



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики  
Кафедра програмного забезпечення комп'ютерних систем

**Лабораторна робота № 2**  
з дисципліни “Бази даних. Частина 2”

Виконав  
студент III курсу  
групи КП-82

Мельничук Олексій Геннадійович  
*(прізвище, ім'я, по батькові)*

варіант № 12

Зарахована  
“ \_\_\_\_ ” “ \_\_\_\_ ” 20\_\_ р.  
викладачем

Петрашенко Андрій Васильович  
*(прізвище, ім'я, по батькові)*

## **Мета роботи**

Здобуття практичних навичок створення ефективних програм, орієнтованих на використання сервера Redis за допомогою мови Python.

## **Завдання**

Реалізувати можливості обміну повідомленнями між користувачами у оффлайн та онлайн режимах із можливістю фільтрації спам-повідомлень.

- 1) Проаналізувавши матеріали ресурсів, наведених у пункті “Джерела”, обрати та обґрунтувати вибір структур даних Redis щодо реалізації наведених вище вимог, обов’язково використати наступні структури даних та інструменти Redis: List, Hash, Sorted List, Set, Pub/Sub.
- 2) Забезпечити роботу програмних засобів у режимі емуляції із можливістю генерації повідомлень від різних користувачів, налаштування кількості виконувачів та часу затримки обробки на спам з можливістю підключення адміністратора для перегляду подій, що відбуваються.
- 3) Перевірку на спам можна проемувати за допомогою затримки на псевдовипадковий час та генерацію псевдовипадкового результату (Так/Ні).

Окремі програмні компоненти та вимоги до них:

- 1) Redis server (RS), що виконує наступні ролі:
  - 1.1. Сховище, що містить: дані користувачів, їхні групи (звичайний користувач та адміністратор), а також повідомлення, що пересилаються між ними.
  - 1.2. Черга повідомлень, які підлягають перевірці на спам та відправленню адресату.
  - 1.3. Інструмент Publish/Subscribe для ведення та розсилання журналу активності користувачів (див. Список активностей для журналювання).
- 2) Інтерфейс користувача (User Interface)

- 2.1. Звичайний користувач має змогу виконувати вхід за ім'ям (без паролю), відправляти та отримувати (переглядати) повідомлення, отримувати дані про кількість своїх повідомлень, згрупованих за статусом (див. Статуси повідомлень).
  - 2.2. Адміністратор має змогу переглядати журнал подій, що відбулись (див. Список активностей для журналювання), переглядати список користувачів, які знаходяться online, переглядати статистику (N найбільш активних відправників повідомлень із відповідною кількістю, N найактивніших “спамерів” із відповідною кількістю)
- 3) Виконувач (worker) призначений для:
- перегляду черги повідомлень, відбору повідомлення, перевірки його вмісту на наявність спаму (у випадку наявності спаму -- додавання запису в журнал)

#### Інші вимоги

- 1) Проаналізувавши матеріали ресурсів, наведений у пункті “Джерела”, обрати та обґрунтувати вибір структур даних Redis щодо реалізації наведених вище вимог, обов'язково використати наступні структури даних та інструменти Redis: List, Hash, Sorted List, Set, Pub/Sub.
- 2) Забезпечити роботу програмних засобів у режимі емуляції із можливістю генерації повідомлень від різних користувачів, налаштування кількості виконувачів та часу затримки обробки на спам з можливістю підключення адміністратора для перегляду подій, що відбуваються
- 3) Перевірку на спам можна проємулювати за допомогою затримки на псевдовипадковий час та генерацію псевдовипадкового результату (Так/Ні).

## Реалізація

В даній роботі були застосовані такі структури даних Redis:

- List – використовується для впорядкованого збереження черги повідомлень
- Hash – використовується для збереження повідомлень та користувачів з допомогою пар ключ-значення
- Sorted list – відсортований список-«рейтинг» спамерів та відправників
- Set – список користувачів онлайн з униканням дублювання (всі значення унікальні)
- Pub/sub – використовується для журналу подій

Програма має дані скрипти:

- main.py – консольний інтерфейс для користувача та адміністратора; є можливість запустити декілька вікон консолі з різними користувачами.
- emulation.py – емуляція що тестує програму, створюючи нових юзерів та надсилаючи повідомлення
- worker.py – переглядувач черги повідомлень, що відокремлює спам

В папці ops:

- connection.py – конфігурація для підключення до бд Redis
- message.py – файл з класом Message що описує всі операції з об'єктом
- user.py – файл з класом User що описує всі операції з об'єктом

## Код програми

### Main.py – консольний інтерфейс

```
import os
import logging
from pathlib import Path

from ops.user import User
from ops.message import Message

import ops.connection as redis
redis.connect()
rconnection = redis.connection

import re

FILE = os.path.join(os.path.dirname(Path(__file__).absolute()), 'logs.txt')
logging.basicConfig(filename="logs.txt", level=logging.INFO)

def start_menu():
    print("\nMAIN MENU")
    print("1. Register")
    print("2. Login")
    print("0. Exit")
    return int(input("Enter the number of action: "))

def user_menu(uid):
    print(f"\nUSER MENU: {User.get_username(uid)}")
    print("1. Inbox")
    print("2. Write new message")
    print("3. Message status")
    print("4. Give admin")
    print("0. Logout")
    return int(input("Enter the number of action: "))

def main():
    while True:
        action = start_menu()

        #register
        if action == 1:
            username = input("Enter new username: ")
            uid = User.register(username)

            if User.is_logged_in(uid):
                user(uid)

        #login
        elif action == 2:
            username = input("Enter username: ")
            uid = User.login(username)

            if User.is_logged_in(uid):
                user(uid)

        #exit
        elif action == 0:
            print("Farewell!")
            break

        #wrong choice
        else:
            print("Enter correct choice (num 0 to 2): ")

def admin_menu(uid):
```

```

        print(f"\nADMIN MENU: {User.get_username(uid)}")
        print("1. Sender rating")
        print("2. Spammer rating")
        print("3. View logs")
        print("4. Users online")
        print("0. Exit")
        return int(input("Enter the number of action: "))

def user(uid):
    while True:
        action = user_menu(uid)

        #inbox
        if action == 1:
            Message.get_inbox(uid)
        #new message
        elif action == 2:
            text = input("Enter message: ")
            receiver = input("Enter username of the receiver: ")

            if Message.create_message(uid, text, receiver):
                print(f"Sent message to {receiver}!")
            else:
                print("Got some trouble with sending a message")
        #message status
        elif action == 3:
            keys = ["queue", "checking", "blocked", "sent", "delivered"]
            view = ["Queued", "Checking", "Blocked", "Sent", "Delivered"]

            user = rconnection.hmget(f"user{uid}", keys)

            for i in range(5):
                print(f"-{view[i]}: {user[i]}")

        #get admin
        elif action == 4:
            return admin(uid)
        #logout
        elif action == 0:
            User.logout(uid)
            break
        #incorrect
        else:
            print("Enter correct choice (num 0 to 4): ")

def admin(uid):
    while True:
        action = admin_menu(uid)

        #get senders
        if action == 1:
            quantity = 10
            active_senders = rconnection.zrange("sent", 0, quantity, desc=True,
withscores=True)

            if len(active_senders) == 0:
                print("No senders found.")
            else:
                print(f"All {len(active_senders)} most active senders: ")
                for index, sender in enumerate(active_senders):
                    sid = re.search(r'\d+', sender[0]).group()
                    uname = User.get_username(sid)
                    print(f"{index + 1}. {uname} - {int(sender[1])} messages")

```

```

        #get spammers
        elif action == 2:
            quantity = 10
            active_spammers = rconnection.zrange("spam", 0, quantity, desc=True,
withscores=True)

            if len(active_spammers) == 0:
                print("No spammers found.")
            else:
                print(f"All {len(active_spammers)} most active spammers: ")
                for index, sender in enumerate(active_spammers):
                    sid = re.search(r'\d+', sender[0]).group()
                    uname = User.get_username(sid)
                    print(f"{index + 1}. {uname} - {int(sender[1])} messages")

        #get logs
        elif action == 3:
            try:
                with open(FILE) as file:
                    print(file.read())
            except Exception as e:
                return f"Error: Log reading: '{e}'"

        #online members
        elif action == 4:
            online_users = rconnection.smembers("online")
            if len(online_users) == 0:
                print("No one's around.")
            else:
                print(f"{len(online_users)} users online: ")
                for user in online_users:
                    print(f"-{user}")

        #logout
        elif action == 0:
            print(f"Farewell, {User.get_username(uid)}!")
            break

        #incorrect
        else:
            print("Enter correct choice (num 0 to 4): ")

if __name__ == '__main__':
    main()

```

## emulation.py – генерування даних

```
import ops.connection as redis

redis.connect()
rconnection = redis.connection

from ops.user import User
from ops.message import Message

from threading import Thread
from faker import Faker
from random import randint
import atexit

class Emulation(Thread):
    def __init__(self, name, users):
        Thread.__init__(self)
        self.conn = rconnection
        self.name = name
        self.users = users
        self.user_id = User.register(name)

    def run(self):
        for i in range(amount):
            sentence = fake.sentence(nb_words=5, variable_nb_words=True,
ext_word_list=None)
            receiver = users[randint(0, amount - 1)]
            print(f"Message {sentence} was sent to {receiver}");
            Message.create_message(self.user_id, sentence, receiver)

    def exit():
        online = rconnection.smembers("online")
        for i in online:
            rconnection.srem("online", i)
            rconnection.publish("logout", f"User {i} signed out.")
            print(f"{i} exits app. Have a good day!")

if __name__ == '__main__':
    fake = Faker()
    atexit.register(exit)
    amount = 5

    users = [fake.profile(fields=["username"], sex=None)["username"] for user in
range(amount)]
    threads = []

    for i in range(amount):
        print(f"User: {users[i]}")
        threads.append(Emulation(users[i], users))

    for t in threads:
        t.start()
```



## Worker.py – переглядувач черги + перевірка на спам

```
import ops.connection as redis
redis.connect()
rconnection = redis.connection

from ops.message import Message
from ops.user import User

from threading import Thread
from random import randint
import time
import random

import logging
import datetime
logging.basicConfig(filename="logs.txt", level=logging.INFO)

def is_spam():
    return random.random() > 0.5

class Worker(Thread):
    def __init__(self):
        Thread.__init__(self)

    def run(self):
        message = rconnection.brpop("queue")

        if message:
            message_id = message[1]
            message_key = f"message{message_id}"
            rconnection.hset(message_key, "status", "checking")

            message = rconnection.hmget(message_key, ["sender_id", "receiver_id"])
            sender_id = message[0]
            receiver_id = message[1]
            sender_name = User.get_username(sender_id)

            rconnection.hincrby(f"user{sender_id}", "queue", -1)
            print("Message enqueued")
            rconnection.hincrby(f"user{sender_id}", "checking", 1)
            time.sleep(randint(0, 3))

            pipeline = rconnection.pipeline(True)
            pipeline.hincrby(f"user{sender_id}", "checking", -1)

            if is_spam():
                print(f"{sender_name} sent spam: id={message_id}")

                message_text = rconnection.hmget(message_key, ["text"])[0]
                logging.info(f"({datetime.datetime.now()}): User {sender_name} sent spam: {message_text}")

                pipeline.zincrby("spam", 1, f"user{sender_id}")
                pipeline.hset(message_key, "status", "blocked")
                pipeline.hincrby(f"user{sender_id}", "blocked", 1)
                pipeline.publish("spam", f"User {sender_name} sent spam: {message_text}.")
            else:
                print(f"Checked and sent message[{message_id}] from {sender_name}.")
                pipeline.hset(message_key, "status", "sent")
                pipeline.hincrby(f"user{sender_id}", "sent", 1)
```

```

        pipeline.sadd(f"sent_to:{receiver_id}", message_id)

    pipeline.execute()

def main():
    handlers = 5
    for i in range(handlers):
        worker = Worker()
        worker.daemon = True
        worker.start()

    while True:
        pass

if __name__ == '__main__':
    main()

```

## ops/User.py – операції з користувачами

```

import ops.connection as redis
import logging
import datetime
logging.basicConfig(filename="logs.txt", level=logging.INFO)

redis.connect()
rconnection = redis.connection

class User:
    def register(username):
        if rconnection.hget("users", username):
            print(f"{username} already exists.")
            return -1

        user_id = rconnection.incr("user_id")
        user_key = f"user{user_id}"
        user_info = {
            "id": user_id,
            "name": username,
            "queue": 0,
            "checking": 0,
            "blocked": 0,
            "sent": 0,
            "delivered": 0
        }

        rconnection.hset("users", username, user_id)
        for key in user_info.keys():
            rconnection.hset(user_key, key, user_info[key])

        rconnection.publish("register", f"User {username} registered")
        rconnection.sadd("online", username)
        logging.info(f"({datetime.datetime.now()}): User {username} registered")
        return user_id

    def login(username):
        user_id = rconnection.hget("users", username)

        if not user_id:
            print(f"{username} does not exist. Register?")

```

```

        return -1

    rconnection.publish("login", f"User {username} logged in")
    rconnection.sadd("online", username)
    logging.info(f"({datetime.datetime.now()}): User {username} logged in")
    return user_id

def logout(user_id):
    username = User.get_username(user_id)
    rconnection.publish("logout", f"User {username} logged out")
    rconnection.srem("online", username)
    logging.info(f"({datetime.datetime.now()}): User {username} logged out")

def get_username(user_id):
    return rconnection.hmget(f"user{user_id}", ["name"])[0]

def is_logged_in(user_id):
    return user_id != -1

```

## ops/Message.py – операції з повідомленнями

```

from os import pipe
import ops.connection as redis

redis.connect()
rconnection = redis.connection
from ops.user import User

import random

class Message:
    def create_message(user_id, message, receiver):
        message_id = rconnection.incr("message_id")
        receiver_id = rconnection.hget("users", receiver)
        if not receiver_id:
            print(f"{receiver} does not exist, can't send a message.")
            return False

        message_key = f"message{message_id}"
        message_info = {
            "id": message_id,
            "text": message,
            "sender_id": user_id,
            "receiver_id": receiver_id,
            "status": "created"
        }

        pipeline = rconnection.pipeline(True)

        for key in message_info.keys():
            pipeline.hset(message_key, key, message_info[key])

        pipeline.lpush("queue", message_id)
        pipeline.hset(message_key, "status", "queue")

        pipeline.hincrby(f"user{user_id}", "queue", 1)
        pipeline.zincrby("sent", 1, f"user{user_id}")

        pipeline.execute()

        return message_id

```

```

def get_inbox(user_id):
    messages = rconnection.smembers(f"sent_to{user_id}")

    if len(messages) == 0:
        print("No messages")
        return

    for message_id in messages:
        message = rconnection.hmget(f"message{message_id}", ["text", "status",
"sender_id"])

        if message[1] != "delivered":
            rconnection.hset(f"message{message_id}", "status", "delivered")
            rconnection.hincrby(f"user{message[2]}", "sent", -1)
            rconnection.hincrby(f"user{message[2]}", "delivered", 1)

    print(f"{message[0]} -> FROM: {User.get_username(message[2])}")

```

### ops/connection.py – конфіг підключення

```

import redis
import sys

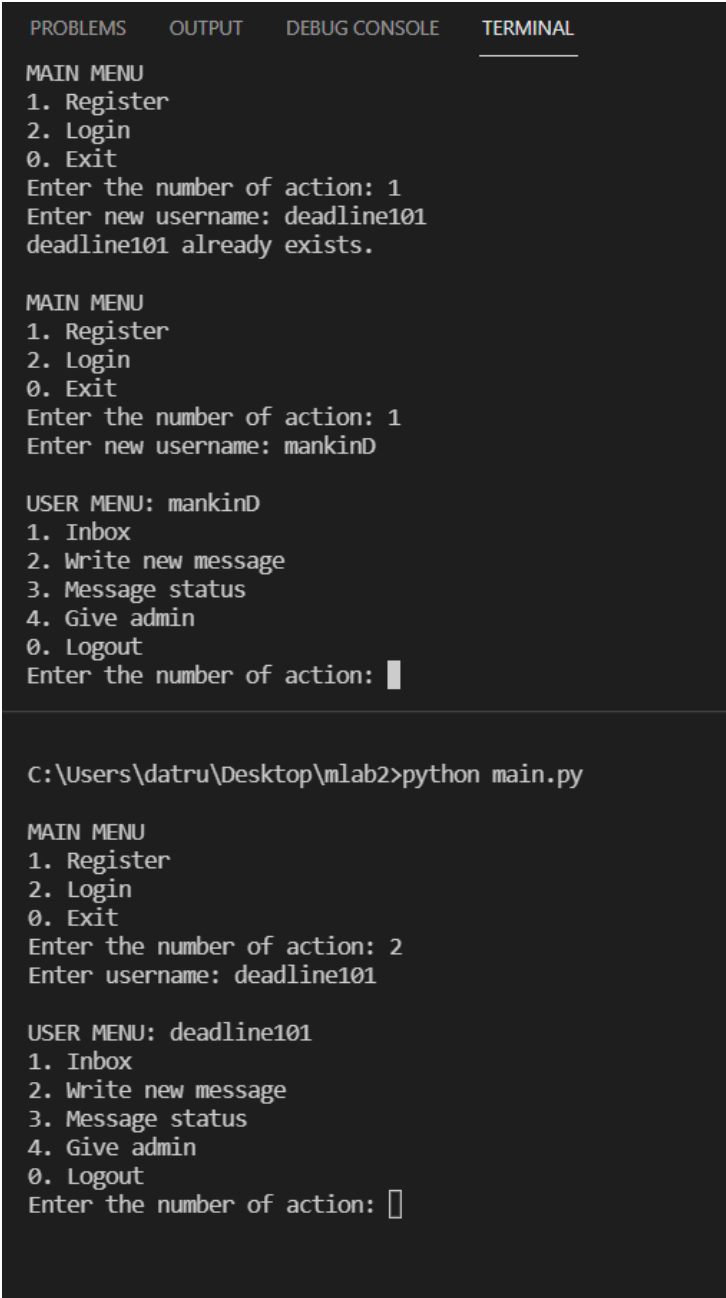
class Redis:
    host = 'localhost'
    port = 6379

connection = redis.Redis(host=Redis.host, port=Redis.port, db=0,
decode_responses=True)

def connect():
    try:
        connection.ping()
    except Exception as err:
        sys.exit(err)

```

## Результати роботи програм



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

MAIN MENU
1. Register
2. Login
0. Exit
Enter the number of action: 1
Enter new username: deadline101
deadline101 already exists.

MAIN MENU
1. Register
2. Login
0. Exit
Enter the number of action: 1
Enter new username: mankind

USER MENU: mankind
1. Inbox
2. Write new message
3. Message status
4. Give admin
0. Logout
Enter the number of action: █

C:\Users\datru\Desktop\mlab2>python main.py

MAIN MENU
1. Register
2. Login
0. Exit
Enter the number of action: 2
Enter username: deadline101

USER MENU: deadline101
1. Inbox
2. Write new message
3. Message status
4. Give admin
0. Logout
Enter the number of action: █
```

Рис. 1. Два користувачі відкриті в двох консолях водночас.

```
USER MENU: mankind
1. Inbox
2. Write new message
3. Message status
4. Give admin
0. Logout
Enter the number of action: 2
Enter message: heloooooooo
Enter username of the receiver: deadline101
Sent message to deadline101!

USER MENU: mankind
1. Inbox
2. Write new message
3. Message status
4. Give admin
0. Logout
Enter the number of action: 3
-Queued: 1
-Checking: 0
-Blocked: 0
-Sent: 0
-Delivered: 0
```

Рис. 2. Надсилання повідомлень.

```
USER MENU: deadline101
1. Inbox
2. Write new message
3. Message status
4. Give admin
0. Logout
Enter the number of action: 4

ADMIN MENU: deadline101
1. Sender rating
2. Spammer rating
3. View logs
4. Users online
0. Exit
Enter the number of action: 4
4 users online:
-deadline101
-Bruh Moment
-mankind
-Shawna Huerta

ADMIN MENU: deadline101
1. Sender rating
2. Spammer rating
3. View logs
4. Users online
0. Exit
Enter the number of action: █
```

Рис. 3. Функція “Give admin” надає права адміністратора користувачу та надає доступ до нових функцій, таких як перегляд користувачів онлайн.

```

USER MENU: deadline101
1. Inbox
2. Write new message
3. Message status
4. Give admin
0. Logout
Enter the number of action: 4

ADMIN MENU: deadline101
1. Sender rating
2. Spammer rating
3. View logs
4. Users online
0. Exit
Enter the number of action: 1
All 11 most active senders:
1. Shawna Huerta - 6 messages
2. Eric Johnson - 5 messages
3. Gina Kelly - 5 messages
4. Nathan Reid - 5 messages
5. Ricardo Buck II - 5 messages
6. alexa30 - 5 messages
7. chadmanning - 5 messages
8. nmunoz - 5 messages
9. tanderson - 5 messages
10. Donald Brown - 5 messages
11. jlee - 5 messages

```

Рис. 4. Перегляд найактивніших відправників.

```

ADMIN MENU: deadline101
1. Sender rating
2. Spammer rating
3. View logs
4. Users online
0. Exit
Enter the number of action: 3
INFO:root:(2021-05-15 00:38:04.069332): User kek321 logged in
INFO:root:(2021-05-15 00:38:49.889452): User tommybaker registered
INFO:root:(2021-05-15 00:38:49.891453): User tphillips registered
INFO:root:(2021-05-15 00:38:49.893453): User sanchezjohn registered
INFO:root:(2021-05-15 00:38:49.895455): User ashleymeadows registered
INFO:root:(2021-05-15 00:38:49.897455): User iwilliams registered
INFO:root:(2021-05-15 00:40:08.122190): User natashaunderwood registered
INFO:root:(2021-05-15 00:40:08.124188): User shawnreyes registered
INFO:root:(2021-05-15 00:40:08.125188): User steven34 registered
INFO:root:(2021-05-15 00:40:08.127187): User carrbarbara registered
INFO:root:(2021-05-15 00:40:08.128192): User mark71 registered
INFO:root:(2021-05-15 00:40:35.331383): User pwalsh registered
INFO:root:(2021-05-15 00:40:35.333395): User mford registered
INFO:root:(2021-05-15 00:40:35.334386): User mjohnson registered
INFO:root:(2021-05-15 00:40:35.336379): User jrogers registered

```

Рис. 5. Перегляд логів

```
logs.txt
5 0:root:(2021-05-15 00:38:49.895455): User ashleymeadows registered
6 0:root:(2021-05-15 00:38:49.897455): User iwilliams registered
7 0:root:(2021-05-15 00:40:08.122190): User natashaunderwood registered
8 0:root:(2021-05-15 00:40:08.124188): User shawnreyes registered
9 0:root:(2021-05-15 00:40:08.125188): User steven34 registered
10 0:root:(2021-05-15 00:40:08.127187): User carrbarbara registered
11 0:root:(2021-05-15 00:40:08.128192): User mark71 registered
12 0:root:(2021-05-15 00:40:35.331383): User pwalsh registered
13 0:root:(2021-05-15 00:40:35.333395): User mford registered
14 0:root:(2021-05-15 00:40:35.334386): User mjohnson registered
15 0:root:(2021-05-15 00:40:35.336379): User jreeves registered
16 0:root:(2021-05-15 00:40:35.338386): User samantha13 registered
17 0:root:(2021-05-15 00:41:42.328267): User kevinmolina registered
18 0:root:(2021-05-15 00:41:42.330268): User biancamoore registered
19 0:root:(2021-05-15 00:41:42.333276): User jamesjohnson registered
20 0:root:(2021-05-15 00:41:42.335266): User simmonsjoshua registered
21 0:root:(2021-05-15 00:41:42.337264): User estradakatherine registered
22 0:root:(2021-05-15 00:43:27.275845): User Ricardo Buck II sent spam: High investment special subject.
23 0:root:(2021-05-15 00:43:27.291853): User Nathan Reid sent spam: Huge beat help friend design respond.
24 0:root:(2021-05-15 00:43:27.448853): User Ricardo Buck II sent spam: Source notice but.
25 0:root:(2021-05-15 00:43:53.359952): User jlee registered
26 0:root:(2021-05-15 00:43:53.361971): User tanderson registered
27 0:root:(2021-05-15 00:43:53.363962): User nmunoz registered
28 0:root:(2021-05-15 00:43:53.365971): User chadmannning registered
29 0:root:(2021-05-15 00:43:53.367954): User alexa30 registered
30 0:root:(2021-05-15 00:45:15.755981): User deadline101 registered
31 0:root:(2021-05-15 00:47:01.855849): User deadline101 logged in
32 0:root:(2021-05-15 00:47:27.936305): User NotReally registered
33 0:root:(2021-05-15 00:47:59.036794): User Shawna Huerta sent spam: Especially sit nothing section.
34 0:root:(2021-05-15 00:48:23.029439): User Eric Johnson sent spam: Hand support gas control series.
35 0:root:(2021-05-15 00:48:38.031360): User Gina Kelly sent spam: Sea what continue their.
36 0:root:(2021-05-15 00:48:38.031360): User Gina Kelly sent spam: One movie beyond better.
37 0:root:(2021-05-15 00:50:18.415197): User deadline101 logged in
38 0:root:(2021-05-15 01:03:47.416824): User deadline101 logged in
39 0:root:(2021-05-15 01:07:43.857433): User deadline101 logged in
40 0:root:(2021-05-15 01:08:33.304299): User deadline101 logged in
```

Рис. 6. Логування



## **Відповіді на контрольні запитання**

- 1) Визначити сфери застосування основних структури даних redis (List, Hash, Sorted List, Set, Pub/Sub).

List – зручні для черг та стеків для швидкого доступу до верхнього/нижнього елементів

Hash – зручна структура для об'єктів та структур з полями з різними значеннями

Sorted List – реалізація «топів» та «рейтингів» з множин за різними параметрами

Set – множини в яких можна швидко перевірити наявність елемента в колекції

Pub/Sub – легковагова розсилка повідомлень, зручна для швидких малих повідомлень

- 2) Визначити основні переваги та недоліки redis.

Переваги:

- Висока швидкість
- Простий в налаштуванні та установці
- Зручність в роботі з різними типами даних
- Гарна документація та підтримка
- Open source

Недоліки

- Потреба великої кількості оперативної пам'яті, оскільки всі дані знаходяться там
- Не має складної мови запитів, лише команди
- Масштабування складне й частіше за все дороге у налаштуванні
- Все навантаження зосереджується на одному master