# Отчет по домашнему заданию № 10 – Сервис счетчиков

Сервис счетчиков

В результате выполнения ДЗ №10 был создан отдельный сервис счетчиков. Сервис в своей БД будет хранить такие счетчики, как число непрочитанных сообщений для каждого пользователя.

В данном задании тренируются навыки:

* разработка отказоустойчивых сервисов;
* использование кешей.

В процессе выполнения ДЗ был переработан проект приложение SocialNet в части диалогов для подключения сервиса счетчиков. Для обеспечения взаимодействия между сервисами диалогов и счетчиков был выбран паттерн **Transactional Outbox**, как наиболее подходящий для высонагруженных систем и обеспечивающий надежную передачу данных и отложенную консистентность. Также для идемпотентности были приняты специальные меры на уровне сервиса счетчиков.

**Transactional Outbox Pattern** (паттерн «транзакционный Outbox») – это шаблон, позволяющий добиться гарантированной доставки событий при межсервисном взаимодействии, избегая потери данных и используя подход *at-least-once delivery* (минимум один раз). Он решает проблему двойной записи и обеспечивает атомарность: событие будет отправлено только если транзакция с записью в базу данных успешна, и никогда – если она откатилась. В контексте CAP-теоремы Outbox-паттерн обеспечивает последовательную (итоговую) согласованность, жертвуя мгновенной доставкой. Он устраняет необходимость в других способах межсервисной транзакционности и снижает связанность сервисов с брокером: каждый сервис отвечает только за свою БД, а доставка событий возлагается на внешний механизм публикации. Паттерн Outbox решает проблему надёжности межсервисной коммуникации: нам не нужно беспокоиться, что брокер или смежный сервис недоступны в момент обработки – сообщение не потеряется, а «полежит» в базе до лучших времён и будет отправлено, когда всё заработает. Также он гарантирует, что сообщение отправится только после успешного коммита транзакции в базе, и никогда – если транзакция откатилась (тем самым предотвращая ложные срабатывания).

В качестве транзакционной БД была выбрана PostgreSQL, в качестве брокера сообщений - RabbitMQ

Реализация функционала:

В приложении SocialNet было добавлено или изменено:

1. Добавлен класс TransactionalOutbox в файлt outbox.py, который поддерживает работу outbox (запись в таблицу БД outbox события, получение списка событий для работы сервиса outbox\_relay, отметка об успешной отправке события в очередь)
2. В файле outbox\_relay.py реализована логика периодического считывания событий из таблицы БД outbox и отправка в очередь сообщений RabbitMQ.
3. Добавлен сервис счетчиков в файле counters\_service.py, который по pub-sub получает события из очереди сообщений и фиксирует их в БД в виде счетчиков событий и непрочитанных сообщений. Этот сервис имеет свой экземпляр БД Postgresql и запускается отдельно.
4. В классе PostgreSupport в файле postgre\_support.py приложения были изменены функции:
   1. add\_dialog\_text – для обеспечения транзакционной логики работы outbox и помещения события в таблицу outbox
   2. get\_dialogs\_by\_user\_id – для чтения сообщений данного пользователя и вызова функции mark\_message\_read, которая помечает сообщение прочитанным, и тразакционно добавляет в таблицу outbox событие о чтении сообщения с последующей передачей события в очередь сообщений RabbitMQ.

Порядок работы:

1. Исполнить docker-compose.yml из каталога HomeWork10, который запускает в докере два экземпляра БД PostgreSQL – для приложения (postgres\_db) и сервиса счетчиков (postgres\_db\_counters), а также Redis.
2. В БД сервиса postgres\_db выполнить sql скрипт users\_dialogs\_outbox.sql для заполнения базы данных пользователями, таблицами диалогов и outbox
3. В БД сервиса postgres\_db\_counters выполнить sql скрипт pg\_counters.sql для заполнения базы данных таблицами событий и счетчиков.
4. Запустить последовательно:
   1. Приложение SocialNet (main.py)
   2. Сервис outbox relay (outbox\_relay.py)
   3. Сервис счетчиков (counters\_esrvice.py)
5. С помощью посылки запросов из коллекции PostMan - send dialog и get list dialogs и наблюдением за содержимым таблицы unread\_counters БД сервиса postgres\_db\_counters убедиться, что данные о количестве непрочитанных сообщения корректно отображаются и паттерн **Transactional Outbox** работает.