Отчет о реализации системы кэширования ленты постов

Выполненные работы

1. Проектирование и реализация базы данных

* Создана структура БД для хранения пользователей, дружеских связей и постов
* Реализованы таблицы: users, friendships, posts
* Добавлены индексы для оптимизации запросов
* Подготовлены SQL-скрипты для наполнения тестовыми данными

2. Разработка API

* Реализованы CRUD операции для постов (/post/create, /post/update, /post/delete, /post/get)
* Добавлены методы управления друзьями (/friend/add, /friend/delete)
* Реализован метод получения ленты постов друзей (/post/feed)

3. Система кэширования

Архитектура:

* Использован Redis как распределенное кэш-хранилище
* Реализован FeedCacheService для управления кэшем лент
* Настроена интеграция через IDistributedCache

Ключевые возможности кэширования:

* Кэширование последних 1000 постов друзей для каждого пользователя
* Автоматическое обновление кэша при добавлении/изменении постов
* Инвалидация кэша при изменении дружеских связей

4. Механизмы инвалидации кэша

* При добавлении поста - асинхронное обновление лент всех друзей
* При удалении/редактировании поста - инвалидация лент друзей
* При изменении дружеских связей - инвалидация лент обоих пользователей

5. Система перестройки кэшей

* Реализован CacheRebuildService для полной перестройки кэшей из БД
* Поддержка постепенной перестройки батчами по 50 пользователей
* API для мониторинга прогресса перестройки
* Возможность перестройки кэшей отдельных пользователей

6. Docker-контейнеризация

* Подготовлен docker-compose.yml для запуска приложения, PostgreSQL и Redis
* Настроены сетевые соединения между контейнерами
* Конфигурация переменных окружения для подключения к БД и Redis

Технические особенности реализации

Использованные технологии:

* Backend: [ASP.NET](https://asp.net/) Core 8, Dapper
* База данных: PostgreSQL
* Кэширование: Redis через StackExchange.Redis
* Контейнеризация: Docker, Docker Compose

Производительность:

* Асинхронные операции для всех I/O операций
* Батчинг при массовых операциях с кэшем
* Оптимизированные SQL-запросы с использованием индексов

Надежность:

* Обработка ошибок и повторные попытки
* Логирование всех критических операций
* Health checks для мониторинга состояния системы

Результат

Создана высокопроизводительная система кэширования ленты постов, которая:

* Обеспечивает быстрый доступ к ленте пользователя через Redis
* Автоматически поддерживает актуальность данных
* Позволяет перестраивать кэши из БД при необходимости
* Масштабируется за счет распределенной архитектуры