

Lab02 - Asynchroniczna komunikacja między mikrousługami z użyciem systemu kolejkowego

1. Bazując na wynikach poprzedniego ćwiczenia lab. WTI uruchom predefiniowany kontener Docker serwera Redis (redis:latest) w trybie lokalnego trybu komunikacji kontenerów docker (--network host) w taki sposób, aby serwer Redis działał na porcie 6381.

```
sudo docker run --name redisque --network host --rm redis:latest --port 6381
1:C 10 Apr 2021 10:54:53.215 # o000o000o000o Redis is starting o000o000o000o
1:C 10 Apr 2021 10:54:53.215 # Redis version=6.2.1, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 10 Apr 2021 10:54:53.215 # Configuration loaded
1:M 10 Apr 2021 10:54:53.216 * monotonic clock: POSIX clock_gettime
1:M 10 Apr 2021 10:54:53.217 * Running mode=standalone, port=6381.
1:M 10 Apr 2021 10:54:53.217 # Server initialized
```

2. Oceń celowość użycia środowiska virtualenv przygotowanego podczas realizacji poprzedniego ćwiczenia lab. WTI.
ODP. Jest to świetne rozwiązanie ponieważ możemy mieć kilka różnych projektów i każdy z inną wersją pythona. Zwalnia nas to też ze zmiany wersji pythona zainstalowanej w systemie.
3. Bazując na wynikach poprzedniego ćwiczenia lab. WTI - mających postać importowalnego modułu/biblioteki Python - przygotuj implementacje dwóch aplikacji klienckich komunikujących się za pośrednictwem kolejki w taki sposób, aby jedna z aplikacji pełniła funkcję tzw. producenta dowolnych (najlepiej "zaślepkowych") wiadomości w formacie JSON (`rpush(queue_to_write_to, json.dumps(message_content))`) a druga pełniła funkcję konsumenta wiadomości (drukującego w konsoli zawartość kolejnej odebranej z użyciem metody `lrange` wiadomości zdekodowanej z użyciem `json.loads(value_read_from_queue)` oraz funkcje inicjalizacji i/lub opróżniania kolejki. Wprowadź odstęp czasowy 10 ms między wysłaniem dwóch kolejnych wiadomości (`time.sleep(0.01)`). Użyj takich metod klasy `StrictRedis` jak `rpush`, `lrange`, `ltrim`, `flushdb`.

```
from wtiproj01_queue import Queue
import time
import datetime

class Consumer:
    def __init__(self):
        self.que = Queue()

    def list(self):
        self.que.list()
```

```

from wtiproj01_queue import Queue

class Producer:
    def __init__(self):
        self.que = Queue()

    def push(self, message):
        message_to_write_to_queue_as_dict = message
        self.que.push(message_to_write_to_queue_as_dict)

if __name__ == "__main__":

    data = pd.read_csv('./userRatedMovies.dat',
                       nrows=100, delimiter="\t")

    row_iterator = data.iterrows()
    producer = Producer()
    diagnostic_row_index = 99999

    for row in row_iterator:
        row_as_dict = row[1].to_dict()
        row_as_dict["diagnostic_index"] = diagnostic_row_index
        producer.push(row_as_dict)
        diagnostic_row_index += 1
        time.sleep(0.25)

```

```

from wtiproj01_queue import Queue
from wtiproj02_consumer import Consumer
from wtiproj02_producer import Producer
import time

producer = Producer()
consumer = Consumer()

que = Queue()
que.pull()

producer.push("message1")
time.sleep(0.10)
producer.push("message2")
consumer.list()

```

```

wtivenv01 > python ./wtiproj02.py
value: message1
value: message2

```

4. Zmodyfikuj kod aplikacji wtiproj02_producer w taki sposób, aby wysyłała ona (w formacie JSON) kolejne wiersze tabeli (typu dataframe) biblioteki Pandas w tempie około 100 wierszy na sekundę. Jako źródło danych tabeli użyj przynajmniej stu wierszy pliku userRatedMovies.dat z (dostępnego w Internecie) zbioru hetrec2011-movieLens-2k-v2. Zastosuj tzw. generator iteracji (df.iterrows()).

```
def zad4():
    producer = Producer()
    consumer = Consumer()

    que = Queue()
    que.pull()

    data = pd.read_csv('./userRatedMovies.dat', nrows=100, delimiter="\t")
    for index, row in data.iterrows():
        producer.push(row[0])

    consumer.list()

zad4()
```

5. Zmodyfikuj kod aplikacji wtiproj02_consumer w taki sposób, aby przez około 10 s odbierała ona wiadomości (wszystkie wiadomości z kolejki - zawierającej wiadomości wysłane przez aplikację wtiproj02_producer) w tempie około 4 Hz.

- wtiproj02_consumer_v2.py

```
from wtiproj01_queue import Queue
import time
import datetime

class Consumer:
    def __init__(self):
        self.que = Queue()

    def list(self):
        self.que.list()

if __name__ == "__main__":
    consumer = Consumer()

    timeout = time.time() + 10

    while True:
        print(datetime.datetime.fromtimestamp(
            time.time()).strftime('%Y-%m-%d %H:%M:%S'))
        consumer.list()
        time.sleep(0.4)
        if time.time() > timeout:
            break
```

```
def zad5():
    consumer = Consumer()
    producer = Producer()

    row_iterator = data.iterrows()

    for row in row_iterator:
        row_as_dict = row[1].to_dict()
        producer.push(row_as_dict)
        time.sleep(0.25)
        consumer.async_list()
```

6. Uruchom, przetestuj i zademonstruj działanie całego dotąd przygotowanego systemu w dwóch konsolach/terminalach. Zidentyfikuj różnice w wyniku działania konsumenta w zależności od częstotliwości wysyłania komunikatów przez producenta

```
th': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 29.0, 'date_second': 16.0, 'diagnostic_index': 100035}
1 {'userID': 75.0, 'movieID': 3889.0, 'rating': 3.0, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 22.0, 'date_second': 43.0, 'diagnostic_index': 100036}
2021-04-24 14:42:57
0 {'userID': 75.0, 'movieID': 3994.0, 'rating': 3.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 25.0, 'date_second': 24.0, 'diagnostic_index': 100037}
2021-04-24 14:42:57
0 {'userID': 75.0, 'movieID': 5107.0, 'rating': 3.0, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 26.0, 'date_second': 11.0, 'diagnostic_index': 100039}
2021-04-24 14:42:57
0 {'userID': 75.0, 'movieID': 5833.0, 'rating': 2.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 29.0, 'date_second': 51.0, 'diagnostic_index': 100040}
1 {'userID': 75.0, 'movieID': 5952.0, 'rating': 3.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 30.0, 'date_second': 40.0, 'diagnostic_index': 100041}
2021-04-24 14:42:55
2021-04-24 14:42:56
2021-04-24 14:42:56
2021-04-24 14:42:57
0 {'userID': 75.0, 'movieID': 4993.0, 'rating': 3.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 30.0, 'date_second': 34.0, 'diagnostic_index': 100038}
2021-04-24 14:42:57
2021-04-24 14:42:57
2021-04-24 14:42:58
0 {'userID': 75.0, 'movieID': 6213.0, 'rating': 4.0, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 26.0, 'date_second': 21.0, 'diagnostic_index': 100042}
2021-04-24 14:42:58
0 {'userID': 75.0, 'movieID': 6225.0, 'rating': 0.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 21.0, 'date_second': 25.0, 'diagnostic_index': 100043}
1 {'userID': 75.0, 'movieID': 6333.0, 'rating': 4.0, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 28.0, 'date_second': 12.0, 'diagnostic_index': 100044}
```

7. Uruchom i przetestuj system o wielu (przynajmniej dwóch) instancjach mikrousługi będącej producentem wiadomości i pojedynczej instancji konsumenta wiadomości odczytującego zawartość kolejki w tempie jej zapełniania przez producenta (w przypadku obu aplikacji wiadomy wiersz kodu: "time.sleep(1.0/4)"). Wskaż naturalne różnice w efekcie działania aplikacji odbiorczej:

- w przypadku o wielu instancjach mikrousługi będącej producentem wiadomości i pojedynczej instancji konsumenta wiadomości (względem przypadku o pojedynczej instancji producenta wiadomości i pojedynczej instancji konsumenta wiadomości).

ODP. Po dodaniu `diagnostic_index`, który jest inkrementowany w każdej iteracji pobierania listy możemy zauważyć różnice w pobranych danych.

8. Uruchom i przetestuj system o jednej instancji mikrousługi będącej producentem wiadomości i wielu instancjach mikrousługi będącej konsumentem wiadomości. Wskaż naturalne różnice w efekcie działania aplikacji odbiorczej względem przypadku z jedną instancją producenta wiadomości i z pojedynczą instancją aplikacji konsumenta wiadomości.

```
~/Studio/wti/wtiProj01 main* f
wslvenv01 > /home/sebastian/Studia/wti/wslvenv01/bin/python /home/sebastian/Studia/wti/wtiProj01/wtiProj01_producer.py
~/Studio/wti/wtiProj01 main* f 25s
wslvenv01 >
~/Studio/wti/wtiProj01 main* f
wslvenv01 >
cond': 3.0, 'diagnostic_index': 100073}
2021-04-24 14:48:43
2021-04-24 14:48:44
0 {'userID': 75.0, 'movieID': 2058.0, 'rating': 4.0, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 29.0, 'date_second': 25.0, 'diagnostic_index': 100026}
2021-04-24 14:48:44
2021-04-24 14:48:45
2021-04-24 14:48:45
0 {'userID': 75.0, 'movieID': 2700.0, 'rating': 4.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 17.0, 'date_second': 52.0, 'diagnostic_index': 100031}
2021-04-24 14:48:45
2021-04-24 14:48:46
0 {'userID': 75.0, 'movieID': 3358.0, 'rating': 1.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 26.0, 'date_second': 8.0, 'diagnostic_index': 100034}
2021-04-24 14:48:46
2021-04-24 14:48:47
~/Studio/wti/wtiProj01 main* f 10s
wslvenv01 >
second': 16.0, 'diagnostic_index': 100035}
2021-04-24 14:48:47
0 {'userID': 75.0, 'movieID': 3889.0, 'rating': 3.0, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 22.0, 'date_second': 43.0, 'diagnostic_index': 100036}
1 {'userID': 75.0, 'movieID': 3994.0, 'rating': 3.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 25.0, 'date_second': 24.0, 'diagnostic_index': 100037}
2021-04-24 14:48:47
0 {'userID': 75.0, 'movieID': 4993.0, 'rating': 3.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 30.0, 'date_second': 34.0, 'diagnostic_index': 100038}
2021-04-24 14:48:47
0 {'userID': 75.0, 'movieID': 5107.0, 'rating': 3.0, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 26.0, 'date_second': 11.0, 'diagnostic_index': 100039}
1 {'userID': 75.0, 'movieID': 5833.0, 'rating': 2.5, 'date_day': 29.0, 'date_month': 10.0, 'date_year': 2006.0, 'date_hour': 23.0, 'date_minute': 29.0, 'date_second': 51.0, 'diagnostic_index': 100040}
~/Studio/wti/wtiProj01 main* f 10s
wslvenv01 >
```

ODP. Konsumerzy "wyszarpują sobie dane".