_ADDRESS))
    # set up channel
    channel = connection.channel()
    # set up queue
    channel.queue_declare(queue=QUEUE,
                          durable=False,
                          exclusive=False,
                          auto_delete=False)

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

        message = body.decode("utf-8")
        counter += 1
```

```
        print(f"Received message: {message}, total number of messages: {counter}")

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

    print(f"Subscribed to the queue '{QUEUE}'")
    channel.start_consuming()


if __name__ == "__main__":
    counter = 0
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemError:
            os._exit(0)
```

Завдання 2.Publisher

```
import pika
import sys
import os
from time import sleep
from random import randint
import threading

# priorities:
#
# high-priority
# medium-priority
# low-priority

IP_ADDRESS = "localhost"
EXCHANGE = "direct_logs"
EXCHANGE_TYPE = "direct"
MIN_TIME_TO_WAIT = 1
priorities = {"high-priority": 2,
              "medium-priority": 4,
              "low-priority": 6}
stop_flag = False

def main(priority_delay, priority_name):
    counter = 0
    while not stop_flag:
        sleep(priority_delay)

        connection = pika.BlockingConnection(pika.ConnectionParameters(host=IP_ADDRESS))
        channel = connection.channel()

        channel.exchange_declare(exchange=EXCHANGE,
                                 exchange_type=EXCHANGE_TYPE)

        counter += 1
        message = f"Number of [{priority_name}] messages: {counter}"

        body = bytes(message, encoding="utf-8")

        channel.basic_publish(exchange=EXCHANGE,
                              routing_key=priority_name,
                              body=body)

        print(f"\t [{priority_name}] message is sent to direct exchange: [N:{counter}]")


if __name__ == "__main__":
    threads = []
    try:
        for priority, max_time_to_wait in priorities.items():
            time_to_wait = randint(MIN_TIME_TO_WAIT, max_time_to_wait)
            threads.append(threading.Thread(target=main, args=(time_to_wait, priority)))

        [t.start() for t in threads]

        # Wait for the user to press Ctrl+C
        while True:
            sleep(1)
    except KeyboardInterrupt:
        stop_flag = True
        [t.join() for t in threads]
        print('Interrupted')
        try:
```

```
            sys.exit(0)
        except SystemError:
            os._exit(0)
```

Завдання 2.Consumer

Наведено приклад consumer’а, що підписаний на чергу з ключем-маршрутизатором “high-priority”, але скрипти для інших черг аналогічні, з відмінністю значення змінної priority_name.

```
import pika
import sys
import os

IP_ADDRESS = "localhost"
EXCHANGE = "direct_logs"
EXCHANGE_TYPE = "direct"
priority_name = "high-priority"

def main():
    connection = pika.BlockingConnection(pika.ConnectionParameters(host=IP_ADDRESS))
    channel = connection.channel()

    channel.exchange_declare(exchange=EXCHANGE,
                             exchange_type=EXCHANGE_TYPE)



    result = channel.queue_declare(queue="", exclusive=True)
    queue_name = result.method.queue
    channel.queue_bind(queue=queue_name,
                       exchange=EXCHANGE,
                       routing_key=priority_name)

    def callback(ch, method, properties, body):
        message = body.decode("utf-8")
        print(f"Received message: [{message}]")

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

    print(f"Waiting for [{priority_name}] messages...")

    channel.start_consuming()



if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print("Interrupted")
        try:
            sys.exit(0)
        except SystemError:
            os._exit(0)
```

Завдання 3.Publisher

```
import pika
import sys
from datetime import date

IP_ADDRESS = "localhost"
EXCHANGE = "topic_logs"
EXCHANGE_TYPE = "topic"

def main():
    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host=IP_ADDRESS))
    channel = connection.channel()

    channel.exchange_declare(exchange=EXCHANGE, exchange_type=EXCHANGE_TYPE)

    routing_key = sys.argv[1] if len(sys.argv) > 1 else "anonimous.info"
```

```
    message = " ".join(sys.argv[2:]) if len(sys.argv) > 2 else "No messages yet, take a hug instead!"
    body = bytes(message, encoding="utf-8")

    channel.basic_publish(exchange=EXCHANGE,
                          routing_key=routing_key,
                          body=body)

    print(f"Sent: [{routing_key}] with content: [{message}]")

    channel.close()

if __name__ == "__main__":
    main()
```

Завдання 3. Consumer

```
import pika
import sys
import os

IP_ADDRESS = "localhost"
EXCHANGE = "topic_logs"
EXCHANGE_TYPE = "topic"

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

    channel.exchange_declare(exchange=EXCHANGE, exchange_type=EXCHANGE_TYPE)

    result = channel.queue_declare(queue="", exclusive=True)
    queue_name = result.method.queue

    binding_keys = sys.argv[1:]

    for b_k in binding_keys:
        channel.queue_bind(queue=queue_name,
                           exchange=EXCHANGE,
                           routing_key=b_k)

    print("Waiting for logs...")

    def callback(ch, method, properties, body):
        message_type = method.routing_key.split(".")[-1]
        body = body.decode("utf-8")
        print(f"Received [{message_type}]:[{body}]")

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

    channel.start_consuming()

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemError:
            os._exit(0)
```