Публичная архитектура приватного мессенджера

FamilyLinkNext

Архитектура продукта и процесс разработки с AI агентами

Автор	Денис Балдин
Роль	Product Owner, архитектор и разработчик
Статус	Работающий продукт, production версия закрыта

Назначение документа. Документ даёт безопасное публичное представление архитектуры мессенджера, ключевых инженерных решений и моего процесса разработки с AI-агентами. Он не содержит исходного кода production-версии, адресов инфраструктуры, секретов, ключей доступа и эксплуатационных деталей.

Кратко о продукте

FamilyLinkNext — приватный веб-мессенджер для доверенных групп. Продукт объединяет личные и групповые чаты, голосовые сообщения, файлы, аудио- и видеозвонки, демонстрацию экрана, push-уведомления и управление устройствами. Основные требования к системе — конфиденциальность, предсказуемое поведение при сбоях и возможность развивать отдельные части продукта без полной переработки архитектуры.

Область	Решение
Клиент	PWA с единым интерфейсом для чатов, контактов, звонков и настроек
Сообщения	Отдельный relay-сервис, доставка по устройствам, статусы доставки и прочтения
Звонки	WebRTC-медиа через LiveKit, отдельный сигнальный и авторизационный контур
Данные	Локальное защищённое состояние клиента и серверные хранилища по назначению
Эксплуатация	Контейнеризация, reverse proxy, журналирование, диагностика и восстановление

Функциональные возможности

- личные и групповые диалоги, контакты и приглашения
- текстовые, голосовые сообщения и передача файлов
- статусы доставки и прочтения, повторная отправка после временных сбоев
- аудио- и видеозвонки, групповые конференции и демонстрация экрана
- push-уведомления и восстановление состояния после возвращения в приложение
- управление доверенными устройствами и отзыв доступа
- подписки и платежный сценарий
- административная диагностика и контроль состояния сервисов

Архитектура верхнего уровня

Система разделена на независимые контуры. Это уменьшает связанность и позволяет по-разному обрабатывать асинхронные сообщения, управляющие запросы и realtime-медиа.

Уровень	Ответственность	Основные технологии
Клиентское приложение	Пользовательские сценарии, локальное состояние, шифрование, обработка ошибок	JavaScript, PWA, Web APIs
Контур данных	Приём и доставка зашифрованных сообщений, очереди, статусы и повторные попытки	Node.js, Express, Redis
Контур управления	Авторизация устройств, контакты, приглашения, токены и служебные операции	REST API, SQLite
Контур звонков	Сигнализация и передача аудио, видео и экрана в реальном времени	LiveKit, WebRTC
Инфраструктура	TLS-терминация, маршрутизация, изоляция сервисов, запуск и наблюдаемость	Docker, Nginx, Linux

Поток личного сообщения

1. Клиент определяет получателя и его активные устройства.
2. Сообщение подготавливается и шифруется на стороне отправителя.
3. Для каждого устройства создаётся адресованная копия зашифрованного payload.
4. Relay принимает данные, сохраняет их до получения и не анализирует содержимое.
5. Клиент получателя забирает сообщение, расшифровывает его локально и обновляет статус.
6. При временной ошибке операция повторяется контролируемо, без создания дублей в интерфейсе.

Поток звонка

Клиент запрашивает краткоживущий токен, уведомляет устройства получателя и подключается к комнате LiveKit. Медиа передаётся по WebRTC, а приложение отдельно управляет состояниями вызова, разрешениями, повторным подключением и завершением сессии. Контур звонков отделён от доставки сообщений, поскольку realtime-медиа имеет другие требования к задержкам и сети.

Ключевые архитектурные решения

Решение	Зачем оно принято
Разделение данных управления и звонков	Каждый контур имеет собственные требования к задержкам, отказам и восстановлению. Сбой звонков не должен останавливать чат.
Адресация человека и устройства	Пользователь остаётся одной логической сущностью, но доставка выполняется на конкретные доверенные устройства.
Fan out на границе системы	Для нескольких устройств формируются отдельные зашифрованные копии без усложнения relay-сервиса.
Локальная криптографическая обработка	Сервер работает с непрозрачными данными и не должен получать открытое содержимое сообщений.
Явная деградация функций	Если недоступен realtime-контур, интерфейс честно отключает звонки, сохраняя доступные функции.
Изоляция инфраструктуры	Сервисы запускаются раздельно, имеют ограниченные зоны ответственности и могут обновляться независимо.

Надёжность и контроль ошибок

- тайм-ауты и ограниченные повторные попытки для внешних и внутренних запросов
- идемпотентная обработка операций, где повтор не должен создавать новую сущность
- сохранение промежуточного состояния при потере сети и восстановление после подключения
- раздельные пользовательские состояния загрузки, успеха, частичной доступности и ошибки
- журналирование технических событий без записи чувствительного содержимого
- проверка доступности зависимостей и диагностические сценарии для администратора

Безопасность и приватность

Архитектура строится по принципу минимального доверия к транспортному и серверному слоям. Ключевые операции выполняются на клиенте, а доступ к функциям ограничивается контекстом пользователя и устройства. Публичное описание намеренно не раскрывает конкретные адреса, секреты, схемы развертывания и внутренние механизмы администрирования.

- end to end шифрование содержимого сообщений
- отдельная идентичность пользователя и доверенных устройств
- отзыв устройства без замены пользовательской идентичности
- проверка доверия к серверу и обнаружение неожиданной смены ключа
- минимизация данных в push-уведомлениях и технических журналах
- секреты и токены передаются через защищённое окружение, а не хранятся в клиентском коде

Модель доверия и идентичности

Идентичность пользователя отделена от конкретного устройства и от ключей отдельного диалога. Это позволяет отзывать потерянное устройство, добавлять новое и развивать криптографию сессий без замены пользовательской идентичности.

Уровень	Назначение	Состояние
Identity Key	Постоянная криптографическая идентичность пользователя. Подписывает доверенные устройства и не применяется напрямую для шифрования сообщений.	Реализовано
Device Key	Отдельный ключ устройства. Может быть добавлен или отозван без смены Identity Key пользователя.	Реализовано
PreKeys и сессионные ключи	Устанавливают защищённый канал с офлайн-получателем и ограничивают последствия компрометации одной сессии.	Базовая схема реализована, усиление в roadmap
Первичное доверие	Контакт подтверждается через QR-рукопожатие или сравнением fingerprint по независимому каналу.	Реализовано на уровне продукта

Границы приватности

Система защищает содержимое сообщений, но не обещает абсолютную анонимность. Публичная модель явно разделяет защищаемые данные и метаданные, которые могут быть видны инфраструктурным участникам.

Участник	Что может быть видно	Что не должно быть доступно
Relay-сервис	IP-адрес, время соединения, размеры и частота зашифрованных payload	Открытое содержимое сообщений и ключи расшифрования
Push-провайдер	Факт доставки технического уведомления конкретному устройству	Текст сообщения, имя отправителя и содержимое диалога
Медиасервис	Факт звонка, технические параметры сессии и сетевые метаданные	История текстовой переписки и криптографические ключи чата
Клиентское устройство	Расшифрованные данные текущего пользователя в рамках его доступа	Данные других идентичностей и отозванных устройств

Реализовано и roadmap

Документ различает работающие механизмы и целевую архитектуру. Это важно для корректной оценки текущей безопасности и следующих этапов развития продукта.

Статус	Возможности
Реализовано	Единая идентичность пользователя, отдельные ключи устройств и их отзыв, QR-доверие, базовая схема PreKeys и X3DH, Sender Key для небольших групп, E2E содержимого сообщений, Server Key с TOFU-проверкой, push, файлы и звонки.
Следующие этапы	Полный X3DH и Double Ratchet, MLS для групп, социальное восстановление Identity Key, отдельный file relay для крупных вложений, relay-to-relay федерация, дополнительные транспорты и нативная оболочка.

Разработка с AI агентами

Я использовал Codex и Claude Code как инженерных исполнителей внутри управляемого процесса. Агенты помогали исследовать кодовую базу, предлагать изменения, реализовывать отдельные задачи и искать причины ошибок. Архитектурные решения, критерии готовности, проверка кода и выпуск оставались моей ответственностью.

Этап	Как организована работа
Формулировка	Проблема описывается через пользовательский сценарий, ограничения, входные данные и критерии результата.
Спецификация	До реализации фиксируются затрагиваемые компоненты, контракты API, модель данных, ошибки и требования к обратной совместимости.
Декомпозиция	Задача делится на небольшие изменения с понятными границами и проверяемым результатом.
Контекст агента	Агент получает релевантные файлы, архитектурные правила и запреты. Секреты и лишние данные в контекст не передаются.
Реализация	Изменения выполняются небольшими итерациями. После каждого этапа анализируется diff и влияние на соседние сценарии.
Верификация	Проверяются сборка, логи, основной и ошибочные сценарии, состояния интерфейса, сеть и сохранность существующих функций.
Выпуск	Изменение разворачивается контролируемо, наблюдается после запуска и имеет понятный путь отката.

Harness и управление контекстом

- единый рабочий репозиторий и архитектурные документы как источник ограничений
- задачи с явными критериями приёмки и списком неизменяемых компонентов
- выбор инструментов агента по задаче: поиск по коду, терминал, тесты, браузерная проверка
- разделение исследования, реализации и ревью на отдельные проходы

- контроль контекста: агент получает только нужную часть системы и актуальные зависимости
- ручная проверка критичных решений, особенно безопасности, данных и сетевого взаимодействия

Тестирование и отладка

Область	Что проверяется
Основные сценарии	Регистрация, вход, контакт, сообщение, файл, звонок, завершение и повторный вход
Негативные сценарии	Неверные данные, просроченные токены, запрет доступа, недоступность зависимости
Сеть	Потеря соединения, повторное подключение, задержка, повтор запроса и восстановление состояния
Несколько устройств	Доставка по устройствам, отзыв устройства, синхронизация статусов и отсутствие дублей
Регрессия	Смежные функции после изменения API, модели данных, маршрутизации или клиентского состояния
Production	Логи запуска, состояние контейнеров, доступность маршрутов и ошибки после релиза

Моя роль и результат

Я самостоятельно прошёл путь от идеи и пользовательских сценариев до архитектуры, реализации и развёртывания продукта. Управлял приоритетами, ставил задачи AI-агентам, проверял изменения, находил ошибки на стыке компонентов и принимал решения по упрощению или переработке архитектуры.

Зона ответственности	Выполненная работа
Продукт	Сценарии, приоритеты, ограничения, критерии готовности и последовательность релизов
Архитектура	Разделение компонентов, API-контракты, идентичность, доставка по устройствам и отказоустойчивость
Разработка	Клиентская логика, интеграции, серверные сервисы и исправление сложных межсервисных ошибок
AI workflow	Спецификации, управление контекстом агентов, постановка задач, review и проверка результата
Инфраструктура	Развёртывание, Docker, reverse proxy, TLS, диагностика и обновление сервисов

Текущее состояние

Рабочая production-версия развёрнута на личной инфраструктуре. Исходный код production-версии закрыт, поскольку содержит продуктовые и инфраструктурные детали. Для публичной проверки доступны демонстрационная версия, описание кейса и этот архитектурный обзор. Работающий продукт и более подробную архитектуру я готов показать на техническом интервью.

Материалы

Портфолио: portfolio.techdao.co.uk

GitHub: демонстрационная версия проекта и ссылка указываются в отклике

Контакт: Telegram @denisbaldin