

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 з принципом роботи брокера повідомлень”

Виконав:

студент групи КМ-12

Прісич К.С.

Керівник:

Асистент

Жук І. С.

ЗМІСТ

МЕТА РОБОТИ	3
ЗАВДАННЯ 1	3
ЗАВДАННЯ 2	5
ЗАВДАННЯ 3	8
ДОДАТОК А	11

МЕТА РОБОТИ

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

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.

Python код `Producer`a` та `Consumer`a` знаходиться в ДОДАТКУ А

Скріншот роботи Producer`а, його повідомлень (запуск у окремому інтерактивному вікні/терміналі):

```
Interactive - consumer.py  Interactive - producer.py X
Interrupt | X Clear All  Restart  Variables  Save  Export  ...  Python 3.9.7

Connected to Python 3.9.7

✓ import pika ...

...  Надіслано повідомлення 1: Практика з RabbitMQ
    Надіслано повідомлення 2: Виконання лабораторної №4
    Надіслано повідомлення 3: Сьогодні хмарна погода
    Надіслано повідомлення 4: Виконання лабораторної №4
    Надіслано повідомлення 5: Виконання лабораторної №4
    Надіслано повідомлення 6: Практика з RabbitMQ
    Надіслано повідомлення 7: Практика з RabbitMQ
    Надіслано повідомлення 8: Практика з RabbitMQ
    Надіслано повідомлення 9: Налаштування брокерів повідомлень
    Надіслано повідомлення 10: Сьогодні хмарна погода
    Надіслано повідомлення 11: Сьогодні хмарна погода
    Надіслано повідомлення 12: Налаштування брокерів повідомлень
    Надіслано повідомлення 13: Налаштування брокерів повідомлень
    Надіслано повідомлення 14: Виконання лабораторної №4
    Надіслано повідомлення 15: Виконання лабораторної №4
    Надіслано повідомлення 16: Виконання лабораторної №4
    Надіслано повідомлення 17: Сьогодні хмарна погода
    Надіслано повідомлення 18: Практика з RabbitMQ
    Надіслано повідомлення 19: Виконання лабораторної №4
    Надіслано повідомлення 20: Сьогодні хмарна погода
```

Скріншот роботи Consumer'a, його повідомлень (запуск у окремому інтерактивному вікні/терміналі):

```

Interactive - consumer.py x Interactive - producer.py
Interrupt | X Clear All Restart Variables Save Export ... Python 3.9.7

Connected to Python 3.9.7

✓ import pika ...

... Очікування повідомлень. Натисніть Ctrl+C для виходу
Отримано повідомлення 1: Практика з RabbitMQ
Отримано повідомлення 2: Виконання лабораторної №4
Отримано повідомлення 3: Сьогодні хмарна погода
Отримано повідомлення 4: Виконання лабораторної №4
Отримано повідомлення 5: Виконання лабораторної №4
Отримано повідомлення 6: Практика з RabbitMQ
Отримано повідомлення 7: Практика з RabbitMQ
Отримано повідомлення 8: Практика з RabbitMQ
Отримано повідомлення 9: Налаштування брокерів повідомлень
Отримано повідомлення 10: Сьогодні хмарна погода
Отримано повідомлення 11: Сьогодні хмарна погода
Отримано повідомлення 12: Налаштування брокерів повідомлень
Отримано повідомлення 13: Налаштування брокерів повідомлень
Отримано повідомлення 14: Виконання лабораторної №4
Отримано повідомлення 15: Виконання лабораторної №4
Отримано повідомлення 16: Виконання лабораторної №4
Отримано повідомлення 17: Сьогодні хмарна погода
Отримано повідомлення 18: Практика з RabbitMQ
Отримано повідомлення 19: Виконання лабораторної №4
Отримано повідомлення 20: Сьогодні хмарна погода

Загальна кількість отриманих повідомлень: 20

```

ЗАВДАННЯ 2

1. Відповідно до варіанту створити необхідні обмінники (exchange), черги (queue) та зв'язки (binding).
2. Навести код Consumer'ів та Publisher'a, який дозволяє перевірити правильність налаштувань.

3. Навести скрін з 15-20 повідомленнями, відправленими Publisher`ом та відповідних отриманих повідомлень Consumer`ів.

Варіант 20. Створіть прямий обмін із чергою на кожен день тижня (Monday, Tuesday, Wednesday тощо). Прив'яжіть кожен чергу до обмінника з відповідним днем тижня як ключ маршрутизації.

Скріншот роботи Publisher`а, його повідомлень (запуск у окремому інтерактивному вікні/терміналі):

- текст повідомлення обирається випадково з-поміж 4 можливих варіантів
- день тижня також обирається випадково

```
Interactive - task2_consumer.py  Interactive - task2_publisher.py × ▶ □ ...
□ Interrupt | × Clear All ↺ Restart 📄 Variables 📄 Save ... 📄 Python 3.9.7
Connected to Python 3.9.7
✓ import pika ...
...  Надіслано повідомлення для Thursday: Звіт за Thursday
    Надіслано повідомлення для Monday: Огляд подій за Monday
    Надіслано повідомлення для Monday: Огляд подій за Monday
    Надіслано повідомлення для Wednesday: Плани на Wednesday
    Надіслано повідомлення для Wednesday: Звіт за Wednesday
    Надіслано повідомлення для Tuesday: Огляд подій за Tuesday
    Надіслано повідомлення для Thursday: Огляд подій за Thursday
    Надіслано повідомлення для Friday: Звіт за Friday
    Надіслано повідомлення для Wednesday: Плани на Wednesday
    Надіслано повідомлення для Wednesday: Огляд подій за Wednesday
    Надіслано повідомлення для Thursday: Плани на Thursday
    Надіслано повідомлення для Tuesday: Звіт за Tuesday
    Надіслано повідомлення для Wednesday: Завдання на Wednesday
    Надіслано повідомлення для Monday: Огляд подій за Monday
    Надіслано повідомлення для Monday: Завдання на Monday
    Надіслано повідомлення для Monday: Плани на Monday
    Надіслано повідомлення для Monday: Звіт за Monday
    Надіслано повідомлення для Thursday: Завдання на Thursday
    Надіслано повідомлення для Sunday: Огляд подій за Sunday
    Надіслано повідомлення для Wednesday: Завдання на Wednesday
```

Скріншот роботи Consumer`а, його повідомлень (запуск у окремому інтерактивному вікні/терміналі):

- відповідний Consumer отримує лише своє повідомлення (за ключем, що є днем тижня)

Connected to Python 3.9.7

✓ `import pika ...`

```
... Consumers запущено. Очікування повідомлень. Ctrl+C для виходу
[Thursday Consumer] Отримано повідомлення: Звіт за Thursday
[Monday Consumer] Отримано повідомлення: Огляд подій за Monday
[Monday Consumer] Отримано повідомлення: Огляд подій за Monday
[Wednesday Consumer] Отримано повідомлення: Плани на Wednesday
[Wednesday Consumer] Отримано повідомлення: Звіт за Wednesday
[Tuesday Consumer] Отримано повідомлення: Огляд подій за Tuesday
[Thursday Consumer] Отримано повідомлення: Огляд подій за Thursday
[Friday Consumer] Отримано повідомлення: Звіт за Friday
[Wednesday Consumer] Отримано повідомлення: Плани на Wednesday
[Wednesday Consumer] Отримано повідомлення: Огляд подій за Wednesday
[Thursday Consumer] Отримано повідомлення: Плани на Thursday
[Tuesday Consumer] Отримано повідомлення: Звіт за Tuesday
[Wednesday Consumer] Отримано повідомлення: Завдання на Wednesday
[Monday Consumer] Отримано повідомлення: Огляд подій за Monday
[Monday Consumer] Отримано повідомлення: Завдання на Monday
[Monday Consumer] Отримано повідомлення: Плани на Monday
[Monday Consumer] Отримано повідомлення: Звіт за Monday
[Thursday Consumer] Отримано повідомлення: Завдання на Thursday
[Sunday Consumer] Отримано повідомлення: Огляд подій за Sunday
[Wednesday Consumer] Отримано повідомлення: Завдання на Wednesday
```

Python код Publisher`а та Consumer`ів знаходиться в ДОДАТКУ А

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 20. Створіть обмінник для платформи соціальних медіа, де ключі маршрутизації мають формат «post.<user_id>. <action> », де user_id — це унікальний ідентифікатор користувача, а action - create, update чи delete.

Створити окремі черги, які підписуються на обмінник для кожного користувача, щоб відстежувати його активність.

Скріншот роботи Publisher`а, його повідомлень (запуск у окремому інтерактивному вікні/терміналі):

- дія користувача обирається випадково з-поміж 3 можливих варіантів (create, update, delete)

```

Interactive - task3_publisher.py
Interrupt | Clear All Restart Variables Save Export ...

Connected to Python 3.9.7

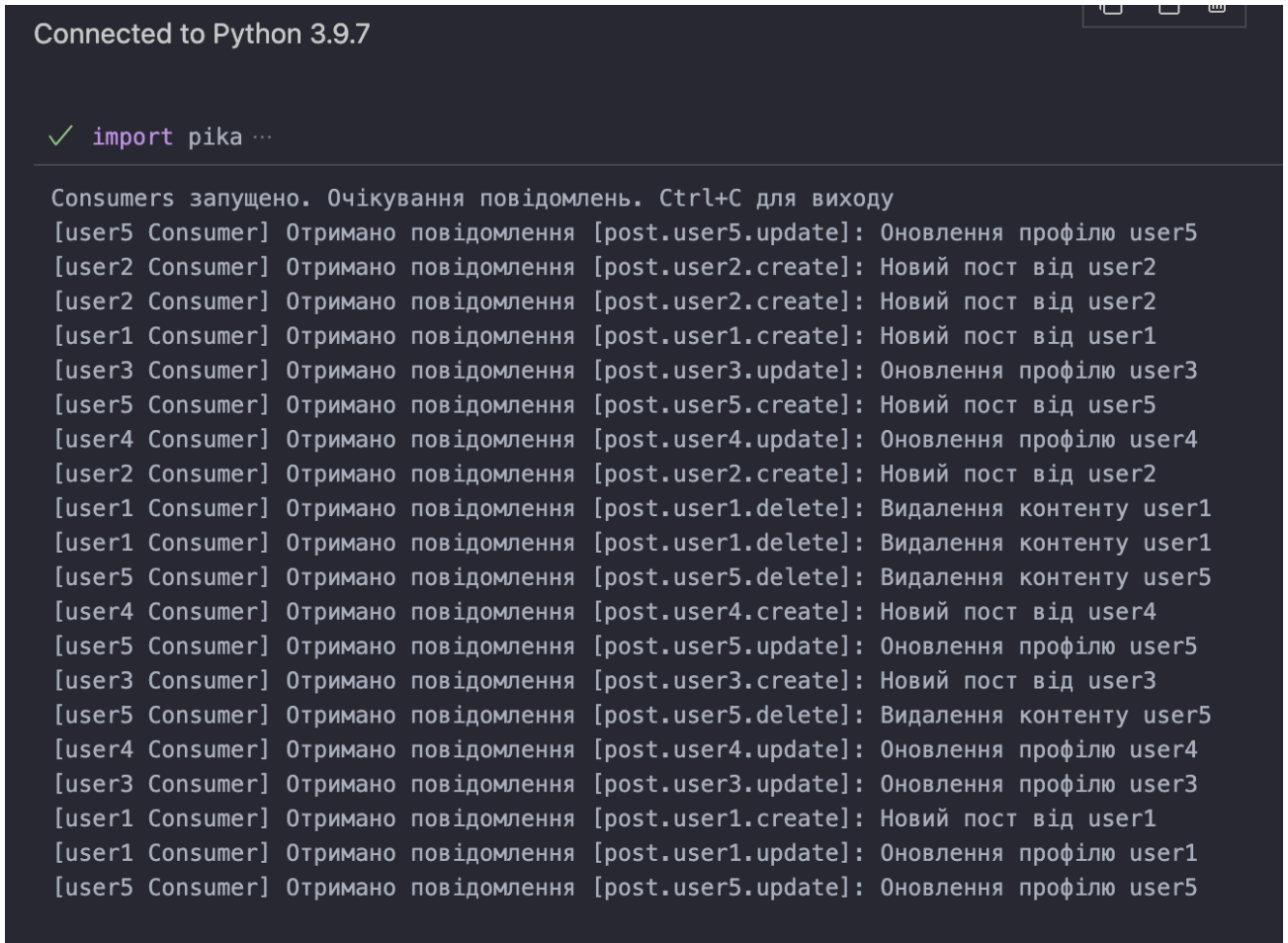
✓ import pika ...

...
Надіслано повідомлення: post.user5.update – Оновлення профілю user5
Надіслано повідомлення: post.user2.create – Новий пост від user2
Надіслано повідомлення: post.user2.create – Новий пост від user2
Надіслано повідомлення: post.user1.create – Новий пост від user1
Надіслано повідомлення: post.user3.update – Оновлення профілю user3
Надіслано повідомлення: post.user5.create – Новий пост від user5
Надіслано повідомлення: post.user4.update – Оновлення профілю user4
Надіслано повідомлення: post.user2.create – Новий пост від user2
Надіслано повідомлення: post.user1.delete – Видалення контенту user1
Надіслано повідомлення: post.user1.delete – Видалення контенту user1
Надіслано повідомлення: post.user5.delete – Видалення контенту user5
Надіслано повідомлення: post.user4.create – Новий пост від user4
Надіслано повідомлення: post.user5.update – Оновлення профілю user5
Надіслано повідомлення: post.user3.create – Новий пост від user3
Надіслано повідомлення: post.user5.delete – Видалення контенту user5
Надіслано повідомлення: post.user4.update – Оновлення профілю user4
Надіслано повідомлення: post.user3.update – Оновлення профілю user3
Надіслано повідомлення: post.user1.create – Новий пост від user1
Надіслано повідомлення: post.user1.update – Оновлення профілю user1
Надіслано повідомлення: post.user5.update – Оновлення профілю user5

```

Скріншот роботи Consumer`а, його повідомлень (запуск у окремому інтерактивному вікні/терміналі):

- відповідний Consumer отримує лише повідомлення для одного з користувачів (за ключем "post.{user_id}.{action}")



```

Connected to Python 3.9.7

✓ import pika ...

Consumers запущено. Очікування повідомлень. Ctrl+C для виходу
[user5 Consumer] Отримано повідомлення [post.user5.update]: Оновлення профілю user5
[user2 Consumer] Отримано повідомлення [post.user2.create]: Новий пост від user2
[user2 Consumer] Отримано повідомлення [post.user2.create]: Новий пост від user2
[user1 Consumer] Отримано повідомлення [post.user1.create]: Новий пост від user1
[user3 Consumer] Отримано повідомлення [post.user3.update]: Оновлення профілю user3
[user5 Consumer] Отримано повідомлення [post.user5.create]: Новий пост від user5
[user4 Consumer] Отримано повідомлення [post.user4.update]: Оновлення профілю user4
[user2 Consumer] Отримано повідомлення [post.user2.create]: Новий пост від user2
[user1 Consumer] Отримано повідомлення [post.user1.delete]: Видалення контенту user1
[user1 Consumer] Отримано повідомлення [post.user1.delete]: Видалення контенту user1
[user5 Consumer] Отримано повідомлення [post.user5.delete]: Видалення контенту user5
[user4 Consumer] Отримано повідомлення [post.user4.create]: Новий пост від user4
[user5 Consumer] Отримано повідомлення [post.user5.update]: Оновлення профілю user5
[user3 Consumer] Отримано повідомлення [post.user3.create]: Новий пост від user3
[user5 Consumer] Отримано повідомлення [post.user5.delete]: Видалення контенту user5
[user4 Consumer] Отримано повідомлення [post.user4.update]: Оновлення профілю user4
[user3 Consumer] Отримано повідомлення [post.user3.update]: Оновлення профілю user3
[user1 Consumer] Отримано повідомлення [post.user1.create]: Новий пост від user1
[user1 Consumer] Отримано повідомлення [post.user1.update]: Оновлення профілю user1
[user5 Consumer] Отримано повідомлення [post.user5.update]: Оновлення профілю user5
  
```

Python код Publisher`а та Consumer`ів знаходиться в ДОДАТКУ А

ДОДАТОК А

Python код для завдань №1, №2 та №3 можна завантажити на Github за посиланням:

https://github.com/Prisych-Kostya/RabbitMQ_Lab4

Завдання 1

producer.py

```
import pika
import time
import random

# Параметри підключення
connection_params = pika.ConnectionParameters('localhost')
connection = pika.BlockingConnection(connection_params)
channel = connection.channel()

# Оголошення черги
queue_name = 'messages'
channel.queue_declare(queue=queue_name)

def generate_message():
    messages = [
        "Сьогодні хмарна погода",
        "Практика з RabbitMQ",
        "Налаштування брокерів повідомлень",
        "Виконання лабораторної №4"
    ]

    return random.choice(messages)
```

```

# Надсилання повідомлень

try:

    for i in range(20):

        message = generate_message()

        channel.basic_publish(

            exchange='',

            routing_key=queue_name,

            body=message

        )

        print(f"Надіслано повідомлення {i+1}: {message}")

        time.sleep(1)  # Інтервал між повідомленнями 1 сек

finally:

    connection.close()

```

consumer.py

```

import pika

# Параметри підключення

connection_params = pika.ConnectionParameters('localhost')

connection = pika.BlockingConnection(connection_params)

channel = connection.channel()

# Оголошення черги (має збігатися з Producer)

queue_name = 'messages'

channel.queue_declare(queue=queue_name)

# Лічильник отриманих повідомлень

```

```

message_count = 0

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

    global message_count

    message_count += 1

    print(f"Отримано повідомлення {message_count}: {body.decode()}")

# Підписка на чергу
channel.basic_consume(

    queue=queue_name,

    on_message_callback=callback,

    auto_ack=True

)

print('Очікування повідомлень. Натисніть Ctrl+C для виходу')

try:

    channel.start_consuming()

except KeyboardInterrupt:

    print(f"\nЗагальна кількість отриманих повідомлень: {message_count}")

    channel.stop_consuming()

connection.close()

```

Завдання 2

task2_publisher.py

```

import pika

import time

import random

```

```

# Підключення до RabbitMQ

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

channel = connection.channel()

# Назва обмінника

EXCHANGE_NAME = 'days_of_week'

# Оголошення обмінника типу direct

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

# Дні тижня - ключі маршрутизації

days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# Функція для генерування (вибору) одного з декількох можливих повідомлення на день

def generate_message(day):

    messages = [

        f"Завдання на {day}",

        f"Плани на {day}",

        f"Огляд подій за {day}",

        f"Звіт за {day}",

    ]

    return random.choice(messages)

# Надсилання повідомлень

try:

    for _ in range(20): # 20 повідомлень

```

```

    day = random.choice(days)

    message = generate_message(day)

    # Надсилання повідомлення з конкретним ключем маршрутизації

    channel.basic_publish(

        exchange=EXCHANGE_NAME,

        routing_key=day,

        body=message

    )

    print(f"Надіслано повідомлення для {day}: {message}")

    time.sleep(1)

finally:

    connection.close()

```

task2_consumer.py

```

import pika

# Підключення до RabbitMQ

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

channel = connection.channel()

# Назва обмінника

EXCHANGE_NAME = 'days_of_week'

# Оголошення обмінника

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

```

```

# Дні тижня

days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

# Функція для створення consumer для конкретного дня

def create_consumer(day):

    # Створення черги для конкретного дня

    queue_name = f'{day}_queue'

    channel.queue_declare(queue=queue_name)

    # Прив'язка черги до обмінника з ключем - день тижня

    channel.queue_bind(

        exchange=EXCHANGE_NAME,

        queue=queue_name,

        routing_key=day

    )

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

        print(f"[{day} Consumer] Отримано повідомлення: {body.decode()}")

    # Підписка на чергу

    channel.basic_consume(

        queue=queue_name,

        on_message_callback=callback,

        auto_ack=True

    )

    return queue_name

```



```

# Створення consumer для кожного дня

consumers = [create_consumer(day) for day in days]

print('Consumers запущено. Очікування повідомлень. Ctrl+C для виходу')

try:

    channel.start_consuming()

except KeyboardInterrupt:

    channel.stop_consuming()

connection.close()

```

Завдання 3

task3_publisher.py

```

import pika

import random

import time

# Підключення до RabbitMQ

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

channel = connection.channel()

# Назва обмінника

EXCHANGE_NAME = 'social_media_events'

# Оголошення обмінника типу topic

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

```

```

# Можливі дії

actions = ['create', 'update', 'delete']

# Можливі user_id

user_ids = ['user1', 'user2', 'user3', 'user4', 'user5']

def generate_message(user_id, action):

    messages = {

        'create': f"Новий пост від {user_id}",

        'update': f"Оновлення профілю {user_id}",

        'delete': f"Видалення контенту {user_id}"

    }

    return messages[action]

# Надсилення повідомлень

try:

    for _ in range(20):

        # Випадковий user_id та дія

        user_id = random.choice(user_ids)

        action = random.choice(actions)

        # Формування ключа маршрутизації ТОЧНО за форматом post.<user_id>.<action>

        routing_key = f"post.{user_id}.{action}"

        message = generate_message(user_id, action)

        # Надсилення повідомлення

        channel.basic_publish(

            exchange=EXCHANGE_NAME,

```

```

        routing_key=routing_key,

        body=message

    )

    print(f"Надіслано повідомлення: {routing_key} - {message}")

    time.sleep(1)

finally:

    connection.close()

```

task3_consumers.py

```

import pika

# Підключення до RabbitMQ

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

channel = connection.channel()

# Назва обмінника

EXCHANGE_NAME = 'social_media_events'

# Оголошення обмінника

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

# Можливі user_id

user_ids = ['user1', 'user2', 'user3', 'user4', 'user5']

def create_consumer(user_id):

    # Створення черги для конкретного користувача

```

```

queue_name = f'{user_id}_queue'

channel.queue_declare(queue=queue_name)

# Прив'язка до обмінника з патерном для всіх подій користувача
# post.<user_id>.* - означає будь-які дії для цього користувача

channel.queue_bind(

    exchange=EXCHANGE_NAME,

    queue=queue_name,

    routing_key=f"post.{user_id}.*"

)

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

    print(f"[{user_id} Consumer] Отримано повідомлення [{method.routing_key}]: {body.decode()}")

# Підписка на чергу

channel.basic_consume(

    queue=queue_name,

    on_message_callback=callback,

    auto_ack=True

)

return queue_name

# Створення consumer для кожного користувача

consumers = [create_consumer(user_id) for user_id in user_ids]

print('Consumers запущено. Очікування повідомлень. Ctrl+C для виходу')

```

```
try:
    channel.start_consuming()
except KeyboardInterrupt:
    channel.stop_consuming()

connection.close()
```