Функциональная спецификация программного обеспечения «Идемпотентный прокси-сервис»

1. Описание

Настоящий документ содержит описание функциональных характеристик программного обеспечения «Идемпотентный прокси-сервис» (далее «Сервис»). Сервис представляет собой HTTP-прокси с поддержкой идемпотентности, предназначенный для обеспечения надежной обработки повторяющихся запросов в распределенных системах.

2. Среда функционирования продукта

Сервис функционирует в любой среде, поддерживающей контейнерную виртуализацию. Предпочтительными средами являются системы оркестрации Docker Swarm или Kubernetes.

3. Функциональные требования

В распределенных системах и микросервисной архитектуре часто возникают ситуации, когда один и тот же запрос может быть отправлен несколько раз из-за сетевых проблем, таймаутов или сбоев клиентов. Сервис предназначен для обеспечения идемпотентности HTTP-запросов методов POST, PUT и PATCH путем их проксирования с сохранением и повторным использованием результатов.

Основная функциональность:

3.1. Обработка HTTP-запросов с идемпотентностью

- Поддержка методов: POST, PUT, PATCH
- Обязательный заголовок Idempotency-Key для идентификации запросов
- Автоматическое проксирование запросов на целевые URL, указанные в параметре URL
- Фильтрация hop-by-hop заголовков при передаче на апстрим
- Удаление заголовка Idempotency-Key при передаче на апстрим

3.2. Механизм кэширования ответов

- Сохранение полного ответа апстрима (статус, заголовки, тело)
- Настраиваемое время жизни кэшированных ответов
- Возврат закэшированных ответов для повторных запросов с тем же ключом

3.3. Распределенные блокировки

- Предотвращение дублирующих вызовов апстрима при конкурентных запросах
- Ключи блокировок: idem:lock:{Idempotency-Key}
- Настраиваемое время удержания блокировки (LOCK_TTL)
- Токен-базированное управление блокировками для безопасного освобождения

3.4. Обработка конкурентных запросов

- Первый запрос получает блокировку и обрабатывается
- Последующие запросы ожидают завершения первого в течение LOCK_TTL
- Автоматическое возвращение результата после завершения первого запроса
- Возврат статуса 409 Conflict при превышении времени ожидания

3.5. Поддержка различных хранилищ

- Redis (рекомендуемый для продакшена): распределенное хранение и блокировки
- In-memory (для тестирования): потокобезопасные мапы с TTL
- Прозрачное API хранилища с единым интерфейсом

3.6. Логирование и мониторинг

- Детальное логирование всех операций: метод, URL, ключ идемпотентности, статус, источник
- Логирование ошибок и таймаутов
- Измерение времени выполнения операций

3.7. Обработка ошибок

- Корректная обработка таймаутов апстрима (статус 504 Gateway Timeout)
- Обработка ошибок сети (статус 502 Bad Gateway)
- Обработка отмененных запросов клиентом
- Гарантированное освобождение блокировок даже при ошибках

3.8. Оптимизация производительности

- Минимальные задержки при попадании в кэш
- Эффективный механизм ожидания для конкурентных запросов
- Настраиваемые таймауты для различных операций

4. Системные требования к ПО

Минимальные аппаратные требования:

- Операционная система, способная запускать контейнеры (предпочтительно Linux)
- Система управления контейнерной виртуализацией (Docker Swarm или Kubernetes)
- Количество логических ядер процессора: 2
- Семейство процессоров: x86
- Частота процессора: 2.0 ГГц
- Объем установленной памяти: 4 Гб

4.1. Минимальные требования к сторонним компонентам и/или системам, необходимым для установки и работы ПО

- **Для хранения данных:**
    - Redis 6.0+ (лицензия BSD)

    - или использование встроенного in-memory хранилища (для тестирования)
- **Для мониторинга и логирования (опционально):**
    - Prometheus + Grafana для метрик
    - ELK Stack или Loki для централизованного логирования
- **Контейнеризация:**
    - Docker 20.10+ (open-source community edition)
    - Podman 3.0+ (альтернатива Docker)

4.2. Языки программирования

При разработке Сервиса был использован язык программирования Go 1.21+ (лицензия BSD).

5. Модули

Модуль обработки HTTP-запросов — отвечает за прием входящих HTTP-запросов, валидацию заголовков, извлечение Idempotency-Key и маршрутизацию.

6. Безопасность

- Заголовок Idempotency-Key не передается на апстрим
- Фильтрация hop-by-hop заголовков для предотвращения утечки информации
- Токен-базированное управление блокировками для предотвращения несанкционированного освобождения
- Поддержка аутентификации Redis при необходимости

7. Масштабируемость

Сервис поддерживает горизонтальное масштабирование при использовании Redis в качестве хранилища. Несколько экземпляров сервиса могут работать параллельно, разделяя состояние через Redis.

8. Ограничения

- Идемпотентность гарантируется только в пределах времени жизни ключа (IDEMPOTENCY_TTL)
- Размер кэшируемых ответов ограничен доступной памятью Redis/процесса
- Не поддерживается для методов GET, DELETE, HEAD, OPTIONS
- Требуется уникальный Idempotency-Key для каждого логически уникального запроса