Комплексный план миграции: Интеграция Rocket.Chat в проект «БЕСЕДКА» с повышенной безопасностью и масштабируемостью

Введение

Проект «БЕСЕДКА» представляет собой многофункциональную платформу, где раздел «Чат» признан ключевым элементом, необходимым для дальнейшего развития проекта. Текущее состояние этого раздела характеризуется значительными ограничениями: он предлагает лишь базовую возможность отправки сообщений, имеет «ужасный дизайн» с сообщениями в виде «плиток», полностью лишен функций ответа, цитирования и прикрепления медиафайлов. Такое положение дел существенно препятствует прогрессу проекта [User Query].

В ответ на эти критические недостатки, было принято стратегическое решение о внедрении Rocket.Chat. Этот шаг рассматривается как своевременное решение, призванное трансформировать текущий «очень печальный» раздел чата в полнофункциональный, безопасный и масштабируемый коммуникационный центр [User Query].

Цель настоящего отчета — предоставить подробный, пошаговый план проекта и набор детализированных инструкций (промптов) для ассистентов на базе искусственного интеллекта. Эти инструкции призваны обеспечить плавный переход к Rocket.Chat, уделяя первостепенное внимание безопасности пользователей (защите сайта от взлома, предотвращению утечек сообщений, шифрованию данных) и масштабируемости, необходимой для поддержки прогнозируемого роста числа пользователей (1000 пользователей к концу текущего года и около 5000 в течение 2-3 лет) [User Query]. Отчет также учитывает готовность к «радикальным архитектурным изменениям» и необходимость сохранения существующей проектной документации, особенно системы ролей и полномочий [User Query].

I. Rocket.Chat: Основа для безопасной и масштабируемой коммуникации

Этот раздел знакомит с Rocket.Chat, подробно описывая его функции, архитектурные особенности для обеспечения производительности и надежную систему безопасности, напрямую отвечающие ключевым требованиям проекта «БЕСЕДКА».

Обзор основных функций и преимуществ Rocket.Chat для «БЕСЕДКИ»

Rocket.Chat — это настраиваемая платформа с открытым исходным кодом, разработанная для общения в реальном времени, что делает ее идеальным решением для устранения текущих недостатков чата «БЕСЕДКИ».[1]

Платформа напрямую решает перечисленные проблемы текущего чата «БЕСЕДКИ»:

- **Полнофункциональная коммуникация:** Rocket.Chat предлагает широкий набор функций, включая прямые сообщения, приватные и публичные каналы/группы, аудио- и видеоконференции, демонстрацию экрана и обмен файлами. Поддерживаются Markdown, эмодзи, упоминания, редактирование и удаление сообщений, предварительный просмотр ссылок, а также полная история сообщений и стенограммы.[1] Эти возможности напрямую устраняют жалобы на отсутствие ответов, цитирования и прикрепления медиафайлов.
- **Дизайн и кастомизация:** Платформа предоставляет различные темы (светлая, темная, черная) и обширные возможности настройки пользовательского интерфейса с помощью пользовательских CSS, JavaScript и специализированной библиотеки UI Fuselage.[1] Это позволяет привести внешний вид чата в соответствие с существующими стандартами пользовательского интерфейса «БЕСЕДКИ» и принципом единого источника истины (SSOT), решая проблему «ужасного дизайна» и «сообщений в виде плиток» [User Query].
- **Открытый исходный код и самоуправление:** Открытый исходный код с лицензией MIT и поддержка концепции «BYOS (bring your own server)» [1]

соответствуют стремлению пользователя к полному контролю над своими данными и инфраструктурой, что критически важно для обеспечения безопасности и суверенитета данных.

Таким образом, комплексный набор функций Rocket.Chat предлагает полноценное решение, значительно превосходящее текущие базовые возможности обмена сообщениями в «БЕСЕДКЕ», что напрямую отвечает выраженному недовольству пользователя существующим чатом.

Таблица 1: Текущие ограничения чата «БЕСЕДКА» и соответствующие функции Rocket.Chat

Текущие ограничения чата «БЕСЕДКА»	Соответствующие функции Rocket.Chat
Ужасный дизайн	Настраиваемый пользовательский интерфейс через CSS/темы и библиотеку Fuselage [5]
Сообщения в виде плиток	Гибкое отображение сообщений, возможность изменения стилей через CSS [5]
Невозможно ответить на сообщения	Ветвящиеся ответы (threaded replies) [3]
Нет цитирования	Функциональность цитирования (как часть ответов и истории)
Нет прикрепления медиафайлов	Обмен файлами/медиафайлами (изображения, видео) [1]
Базовая отправка сообщений (и всё)	Полнофункциональная коммуникация: прямые сообщения, публичные/приватные каналы, аудио/видеоконференции, демонстрация экрана, Markdown, эмодзи, упоминания, редактирование/удаление сообщений, предпросмотр ссылок, история сообщений [1]

Архитектура Rocket.Chat и варианты развертывания для высокой доступности и производительности

Архитектура Rocket.Chat представляет собой комплексную, многокомпонентную систему, разработанную для обеспечения надежности, гибкости и масштабируемости.[8] Ключевые компоненты включают:

- **Сервер:** Может быть развернут как монолит (централизованное, унифицированное управление) или как микросервисная архитектура (независимые компоненты для повышения масштабируемости, балансировки нагрузки, высокой доступности и независимых обновлений).[8]
- **База данных:** Rocket.Chat использует MongoDB для хранения сообщений чата, информации о пользователях и системных конфигураций.[9]
- **Файловое хранилище:** Поддерживает локальное хранение на сервере или сетевые сервисы, такие как Amazon S3.[8]
- **Клиенты:** Веб-, настольные и мобильные приложения подключаются к серверу в основном через HTTP-запросы и WebSockets.[8]
- **API:** Все функции доступны через REST API и WebSockets, что упрощает интеграцию Rocket.Chat в существующие пользовательские интерфейсы.[8]

Rocket.Chat поддерживает различные методы развертывания, что критически важно для удовлетворения требований «БЕСЕДКИ» к масштабируемости и безопасности [11]:

- **Самостоятельное управление (On-Premise):** Обеспечивает полный контроль над инфраструктурой и данными. Рекомендуемые методы включают:
    - **Docker/Docker Compose:** Упрощает развертывание и обновления, популярен благодаря простоте использования.[11] Подходит для начальной настройки и небольших развертываний.
    - **Kubernetes (Helm Charts):** Обеспечивает высокую масштабируемость, позволяя масштабировать отдельные сервисы с помощью микросервисной архитектуры. Рекомендуется для пользователей с опытом работы с Kubernetes.[11]
    - **Snaps:** Быстрая и безопасная настройка для небольших развертываний, но *не рекомендуется для производственных или крупномасштабных развертываний* из-за отсутствия горизонтальной масштабируемости.[11]
- **Облачное размещение:** Доступны премиальные выделенные или общие облачные хостинги, однако акцент пользователя на суверенитете данных и безопасности указывает на предпочтительность самостоятельного размещения.[11]
- **Изолированное развертывание (Air-Gapped):** Для максимальной безопасности, работа без доступа к интернету.[11]

Пользователь прогнозирует **1000 пользователей к концу года и 5000 пользователей в течение 2-3 лет** [User Query]. Для развертываний, приближающихся к 1000 одновременных пользователей и выше, **рекомендуется микросервисная архитектура для оптимальной масштабируемости**.[15] Это критически важный аспект, поскольку прогнозы роста числа пользователей проекта «БЕСЕДКА» попадают именно в эту категорию. Хотя монолит, масштабированный с помощью нескольких экземпляров Docker, может обрабатывать тысячи одновременных пользователей, это может привести к «плато масштабирования» и увеличению потребления ЦП/памяти на каждый экземпляр.[16]

Следовательно, выбор архитектуры напрямую определяется ожидаемым ростом числа пользователей. Для достижения целевых показателей масштабируемости и производительности, особенно при прогнозируемых 5000 пользователях, необходимо использовать более сложную, но более надежную архитектуру.

Рекомендуемые аппаратные требования для 500-5000 одновременных пользователей (микросервисы, высокая доступность) включают:

- Rocket.Chat: 4-16 vCPU, 4-12 GiB ОЗУ, 40 ГБ хранилища.[17]
- MongoDB (на реплику): 2-8 vCPU, 4-16 GiB ОЗУ, 20-80 ГБ хранилища, при этом рекомендуется 3 реплики.[17]
- Файловое хранилище: Приблизительно 120 ГБ/год для 5000 пользователей.[17]

Учитывая прогнозируемый рост числа пользователей и готовность пользователя к «радикальным архитектурным изменениям», развертывание Rocket.Chat на **Kubernetes с микросервисной архитектурой** является наиболее подходящим и перспективным вариантом. Это обеспечивает высокую доступность и позволяет гранулированно масштабировать отдельные сервисы.[14] Для производственной среды рекомендуется использовать неконтейнеризованный набор реплик MongoDB.[14]

Необходимо отметить, что стремление пользователя к «максимально безопасному чату» и полному контролю над данными [User Query] указывает на предпочтение самостоятельного размещения. Хотя самостоятельное размещение обеспечивает «полный суверенитет данных» [12], оно также означает, что пользователь (или его AI-ассистенты) будет нести ответственность за все аспекты управления инфраструктурой, включая обновления, резервное копирование, мониторинг и усиление безопасности.[13] Это создает значительную операционную нагрузку по сравнению с SaaS или управляемым облачным

решением, и эти обязанности должны быть четко обозначены в плане миграции.

Таблица 2: Варианты развертывания Rocket.Chat и матрица масштабируемости для «БЕСЕДКИ»

Сценарий масштаба пользователей	Рекомендуемый метод развертывания	Ключевые аппаратные ресурсы (ЦП, ОЗУ, хранилище)	Обоснование/Преимущества	Связанная сложность/Соображения
Текущий (малый масштаб)	Docker Compose (для PoC/разработки)	Одноядерный (2 ГГц), 1 ГБ ОЗУ, 30 ГБ SSD [21]	Простота настройки, быстрое развертывание [12]	Не подходит для производственного масштаба [12]
Конец года (1000 пользователей)	Kubernetes + Микросервисы (Производство)	Rocket.Chat: 4-16 vCPU, 4-12 GiB ОЗУ, 40 ГБ хранилища; MongoDB (на реплику): 2-8 vCPU, 4-16 GiB ОЗУ, 20-80 ГБ хранилища (3 реплики) [17]	Высокая масштабируемость, устойчивость, гранулированный контроль, распределение нагрузки [8]	Требуется опыт работы с Kubernetes, более сложная начальная настройка [14]
2-3 года (5000 пользователей)	Kubernetes + Микросервисы (Производство)	Rocket.Chat: 16 vCPU, 12 GiB ОЗУ, 40 ГБ хранилища; MongoDB (на реплику): 4 vCPU, 16 GiB ОЗУ, 80 ГБ хранилища (3 реплики); Файловое хранилище: ~120 ГБ/год [17]	Оптимальная производительность для больших нагрузок, непрерывное масштабирование [12]	Непрерывный мониторинг и оптимизация ресурсов, регулярное тестирование под нагрузкой [12]

Углубленный анализ возможностей безопасности Rocket.Chat: Шифрование,

аутентификация и защита данных

Rocket.Chat разработан с учетом требований безопасности корпоративного уровня, предлагая полный контроль над суверенитетом данных и соответствием нормативам.[2]

Комплексные функции безопасности:

- **Сквозное шифрование (E2EE):** Позволяет читать сообщения только общающимся пользователям, предотвращая прослушивание. Может быть включено для приватных комнат и прямых сообщений по умолчанию, с возможностью шифрования загруженных файлов.[2]
    - Критическое ограничение, которое необходимо учитывать: зашифрованные сообщения в зашифрованных комнатах *не будут найдены* при поиске на стороне сервера.[23] Это существенный компромисс, который пользователь должен принять во внимание. Механизм E2EE включает хранение публичного ключа на сервере, а приватный ключ шифруется с помощью Мастер-ключа (полученного из пароля E2EE) и также хранится на сервере. Для каждой комнаты генерируются сессионные ключи, которые шифруются для каждого пользователя.[24]
- **Аутентификация и контроль доступа:**
    - **Двухфакторная аутентификация (2FA):** Обеспечивает дополнительный уровень защиты учетных записей пользователей.[1] Настоятельно рекомендуется включить.
    - **Единый вход (SSO):** Поддерживает интеграцию с LDAP, Active Directory, SAML и провайдерами OAuth (Google, GitHub, пользовательские OAuth) для централизованной аутентификации.[1] Это жизненно важно для интеграции с существующей системой управления пользователями «БЕСЕДКИ».
    - **Управление доступом на основе ролей (RBAC) / Пользовательские роли и разрешения:** Позволяет администраторам создавать новые роли и назначать им определенные разрешения, придерживаясь принципа наименьших привилегий (POLP).[2] Это имеет решающее значение для управления доступом к приватным чатам и другим функциям на основе существующего документа «БЕСЕДКИ» о ролях и полномочиях.
- **Защита данных:**
    - **Шифрование данных при передаче (TLS/HTTPS):** Необходимо для защиты веб-трафика между клиентами и сервером.[19] Требует обратного прокси-сервера (например, Nginx) для управления SSL.[18]

    - **Шифрование данных в состоянии покоя:** Внедрение шифрования диска для базы данных MongoDB является критически важным.[19] MongoDB Enterprise предлагает собственное шифрование.[29] Управление ключами может осуществляться через KMIP или локальный файл ключей.[29]
    - **Предотвращение потери данных (DLP):** Доступно через интеграцию приложений, позволяя администраторам устанавливать правила, ключевые слова и управлять каналами для защиты конфиденциальных данных.[3]
    - **Модерация контента:** Фильтрует вредоносный контент (спам, разжигание ненависти) через интеграцию приложений.[2]
    - **Гранулированные политики хранения данных:** Автоматически удаляют старые сообщения и файлы на основе пользовательских правил, глобально или для каждой комнаты.[3]
    - **Аудит сообщений:** Позволяет авторизованным пользователям проверять разговоры.[3]

Рекомендации по безопасности:

- **Межсетевые экраны и сегментация сети:** Защитите серверы Rocket.Chat и MongoDB строгими правилами межсетевого экрана, разрешая только необходимые порты. Сегментируйте сети (DMZ/VLAN) для ограничения горизонтального перемещения в случае компрометации части сети.[19]
- **Регулярные обновления и усиление защиты:** Поддерживайте актуальность Rocket.Chat, Jitsi (если интегрировано для видео) и операционной системы. Обновления устраняют уязвимости безопасности. Настройте автоматическое применение патчей безопасности или напоминания об обновлениях.[19]
- **Надежная политика паролей:** Введите требование к паролям длиной не менее 12 символов, включающих строчные, прописные буквы, цифры и символы.[20]
- **Удаление метаданных EXIF:** Включите эту опцию, если изображения не должны загружаться с метаданными, которые могут содержать конфиденциальную информацию, такую как геолокация.[20]
- **Ограничение типов файлов:** Разрешайте загрузку только необходимых форматов файлов; блокируйте потенциально опасные типы (.exe,.sh,.js), которые могут содержать вредоносный код.[20]

Важно отметить, что хотя Rocket.Chat является продуктом с открытым исходным кодом, некоторые расширенные функции безопасности и управления, такие как «пользовательские роли и разрешения» [3] и некоторые «расширенные функции управления идентификацией» в изолированных средах [25], привязаны к

премиальным/корпоративным планам.[27] Это означает, что для достижения максимально высокого уровня безопасности и гранулированного контроля может потребоваться подписка на премиальные планы Rocket.Chat, что является потенциальным дополнительным расходом помимо расходов на хостинг инфраструктуры.

Таблица 3: Контрольный список реализации ключевых функций безопасности для «БЕСЕДКИ»

Функция безопасности	Возможности Rocket.Chat	Реализация/Конфигурация в «БЕСЕДКЕ»	Ссылки на источники
Сквозное шифрование (E2EE)	Встроено, настраивается для каждой комнаты [23]	Включить для приватных комнат и прямых сообщений; учесть ограничение поиска [23]	[23]
Двухфакторная аутентификация (2FA)	Поддерживает несколько методов [3]	Обязательно для всех пользователей; настроить SMTP для 2FA по электронной почте [20]	[3]
Управление доступом на основе ролей (RBAC)	Настраиваемые роли/разрешения [27]	Сопоставить существующие роли «БЕСЕДКИ»; следовать принципу наименьших привилегий [20]	[27]
Шифрование данных в состоянии покоя	Функция MongoDB Enterprise [29]	Внедрить шифрование диска для MongoDB; рассмотреть KMIP или файл ключей [19]	[19]
Модерация контента и DLP	Через интеграцию приложений [3]	Интегрировать соответствующие приложения для фильтрации и предотвращения утечек [3]	[3]

Сегментация сети	Внешняя инфраструктура [19]	Разместить чат-серверы в DMZ или отдельном VLAN [19]	[19]
Шифрование данных при передаче (TLS/HTTPS)	Требует обратного прокси [20]	Настроить обратный прокси (Nginx) для принудительного использования HTTPS [18]	[18]
Удаление метаданных EXIF	Настраиваемая опция [20]	Включить для всех загружаемых изображений [20]	[20]
Ограничение типов файлов	Настраиваемая опция [20]	Разрешить только необходимые типы файлов; блокировать потенциально опасные [20]	[20]

II. Стратегические фазы миграции: От концепции к реализации

Этот раздел подробно описывает практические шаги по миграции функционала чата «БЕСЕДКИ» на Rocket.Chat, структурированные по трем отдельным фазам.

Фаза 1: Обнаружение и подготовка

Начальный этап миграции включает глубокий анализ требований и тщательную подготовку инфраструктуры.

- **Детальный анализ требований:**
  - **Функциональные:** Необходимо подтвердить конкретные потребности: публичный чат, приватный чат (с контролируемым администратором

доступом), ответы, цитирование, прикрепление медиафайлов (изображения, видео), форматирование текста (Markdown), история сообщений, упоминания пользователей, эмодзи.[1]
    - **Нефункциональные:**
        - **Масштабируемость:** Поддержка 1000 пользователей к концу года и 5000 пользователей в течение 2-3 лет [User Query]. Это означает необходимость в надежной инфраструктуре, способной обрабатывать 100-500+ одновременных пользователей.[15]
        - **Безопасность:** Максимальная безопасность пользователей, защита сайта от взлома, предотвращение утечки сообщений из чата, обеспечение шифрования сообщений, безопасный доступ к приватным/публичным чатам [User Query].
        - **Производительность:** Низкая задержка, высокая отзывчивость для общения в реальном времени.
        - **Доступность:** Высокое время бесперебойной работы для постоянного доступа к чату.
    - **UI/UX:** Соблюдение стандартов пользовательского интерфейса «БЕСЕДКИ» и SSOT для визуальной согласованности [User Query].
- **Выбор оптимальной модели развертывания Rocket.Chat для масштаба «БЕСЕДКИ»:**
    - **Рекомендация:** Развертывать Rocket.Chat с использованием **Kubernetes с микросервисной архитектурой** для производственных сред, особенно учитывая прогнозируемый рост числа пользователей до 5000 и потребность в высокой доступности и производительности.[12]
    - **Обоснование:** Микросервисы позволяют независимое масштабирование отдельных компонентов, распределяя нагрузку и улучшая производительность без единой точки отказа.[8] Kubernetes обеспечивает оркестрацию для этого.[12] Это также повышает устойчивость за счет балансировки нагрузки и высокой доступности [8], а также упрощает обслуживание, позволяя обновлять или заменять отдельные компоненты без влияния на всю систему.[8] Самостоятельное развертывание обеспечивает полный контроль над данными.[12]
    - **Предварительные условия для развертывания Kubernetes:** Настроенный кластер Kubernetes, установленный Helm v3 [14], зарегистрированное доменное имя, указывающее на IP-адрес сервера [14], правильная конфигурация межсетевого экрана для разрешения HTTPS-трафика [14], и неконтейнеризованный набор реплик MongoDB для производственной среды.[14]
    - **Альтернатива (для тестирования/начальной фазы):** Docker Compose

может быть использован для быстрой локальной настройки, чтобы ознакомиться с Rocket.Chat, но он не рекомендуется для прогнозируемого производственного масштаба.[12]

- **Начальная настройка и конфигурация экземпляра Rocket.Chat:**
  - **Получение Rocket.Chat:** Rocket.Chat — это серверное приложение, а не просто «файлы». Его можно развернуть из исходного кода (репозиторий GitHub[31]), образов Docker [11] или Helm-чартов Kubernetes.[11] Пользователь должен понимать, что он развертывает *сервер*, а не просто скачивает архив.
  - **Шаги развертывания (высокоуровневые для промптов AI):**
    - Предоставить сервер Linux (например, Ubuntu, CentOS, Debian), соответствующий рекомендованным аппаратным спецификациям для 500-5000 пользователей.[17]
    - Установить Docker и Docker Compose V2 (если используется для начального тестирования/PoC).[30]
    - Установить Kubernetes и Helm (для производственного развертывания).[14]
    - Развернуть Rocket.Chat с использованием официального Helm-чарта для Kubernetes.[14]
    - Настроить MongoDB (предпочтительно как отдельный, неконтейнеризованный набор реплик для производственной среды).[14]
    - **Начальный доступ:** Доступ к Rocket.Chat через localhost:3000/ (для локальной настройки Docker) или настроенное доменное имя для Kubernetes.[18]
    - **Настройка администратора:** Завершить начальный мастер настройки, создав учетную запись основного администратора и информацию об организации.[30]
    - **Критический шаг безопасности: Настройка обратного прокси (Nginx/Apache) с TLS/HTTPS:** Rocket.Chat не обрабатывает SSL напрямую. Обратный прокси необходим для шифрования всего веб-трафика.[18] Настроить параметры ROOT_URL и SITE_URL в Rocket.Chat так, чтобы они соответствовали домену, используемому обратным прокси, для обеспечения корректных ссылок в электронных письмах и пользовательском интерфейсе.[18]
    - **Первоначальное рассмотрение 2FA:** По умолчанию Rocket.Chat требует 2FA для новых пользователей. Если SMTP-сервер для подтверждения по электронной почте не настроен немедленно, 2FA может потребоваться временно отключить на этапе начальной

настройки.[18]
Повторное включение и настройка 2FA с правильно работающим SMTP-сервером является критически важной последующей задачей по обеспечению безопасности.

Фаза 2: Основная интеграция и кастомизация

Эта фаза фокусируется на интеграции Rocket.Chat с существующими системами «БЕСЕДКИ» и настройке его внешнего вида.

- **Интеграция аутентификации и авторизации пользователей с существующей системой «БЕСЕДКИ» (SSOT, RBAC):**
  - **Синхронизация пользователей:** Определить оптимальный метод синхронизации пользователей «БЕСЕДКИ» с Rocket.Chat. Rocket.Chat поддерживает:
    - **LDAP/Active Directory:** Для синхронизации имен пользователей, уникальных идентификаторов и управления учетными записями пользователей (создание, обновление, удаление).[25] Это сильный кандидат, если «БЕСЕДКА» использует каталог, совместимый с LDAP.
    - **OAuth/SAML:** Для единого входа (SSO), позволяющего пользователям входить в «БЕСЕДКУ» один раз и получать доступ к Rocket.Chat без отдельных учетных данных.[25] Это обеспечивает бесшовный пользовательский опыт.
    - **Пользовательская интеграция API:** Если «БЕСЕДКА» имеет пользовательскую базу данных, используйте REST API Rocket.Chat (/api/v1/import.addUsers, /api/v1/users.create) для создания/управления пользователями.[10]
  - **Сопоставление управления доступом на основе ролей (RBAC):**
    - Существующий документ пользователя «система распределения ролей и полномочий» является первостепенным и *не должен быть изменен* [User Query].
    - Сопоставить существующие роли «БЕСЕДКИ» (например, «владелец», «админ», «пользователь») с пользовательскими ролями Rocket.Chat.[3]
    - Создать пользовательские роли в Rocket.Chat (Администрирование > Рабочее пространство > Разрешения > Новая роль) и назначить им определенные разрешения.[27]
    - Убедиться, что разрешения, такие как create-p (создание приватных

групп), add-user-to-any-p-room (добавление пользователей в приватные комнаты) и assign-roles, правильно назначены ролям владельца/администратора «БЕСЕДКИ».[28]
    - Реализовать принцип наименьших привилегий (POLP).[20]
- **Реализация функционала приватного и публичного чата с административным контролем:**
    - **Публичные чаты:** Создавать публичные каналы с использованием разрешения create-c.[28] Они будут видны всем пользователям.
    - **Приватные чаты:**
        - Создавать приватные группы с использованием метода Realtime API createPrivateGroup или REST API.[34] Это позволяет указать имя и начальных участников.
        - **Контроль доступа администратором:** Владелец «БЕСЕДКИ» может управлять доступом к приватным чатам через свою админ-панель. Этот функционал будет использовать API Rocket.Chat:
            - createPrivateGroup для создания новых приватных комнат.[34]
            - /api/v1/channels.invite или разрешение add-user-to-any-p-room для добавления конкретных пользователей в приватный канал.[28]
            - Это гарантирует, что владелец «раздает доступ» [User Query], добавляя пользователей в приватный чат.

Таблица 4: Точки API/интеграции Rocket.Chat для управления пользователями и комнатами в «БЕСЕДКЕ»

Требование «БЕСЕДКИ»	API/Метод Rocket.Chat	Назначение/Описание	Требуемые разрешения	Ссылки на источники
Синхронизация пользователей	/api/v1/import.addUsers (REST API)	Импорт существующих пользователей из «БЕСЕДКИ» в Rocket.Chat [33]	run-import [33]	[33]
Создание приватного чата	createPrivateGroup (Realtime API)	Программное создание приватных комнат [34]	create-p [34]	[34]
Предоставление доступа к приватному чату	/api/v1/channels.invite (REST API)	Добавление конкретных пользователей в приватный	add-user-to-any-p-room [28]	[36]

		канал [36]		
Управление ролями пользователей	assign-roles (разрешение)	Назначение пользовательских ролей синхронизированным пользователям [28]	assign-roles, view-user-administration [28]	[28]

- **Настройка UI/UX Rocket.Chat в соответствии со стандартами UI «БЕСЕДКИ»:**
  - **Метод интеграции:** Для начальной интеграции, которая требует минимальных изменений в существующей архитектуре, рекомендуется встраивание Rocket.Chat с использованием **iframe**.[5] Это позволяет отображать весь пользовательский интерфейс Rocket.Chat на странице «БЕСЕДКИ», минимизируя прямые модификации существующего кода фронтенда «БЕСЕДКИ». Интеграция iframe также поддерживает аутентификацию пользователей через существующую страницу входа «БЕСЕДКИ» [37], обеспечивая бесшовный пользовательский путь.
  - **Темизация и стилизация:**
    - Доступ к настройкам макета Rocket.Chat (Администрирование > Рабочее пространство > Настройки > Макет) позволяет настраивать цвета с использованием существующих выражений или палитры цветов.[5]
    - Применение пользовательских CSS-стилей позволяет переопределить стандартные стили и привести их в соответствие со стандартами пользовательского интерфейса «БЕСЕДКИ».[5] Это включает настройку цветов, шрифтов, отступов и внешнего вида компонентов, чтобы они соответствовали существующей эстетике «БЕСЕДКИ» (например, «плитки» для сообщений могут быть перестилизованы).
    - Использование переменных темизации Rocket.Chat, таких как primary-background-color, primary-font-color, success-color [6], позволяет напрямую связать их с существующими определениями в документе SSOT «БЕСЕДКИ», обеспечивая программную согласованность дизайна на всей интегрированной платформе.
    - Использование компонентов библиотеки Fuselage Rocket.Chat рекомендуется для обеспечения согласованности, если в дальнейшем будет осуществляться более глубокая интеграция пользовательского интерфейса.[7]

        - Обеспечение согласованности с документами «унифицированный интерфейс» и «единый источник истины, SSOT» [User Query].
- **Включение основных функций чата: Ответы, цитирование, обмен медиафайлами и многое другое:**
    - Большинство желаемых функций являются **встроенными** в Rocket.Chat и будут доступны сразу после развертывания:
        - **Загрузка/обмен файлами:** Поддерживается для различных медиафайлов.[1]
        - **Ответы и ветвящиеся чаты:** Rocket.Chat поддерживает организованные беседы с командами, каналами, обсуждениями или ветками.[3]
        - **Цитирование:** Неявно поддерживается через ответы на сообщения и историю.
        - **Упоминания:** Использование символа @ для уведомления пользователей.[1]
        - **Эмодзи и реакции:** Доступны для выразительного общения.[1]
        - **Предварительный просмотр ссылок:** Автоматически генерирует предварительный просмотр для общих ссылок.[1]
        - **Отчеты о прочтении:** Показывает список пользователей, прочитавших сообщение, с отметкой времени.[3]
        - **Функция поиска:** Расширенные возможности поиска [1], однако необходимо помнить об ограничении E2EE.[23]

Фаза 3: Усиление безопасности, оптимизация производительности и тестирование

Эта фаза направлена на обеспечение максимальной безопасности и производительности системы, а также на комплексное тестирование.

- **Расширенная конфигурация безопасности:**
    - **Включение сквозного шифрования (E2EE):** Активировать E2EE в настройках администрирования.[23] Включить по умолчанию для новых приватных комнат и прямых сообщений, а также включить шифрование файлов.[23]
      Критически важно, что сообщения E2EE не подлежат поиску на стороне сервера.[23]
    - **Принудительное использование двухфакторной аутентификации**

(2FA): Убедиться, что 2FA включена для всех пользователей и/или определенных ролей.[3] Настроить SMTP-сервер для 2FA на основе электронной почты или интегрировать с другими методами 2FA.
    - **Внедрение строгих политик паролей:** Настроить политики паролей, требующие не менее 12 символов, включая строчные, прописные буквы, цифры и символы.[20]
    - **Настройка модерации контента и предотвращения потери данных (DLP):** Изучить и интегрировать доступные приложения для модерации контента (фильтрация спама/ненавистнических высказываний) и DLP (предотвращение утечки конфиденциальных данных).[3]
    - **Установка гранулированных политик хранения данных:** Определить правила автоматического удаления старых сообщений и файлов, глобально или для каждой комнаты, в соответствии с потребностями «БЕСЕДКИ» в хранении данных.[3]
    - **Сетевая безопасность:**
        - Усилить строгие **правила межсетевого экрана** для серверов Rocket.Chat и MongoDB, разрешая только необходимые порты (например, 80/443 для веб, порт по умолчанию MongoDB).[19]
        - Внедрить **сегментацию сети** (DMZ или отдельный VLAN) для чат-серверов, чтобы изолировать их от других частей «БЕСЕДКИ» и ограничить горизонтальное перемещение в случае взлома.[19]
        - Обеспечить принудительное использование **HTTPS/TLS** для всей связи между клиентом и сервером через обратный прокси.[19]
    - **Усиление защиты сервера:** Отключить ненужные службы, удалить учетные записи по умолчанию, защитить доступ по SSH (аутентификация на основе ключей, отключение входа root).[19]
    - **Удаление метаданных EXIF:** Включить эту настройку для удаления конфиденциальных метаданных из загружаемых изображений.[20]
    - **Ограничение типов файлов:** Настроить разрешенные типы файлов для загрузки, чтобы предотвратить выполнение вредоносного кода.[20]
    - **Принцип наименьших привилегий (POLP):** Регулярно проверять и аудировать роли и разрешения пользователей, гарантируя, что пользователи имеют только минимально необходимый доступ для выполнения своих задач.[20]
- **Оптимизация масштабируемости для 1000-5000 пользователей:**
    - **Развертывание микросервисов:** Подтвердить правильность настройки Kubernetes/микросервисной архитектуры для обработки прогнозируемой нагрузки.[14]
    - **Мониторинг ресурсов:** Внедрить непрерывный мониторинг

производительности сервера (ЦП, ОЗУ, ввод/вывод диска, сеть) как для экземпляров Rocket.Chat, так и для MongoDB.[15] Это поможет выявить узкие места по мере роста числа пользователей.
    - **Горизонтальное масштабирование:** Быть готовым к горизонтальному масштабированию служб Rocket.Chat путем добавления дополнительных экземпляров/подов в кластере Kubernetes по мере необходимости.[9]
    - **Оптимизация MongoDB:** Убедиться, что MongoDB настроена для высокой производительности и доступности (набор реплик, достаточные ресурсы).[14]
    - **Нагрузочное тестирование:** Использовать инструменты, такие как Gatling, JMeter, Locust или K6, для имитации пользовательской нагрузки (1000 и 5000 одновременных пользователей) и оценки устойчивости и производительности системы при стрессовых условиях.[12] Это поможет выявить и устранить узкие места в производительности заблаговременно.
- **Комплексная стратегия тестирования (функциональное, безопасность, производительность, приемочное тестирование):**
    - **Функциональное тестирование:** Проверить правильность работы всех функций Rocket.Chat (обмен сообщениями, обмен файлами, приватные/публичные комнаты, ответы, цитирование) и их бесшовную интеграцию с «БЕСЕДКОЙ».
    - **Тестирование безопасности:** Провести сканирование уязвимостей и тестирование на проникновение [19] на развернутом экземпляре Rocket.Chat и его точках интеграции с «БЕСЕДКОЙ». Проверить правильность работы всех настроек безопасности (2FA, E2EE, RBAC).
    - **Тестирование производительности:** Выполнить нагрузочные тесты, имитирующие 1000 и 5000 пользователей, чтобы убедиться, что система остается отзывчивой и стабильной при ожидаемых пиковых нагрузках.[12]
    - **Приемочное тестирование пользователями (UAT):** Привлечь пользователей «БЕСЕДКИ» для тестирования интегрированного чата, уделяя особое внимание удобству использования, согласованности дизайна (со стандартами UI «БЕСЕДКИ») и общему пользовательскому опыту.

III. Архитектурное влияние и эволюция документации

Этот раздел анализирует необходимые изменения в существующей архитектуре «БЕСЕДКИ» и описывает комплексный план обновления проектной документации, с акцентом на сохранение критически важного документа о ролях и полномочиях.

Анализ требуемых архитектурных изменений в «БЕСЕДКЕ» для бесшовной интеграции

Интеграция Rocket.Chat потребует изменений как во фронтенде, так и в бэкенде проекта «БЕСЕДКА».

- **Влияние на фронтенд:**
  - **Точка интеграции:** Наиболее непосредственным и наименее «радикальным» архитектурным изменением для фронтенда «БЕСЕДКИ» будет встраивание клиента Rocket.Chat с использованием **iframe**.[5] Это позволяет отображать весь пользовательский интерфейс Rocket.Chat на странице «БЕСЕДКИ», минимизируя прямые модификации существующего кода фронтенда «БЕСЕДКИ». Такой подход позволяет проверить работоспособность решения Rocket.Chat и его функций, прежде чем приступать к более глубокой и сложной архитектурной перестройке существующего фронтенда «БЕСЕДКИ».
  - **Поток аутентификации:** Интеграция iframe поддерживает аутентификацию пользователей через существующую страницу входа «БЕСЕДКИ» [37], обеспечивая бесшовный пользовательский путь. Это означает, что «БЕСЕДКА» останется основным центром аутентификации.
  - **Кастомизация пользовательского интерфейса:** Хотя iframe содержит пользовательский интерфейс Rocket.Chat, его внешний вид может быть значительно настроен с помощью внутренних CSS/темизационных опций Rocket.Chat [5] для соответствия внешнему виду «БЕСЕДКИ», что снижает необходимость для «БЕСЕДКИ» напрямую манипулировать рендерингом Rocket.Chat.
  - **Потенциальная более глубокая интеграция (будущее/опционально):** Если пользователь желает более гранулированного контроля над отдельными элементами пользовательского интерфейса или действительно нативного ощущения, более глубокая интеграция будет включать использование библиотеки Fuselage Rocket.Chat [7] для создания нативных компонентов чата «БЕСЕДКИ», которые взаимодействуют с API

Rocket.Chat. Это более значительный архитектурный сдвиг для фронтенда «БЕСЕДКИ».

- **Влияние на бэкенд:**
  - **Управление пользователями и синхронизация:** Бэкенд «БЕСЕДКИ» должен будет интегрироваться с API Rocket.Chat для управления пользователями. Это включает:
    - **Первоначальный импорт пользователей:** Одноразовый или постоянный процесс синхронизации для создания существующих пользователей «БЕСЕДКИ» в Rocket.Chat.[33]
    - **Постоянное управление жизненным циклом пользователей:** API для создания новых пользователей, обновления профилей и деактивации/удаления пользователей в Rocket.Chat при изменении в основной базе данных пользователей «БЕСЕДКИ».[28]
    - **Интеграция аутентификации:** Внедрение SSO (LDAP, OAuth, SAML) между системой аутентификации «БЕСЕДКИ» и Rocket.Chat.[25]
  - **Контроль доступа к приватным чатам:** Админ-панель «БЕСЕДКИ» должна будет вызывать API Rocket.Chat для:
    - Создания приватных комнат (Realtime API createPrivateGroup).[34]
    - Добавления/удаления пользователей из приватных комнат (REST API /api/v1/channels.invite).[36]
  - **Хранение данных:** Rocket.Chat использует MongoDB.[9] Существующие данные «БЕСЕДКИ» останутся в ее текущей базе данных. Только данные, связанные с чатом (сообщения, профили пользователей, конфигурации комнат), будут храниться в MongoDB Rocket.Chat. Прямая миграция основных данных «БЕСЕДКИ» в MongoDB не подразумевается.

Предлагаемые обновления ключевой проектной документации

Обновление документации является критически важным для поддержания актуальности и согласованности проекта.

- **Мастер-документация (общий план проекта):**
  - **Добавить:** Новый раздел, подробно описывающий интеграцию Rocket.Chat: архитектуру (Kubernetes/микросервисы), модель развертывания, компоненты сервера и поток данных. Обзор роли Rocket.Chat как основного коммуникационного модуля. Высокоуровневая архитектура безопасности, включая вклад функций безопасности

Rocket.Chat в общую безопасность сайта. План масштабирования и прогнозируемые требования к ресурсам для 1K и 5K пользователей. Точки интеграции с существующим управлением пользователями, аутентификацией и админ-панелью «БЕСЕДКИ».
    - **Удалить:** Любые подробные архитектурные описания старой, рудиментарной системы чата.
- **Стандарты UI «БЕСЕДКА» (специфично для раздела чата):**
    - **Добавить:** Новые рекомендации по интеграции и настройке пользовательского интерфейса Rocket.Chat для обеспечения визуальной согласованности с брендом «БЕСЕДКИ». Конкретные переменные CSS и точки настройки в параметрах макета Rocket.Chat [5], которые соответствуют принципам дизайна «БЕСЕДКИ». Примеры того, как нативные компоненты Rocket.Chat (например, пузырьки сообщений, поля ввода, списки пользователей) будут стилизованы в соответствии с эстетикой «БЕСЕДКИ». Инструкции по использованию пользовательского JavaScript в макете Rocket.Chat для конкретного поведения пользовательского интерфейса.[5]
    - **Удалить:** Любые спецификации UI/UX, относящиеся исключительно к «плиткам» или устаревшим элементам дизайна старого чата.
- **Единый источник истины (SSOT) для элементов интерфейса:**
    - **Добавить:** Обновить SSOT, чтобы включить основные компоненты пользовательского интерфейса Rocket.Chat и их соответствующие токены/переменные дизайна.[6] Сопоставить переменные темизации Rocket.Chat (например, primary-background-color, primary-font-color, success-color из [6]) с установленной палитрой цветов и типографикой «БЕСЕДКИ». Документировать, как компоненты библиотеки Fuselage Rocket.Chat [7] соответствуют SSOT, особенно если планируется более глубокая интеграция на уровне компонентов.
    - **Удалить:** Ссылки на любые элементы пользовательского интерфейса из старого чата, которые не переносятся.
- **Стратегия перекрестных ссылок между документами:**
    - Внедрить четкие перекрестные ссылки (например, гиперссылки) между Мастер-документацией, Стандартами UI и SSOT. Например, раздел чата в Мастер-документации должен напрямую ссылаться на соответствующие записи в Стандартах UI и SSOT для получения подробной информации о дизайне.
    - Убедиться, что документ о ролях и полномочиях правильно ссылается из

Мастер-документации и новых разделов интеграции чата.

Таблица 5: План обновления проектной документации для «БЕСЕДКИ»

Название документа	Требуемое действие	Конкретное содержание/Изменения	Стратегия перекрестных ссылок
Мастер-документация	Обновить	Добавить новый раздел об архитектуре Rocket.Chat (Kubernetes/микросервисы), ключевых компонентах, потоке данных и обзоре безопасности. Удалить устаревшую информацию о предыдущей системе чата.	Ссылки на Стандарты UI и SSOT
Стандарты UI «БЕСЕДКА»	Пересмотреть	Добавить подраздел для интерфейса чата, детализирующий стилизацию элементов UI Rocket.Chat с использованием пользовательских CSS и опций темизации для соответствия визуальным рекомендациям «БЕСЕДКИ». Включить примеры сопоставленных переменных цвета и типографики.	Ссылки из Мастер-документации
Единый источник истины (SSOT)	Включить	Включить основные компоненты UI Rocket.Chat и их соответствующие	Ссылки из Мастер-документации и Стандартов UI

		токены/переменные дизайна. Явно сопоставить переменные темизации Rocket.Chat (например, primary-background-color из [6]) с установленными определениями системы дизайна «БЕСЕДКИ».	
Документ о ролях и полномочиях	Сохранить/ссылаться	Без изменений; ссылаться как на неизменный источник логики доступа.	Ссылаться из Мастер-документации

Сохранение целостности документа о ролях и полномочиях

Пользователь явно указывает, что этот документ «очень важен» и «не изменен в базовом виде», что означает его неизменность [User Query].

- **Стратегия интеграции:** Вместо изменения этого документа процесс миграции будет сосредоточен на **сопоставлении** его определений с настраиваемой системой управления доступом на основе ролей (RBAC) Rocket.Chat.
  - **Сопоставление:** Каждая роль и связанные с ней разрешения, определенные в документе «БЕСЕДКИ», будут тщательно переведены в пользовательские роли и разрешения в панели администрирования Rocket.Chat.[27]
  - **API для управления:** AI-ассистенты будут использовать API Rocket.Chat (например, разрешение assign-roles[28]) для программного назначения этих сопоставленных ролей пользователям.
  - **Без прямых модификаций:** Оригинальный документ о ролях и полномочиях будет служить «единым источником истины» для логики доступа, но Rocket.Chat будет настроен на *принудительное применение*

этих правил, а не на их определение.

Это обеспечивает, что система управления доступом на основе ролей (RBAC) становится уровнем соответствия и контроля. Настойчивость пользователя в отношении неизменности документа о ролях/разрешениях, в сочетании с необходимостью «доступа к приватному чату, распределяемого владельцем», указывает на сильные требования к управлению и соответствию. Гранулированная RBAC Rocket.Chat [27] напрямую поддерживает это, позволяя создавать пользовательские роли и разрешения. Это не просто вопрос функций; это вопрос поддержания организационного контроля и соблюдения заранее определенных политик доступа. Следовательно, инструкции для AI-ассистентов по управлению пользователями и комнатами должны строго соответствовать логике, вытекающей из этого неизменного документа, гарантируя, что доступ к приватному чату, например, предоставляется только пользователям с определенными ролями «БЕСЕДКИ», которые затем сопоставляются с соответствующими разрешениями Rocket.Chat.

IV. Структура промптов для AI-ассистентов

В этом разделе излагаются общие принципы создания эффективных промптов для технических задач и предоставляются структурированные шаблоны для ключевых фаз и задач миграции. Промпты будут достаточно детализированы для выполнения AI, оставаясь при этом понятными для пользователя, не являющегося программистом.

Общие принципы создания эффективных промптов для технических задач

- **Ясность и специфичность:** Каждая инструкция должна быть однозначной, избегая жаргона, где это возможно, или четко его объясняя. Определить входные данные, ожидаемые результаты и критерии успеха.
- **Пошаговая детализация:** Разбить сложные задачи на атомарные, выполнимые шаги.
- **Насыщенность контекстом:** Предоставить достаточную справочную информацию (например, текущее состояние системы, желаемый результат,

ссылки на соответствующую документацию Rocket.Chat).
- **Итеративность и модульность:** Разработать промпты для отдельных задач или небольших модулей, что позволит тестировать и проверять на каждом этапе.
- **Обработка ошибок и проверка:** Включить инструкции по проверке ошибок, логированию вывода и проверке успешного выполнения задач.
- **Безопасность прежде всего:** Явно включить соображения безопасности и лучшие практики в соответствующие промпты.
- **Ссылки на документацию:** Направлять AI на конкретные URL-адреса документации Rocket.Chat (фрагменты), где можно найти подробную информацию.
- **Цикл обратной связи с пользователем:** Включить шаги для сообщения о прогрессе, проблемах и запроса разъяснений у пользователя.

Структурированные шаблоны промптов для каждой фазы миграции и ключевой задачи

Шаблон промпта 1: Развертывание сервера Rocket.Chat (Kubernetes с микросервисами)

- **Цель:** Развернуть высокодоступный и масштабируемый экземпляр Rocket.Chat в кластере Kubernetes, настроенный для производственного использования.
- **Контекст:** «БЕСЕДКА» нуждается в чат-решении, способном поддерживать до 5000 пользователей с высокой безопасностью и производительностью. Kubernetes с микросервисной архитектурой является выбранной моделью развертывания.
- **Предварительные условия:** Доступ к кластеру Kubernetes, установленный Helm v3, зарегистрированное доменное имя (например, chat.besedka.com) и административные учетные данные для сервера.
- **Шаги для AI-ассистента:**
    1. **Предоставление сервера:** «Убедитесь, что базовая серверная инфраструктура для кластера Kubernetes соответствует рекомендованным спецификациям для 5000 одновременных

пользователей: Rocket.Chat (4-16 vCPU, 4-12 GiB ОЗУ, 40 ГБ хранилища) и MongoDB (3 реплики, каждая по 2-8 vCPU, 4-16 GiB ОЗУ, 20-80 ГБ хранилища). См. [17] для подробных аппаратных спецификаций».

2. **Добавление Helm-чарта:** «Добавьте официальный репозиторий Helm-чартов Rocket.Chat в Helm. Используйте команду: helm repo add rocketchat https://rocketchat.github.io/helm-charts. Проверьте успешное добавление. (См. [14])».
3. **Определение конфигурации:** «Создайте файл values.yaml для определения конфигураций развертывания. Этот файл должен указывать доменное имя, класс контроллера ingress и отключать развертывание MongoDB по умолчанию в Helm-чарте, так как будет использоваться внешний набор реплик MongoDB. (См. [14])».
4. **Настройка внешней MongoDB:** «Настройте отдельный, неконтейнеризованный набор реплик MongoDB (3 реплики) на выделенных серверах, убедившись, что он соответствует требованиям к производительности и безопасности для 5000 пользователей. Настройте его для высокой доступности. (См. [14])».
5. **Установка Rocket.Chat:** «Установите Rocket.Chat в кластере Kubernetes, используя настроенный файл values.yaml и Helm-чарт. Используйте команду: helm install rocketchat -f values.yaml rocketchat/rocketchat. (См. [14])».
6. **Завершение мастера начальной настройки:** «Получите доступ к развернутому экземпляру Rocket.Chat через настроенный домен (например, https://chat.besedka.com) и завершите мастер начальной настройки, создав основную учетную запись администратора».
7. **Конфигурация обратного прокси:** «Настройте обратный прокси-сервер (например, Nginx) перед Rocket.Chat для обработки SSL/TLS-шифрования (HTTPS). Убедитесь, что параметры ROOT_URL и SITE_URL в администрировании Rocket.Chat соответствуют публичному домену. (См. [18])».
8. **Проверка:** «Убедитесь, что все поды Rocket.Chat запущены (kubectl get pods) и проверьте логи на наличие ошибок (kubectl logs <pod-name>). Подтвердите, что Rocket.Chat доступен по HTTPS на указанном домене. (См. [14])».
9. **Отчет:** «Предоставьте сводку состояния развертывания, включая URL-адрес доступа и любые возникшие проблемы».

Шаблон промпта 2: Интеграция аутентификации пользователей и RBAC

- **Цель:** Интегрировать существующую аутентификацию пользователей и роли «БЕСЕДКИ» с системой Rocket.Chat.
- **Контекст:** Проект «БЕСЕДКА» имеет существующую базу пользователей и критически важный, неизменяемый документ о ролях и полномочиях.
- **Предварительные условия:** Развернутый экземпляр Rocket.Chat, доступ к базе данных пользователей/системе аутентификации «БЕСЕДКИ» (например, данные провайдера LDAP/OAuth).
- **Шаги для AI-ассистента:**
  1. **Выбор метода аутентификации:** «Проанализируйте текущую систему аутентификации «БЕСЕДКИ» (например, LDAP, OAuth, пользовательская база данных). Рекомендуйте наиболее подходящий метод интеграции Rocket.Chat (LDAP, SAML, OAuth или пользовательская интеграция API). (См. [25])».
  2. **Конфигурация аутентификации:** «Настройте выбранный метод аутентификации в настройках администрирования Rocket.Chat (например, Администрирование > Рабочее пространство > Настройки > OAuth для Google/GitHub или настройки LDAP). (См. [25])».
  3. **Первоначальная синхронизация/импорт пользователей:** «Разработайте скрипт или используйте API импорта Rocket.Chat (/api/v1/import.addUsers) для импорта существующих пользователей «БЕСЕДКИ» в Rocket.Chat. Убедитесь в уникальности имен пользователей и адресов электронной почты. (См. [33])».
  4. **Создание пользовательских ролей:** «На основе неизменяемого документа «БЕСЕДКИ» о ролях и полномочиях создайте соответствующие пользовательские роли в Rocket.Chat (например, «BESEDKA_Owner», «BESEDKA_User»). Назначьте этим ролям определенные разрешения, такие как create-p для приватных групп, add-user-to-any-p-room, delete-message и т. д., в соответствии с документом. (См. [3])».
  5. **Назначение ролей:** «Разработайте механизм (например, вызовы API) для назначения этих пользовательских ролей Rocket.Chat синхронизированным пользователям на основе их ролей в «БЕСЕДКЕ». Убедитесь, что соблюдается принцип наименьших привилегий (POLP). (См. [20])».
  6. **Конфигурация 2FA:** «Настройте двухфакторную аутентификацию (2FA) в Rocket.Chat и убедитесь, что она включена для всех пользователей или определенных ролей, в соответствии с политикой безопасности

«БЕСЕДКИ». Настройте SMTP-сервер для 2FA по электронной почте, если это еще не сделано. (См. [3])».

7. **Проверка:** «Проверьте вход пользователя через интегрированный метод аутентификации. Убедитесь, что пользователям назначены правильные роли и они обладают соответствующими разрешениями в Rocket.Chat. Проверьте функциональность 2FA».
8. **Отчет:** «Предоставьте подробный отчет об интеграции аутентификации и RBAC, включая любые проблемы и результаты проверки».

Шаблон промпта 3: Настройка UI и встраивание

- **Цель:** Встроить Rocket.Chat на сайт «БЕСЕДКИ» и настроить его пользовательский интерфейс в соответствии со стандартами UI «БЕСЕДКИ».
- **Контекст:** «БЕСЕДКА» требует визуально унифицированного опыта, а текущий дизайн чата «ужасен».
- **Предварительные условия:** Развернутый экземпляр Rocket.Chat, документ «Стандарты UI БЕСЕДКА», документ SSOT.
- **Шаги для AI-ассистента:**
  1. **Встраивание iframe:** «Встройте рабочее пространство Rocket.Chat в раздел «Чат» веб-сайта «БЕСЕДКА» с помощью iframe. Настройте аутентификацию пользователей iframe для использования существующего входа в «БЕСЕДКУ». (См. [5])».
  2. **Темизация UI – Цветовая палитра:** «Получите доступ к Администрирование > Рабочее пространство > Настройки > Макет > Цвета Rocket.Chat. Сопоставьте основные, второстепенные и акцентные цвета «БЕСЕДКИ» из документа SSOT с настройками Major Colors (например, primary-background-color, primary-font-color, primary-action-color) и Minor Colors Rocket.Chat. Примените эти изменения. (См. [5])».
  3. **Темизация UI – Пользовательский CSS:** «Напишите и примените пользовательский CSS в настройках макета Rocket.Chat (раздел Custom CSS) для переопределения стандартных стилей. Сосредоточьтесь на:
     - Перестилизации «плиток» сообщений в более современный вид чат-пузырей.
     - Настройке шрифтов и типографики в соответствии с брендом «БЕСЕДКИ».
     - Изменении стилей боковой панели, заголовка и полей ввода для

согласованности.
    - Обратитесь к документации Rocket.Chat по цветам UI для названий переменных. (См. [5])».
4. **Пользовательский JavaScript (опционально):** «Если требуются специфические поведения или взаимодействия пользовательского интерфейса, которые не могут быть достигнуты с помощью CSS, реализуйте пользовательский JavaScript в разделе Custom Script for Logged In Users. (См. [5])».
5. **Проверка:** «Визуально осмотрите встроенный чат на веб-сайте «БЕСЕДКА». Убедитесь, что элементы дизайна (цвета, шрифты, внешний вид сообщений) соответствуют стандартам UI «БЕСЕДКА». Проверьте отзывчивость на разных устройствах».
6. **Отчет:** «Задокументируйте все примененные настройки CSS и JavaScript, а также скриншоты нового внешнего вида чата и сравнение со стандартами UI «БЕСЕДКА»».

Шаблон промпта 4: Управление приватными комнатами для админ-панели

- **Цель:** Позволить владельцу «БЕСЕДКИ» создавать приватные чат-комнаты и управлять доступом к ним непосредственно из своей админ-панели.
- **Контекст:** Владельцу необходимо «раздавать доступ» к приватным чатам через «выдачу пароля или ключа».
- **Предварительные условия:** Развернутый экземпляр Rocket.Chat, токены доступа API для бэкенда «БЕСЕДКИ» для взаимодействия с Rocket.Chat.
- **Шаги для AI-ассистента:**
    1. **Генерация ключа API:** «Сгенерируйте ключ API для бэкенд-системы «БЕСЕДКИ» для безопасного взаимодействия с REST и Realtime API Rocket.Chat. Убедитесь, что этот ключ имеет необходимые разрешения (например, create-p, add-user-to-any-p-room). (См. [10])».
    2. **Интеграция API создания приватных комнат:** «Интегрируйте метод Realtime API Rocket.Chat createPrivateGroup в админ-панель «БЕСЕДКИ». Это позволит владельцу указать имя для приватного чата и опционально добавить начальных участников. (См. [34])».
    3. **Интеграция API приглашения пользователей:** «Интегрируйте конечную точку REST API Rocket.Chat /api/v1/channels.invite в админ-панель «БЕСЕДКИ». Это позволит владельцу добавлять или удалять конкретных пользователей из существующих приватных комнат по их ID пользователя.

(См. [36])».

4. **Разработка UI админ-панели:** «Разработайте или модифицируйте UI админ-панели «БЕСЕДКИ», чтобы включить:
   - Форму для создания новых приватных чат-комнат (поле для имени, опциональный выбор пользователей).
   - Список существующих приватных комнат.
   - Функциональность для просмотра участников приватной комнаты.
   - Кнопки/опции для добавления/удаления пользователей из выбранной приватной комнаты».
5. **Проверка:** «Проверьте функциональность создания приватных комнат из админ-панели «БЕСЕДКИ». Убедитесь, что приватные комнаты создаются в Rocket.Chat. Проверьте добавление и удаление пользователей из приватных комнат и убедитесь, что только авторизованные пользователи могут получить к ним доступ».
6. **Отчет:** «Предоставьте фрагменты кода для интеграции API и пошаговое описание новых функций управления приватным чатом в админ-панели «БЕСЕДКИ»».

Шаблон промпта 5: План обновления документации

- **Цель:** Обновить ключевые проектные документы «БЕСЕДКИ» для отражения миграции Rocket.Chat, обеспечивая перекрестные ссылки и сохраняя документ о ролях/полномочиях.
- **Контекст:** Пользователь явно спрашивает об обновлении Мастер-документации, Стандартов UI, SSOT и строгом сохранении документа о ролях/полномочиях.
- **Предварительные условия:** Доступ ко всем файлам проектной документации «БЕСЕДКИ».
- **Шаги для AI-ассистента:**
  1. **Обновление Мастер-документации:** «Обновите Мастер-документацию «БЕСЕДКИ». Добавьте новый раздел, подробно описывающий интеграцию Rocket.Chat, охватывающий выбранную архитектуру Kubernetes/микросервисов, ключевые компоненты, поток данных и высокоуровневый обзор безопасности. Удалите любую устаревшую информацию, относящуюся к предыдущей системе чата».
  2. **Обновление Стандартов UI «БЕСЕДКА»:** «Пересмотрите документ «Стандарты UI БЕСЕДКА». Добавьте специальный подраздел для

интерфейса чата, подробно описывающий, как элементы UI Rocket.Chat (например, пузырьки сообщений, поля ввода, списки пользователей) стилизуются с использованием пользовательских CSS и опций темизации Rocket.Chat для соответствия визуальным рекомендациям «БЕСЕДКИ». Включите примеры сопоставленных переменных цвета и типографики».

3. **Обновление Единого источника истины (SSOT):** «Обновите документ SSOT «БЕСЕДКИ». Включите основные компоненты UI Rocket.Chat и их соответствующие токены/переменные дизайна. Явно сопоставьте переменные темизации Rocket.Chat (например, primary-background-color из [6]) с установленными определениями системы дизайна «БЕСЕДКИ»».
4. **Реализация перекрестных ссылок:** «Реализуйте четкие перекрестные ссылки (например, гиперссылки или явные ссылки) между обновленной Мастер-документацией, Стандартами UI и SSOT. Например, убедитесь, что раздел чата в Мастер-документации ссылается на соответствующие записи в Стандартах UI и SSOT для получения подробной информации о дизайне».
5. **Обработка документа о ролях и полномочиях:** «Подтвердите, что документ «БЕСЕДКИ» о ролях и полномочиях остается *неизменным*. Добавьте явные ссылки в обновленную Мастер-документацию на этот неизменяемый документ как на авторитетный источник логики доступа, объясняя, как RBAC Rocket.Chat настроен для отражения его правил».
6. **Проверка и валидация:** «Проверьте все обновленные документы на точность, согласованность и полноту. Убедитесь, что весь устаревший контент, связанный с чатом, удален или четко помечен как устаревший».
7. **Отчет:** «Предоставьте сводку всех изменений в документах, включая список обновленных разделов и нового контента, а также подтверждение того, что документ о ролях и полномочиях был сохранен».

Заключение и перспективы на будущее

Миграция на Rocket.Chat представляет собой фундаментальное преобразование функциональности чата «БЕСЕДКИ», выводя его из текущего «очень печального состояния» в безопасную, масштабируемую и многофункциональную коммуникационную платформу. Этот переход напрямую решает существующие проблемы, такие как устаревший дизайн, отсутствие функций ответов и цитирования, а также невозможность обмена медиафайлами, и закладывает

прочную основу для будущего роста проекта.

Предложенное решение уделяет первостепенное внимание ключевым требованиям пользователя в отношении безопасности, включая сквозное шифрование, надежную аутентификацию и защиту данных, а также масштабируемости, что позволяет эффективно обрабатывать прогнозируемый рост числа пользователей от 1000 до 5000. Выбор архитектуры на основе Kubernetes и микросервисов является стратегическим шагом, обеспечивающим высокую доступность и производительность, необходимые для долгосрочного успеха проекта.

Эта миграция — не просто техническое обновление, а стратегический шаг, который устраняет препятствия для дальнейшего развития проекта «БЕСЕДКА». Она позволит продолжить работу над другими разделами, такими как «Галерея», «Гроу-репорты» и «Новости», которые уже практически готовы [User Query].

В будущем можно рассмотреть следующие улучшения:

- Изучение возможностей Rocket.Chat, основанных на искусственном интеллекте, таких как автоматическое суммирование бесед или чат-боты.[2]
- Более глубокая интеграция компонентов Rocket.Chat во фронтенд «БЕСЕДКИ» с использованием библиотеки Fuselage, если подход с iframe окажется недостаточным для долгосрочных целей пользовательского интерфейса.
- Изучение возможностей Rocket.Chat по объединению для кросс-платформенной коммуникации.[8]
- Непрерывный мониторинг производительности и оптимизация по мере приближения числа пользователей к 5000 и более.
- Регулярные аудиты безопасности и обновления для поддержания безопасной среды.

Источники

1. Rocket.Chat - Apps on Google Play, дата последнего обращения: июня 17, 2025, https://play.google.com/store/apps/details?id=chat.rocket.android
2. Secure, Scalable, and Customizable for Mission-Critical Operations - Rocket.Chat, дата последнего обращения: июня 17, 2025, https://www.rocket.chat/platform-overview
3. Pricing for secure team collaboration - Rocket.Chat, дата последнего обращения: июня 17, 2025, https://www.rocket.chat/pricing
4. 7 essential messaging SDK components - Rocket.Chat, дата последнего обращения: июня 17, 2025, https://www.rocket.chat/blog/messaging-sdk

5. Customize Workspace Layout - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/layout
6. guides/developer/ui-and-theming/colors.md · 979f90667ecd1de4858843463eb00c42aff75c96 - GitLab, дата последнего обращения: июня 17, 2025, https://gitlab.ow2.org/RocketChat/docs/-/blob/979f90667ecd1de4858843463eb00c42aff75c96/guides/developer/ui-and-theming/colors.md
7. Componentization - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/docs/componentization
8. Architecture and Components - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/docs/architecture-and-components
9. Server Architecture - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/docs/server-architecture
10. Rocket.Chat API, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/apidocs/rocketchat-api
11. Deploy Rocket.Chat, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/deploy-rocketchat
12. Deploy Rocket.Chat — Complete Data Sovereignty, дата последнего обращения: июня 17, 2025, https://www.rocket.chat/platform/deploy
13. quick-start/installing-and-updating/introduction/choosing-a-deployment-method.md - GitLab, дата последнего обращения: июня 17, 2025, https://gitlab.ow2.org/RocketChat/docs/-/blob/ea8701d691374e8b214d754be2f66adf8a1206af/quick-start/installing-and-updating/introduction/choosing-a-deployment-method.md
14. Deploy with Kubernetes - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/deploy-with-kubernetes
15. Scaling Rocket.Chat, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/scaling-rocketchat
16. Scaling Open Source Realtime Messaging System for Millions - FOSDEM 2023, дата последнего обращения: июня 17, 2025, https://archive.fosdem.org/2023/schedule/event/scaling_rtc_messaging/attachments/slides/5967/export/events/attachments/scaling_rtc_messaging/slides/5967/FOSDEM23_RTC_RC
17. System Requirements - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/system-requirements
18. First steps with self-hosted Rocket Chat: Watch the holes (bugs)! - Claudio Kuenzler, дата последнего обращения: июня 17, 2025, https://www.claudiokuenzler.com/blog/1290/first-steps-rocket-chat-watch-the-holes
19. Best Practices for Secure Rocket.Chat + Jitsi Deployments, дата последнего обращения: июня 17, 2025, https://jitsi.support/how-to/secure-rocketchat-jitsi-deployment/
20. Security Guidelines - Rocket-Chat Documentation, дата последнего

обращения: июня 17, 2025, https://docs.rocket.chat/docs/security-guidelines
21. deploy/deploy-rocket.chat/system-requirements.md · e62281b8b1ba8ad5d9e46b94397ff7303a80f54e · undefined · GitLab, дата последнего обращения: июня 17, 2025, https://gitlab.ow2.org/RocketChat/docs/-/blob/e62281b8b1ba8ad5d9e46b94397ff7303a80f54e/deploy/deploy-rocket.chat/system-requirements.md
22. Rocket.Chat Hosting | Deploy Rocket.Chat Cloud Servers - Kamatera, дата последнего обращения: июня 17, 2025, https://www.kamatera.com/applications/rocketchat/
23. E2E Encryption - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/e2e-encryption
24. End-to-End Encryption Specifications - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/end-to-end-encryption-specifications
25. Authentication and Identity Management FAQ, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/identity-management-faq
26. OAuth - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/oauth
27. Create Custom Roles - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/custom-roles
28. Permissions in Rocket.Chat, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/permissions
29. Encryption at Rest - Database Manual - MongoDB Docs, дата последнего обращения: июня 17, 2025, https://www.mongodb.com/docs/manual/core/security-encryption-at-rest/
30. How to host a private Rocket.Chat server - Meshnet docs - NordVPN, дата последнего обращения: июня 17, 2025, https://meshnet.nordvpn.com/how-to/joint-projects/private-rocket.chat-server
31. Install Desktop & Mobile App - Rocket-Chat Documentation, дата последнего обращения: июня 17, 2025, https://docs.rocket.chat/docs/desktop-mobile-apps
32. Linux and Windows Rocket.Chat Development Environment, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/docs/linux-and-windows-rocketchat-development-environment
33. Add Users - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/apidocs/add-users
34. Create Private Room (Realtime) - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/apidocs/create-private-room-realtime
35. Rooms - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/docs/rooms
36. Add Users to Channel - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025, https://developer.rocket.chat/apidocs/add-users-to-channel
37. Iframe Integration - Rocket.Chat Developer, дата последнего обращения: июня 17, 2025,

https://developer.rocket.chat/docs/customize-and-embed-iframe-integration
38. Flexibility and scalability - Rocket.Chat, дата последнего обращения: июня 17, 2025, https://www.rocket.chat/platform/flexibility-scalability