# Отчет по домашнему заданию № 8 – Разделение монолита на сервисы

В результате выполнения ДЗ было перенесен бизнес-домен монолитного приложения Диалоги в отдельный сервис и тренировались навыки:

* декомпозиции предметной области;
* разделение монолитного приложения;
* работа с HTTP;
* работа с REST API.

В процессе выполнения ДЗ был переделан модуль диалогов с целью перенесения его из монолита в отдельный сервис с которым основное приложение взаимодействует через REST API.

В приложении SocialNet были созданы, добавлены или изменены:

* + Создано отдельное приложение на FastAPI в виде отдельного сервиса (находится в папке dialog\_service)
  + В новом сервисе созданы ендпойнты (routes.py) для запросов REST API:
    - /dialog/send – для создания диалога между двумя пользователями
    - /dialog/list/{id\_user} – для получения листа диалогов для заданного пользователя
  + В сервисе переделан класс RedisSupport, обслуживающий Redis, который используется в качестве БД (как в ДЗ7):
    - register\_lua\_func – загружает в Redis функции языка Lua
    - add\_dialog\_text –добавляет информацию о диалоге в Redis
    - get\_dialogs\_by\_user\_id - выводит лист диалогов заданного пользователя из Redis с помощью функции Lua
  + В основном приложении добавлены новые ендпойнты для версии 2, которые вызывают по REST API соответствующие ендпойнты из нового сервиса:
    - /v2/dialog/send – для создания диалога между двумя пользователями
    - /v2/dialog/list/{id\_user}
  + 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.

Сначала нужно запустить скрипт dialogs.py из папки нового сервиса, затем запустить основное приложение SocialNet из main.py

Описание работы приложения SocialNet:

При запуске нового сервиса вызывается метод register\_lua\_func класса RedisSupport, который загружает функции Lua в Redis.

Функции Lua те же, что и в задании 7. Для совместимости со старым АПИ БД Redis используется совместно, как для старой версии диалогов, так и для новой версии на базе сервиса. Это не совсем микросервис, т.к. БД используется совместно.

Далее для записи диалога двух пользователей вызывается АПИ POST <http://localhost:8070/v2/dialog/send>, вызов перенаправляется в сервис dialog\_service, где обрабатывается, диалог запоминается в БД и ответ возвращается в основное приложение.

Для чтения списка диалогов заданного пользователя вызывается АПИ GET [http://localhost:8070/v2/dialog/list/{id\_user}](http://localhost:8070/v2/dialog/list/%7bid_user%7d%20) , вызов перенаправляется в сервис dialog\_service, где обрабатывается, список диалогов из БД возвращается в виде ответа в основное приложение.

Старые ендпойнты работают как обычно в монолитном приложении.