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”

тема “Практика використання сервера Redis”

Виконала
студентка III курсу
групи КП-81
Мозгова Катерина
Олегівна
варіант №12

Перевірив
“____” “_____” 20___р.
Петрашенко Андрій Васильович

Київ 2021

Завдання

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 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

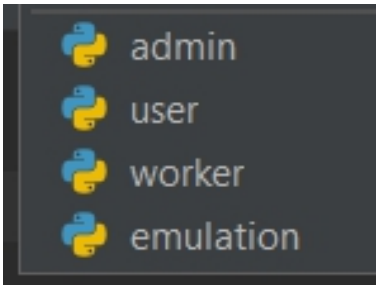
GitHub посилання

1. Структури даних Redis та їх використання:

- List - черга повідомлень (використовуємо вставку та вилучення з кінців цієї структури для реалізації роботи з повідомленнями)
- Hash - об’єкти користувача та повідомлення (зберігання відбувається у форматі ключ-значення, що є найкращим варіантом в даному випадку)
- Sorted List - список найчастіших відправників та спамерів (так як маємо сортувати такі списки)
- Set - користувачі онлайн (порядок записів нам неважливий, але кожен з записів має бути унікальним)
- Pub/Sub - журналювання подій (виконувач події виступає у ролі публиканта, і отримувач слухає на подію)

2. Дана робота має 4 скрипти для запуску:

- admin - інтерфейс адміністратора
- user - інтерфейс користувача
- worker - переглядач черги повідомлень (виконувач)
- emulation - емуляція, що реєструє нових користувачів та надсилає від їх імені повідомлення



3. Приклад записів про певні події.

activity_logs.txt

INFO:root:(2021-04-16 14:55:43.393571) User dragon signed up.
INFO:root:(2021-04-16 14:55:46.075801) User dragon signed out.
INFO:root:(2021-04-16 14:55:57.111797) User fakeUser signed up.
INFO:root:(2021-04-16 14:56:11.307379) User fakeUser signed out.
INFO:root:(2021-04-16 14:56:24.403028) User fakeUser sent spam: hello :).
INFO:root:(2021-04-16 14:56:34.473134) User Shane Andrews signed up.
INFO:root:(2021-04-16 14:56:34.475667) User Michelle Roth signed up.
INFO:root:(2021-04-16 14:56:34.477136) User Sheila Smith signed up.
INFO:root:(2021-04-16 14:56:34.497084) User Sheila Smith signed out.
INFO:root:(2021-04-16 14:56:34.497084) User Shane Andrews signed out.
INFO:root:(2021-04-16 14:56:34.498080) User Michelle Roth signed out.
INFO:root:(2021-04-16 14:56:40.773900) User Michelle Roth sent spam: Sometimes charge better them role..
INFO:root:(2021-04-16 14:56:48.523182) User Shane Andrews sent spam: Child research heart about language else..
INFO:root:(2021-04-16 14:56:48.539159) User Sheila Smith sent spam: Bring law try pressure..
INFO:root:(2021-04-16 14:56:48.729627) User Sheila Smith sent spam: Federal today long capital time..
INFO:root:(2021-04-16 14:56:56.681380) User Sheila Smith sent spam: Lot peace cost before he test..
INFO:root:(2021-04-16 14:56:56.793082) User Sheila Smith sent spam: Size wide rise..

4. Головний інтерфейс.

- Exit - вихід з програми
- Sign up - реєстрація за ім’ям
- Sign in - вхід за ім’ям

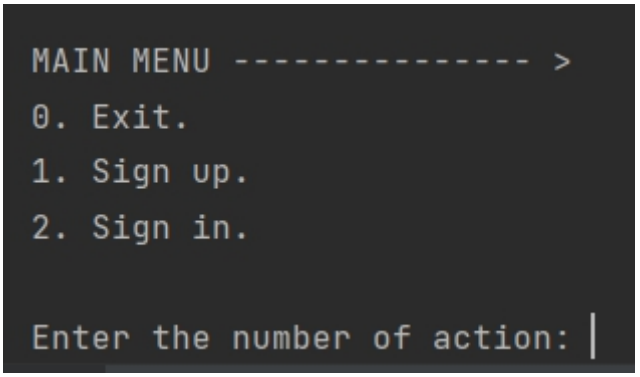


Рис. 1. Приклад головного інтерфейсу.

5. Інтерфейс користувача.

- Log out - вихід з облікового запису
- Inbox messages - перегляд повідомлень, що надійшли до користувача
- Send message - можливість надіслати повідомлення
- Statistics - статистика поточного користувача

```
Enter username: dragon

USER MENU --------------- >
0. Log out.
1. Inbox messages.
2. Send message.
3. Statistics.

Enter the number of action: 2
Enter message: hi!
Enter username of the receiver: fakeUser
Sent your message to fakeUser

USER MENU --------------- >
0. Log out.
1. Inbox messages.
2. Send message.
3. Statistics.

Enter the number of action: 3
in_queue: 2
checking: 0
blocked: 0
sent: 0
delivered: 0
```

```
USER MENU --------------- >
0. Log out.
1. Inbox messages.
2. Send message.
3. Statistics.

Enter the number of action: 1
Oh. You have empty inbox. Let's type someone? :)
```

Рис. 2 та 3. Приклад інтерфейсу користувача.

6. Інтерфейс адміністратора.

- Exit - вихід з програми
- Most active senders - список найактивніших листувальників
- Most active spammers - список найвктивніших спамерів
- Users online - список користувачів, що онлайн наразі

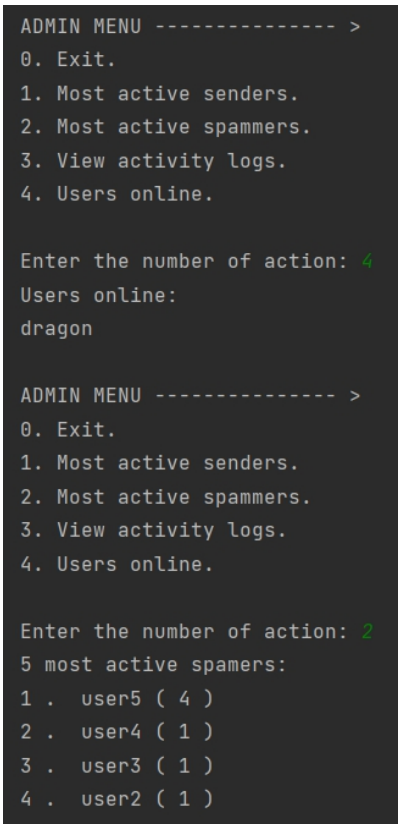


Рис. 4. Приклад інтерфейсу адміністратора.

7. Генерування даних.

Генерування даних полягає у генеруванні певної кількості користувачів, їх реєстрації, надсилання від їх імені в системі та вихід з онлайн режиму.

Emulation.py

```
class Emulation(Thread):
   def___init__(self, name, users):
       Thread.__init__(self)
       self.name = name
       self.users = users
       self.user_id = sign_up(name)

   def run(self):
       for i in range(5):
           text = fake.sentence(nb_words=5, variable_nb_words=True,
ext_word_list=None)
           recipient = users[randint(0, quantity_of_users - 1)]
           create_message(self.user_id, text, recipient)

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

if___name__== '__main__':
   atexit.register(on_emulation_off)
   quantity_of_users = 3
   fake = Faker()
   users = [fake.profile(fields=["name"], sex=None)["name"] for user in
range(quantity_of_users)]
   threads = []

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

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

8. Перевірка на спам полягає у псевдовипадковому генеруванні відповіді Так/Ні

message.py

```
def message_is_spam():
    return random.random() > 0.5
```

Переваги Redis:

- легкий у використанні
- швидкий у роботі
- підтримка багатьох структур даний
- відкритий код
- має власний механізм хешування

Недоліки Redis:

- потребує великої кількості оперативної пам’яті через особливості реалізації
- збільшення необхідної кількості ресурсів для роботи при масштабуванні
- при підключенні клієнти повинні знати топологію кластера, що збільшує кількість налаштувань на клієнті