

Домашнее задание 05: Оптимизация производительности через кеширование и rate limiting

Цель работы: Получить практические навыки проектирования систем с учетом производительности, реализации кеширования и rate limiting.

Задание

Для своего варианта задания выполните следующие задачи:

1. Анализ производительности

- Изучите выбранный вариант задания
- Определите операции, которые будут выполняться часто (hot paths)
- Определите операции, которые могут быть медленными (обращения к БД, внешние API)
- Определите требования к производительности (время отклика, пропускная способность)

2. Проектирование стратегии кеширования

- Определите, какие данные можно кешировать:
 - Часто читаемые данные
 - Результаты сложных вычислений
 - Данные, которые редко изменяются
- Выберите стратегию кеширования для каждого типа данных:
 - Cache-Aside (Lazy Loading)
 - Read-Through
 - Write-Through
 - Write-Back (Write-Behind)
- Определите TTL (Time To Live) для кешируемых данных
- Определите стратегию инвалидации кеша

3. Реализация кеширования

- Реализуйте простое кеширование в вашем REST API (можно использовать Redis или in-memory cache)
- Примените кеширование минимум для 2 endpoints
- Реализуйте инвалидацию кеша при обновлении данных

4. Проектирование rate limiting

- Определите, какие endpoints требуют rate limiting
- Выберите алгоритм rate limiting для каждого endpoint:
 - Token Bucket
 - Leaking Bucket
 - Fixed Window Counter
 - Sliding Window Log/Counter
- Определите лимиты (например, 100 запросов в минуту для обычных пользователей, 1000 для премиум)

5. Реализация rate limiting

- Реализуйте простой rate limiting для минимум одного endpoint
- Используйте выбранный алгоритм
- Возвращайте правильные HTTP статус-коды (429 Too Many Requests)

- Добавьте заголовки с информацией о лимитах (X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Reset)

6. Анализ производительности

- Опишите, как кеширование и rate limiting улучшают производительность системы
- Определите метрики для мониторинга производительности
- Опишите, как измерить эффективность кеширования (hit rate)

Результат

Результат должен быть оформлен в виде следующих файлов, размещенных в вашем GitHub репозитории:

- ``performance_design.md`` – описание стратегии кеширования и rate limiting
- Исходный код с реализацией кеширования и/или rate limiting
- ``README.md`` – описание проекта и реализованных оптимизаций
- Dockerfile и docker-compose.yaml для запуска приложения

Критерии оценки:

- Обоснованность выбора стратегий кеширования
- Корректность проектирования rate limiting
- Качество реализации (если реализовано)
- Анализ влияния на производительность
- Качество документации