

Анализ оставшихся задач DHT проекта по методике MoSCoW

Дата: 6 июня 2025

Проект: Distributed Hash Table (DHT) на основе Kademlia

Текущий прогресс: 4.5 из 7 спринтов завершено (64%)

Методика: MoSCoW (Must have, Should have, Could have, Won't have)



Текущее состояние проекта



Завершенные компоненты:

- Базовая архитектура DHT и протокол Kademlia
- Визуализация сети и симулятор
- RPC операции (PING, STORE, FIND_NODE, FIND_VALUE)
- Система репликации и отказоустойчивости
- Балансировка нагрузки (8 стратегий)
- Комплексная система мониторинга
- Веб-дашборд для визуализации



Оценка готовности: 64% (готов к базовому использованию)



MUST HAVE (Обязательно к реализации)

Критически важные функции для production-ready системы

Спринт 5: Базовая безопасность

Приоритет: Критический

Время: 4-5 часов

Обоснование: Без безопасности система непригодна для промышленного использования

Обязательные задачи:

1. Аутентификация узлов (2 часа)

2. Система цифровых подписей для узлов
3. Базовая PKI инфраструктура
4. Проверка подлинности при подключении к сети
5. **TLS шифрование** (1.5 часа)
6. Шифрование всех RPC соединений
7. Защита передачи данных между узлами
8. Настройка TLS 1.3 для максимальной безопасности
9. **Базовая защита от атак** (1.5 часа)
10. Защита от простых DDoS атак (rate limiting)
11. Базовая защита от Sybil атак
12. Валидация входящих запросов

Спринт 7: Базовая интеграция

Приоритет: Критический

Время: 4-5 часов

Обоснование: Необходимо для использования системы внешними приложениями

Обязательные задачи:

1. **REST API** (2 часа)
2. Основные операции: GET, PUT, DELETE данных
3. Получение статуса сети и узлов
4. Базовая OpenAPI документация
5. **Docker контейнеризация** (1.5 часа)
6. Docker образы для DHT узлов
7. Docker Compose для локального развертывания
8. Базовые инструкции по запуску
9. **Документация пользователя** (1.5 часа)
10. Руководство по установке и настройке
11. Примеры использования API
12. Базовое руководство администратора

Итого MUST HAVE: 8-10 часов

SHOULD HAVE (Желательно к реализации)

Важные функции, значительно улучшающие систему

Спринт 5: Расширенная безопасность

Приоритет: Высокий

Время: 2-3 часа

Желательные задачи:

1. **Шифрование данных в покое** (1 час)
2. AES-256 шифрование хранимых данных
3. Управление ключами шифрования
4. **Расширенная защита от атак** (1-2 часа)
5. Защита от Eclipse атак
6. Продвинутое защита от Sybil атак
7. Система репутации узлов

Спринт 6: Базовая оптимизация

Приоритет: Высокий

Время: 3-4 часа

Желательные задачи:

1. **Профилирование производительности** (1 час)
2. Анализ узких мест в коде
3. Измерение производительности операций
4. **Оптимизация критических путей** (2-3 часа)
5. Оптимизация алгоритма маршрутизации
6. Улучшение поиска в k-buckets
7. Асинхронная обработка запросов

Спринт 7: Расширенная интеграция

Приоритет: Высокий

Время: 2-3 часа

Желательные задачи:

1. **CI/CD Pipeline** (1.5 часа)
2. Автоматическое тестирование
3. Автоматическая сборка Docker образов
4. **Расширенная документация** (1.5 часа)
5. Архитектурная документация
6. Примеры интеграции с популярными фреймворками

Итого SHOULD HAVE: +7-10 часов

COULD HAVE (Можно добавить при наличии времени)

Полезные функции, но не критичные для основного функционала

Спринт 6: Продвинутая оптимизация

Приоритет: Средний

Время: 2-3 часа

Дополнительные задачи:

1. **Многоуровневое кэширование** (1 час)
2. Кэширование часто запрашиваемых данных
3. Предварительная загрузка популярных ключей
4. **Оптимизация для больших сетей** (1-2 часа)
5. Поддержка сетей из 1000+ узлов
6. Оптимизация использования памяти

Спринт 7: Дополнительные интеграции

Приоритет: Средний

Время: 2-3 часа

Дополнительные задачи:

1. **Интеграция с базами данных** (1.5 часа)

2. Поддержка PostgreSQL как backend
3. Синхронизация с внешними хранилищами
4. **SDK для разных языков** (1.5 часа)
5. Python SDK
6. JavaScript/Node.js SDK

Итого COULD HAVE: +4-6 часов

WON'T HAVE (Не будет реализовано в текущей версии)

Функции, которые откладываются на будущие версии

Отложенные функции:

1. **Мобильные клиенты** - Android/iOS SDK
2. **Blockchain интеграция** - использование DHT для блокчейн данных
3. **IoT поддержка** - легковесные клиенты для IoT устройств
4. **Машинное обучение** - оптимизация маршрутизации с помощью ML
5. **Квантово-устойчивое шифрование**
6. **WebRTC поддержка** для браузерных клиентов
7. **Федеративное обучение** через DHT




Обоснование:

- Требуют значительных дополнительных исследований
 - Выходят за рамки базового DHT функционала
 - Могут быть реализованы в отдельных проектах/версиях
-







Рекомендуемые сценарии реализации

Сценарий 1: Минимальный MVP (8-10 часов)





Цель: Получить работающую production-ready систему -  Базовая безопасность (аутентификация + TLS) -  REST API и документация -  Docker контейнеризация

- **Результат:** Готовая к использованию DHT система

Сценарий 2: Полнофункциональная система (15-20 часов)

Цель: Получить конкурентоспособную DHT платформу -  Все из Сценария 1 - 
Расширенная безопасность -  Оптимизация производительности -  CI/CD и
расширенная документация - **Результат:** Профессиональная DHT платформа

Сценарий 3: Расширенная платформа (19-26 часов)

Цель: Получить DHT платформу с дополнительными возможностями -  Все из
Сценария 2 -  Продвинутая оптимизация -  Интеграция с внешними системами
-  SDK для разных языков - **Результат:** Комплексная DHT экосистема

Итоговые рекомендации

Приоритетный план (рекомендуемый):

1. **Неделя 1:** Спринт 5 (MUST HAVE) - Базовая безопасность (4-5 часов)
2. **Неделя 2:** Спринт 7 (MUST HAVE) - Базовая интеграция (4-5 часов)
3. **Неделя 3:** Спринт 5+6 (SHOULD HAVE) - Расширения (7-10 часов)

Критерии готовности:

- **MVP готов:** После недели 2 (8-10 часов)
- **Production готов:** После недели 3 (15-20 часов)
- **Полная платформа:** При реализации COULD HAVE (+4-6 часов)

Риски и митигация:

- **Технический риск:** Сложность интеграции безопасности
 - Митигация: Начать с базовых решений, постепенно усложнять
 - **Временной риск:** Недооценка времени на документацию
 - Митигация: Писать документацию параллельно с кодом
 - **Качественный риск:** Недостаточное тестирование
 - Митигация: Включить автоматические тесты в CI/CD
-

Анализ подготовлен: 6 июня 2025

Методика: MoSCoW приоритизация

Автор: Manus AI Agent