# Отчет по домашнему заданию № 7 – Применение In-Memory СУБД

В результате выполнения ДЗ было перенесено хранение одного из модулей моего приложения в In-Memory СУБД и тренировались навыки:

* администрирование In-Memory СУБД;
* разработка хранимых процедур для In-Memory СУБД.

В качестве In-Memory СУБД была выбрана БД Redis.

В процессе выполнения ДЗ был переделан модуль диалогов с целью перевода его с PostgreSQL на in-memory СУБД Redis.

В приложении SocialNet были созданы, добавлены или изменены:

* + В класс CacheSupport, обслуживающий Redis, были добавлены методы:
    - register\_lua\_func – загружает в Redis функции языка Lua
    - add\_dialog\_text – по сигнатуре идентичен методу PostgreSQL и добавляет информацию о диалоге в Redis
    - get\_dialogs\_by\_user\_id - по сигнатуре идентичен методу PostgreSQL и выводит лист диалогов заданного пользователя из Redis с помощью функции Lua
  + Изменен функции ендпойнтов в файле handlers.py, которые сейчас вызывают не методы класса PostgreSupport, а идентичные методы класса CacheSupport
  + Остальной код приложения не изменился

Для работы приложения SocialNet использовался docker-compose.yml из задания № 7.

После запуска БД PostgreSQL необходимо выполнить скрипт only\_100\_users.sql из каталога HomeWork 7 – он создает все необходимые таблицы и заполняет данными таблицу пользователей – первыми 100.

Описание работы приложения SocialNet:

При запуске приложения вызывается метод register\_lua\_func класса CacheSupport, который загружает функции Lua в Redis.

Функции Lua:

* store\_string\_with\_hash – принимает как аргументы 3 значения – id\_from\_user, id\_to\_user и текст диалога. Затем она сортирует ид пользователей, конкатинирует их в строку и вычисляет хэш от этой строки – это значит, что все диалоги этих пользователей будут идентифицироваться этим хэшем. Потом берется текущее время и вызывается функция Redis ZADD, которая сохраняет диалог пользователей по хэшу.
* get\_string\_by\_hash – получает все диалоги пользователя по сохраненному хэшу, который объединяет двух пользователей с помощью функции Redis ZRANGEBYSCORE. Здесь выводятся диалоги пользователя не ранее 1 месяца.

Все связки двух пользователей сохраняются в виде хэшей также в Redis в методе add\_dialog\_text, здесь также используются функции Redis – ZADD и ZRANGEBYSCORE, чтобы вывести диалоги не раньше одного месяца.

При добавлении диалога между какими-то двумя пользователями (например, с id=2 и 3) в методе Postman POST /dialog/send функции add\_dialog\_text выполняются следующие действия:

1. В методе cache.add\_post вызывается функция Lua store\_string\_with\_hash и добавляется запись в базу данных Redis для данного id пользователя.
2. Также запоминаются хэши связки двух пользователей

При чтении списка диалогов GET /dialog/list/{id\_user} в методе get\_dialog\_by\_user\_id выполняются действия:

1. Вычисляется timestamp времени на 1 месяц назад
2. Получается список всех хэшей связок данного пользователя с другими по id\_user
3. Производится проход по списку хэшей и для каждого из них вызывается функция Lua get\_sttring\_by\_hash, которая выводит из Redis список всех диалогов данного пользователя со связанным по хэшу пользователем.
4. Далее список упорядочивается и выводится в ответе Response