МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 2

з дисципліни “Бази даних 2. БД на основі XML”
тема “Практика використання сервера Redis ”

Виконала
студентка III курсу
групи КП-83
Виноградова Анастасія Сергіївна
(*прізвище, ім’я, по батькові*)

Зарахована
“____” “_____” 20___ р.
викладачем
Петрашенко А. В.
(*прізвище, ім’я, по батькові*)

варіант № 3

Київ 2021

Вступ

Метою роботи є 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 режамах із можливістю фільтрації спам-повідомлень.

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 обгрунтувати 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 проемулювати за допомогою затримки на псевдовипадковий час та генерацію псевдовипадкового результату (Так/Ні).

Список активностей для журналювання

Вхід/вихід користувача, наявність спаму у повідомленні.

Статуси повідомлень

“Створено”, “У черзі”, “Перевіряється на спам”, “Заблоковано через спам”, “Відправлено адресату”, “Доставлено адресату”.

Реалізація

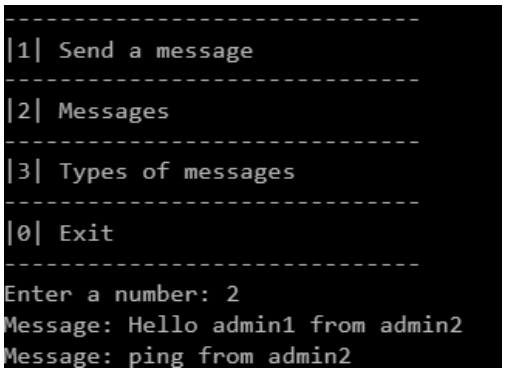
1. Обгрунтування вибору структур даних redis:
    - List - черга повідомлень для перевірки Виконувача
    - Hash - об’єкти Звичайного користувача та відповідні повідомлення
    - Sorted List - статистика (найчастіші відправники та спамери) для Адміністратора
    - Set - Звичайні користувачі онлайн
    - Pub/Sub - журнал івентів активності користувачів

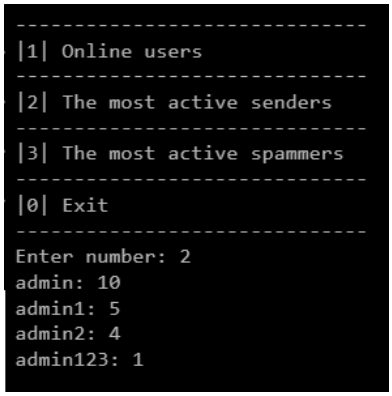
2. Посилання на репозиторій GitHub.

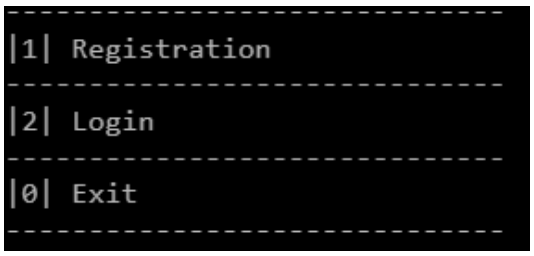
3. Опис умов експерименту:
    - черга повідомлень (handler.py) - повідомлення від Звичайних користувачів, що підлягають перевірці. Перевірка проводиться на основі генерації випадкових чисел (random). При перевірці використовується умовна затримка (delay);
    - інтерфейс звичайного користувача (main.py) - авторизація та реєстрація Звичайного користувача за логіном, перевірка та надсилання повідомлень;
    - адміністратор (admin.py) - перевірка статистики (повідомлень та спамерів) після авторизації як адміністратор сервісу;

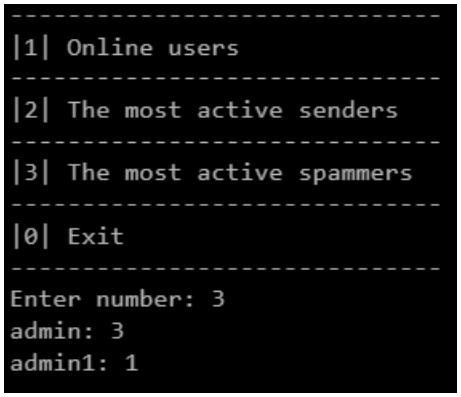
- емулятор (emulator.py) - скрипт, щоб генерувати повідомлення від користувачів з деякими налаштуваннями;
- сервіс (service.py) - основний програмний інтерфейс для роботи із сервісом;

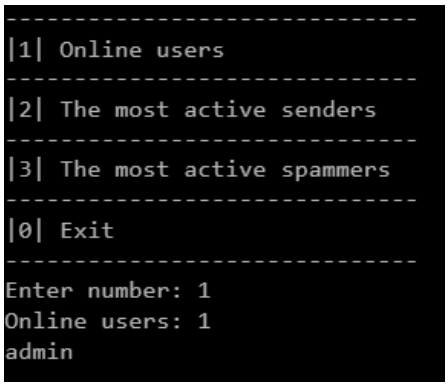
4.

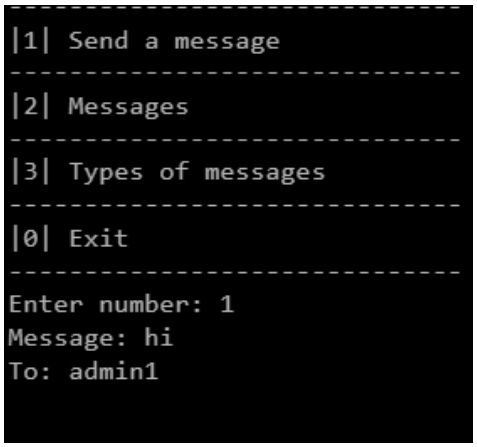


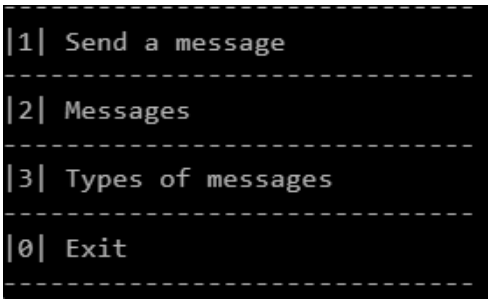





|1| Online users
|2| The most active senders
|3| The most active spammers
|0| Exit
Enter number: 3
admin: 3
admin1: 1


|1| Online users
|2| The most active senders
|3| The most active spammers
|0| Exit
Enter number: 1
Online users: 1
admin


|1| Send a message
|2| Messages
|3| Types of messages
|0| Exit
Enter number: 1
Message: hi
To: admin1


|1| Send a message
|2| Messages
|3| Types of messages
|0| Exit

5.

1. Плюси:
    - Швидкий(дуже швидкий)
    - Простий у використанні
    - Підтримує майже всі структури данних
    - Дозволяє зберігати ключі та значення у розмірі до 512мб
    - Open source(можливо покататись та подивитися, як він влаштований)

    Мінуси:

    - Має бути багато RAM пам’яті.
    - Розмір БД обмежений доступною пам'яттю
    - NoSQL(no joins or query language)
    - Потрібно вивчати Lua
2. **Списки** гарні коли в основному ви працюєте з крайніми елементами: близько хвоста, або близько голови. Списки не найкращий вибір для поділу чого-небудь на сторінки, через повільне випадкового доступу, O (N). Хорошим використанням списків будуть прості черзі і стеки, або циклічна обробка елементів командою RPOPLPUSH, параметрами якої буде один і той же список.
   Множина - це не впорядкований набір даних, воно ефективно коли у вас є колекція елементів, і важливо дуже швидко перевірити присутність елемента в колекції, або отримати її розмір. Ще одна «фішка» множин - це можливість отримати випадковий елемент (команди SRANDMEMBER і SPOP).

Хеші відмінна структура для представлення об'єктів, складених з полів і значень. Поля хеш можуть бути атомарному інкрементіровать командою HINCRBY. Якщо у вас є об'єкти, такі як користувачі, записи в блозі, або інші види елементів, хеші - це те, що вам потрібно, якщо ви не хочете використовувати свій власний формат, такий як JSON або будь-який інший.

Впорядковане Множина - це єдина структура даних, крім списку, що підтримує роботу з впорядкованими елементами. З впорядкованими множинами можна робити багато крутих речей. Наприклад, ви можете реалізувати всі види Топа Чогось в вашому веб-додатку. Топ користувачів по рейтингу, топ постів по числу переглядів, топ чого завгодно, і один екземпляр Redis обслуговуватиме тонни вставок і запитів в секунду.

Pub/Sub - підписники висловлюють інтерес до одного або декількох каналів, і отримують лише повідомлення, які їх потрібні, без знання того, хто є publisher. Це розділення publisher та subscriber може легко дозволити більшу масштабування.

Висновки

Я здобула практичні навички створення ефективних програм, орієнтованих на використання сервера Redis за допомогою мови Python. Завдяки даній роботі я дізналася про потоки у Python та відкрила для себе програмні рішення з декількома скриптами для запуску, що дозволяє розпаралелити будь-яку роботу.