Инструкция по запуску и тестированию

Предварительные требования

- Docker и Docker Compose
- Postman для тестирования API

Запуск системы

Репозиторий - https://github.com/artstesh/otus-highload/tree/saga

Перейдите в папку \deployment и запустите файл start.bat.

Коллекция Postman доступна по адресу https://www.postman.com/orange-satellite-666437/otus-highload/collection/8uv29vb/otus-highload

Архитектура решения

Технологический стек

- **Backend**: .NET 8, C#
- **База данных**: PostgreSQL + Dapper
- **Кеширование**: Redis
- **Контейнеризация**: Docker
- **Тестирование**: k6 для нагрузочного тестирования

Ключевые компоненты

2.1. Сервис счетчиков

```
public class CounterService : ICounterService
{
   // Основные методы:
   - GetUnreadCountAsync() - получение счетчика с кешированием
   - IncrementUnreadCountAsync() - инкремент счетчика
   - DecrementUnreadCountAsync() - декремент счетчика
}
```

2.2. SAGA координатор

```
public class SagaCoordinator : ISagaCoordinator
{
   // Процесс пометки сообщения как прочитанного:
   1. MarkMessageAsRead → 2. DecrementCounter → 3. Completion
   // Компенсация при ошибках
```

```
}
```

2.3. Кеширование

- Redis для хранения актуальных значений счетчиков
- TTL 30 минут для баланса между актуальностью и производительностью
- Стратегия Cache-Aside pattern

4. Производительность и кеширование

Стратегия кеширования

- **Чтение**: Cache-Aside → сначала кеш, потом БД
- **Запись**: Write-Through → инвалидация кеша при изменении
- **Обновление**: Lazy Loading → кеш обновляется при следующем чтении

Процесс тестирования сервиса счетчиков

Методология тестирования

Для проверки функциональности и надежности сервиса счетчиков было проведено комплексное тестирование с использованием инструмента **k6** для нагрузочного тестирования. Все тестовые скрипты расположены в директории \Counter\stress.

Тестовые сценарии

Сценарий 1: Тестирование инкремента счетчика

Цель: Проверить корректность увеличения счетчика непрочитанных сообщений при получении новых сообщений.

Процесс:

1. Запускался скрипт sendtest.js, который выполняет массовую отправку случайных сообщений тестовому пользователю с идентификатором b90940ae-ae81-4f6c-b4f2-6986d0b91d4c
2. После выполнения скрипта вызывался REST API метод Get unread (документированный в Postman-коллекции) для получения текущего значения счетчика
3. Проводилась валидация: значение поля unread_messages_count в ответе должно соответствовать количеству успешно отправленных сообщений

Сценарий 2: Тестирование декремента счетчика через SAGA

Цель: Проверить корректность уменьшения счетчика при пометке сообщений как прочитанных и обеспечение консистентности через паттерн SAGA.

Процесс:

1. Запускался скрипт markreadtest.js, который для тестового пользователя:
    - Получал список непрочитанных сообщений

- Для первого найденного непрочитанного сообщения инициировал процесс пометки как прочитанного через SAGA

2. После выполнения скрипта повторно вызывался метод Get unread для проверки обновленного значения счетчика
3. Проводилась валидация: значение unread_messages_count должно уменьшиться ровно на количество успешно обработанных SAGA-транзакций

Метрики валидации

Для сценария отправки сообщений:

```
Ожидаемое значение счетчика = Исходное значение + Количество успешных запросов отправки
```

Для сценария пометки как прочитанного:

```
Ожидаемое значение счетчика = Исходное значение - Количество успешных SAGA-транзакций
```

Результаты тестирования

Тестирование подтвердило:

- Корректную работу инкремента/декремента счетчиков
- Сохранение консистентности данных между сервисом сообщений и сервисом счетчиков
- Надежность SAGA-транзакций при распределенных операциях
- Соответствие фактических значений счетчика ожидаемым математическим расчетам