|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Приднестровский государственный университет им. Т.Г. Шевченко | | | | | | | | | |
| Физико-технический институт | | | | | | | | | |
| Факультет информатики и вычислительной техники | | | | | | | | | |
| Кафедра информационных технологий | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| **ОТЧЕТ ПО ПРАКТИКЕ** | | | | | | | | | |
| Производственная практика | | | | | | | | | |
| Научно-исследовательская работа | | | | | | | | | |
|  | | | | | | | | | |
| Направление 2.09.03.01 Информатика и вычислительная техника | | | | | | | | | |
| профиль «Вычислительные машины, комплексы, системы и сети» | | | | | | | | | |
| Студента 5 курса ИТ20ВР62ИВ группа | | | | | | | | | |
| форма обучения заочная | | | | | | | | | |
| Заведий Владислав Сергеевич | | | | | | | | | |
| Место прохождения практики ПГУ им. Т.Г. Шевченко, ФТИ ИВТ кафедра информационных технологий | | | | | | | | | |
| Сроки прохождения практики | | | | | | | | | |
| с «18» ноября 2024 г. по «14» декабря 2024 г. | | | | | | | | | |

|  |  |  |
| --- | --- | --- |
| от профильной организации | | |
| зав. кафедрой ИТ |  | Ю.А. Столяренко |
| от университета | | |
| Ст.преп. |  | С.Л.Чирвина |
| Итоговая оценка (зачет) по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Тирасполь, 2024 г. | | |

Оглавление

[ВВЕДЕНИЕ 3](#_Toc184809117)

[**1 ИССЛЕДОВАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 5](#_Toc184809118)

[**1.1** **Описание поставленной задачи, ее обоснование** 5](#_Toc184809119)

[1.2 Обоснование актуальности исследуемой задачи. 6](#_Toc184809120)

[1.3 Современное состояние исследуемой задачи. 11](#_Toc184809121)

[1.4 Обзор методов решения подобных задач 16](#_Toc184809122)

[1.5 Постановка задачи, системные требования, требования к входным данным и выходным формам 24](#_Toc184809123)

[1.6 Выводы 27](#_Toc184809124)

[2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ И АРХИТЕКТУРЫ ПРОГРАММНОГО ПРОДУКТА 28](#_Toc184809125)

[2.1 Выбор методов и средств для реализации, его обоснование 28](#_Toc184809126)

[2.2 Структура, архитектура программного продукта 32](#_Toc184809127)

[2.3 Проектирование базы данных 38](#_Toc184809128)

[Пример переработанного текста: 38](#_Toc184809129)

[2.4 Описание логической структуры программного продукта 38](#_Toc184809130)

[(сетевое взаимодействие, используемые протоколы, организация 39](#_Toc184809131)

[соединения, получение и отправка данных, закрытие соединения, 39](#_Toc184809132)

[разработанные классы и/ или редко используемые библиотечные 39](#_Toc184809133)

[классы, взаимодействие классов) 39](#_Toc184809134)

[Основные отличия между пунктами 39](#_Toc184809135)

[2.5 Функциональная схема, функциональное назначение программного 40](#_Toc184809136)

[продукта (функции клиента, функции сервера) 40](#_Toc184809137)

[2.4 Выводы 40](#_Toc184809138)

[**ЗАКЛЮЧЕНИЕ** 41](#_Toc184809139)

[**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, ЕДИНИЦ** 41](#_Toc184809140)

[**И ТЕРМИНОВ** 41](#_Toc184809141)

[**СПИСОК ЛИТЕРАТУРЫ** 41](#_Toc184809142)

# ВВЕДЕНИЕ

В условиях быстрого технологического прогресса и развития цифровых коммуникаций, эффективная организация внутренней связи в корпоративной среде становится неотъемлемой частью успешного функционирования организаций. В частности, роль мессенджеров в современном бизнесе сложно переоценить: они позволяют обеспечивать моментальную коммуникацию, обмен информацией и документами в реальном времени. Важность таких инструментов растет с каждым днем, так как они способствуют ускорению обмена данными между сотрудниками, отделами и рабочими группами, что помогает оперативно реагировать на изменения внешней среды и экономической ситуации.

В рамках данной работы рассматривается проектирование корпоративного клиент-серверного мессенджера, который служит инструментом для улучшения коммуникации и совместной работы внутри организации. Мессенджер будет направлен на автоматизацию и улучшение процессов обмена информацией в корпоративной среде, поддерживая требования безопасности, гибкости и эффективности.

Актуальность выбранной темы подтверждается необходимостью внедрения удобных и безопасных средств связи, особенно в условиях дистанционной работы, где мгновенная доставка сообщений и возможность обмена различными материалами становятся важнейшими факторами. Современные мессенджеры играют ключевую роль в бизнес-коммуникациях, обеспечивая связь в реальном времени между участниками.

Целью работы является разработка корпоративного мессенджера, который позволит организовать быструю и безопасную рассылку информационных сообщений, а также обмен различными типами файлов и документов внутри компании. Важной частью работы является также анализ существующих решений на рынке корпоративных мессенджеров и выявление их недостатков, что дает возможность создать более эффективный инструмент для организации корпоративной коммуникации.

Объектом исследования является коммуникационная информационная среда предприятия, а предметом — система автоматизированной рассылки информационных сообщений. В ходе работы будут рассмотрены методы проектирования, особенности разработки и реализации мессенджера, а также способы интеграции с существующими корпоративными системами.

Таким образом, цель и задачи исследования направлены на создание эффективной и инновационной информационной системы, ориентированной на решение актуальных проблем в области корпоративной связи.

# **1 ИССЛЕДОВАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

* 1. **Описание поставленной задачи, ее обоснование**

В условиях стремительного технологического прогресса и динамичных изменений в организационных структурах предприятий, эффективное управление коммуникациями становится фундаментальным фактором успеха. Постоянный рост числа дистанционных рабочих групп, географическое разделение персонала, а также необходимость мгновенного обмена информацией подчеркивают актуальность проблемы эффективной корпоративной коммуникации.

В рамках выполнения выпускной квалификационной работы поставлена задача разработать корпоративный мессенджер для осуществления обмена служебными данными в виде текстовых сообщений, изображений, документов и файлов внутри рабочей среды предприятия.

Целью данного исследования является разработка и внедрение корпоративного клиент-серверного мессенджера . Эта задача обусловлена необходимостью создания инновационного инструмента для улучшения внутренней коммуникации в организации. Временные задержки в обмене информацией и несвоевременное принятие решений могут привести к потере конкурентоспособности компании в современной бизнес-среде.

Разработка корпоративного мессенджера оправдывается также тем, что традиционные средства связи, такие как электронная почта, становятся менее эффективными при высоком темпе работы и необходимости оперативного взаимодействия. К тому же использование сторонних мессенджеров может создавать риски утечки конфиденциальной информации и зависимость от внешних сервисов. Создание собственного мессенджера позволит не только повысить скорость и надежность коммуникаций, но и обеспечить защиту корпоративных данных, а также соответствие специфическим требованиям бизнеса.

Обоснование данной задачи также основывается на стремлении повысить эффективность внутренних бизнес–процессов, снизить издержки и улучшить общий опыт работы персонала. Создание интегрированной системы коммуникации может способствовать формированию единого информационного пространства, обеспечивая сотрудникам более оперативное и эффективное взаимодействие, что является критически важным аспектом в современном корпоративном мире.

Основываясь на анализе современного состояния организаций, можно выделить растущую потребность в усовершенствованных инструментах, способных улучшить взаимодействие между сотрудниками. Корпоративный мессенджер представляет собой ответ на этот вызов, обеспечивая более быстрый, гибкий и интерактивный обмен информацией, что, в свою очередь, способствует более эффективному принятию решений.

Дополнительным обоснованием является рост числа компаний, в которых сотрудники работают удаленно или в различных географически распределенных отделах. Корпоративный мессенджер, ориентированный на современные потребности, обеспечит связь и синхронизацию между удаленными командами, способствуя единообразному восприятию информации и обеспечивая централизованный доступ ко всем необходимым ресурсам.

Таким образом, разработка корпоративного клиент–серверного мессенджера нацелена не только на решение текущих проблем внутренней коммуникации, но и на создание стратегического резерва для адаптации к будущим требованиям компаний.

## 1.2 Обоснование актуальности исследуемой задачи.

За последние пару лет были отмечены заметные изменения в динамике корпоративного общения, где мессенджеры стали неотъемлемой частью ежедневного быта. Тем не менее, частые сбои в функционировании ведущих платформ, таких как *WhatsApp* и *Telegram*, поднимают вопросы относительно надежности этих средств коммуникации. Эти технические неполадки могут серьезно воздействовать на бизнес-процессы, стимулируя крупные компании обращать внимание на важность создания собственных корпоративных мессенджеров.

Современные проблемы с надежностью популярных мессенджеров вынуждают компании рассматривать альтернативные решения. Поскольку приложение не принадлежит предприятию , перестроить процесс обеспечения безопасности для защиты данных сотрудников и клиентов не получится а так же данные могут попасть к злоумышленникам и быть скомпрометированы.

На примере *Telegram* можно отметить, что случаи компрометации аккаунтов предпринимателей стали обыденным явлением. Наиболее серьезной угрозой в таких ситуациях является то, что злоумышленники могут не только инициировать запросы о переводе денежных средств от имени владельца аккаунта, но и пытаться получить доступ к внутренней документации компании, корпоративным ресурсам и конфиденциальной информации через сотрудников.

Следующим фактором является разгрузка информационного шума, так как в едином пространстве мессенджера объединены личные и групповые чаты, сообщества и каналы. Это затрудняет разграничение рабочего и личного пространства, а также сосредоточение исключительно на профессиональных задачах. Сотрудники оказываются в условиях, где отвлечения становятся неизбежными: поток сообщений от друзей и родственников пересекается с деловой перепиской, а кликбейтные заголовки новостных сообществ лишь усиливают соблазн отвлечься. В результате создается риск, что наименее дисциплинированные сотрудники будут терять продуктивность. Кроме того, отсутствие четкого разделения контента может привести к утечке информации, когда рабочие данные случайно отправляются в личные чаты или сторонние группы.

Подчеркивая важность не только средства общения, но и комплексного инструмента, включающего в себя корпоративный документооборот и управление бизнес-процессами, корпоративный мессенджере можно не только обеспечивать коммуникацию, но и формировать корпоративную культуру взаимодействия. В общедоступных сервисах нельзя выстроить нужную для команды структуру: например, создать ветки для обсуждения конкретных тем и взаимодействия ограниченного количества отделов

Процесс внедрения нового корпоративного программного обеспечения , в том числе мессенджеров, требует времени, особенно в условиях обеспечения высокого уровня безопасности. Для успешного перехода к новым технологиям, необходимо предоставить сотрудникам мобильное рабочее место с широким функционалом, включая не только коммуникацию, но и работу с документами и бизнес-процессами. В целом, внедрение корпоративного мессенджера, не просто технологическое обновление, но и стратегически важный шаг для компаний, стремящихся обеспечить эффективное и безопасное корпоративное взаимодействие в условиях современно.

Разработка корпоративного клиент-серверного мессенджера представляет собой задачу, которая становится весьма актуальной в современной деловой среде. Развитие информационных технологий и их влияние на организации делают такие мессенджеры неотъемлемой частью корпоративной коммуникации. Обоснование актуальности этой задачи опирается на ряд факторов, которые определены ниже.

В условиях современного бизнеса активность коммуникаций между сотрудниками и клиентами играет ключевую роль в успешной деятельности компании. Корпоративные мессенджеры, спроектированные с учетом специфики бизнес-процессов, становятся эффективным инструментом для оперативного обмена информацией и принятия стратегических решений.

Безопасность и конфиденциальность: когда речь идет о корпоративной среде, особое внимание уделяется вопросам безопасности и защиты конфиденциальной информации. Разработка собственного мессенджера позволяет интегрировать высокие стандарты шифрования и механизмы контроля доступа, обеспечивая таким образом надежную защиту бизнес-данных.

Интеграция с другими инструментами: корпоративный мессенджер, разработанный с учетом потребностей компании, предоставляет возможность интеграции с другими корпоративными инструментами, такими как системы управления задачами, электронные почты и документооборот. Это способствует улучшению рабочих процессов и повышению эффективности внутри компании.

Гибкость и масштабируемость: разработка собственного мессенджера предоставляет возможность создать инструмент, который легко адаптируется под специфику компании. Гибкость в настройке функционала и масштабируемость в соответствии с ростом бизнеса делают такое решение более перспективным в долгосрочной перспективе.

Соблюдение регулировании: в условиях строгих регуляций, таких как законы о защите данных и прозрачности в бизнесе, разработка собственного корпоративного мессенджера позволяет лучше соответствовать требованиям законодательства и обеспечивать контроль над обработкой информации.

Оптимизация рабочих процессов: корпоративный мессенджер, разработанный под нужды организации, может включать в себя функционал для оптимизации рабочих процессов, таких как автоматизированные уведомления, управление задачами и аналитика в реальном времени. Это способствует повышению производительности и эффективности труда.

Контроль над коммуникацией:  корпоративный мессенджер предоставит возможность создавать группы и чаты с различными уровнями доступа, что позволит эффективно управлять взаимодействием между сотрудниками. Он обеспечит быстрый доступ к каждому сотруднику, исключая необходимость длительных поисков в списке контактов. Кроме того, есть возможность предусмотреть изолированные каналы связи, которые могут быть использованы в ситуациях, требующих конфиденциальности. Для повышения удобства взаимодействия, мессенджер также позволит интегрировать клиентов, с которыми ведется постоянное сотрудничество, в рабочее пространство, что упростит работу проектных команд и повысит качество коммуникации с клиентами.

Адаптивность: в рамках разработки корпоративного мессенджера, компания имеет возможность самостоятельно определить, под какие устройства и операционные системы будет создано решение. Разработчик, в свою очередь, окажет поддержку в выборе технологического стека, который наилучшим образом соответствует целям и специфике бизнеса, обеспечивая полную совместимость и эффективную работу приложения на различных платформах. Это позволит максимально адаптировать решение под нужды компании и сотрудников, обеспечивая гибкость и расширяемость в дальнейшем.

Большие возможности для масштабирования: с развитием компании и ростом числа сотрудников корпоративный мессенджер позволит легко расширять его возможности. Компания может увеличить место для хранения данных или масштабировать структуру по мере необходимости. Это обеспечит гибкость в управлении ресурсами и возможностями мессенджера. Кроме того, функционал можно будет постоянно расширять, превращая простой мессенджер в полноценную платформу для корпоративной коммуникации и взаимодействия.

Экономия ресурсов в долгосрочной перспективе: хотя первоначальная разработка корпоративного мессенджера может потребовать больших затрат по сравнению с годовой подпиской на готовый сервис, в долгосрочной перспективе это будет более выгодным решением. Разовая оплата разработки даст компании гораздо больше свободы, гибкости и контроля, позволяя создать решение, полностью соответствующее потребностям бизнеса. В отличие от шаблонных решений, корпоративный мессенджер можно адаптировать и развивать, что обеспечит большую ценность и эффективность в будущем.

Таким образом, разработка корпоративного клиент-серверного мессенджера является обоснованной и стратегически важной для современных компаний. В условиях постоянно меняющихся требований бизнеса и роста числа сотрудников и клиентов, собственное решение позволяет эффективно управлять коммуникациями, обеспечивать безопасность данных, интегрировать различные инструменты и поддерживать гибкость в масштабировании. Все эти факторы способствуют улучшению производительности, повышению качества взаимодействия как внутри компании, так и с внешними партнерами, что делает внедрение корпоративного мессенджера не только целесообразным, но и необходимым шагом для достижения долгосрочного успеха.

1.3 Современное состояние исследуемой задачи.

Современные корпоративные мессенджеры становятся неотъемлемой частью бизнес-процессов, играя ключевую роль в обеспечении эффективной коммуникации внутри компаний. Однако развитие этой сферы не ограничивается лишь улучшением связи. Стремление к повышению производительности и оптимизации рабочего процесса приводит к интеграции корпоративных мессенджеров с другими инструментами.

Интеграция с электронной почтой, календарями, системами управления проектами и облачными хранилищами облегчает обмен информацией между различными приложениями, устраняя необходимость переключения между ними. Это не только повышает эффективность, но и улучшает общую координацию и взаимодействие внутри команды.

С ростом мобильности и многоплатформенности современные корпоративные мессенджеры становятся доступными на различных устройствах и операционных системах. Это позволяет сотрудникам поддерживать связь и оставаться продуктивными в любой точке мира. Мобильные приложения для корпоративных мессенджеров обеспечивают гибкость работы и возможность оперативного реагирования на запросы и обновления.

Безопасность является одним из приоритетов современных корпоративных мессенджеров. Внедрение шифрования конечного устройства, двухфакторной аутентификации и управления правами доступа помогает защитить конфиденциальность данных. В свете угроз кибербезопасности, этот подход становится стандартной практикой для обеспечения безопасности бизнес-коммуникаций.

Использование искусственного интеллекта также трансформирует функционал корпоративных мессенджеров. Исскуственный интелект улучшает поиск информации, анализ данных и даже автоматизацию рутинообразных задач. Коллаборативные функции, такие как обмен файлами, создание групповых чатов и проведение видеоконференций, становятся более интеллектуальными и адаптивными.

Аналитика и отчетность становятся важным элементом корпоративных мессенджеров. Инструменты оценки эффективности коммуникаций и мониторинга активности пользователей помогают компаниям адаптироваться и улучшать свои бизнес–процессы. Облачные решения обеспечивают гибкую инфраструктуру для хранения данных и позволяют легко масштабировать использование мессенджеров при необходимости.

Пользовательский опыт также остается в центре внимания. Интерфейсы становятся более интуитивными, функционал более гибким, а возможности настройки персональных предпочтений расширяются. Это направление развития способствует более быстрому освоению платформ и повышению их эффективности в повседневной работе.

С учетом постоянных изменений в форматах работы, связанных с удаленной работой и гибкими графиками, корпоративные мессенджеры становятся важным инструментом для поддержания связи и синхронизации команд. Виртуальные встречи и видеоконференции помогают преодолевать расстояния, создавая более тесные и эффективные рабочие отношения.

С разнообразием предоставляемых функций корпоративные мессенджеры не только облегчают оперативные обсуждения проектов, но и способствуют развитию корпоративной культуры. Тематические чаты, форумы для обмена идеями и виртуальные офисы создают атмосферу командной работы, способствуя эффективному взаимодействию сотрудников.

В целом, современные корпоративные мессенджеры продолжают эволюционировать, предоставляя более удобные, безопасные и гибкие инструменты для коммуникаций в рабочей среде. Они активно адаптируются к изменяющимся требованиям современного рынка труда, поддерживая инновации и обеспечивая эффективное взаимодействие внутри организаций.

На российском рынке мессенджеров в 2023 году продолжается рост как на мировом, так и на отечественном уровне. Прогнозируется, что объем рынка иностранных сервисов для корпоративных коммуникаций достигнет 2,5 млрд долларов и вырастет до 10% к 2031 году. В этом контексте, ведущие платформы, такие как *Microsoft Teams*, *Cisco Webex*, *Jira* и *Slack*, остаются ключевыми участниками рынка.

Доля отечественных решений в 2022 году достигла 20–25%, что свидетельствует о растущем интересе компаний к локальным продуктам. Этот тренд подтверждается исследованием *TGStat*: хотя популярность зарубежных платформ, таких как *Telegram* и *Slack*, сохраняется, впервые за несколько лет в рейтинге корпоративных мессенджеров начали активно выделять российские разработки. Несмотря на то что их доля пока составляет лишь несколько процентов, это указывает на постепенный переход бизнеса на защищенные отечественные сервисы, что способствует укреплению позиций местных производителей. На рисунке 1.1 отображен рост корпоративных мессенджеров Российского производства.

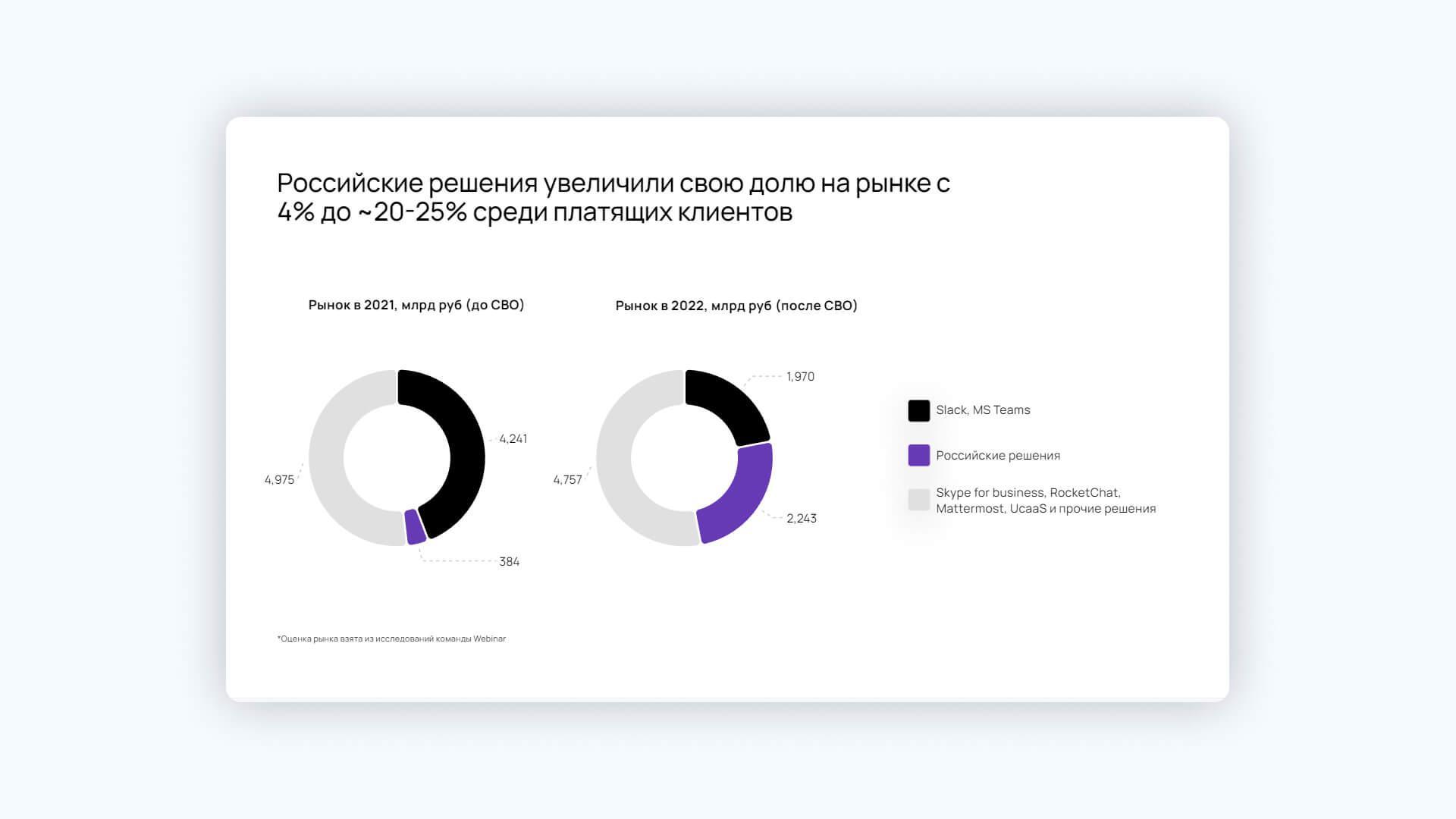
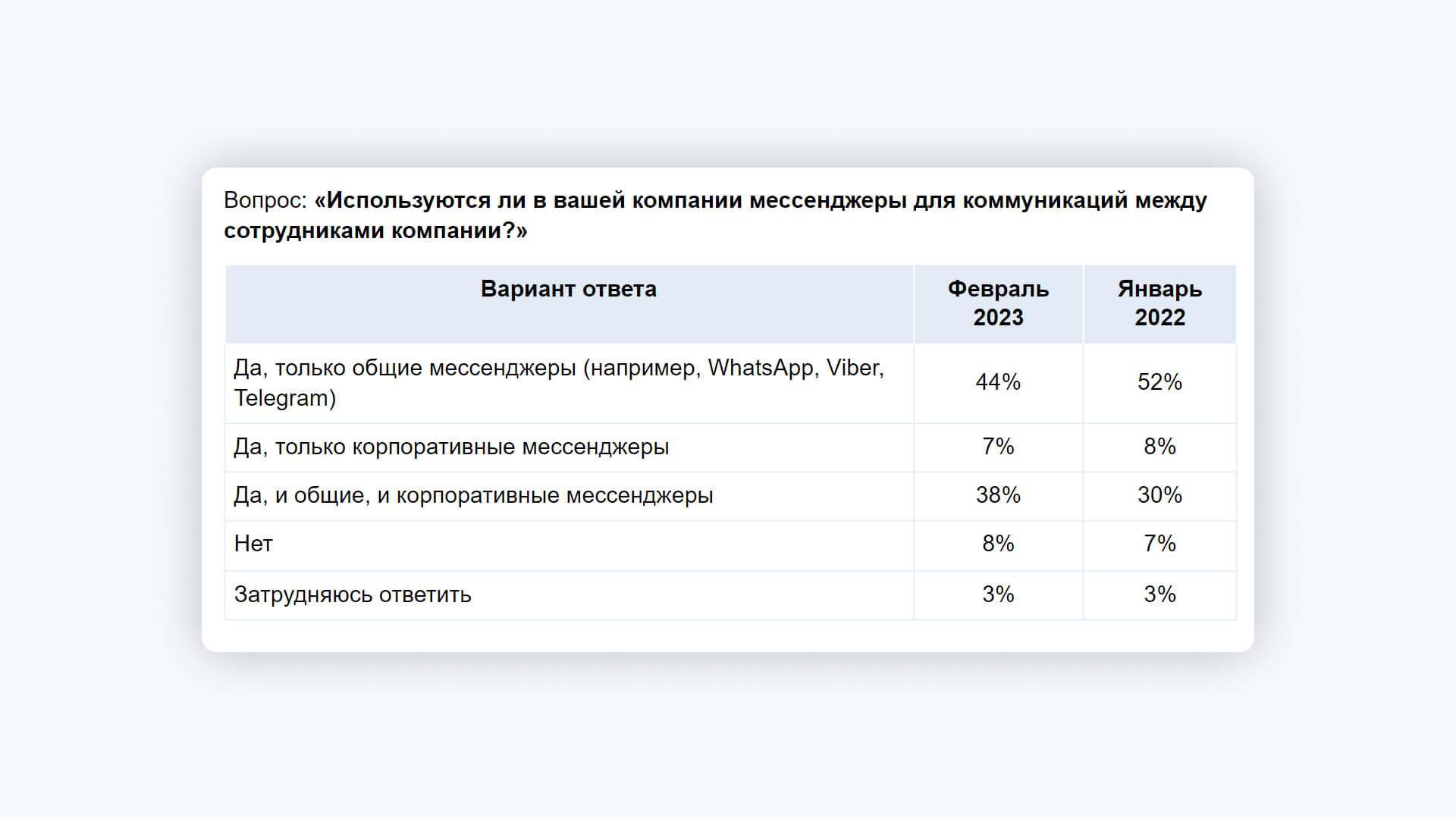


Рисунок 1.1 – Статистика Российского рынка

Такой переход понятен – крупных западных сервисов официально больше нет, а вести рабочую переписку где–то нужно. Поэтому дальше основной рост будет обеспечиваться уходом западных игроков и переходу к более современным отечественным системам.

Долгое время в России лидерами в платных корпоративных коммуникациях были представители зарубежных сервисов. Популярность *Microsoft Teams* в корпоративных коммуникациях была обеспечена пакетом *Office* в рамках продуктов *Microsoft* – они поставлялись вместе, что позволило компании занимать около 20% рынка.

В связи с уходом иностранных решений, рынок начал меняться, но спросом пользуются не корпоративные мессенджеры, а бесплатные публичные. Согласно  исследованию *SuperJobs* в феврале 2023 года в лидерах все еще оставались *Telegram*, *WhatsApp*, *Viber* и другие. Их используют в компаниях почти половина сотрудников. В 4 из 10 компаний общаются в смешанном режиме: часть рабочих переписок приходилось на публичные, часть – на корпоративные. Только 7% полностью ведут диалог в бесплатных или платных корпоративных мессенджерах. На рисунке 1.2 отображен опросник использования мессенджеров для коммуникации между сотрудниками компании.



1. Рисунок 1.2 – Опросник использования мессенджеров

Но тренд движется к тому, что крупный бизнес постепенно отказывается от публичных мессенджеров и переходит в защищенные сервисы по обмену сообщениями. Это косвенно подтверждает исследование  *TGStat*. Да, у половины опрошенных есть несколько рабочих групповых чатов с коллегами в *Telegram*, но кроме этого они используют *MS Teams*, *Discord*, *Slack*. Впервые за несколько лет в рейтинге начали отдавать предпочтение российскому продукту. На него хоть и приходится только несколько процентов, но это еще один пример положительного тренда.

Следует отметить, что развитие технологий искусственного интеллекта, машинного обучения и блокчейна открывает новые перспективы для обеспечения безопасности корпоративных мессенджеров. Эти передовые технологии позволяют значительно усилить меры защиты за счет внедрения адаптивных систем, которые в режиме реального времени анализируют поведение пользователей и паттерны передачи данных. Это помогает выявлять подозрительные действия, предотвращать угрозы фишинга, вирусные атаки и утечки данных.

Блокчейн вносит революционные изменения в обеспечение безопасности, создавая распределенные реестры, которые обеспечивают непрерывную запись и проверку всех транзакций. Это исключает возможность подделок, несанкционированного изменения или удаления данных. Кроме того, технологии блокчейна позволяют эффективно управлять цифровыми идентификаторами, упрощая процесс верификации пользователей и устройств.

Таким образом, использование искусственного интеллекта и блокчейна не только повышает уровень безопасности корпоративных коммуникаций, но и обеспечивает новые возможности для автоматизации бизнес-процессов и улучшения рабочих процессов.

## 1.4 Обзор методов решения подобных задач

В современном бизнес–мире эффективное внутреннее общение играет ключевую роль в успешной работе организаций. Существует широкий спектр программ корпоративных мессенджеров, предназначенных для обеспечения эффективной коммуникации и сотрудничества внутри компаний. В данном обзоре буду рассмотрены существующие методы решения предназначенные для корпоративных коммуникаций, будут выявлены их особенности, преимущества и недостатки.

Одним из основных методов является использование специализированных платформ, таких как *Microsoft Teams*, *Slack*, *Cisco Webex* и другие. Эти приложения предоставляют обширные функциональные возможности, включая текстовые чаты, видеоконференции, обмен файлами и интеграцию с другими рабочими инструментами. Преимущества таких платформ в их многогранности и широком спектре инструментов для коллективной работы.

Еще одним методом является внедрение внутренних корпоративных социальных сетей. Подобные платформы создают виртуальное пространство для обмена информацией и идеями, позволяя сотрудникам взаимодействовать вне зависимости от местоположения. Однако, такие решения могут столкнуться с проблемой масштабируемости и требовать внимательного управления контентом.

Интеграция мессенджеров с другими инструментами бизнес-процессов также является распространенным методом. Это позволяет сократить время переключения между приложениями и повысить эффективность внутрикорпоративного взаимодействия.

С учетом повышенной озабоченности защитой данных, методы шифрования, двухфакторной аутентификации и управления правами доступа становятся важными составляющими решений в сфере корпоративных коммуникаций. Это направление призвано обеспечить конфиденциальность информации и предотвратить утечки данных.

Искусственный интеллект вносит свой вклад в современные методы решения задач корпоративных коммуникаций, улучшая поиск информации, анализ данных и автоматизацию повседневных задач. Автоматическое определение контекста и предложение релевантных решений содействуют более эффективному общению внутри команд.

Линк Чаты (ранее *[We.Teams](https://mts-link.ru/products/messenger/)*[)](https://mts-link.ru/products/messenger/) – собственный корпоративный мессенджер от МТС Линк, в котором можно совершать звонки как один на один, так и на всю команду. Он создан, чтобы помогать командам выстраивать коммуникации, оставаться на связи и продуктивно работать не отвлекаясь на инфошум [1].

Особое внимание уделено аспекту продуктивности – Линк Чаты спроектированы так, чтобы команды могли эффективно взаимодействовать, не отвлекаясь на информационный шум. Возможности мессенджера позволяют структурировать общение, устанавливать приоритеты и легко находить необходимую информацию. Таким образом, Линк Чаты становятся надежным инструментом для тех, кто стремится создать прозрачную и результативную рабочую обстановку в своей компании. На рисунке 1.3 отображен интерфейс Линк Чата.

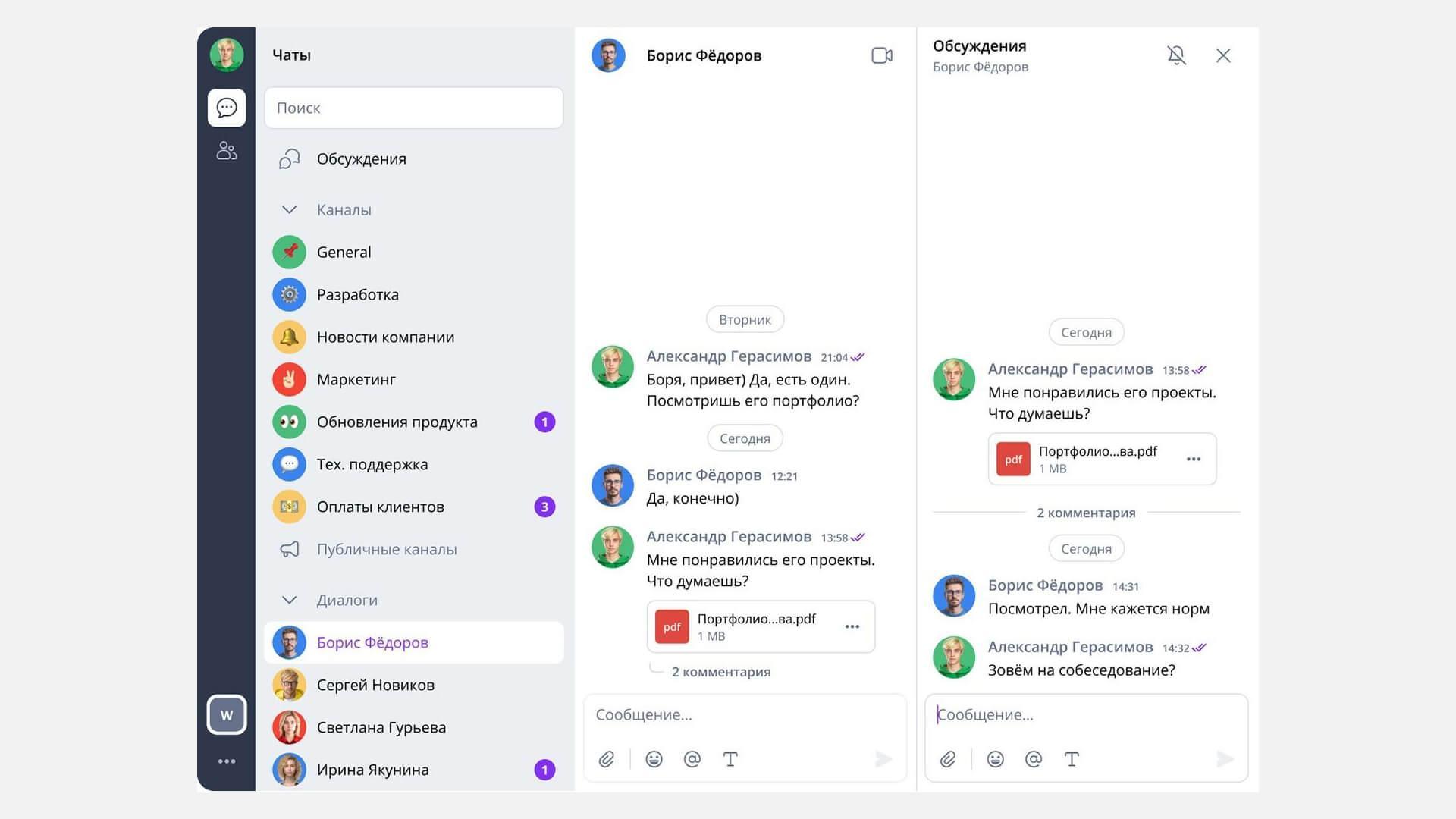


Рисунок 1.3 – Линк Чат

Ключевые преимущества:

* Все функции мессенджера: чаты, каналы, треды, передача файлов, реакции и упоминания. Адресная книга для быстрого поиска коллег и рабочие области для переключения между организациями, в которых состоит пользователь.
* Быстрый вход: мессенджер доступен сразу после прохождения простой регистрации. В бесплатной версии будет *Active Directory* для быстрого переноса рабочих профилей из баз данных компании.
* Защищенность: встроенный антивирус и протоколы шифрования для безопасного общения и передачи файлов.
* Кроссплатформенность: мобильное приложение, десктоп-версия, браузер — мессенджер можно будет открывать там, где удобно.
* Панель администратора:  поможет назначить роли, выдать уровень доступа и разрешения, настроить поля в профиле, а также быстро удалить или добавить сотрудника. Там же можно будет выдавать гостевые доступы для подрядчиков или клиентов.

Поддержка интеграций  корпоративного мессенджера Линк Чаты можно будет связать с *CRM*, системами документооборота и другими рабочими сервисами. На таблице 1.1 изображены плюсы и минусы данного продукта.

Таблица 1.1 – Плюсы и минусы Линк Чата

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Бесплатный | Календарь |
| Полнофункциональный: чаты, каналы, треды, рабочие области, упоминания | Отсутствуют таск–трекеры |
| Панель администратора, *LDAP*, *AD* | Отсутствует телефония |
| Есть видео и аудиозвонки |  |
| Интеграция с корпоративными сервисами |  |

*Dialog* – это мессенджер от Сбера, разработанный специально для нужд бизнеса. Он обладает широким спектром функций, делающих его идеальным инструментом для корпоративного общения. *Dialog* может быть развернут только на сервере компании клиента или в частном облаке, что обеспечивает полный контроль над данными[1].

Одним из ключевых преимуществ *Dialog* является гибкость в развертывании – он может быть реализован непосредственно на сервере компании-клиента или в частном облаке. Это обеспечивает организациям полный контроль над своими данными, что особенно важно в условиях повышенной важности безопасности информации. Кроме того, *Dialog* предлагает расширенные возможности настройки, позволяя адаптировать мессенджер под конкретные потребности и требования компании. На рисунке 1.4 отображен интерфейс мессенджера [2].

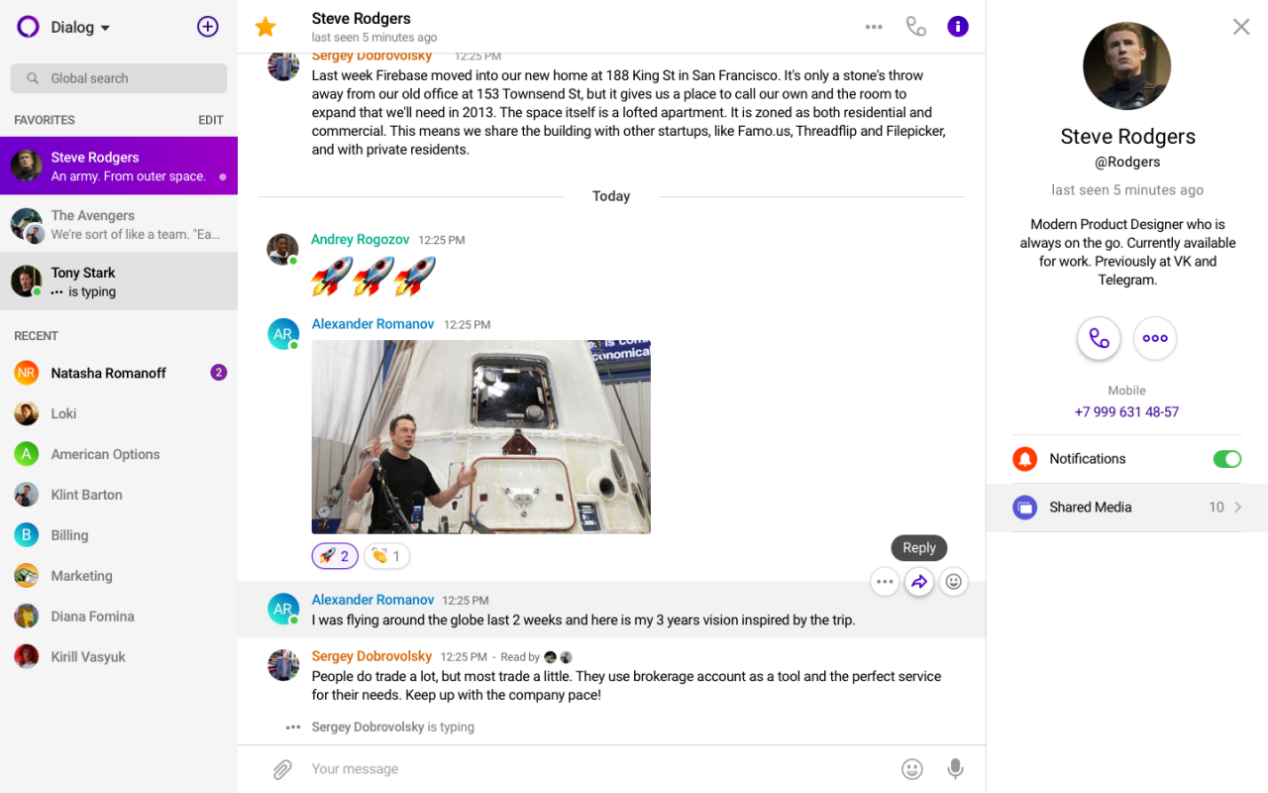


Рисунок 1.4 – Dialog

Все эти особенности делают *Dialog* привлекательным выбором для компаний, стремящихся к эффективной и безопасной платформе для внутреннего общения, спроектированной с учетом требований современного бизнеса.

Функциональность *Dialog*:

* Чаты: общайтесь с коллегами один на один или в группах.
* Каналы: создавайте каналы для трансляции важных новостей и информации всей команде.
* Видеозвонки: проводите онлайн-совещания и встречи с коллегами, клиентами и партнерами.
* Отправка файлов:  делитесь документами, изображениям и  видео а также другими файлами с пользователями.
* Интеграции: подключайте другие приложения и сервисы для повышения эффективности работы.
* Безопасность: *Dialog* соответствует всем требованиям безопасности и конфиденциальности данных.

На таблице 1.2 отображены плюсы и минусы программного продукта Dialog.

Таблица 1.2 – Плюсы и минусы Dialog

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Простой и понятный интерфейс | Нет бесплатной версии |
| Есть видео и аудиозвонки плюс телефония по внутренним номерам | Видеозвонки доступны в более дорогом тарифе «Корпоративный» |
| Возможны интеграции через *Dialog Bot SDK и Webhook* | Только коробочная версия |

*Samoware*– представляет собой инновационный корпоративный мессенджер, обеспечивающий широкий спектр функциональности для эффективной коммуникации и совместной работы в рабочей среде. Этот многофункциональный инструмент объединяет в себе возможности ведения переписки по электронной почте, осуществления видеозвонков с использованием интеграции с *TrueConf*, а также обмена текстовыми сообщениями[4].

Обмен сообщениями в *Samoware* предоставляет пользователям гибкость ведения общения как в индивидуальном порядке, так и в группах. Создание чатов для обсуждения рабочих вопросов способствует эффективной командной работе, а возможность вести переписку прямо в интерфейсе почты устраняет необходимость переключения между приложениями.

Интерфейс мессенджера *Samoware*, представленный на рисунке 1.5, демонстрирует удобный и интуитивно понятный дизайн, обеспечивающий пользователей удобством использования платформы. Однако, функциональность *Samoware* не ограничивается лишь перепиской и видеозвонками. Давайте подробнее рассмотрим его многообразные возможности.

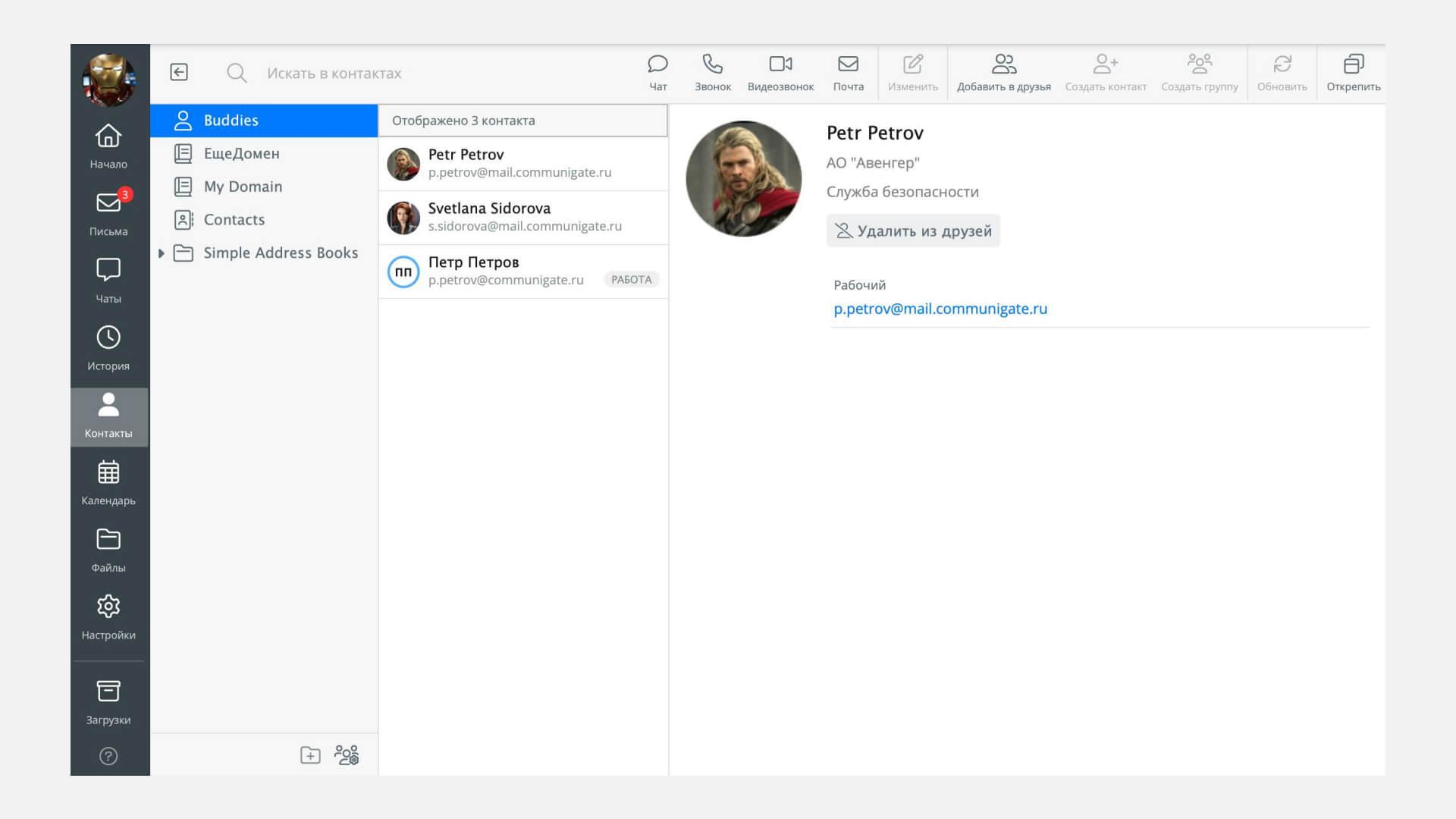


Рисунок 1.5 – *Samoware*

Функциональность *Samoware*:

* Обмен сообщениями: общайтесь с коллегами один на один или в группах, создавайте чаты для обсуждения рабочих вопросов.
* Электронная почта: ведите переписку по электронной почте прямо в интерфейсе *Samoware*, без необходимости переключаться между приложениями.
* Видеозвонки: совершайте видеозвонки с коллегами, клиентами и партнерами через интеграцию с *TrueConf*.
* Файловый обмен: документами, изображениями, видео и другими файлами с другими пользователями.
* Интеграции: подключайте другие приложения и сервисы, такие как *CRM*, календари, системы управления проектами и т.д.
* Безопасность: *Samoware* соответствует всем требованиям безопасности и конфиденциальности данных.

На таблице 1.3 отображены плюсы и минусы программного продукта *Samoware*.

Таблица 1.3 – Плюсы и минусы Dialog

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Интеграции с онлайн-документами и *CRM*–системами | Нет облачной версии, можно развернуть только на своем сервере |
| Чат-боты | Нет бесплатной версии |
| Продукт формата «всё в одном»: почта, телефония, корпоративный портал, видеосвязь | Нет тредов и каналов |
| Работает в локальных сетях | Закупка лицензии только через посредников, мобильные версии оплачиваются отдельно |
| Поддерживает российские операционные системы |  |

*Tada.Team* представляет собой инновационный корпоративный мессенджер, созданный с учетом потребностей в эффективном общении и управлении командами. Этот мессенджер не ограничивается простым обменом текстовыми сообщениями в чатах – он предоставляет уникальные функции для более глубокого взаимодействия и сотрудничества [5] .

Одной из ключевых особенностей *Tada.Team* является интеграция с таск-трекерами, позволяя пользователям создавать и отслеживать задачи прямо внутри мессенджера. Это существенно упрощает процесс координации и повышает прозрачность в выполнении проектов. Кроме того, мессенджер предоставляет возможность назначать собрания и управлять календарными событиями, обеспечивая комплексный подход к управлению временем и задачами.

Интерфейс мессенджера *Tada.Team*, представленный на рисунке 1.6, демонстрирует его удобство использования и интуитивно понятный дизайн. Пользователи могут легко переключаться между чатами, задачами и календарными событиями, обеспечивая централизованное место для ведения проектов и командной работы.

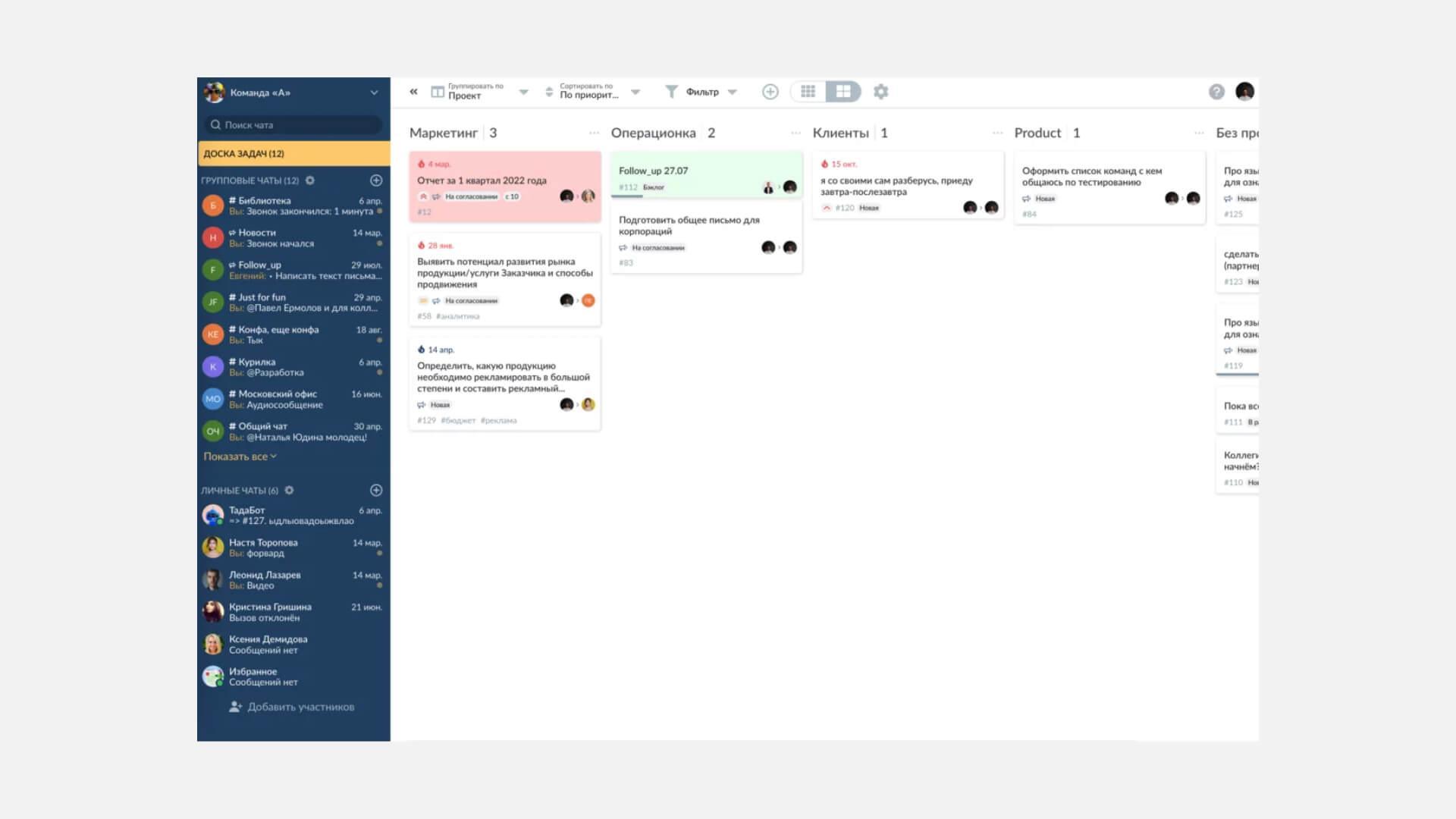


Рисунок 1.6 – *Tada.Team*

Таким образом, *Tada.Team* не только обеспечивает эффективное коммуникационное окружение для команд, но и интегрирует в себя инструменты управления задачами и временем, делая его мощным решением для комплексного ведения проектов и командной работы. На таблице 4.4 отображены плюсы и минусы программного продукта *Tada.Team*.

Таблица 4.4 – Плюсы и минусы *Tada.Team*

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Полноценное управление задачами на канбан-доске | Устаревший интерфейс |
| Чат-боты | Нет записи звонка |
| Календарь | Нет *SSO* |
| Полнофункциональный мессенджер: чаты, каналы, реакции, закрепления | Не поддерживает телефонию |
| Панель администратора |  |

## 1.5 Постановка задачи, системные требования, требования к входным данным и выходным формам

Разработать корпоративное приложение для обмена сообщениями между сотрудниками, обеспечивающее удобную и безопасную коммуникацию, управление доступом, а также расширенные функции взаимодействия и администрирования.

Основные цели разработки включают:

* Обеспечение эффективного обмена текстовыми сообщениями, мультимедийными файлами и документами.
* Реализация возможностей для индивидуальных и групповых коммуникаций.
* Предоставление инструментов для удобного взаимодействия между пользователями, включая использование эмодци , реакций и ответов на сообщения.
* Гарантия высокого уровня безопасности данных и контроль доступа на уровне пользователя.

Системные требования для приложения чата:

**Технические требования:**

1. Серверная часть:

* Операционная система: *Linux* (рекомендуется *Ubuntu* 20.04 и выше) или *Windows Server* 2019+.
* Язык программирования: *Python* (*Djang*o).
* Реализация асинхронной работы: *Django Channels Rest Framework*.
* База данных: *SQLite* версии 3.
* Среда выполнения *WebSocket*: *Redis* для поддержки реального времени.
* Хранилище медиафайлов: локальное хранилище.

1. Клиентская часть:

* Фреймворк: *React* версии 17+.
* Поддержка современных браузеров*: Chrome*, *Firefox, Edge*

**2. Функциональные требования:**

1. Сообщения:

* Текстовые сообщения.
* Отправка фотографий и документов.
* Поддержка эмодзи.
* Возможность ставить реакции на сообщения.
* Ответы и пересылка сообщений.

1. Чаты:

* Создание групповых чатов.
* Управление участниками (добавление/удаление участников администратором).

1. Безопасность:

* Авторизация пользователей через *JWT.*
* Ограничение доступа к серверным *API* только для авторизованных пользователей.
* Разграничение прав доступа (администратор/обычный пользователь).
* Ограничение доступа к административной панели только для администратора.

Управление пользователями:

* Регистрация пользователей только через администратора.
* Возможность изменения профиля пользователем.

Интерфейсные требования:

* Интуитивно понятный пользовательский интерфейс.
* Адаптивность для работы на компьютерах.

**Условия развертывания:**

1. Минимальные ресурсы для сервера:

* *CPU, 4 GB RAM,* 50 *GB SSD*.

1. Рекомендуемые ресурсы:

*CPU*, 8 *GB* *RAM*, 100 *GB* *SSD*, *Redis* для кэширования и работы *WebSocket*.

Эти системные требования обеспечивают стабильную и безопасную работу приложения для корпоративного общения.

# 2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ И АРХИТЕКТУРЫ ПРОГРАММНОГО ПРОДУКТА

## 2.1 Выбор методов и средств для реализации, его обоснование

Для разработки корпоративного мессенджера были выбраны следующие инструменты и технологии:

**Клиентская часть разработки.** Для клиентской реализации проекта была выбрана библиотека React (также известная как *React.js* или *ReactJS*) — популярная *JavaScript*-библиотека с открытым исходным кодом для создания пользовательских интерфейсов. Разработка и поддержка *React* осуществляются компанией *Facebook*, *Instagram*, а также сообществом независимых разработчиков и организаций.

*React* идеально подходит для создания одностраничных и мобильных приложений благодаря своей высокой скорости разработки, удобству использования и возможности масштабирования. Библиотека предоставляет мощный набор инструментов для построения динамических интерфейсов и эффективного управления состоянием компонентов.

Для расширения функциональности *Reac*t часто используется вместе с другими библиотеками, такими как *MobX*, *Redux* или *GraphQL*, что позволяет адаптировать технологический стек под потребности конкретного проекта.

**Серверная часть разработки.** Для серверной части проекта был выбран фреймворк *Django*, представляющий собой высокоуровневый веб-фреймворк для языка *Python*. *Django* способствует быстрой и эффективной разработке приложений, следуя принципам чистого и прагматичного дизайна. Он создан опытными разработчиками и значительно упрощает процесс веб-разработки, беря на себя реализацию типовых задач, таких как обработка запросов, управление базами данных, аутентификация пользователей и безопасность.

Одним из ключевых преимуществ *Django* является его бесплатность и открытый исходный код, что делает его доступным для использования в широком спектре проектов. Благодаря встроенным инструментам и модульной архитектуре, *Django* позволяет разработчикам сосредоточиться на функциональности приложения, избегая необходимости создания базовых компонентов "с нуля".

**Клиент-серверное взаимодействие.** В фреймворке *Django* можно реализовать как серверную так и клиентскую часть приложения. Однако в данном проекте для клиентской реализации используется *JavaScript*-библиотека *React*, поэтому для организации взаимодействия между клиентом и сервером был выбран *Django REST Framework* (*DRF*).

*DRF* является мощным и гибким инструментом для создания веб-*API*, который работает поверх фреймворка *Django*. Он предоставляет готовые решения для разработки *RESTful API*, обеспечивая удобство работы с запросами и ответами, а также обработку данных.

Ключевые преимущества *DRF* включают поддержку сериализации данных, интеграцию с *Django ORM* для взаимодействия с базой данных, а также встроенные механизмы аутентификации и авторизации. Эти возможности упрощают реализацию безопасной и масштабируемой архитектуры взаимодействия между клиентом и сервером. Использование *DRF* позволяет эффективно организовать обмен данными и интеграцию между компонентами приложения.

**Реализация реального времени.** Для обеспечения обмена сообщениями в реальном времени в проекте использовались *Django Channels* и протокол *WebSocket*. Это решение позволяет реализовать асинхронное взаимодействие между клиентами и сервером, что является ключевым элементом современных приложений для обмена сообщениями. Основные преимущества использования данного подхода:

Мгновенная доставка сообщений: пользователи могут отправлять и получать сообщения без необходимости обновления страницы, что улучшает пользовательский опыт.

Высокая производительность: *Django Channels* поддерживает одновременное взаимодействие с большим количеством пользователей, что делает его подходящим для масштабируемых приложений.

Для упрощения разработки был использован *Django Channels Rest Framework (DCRF*), который предоставляет интерфейс, похожий на *Django* *REST Framework*, для создания потребителей веб-сокетов. Это значительно сокращает время разработки и облегчает интеграцию функционала реального времени с остальной архитектурой приложения.Использование данной технологии делает систему более интерактивной и надежной, обеспечивая устойчивую работу даже при высокой нагрузке.

**Авторизация и безопасность.** Для реализации авторизации пользователей в проекте была выбрана *Token Authentication*. Этот подход использует уникальные токены, которые генерируются при аутентификации пользователя и передаются в заголовках запросов для идентификации пользователя. Использование *Token* *Authentication* позволяет обеспечивать:

Безопасную идентификацию пользователей при выполнении запросов, гарантируя, что только авторизованные пользователи могут получать доступ к защищенным данным и функционалу.

Удобство интеграции с клиентской частью, так как токен можно легко передавать с запросами между клиентом и сервером, а также хранить в локальном хранилище на стороне клиента.

Для управления доступами в приложении была использована система ролей *Django*, которая позволяет гибко ограничивать доступ к функционалу в зависимости от роли пользователя. Это решение дает возможность точно настроить права доступа, где только администраторы могут выполнять действия, связанные с регистрацией пользователей и администрированием системы.

Для реализации системы аутентификации был использован *Django* *Rest Framework (DRF*) и библиотека *Djoser*, которая предоставляет набор готовых представлений для базовых операций с пользователями, таких как:

* Регистрация нового пользователя
* Вход в систему
* Выход из системы
* Сброс пароля
* Активация учетной записи

Эта библиотека работает с пользовательской моделью пользователя, что позволяет настроить аутентификацию в соответствии с требованиями проекта и обеспечить безопасность взаимодействия между клиентом и сервером.

**Контроль версий.** В процессе разработки приложения был использован контроль версий *Git*, что позволило эффективно управлять изменениями в коде, отслеживать историю разработок и обеспечивать командное взаимодействие при разработке.

Для разделения клиентской и серверной частей проекта были созданы два отдельных репозитория:

Один репозиторий был предназначен для клиентской части, содержащей код, реализующий интерфейс и взаимодействие с сервером с использованием *React*.

Второй репозиторий был использован для серверной части, где реализована логика обработки запросов, а также взаимодействие с базой данных с помощью *Django* и *Django REST Framework*.

Все изменения в коде проектировались и сохранялись с использованием *Git*. Для дальнейшего хранения и совместной работы с кодом, а также для обеспечения его доступности для команды, оба репозитория были размещены на *GitHub*. Это позволило эффективно управлять версионностью, делиться кодом с коллегами и реализовать возможности для совместной работы и ревью изменений.

Среда разработки. В процессе работы над проектом использовались две разные среды разработки, каждая из которых была выбрана с учётом специфики клиентской и серверной частей приложения.

Для разработки клиентской части проекта использовалась среда разработки Visual Studio Code (VS Code), на рисунке 2.1 отображен интерфейс среды разработки.

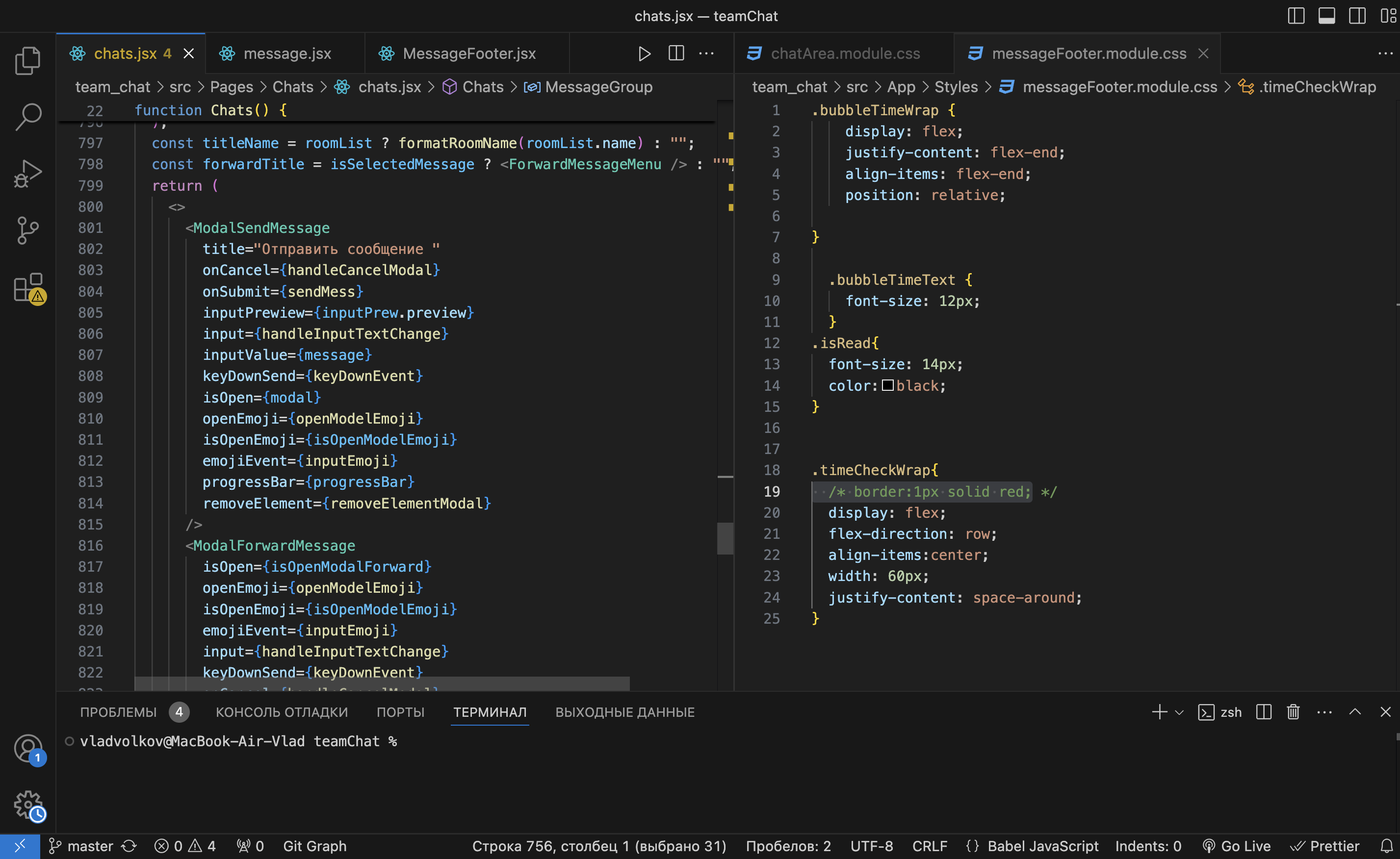


Рисунок 2.1 – Среда разработки *VS Code*

Редактор кода *VS Code* предоставляет широкий набор инструментов, необходимых для эффективного создания пользовательского интерфейса. Среди основных преимуществ VS Code можно выделить:

* Поддержка расширений: наличие множества плагинов для работы с React, JavaScript, HTML и CSS.
* Интеграция с системами контроля версий: встроенная поддержка Git позволила оперативно управлять изменениями в коде.
* Лёгкость настройки: гибкость конфигураций и возможность настройки рабочей среды под индивидуальные нужды разработчика.
* Функциональные дополнения: такие инструменты, как IntelliSense для подсказок кода, дебаггер и терминал, встроенный в редактор.

VS Code обеспечил комфортную работу над проектом и ускорил процесс разработки за счёт удобного интерфейса и возможностей для автоматизации задач.

Для разработки серверной части проекта была выбрана среда PyCharm, на рисунке 2.2 отображен интерфейс среды разработки.

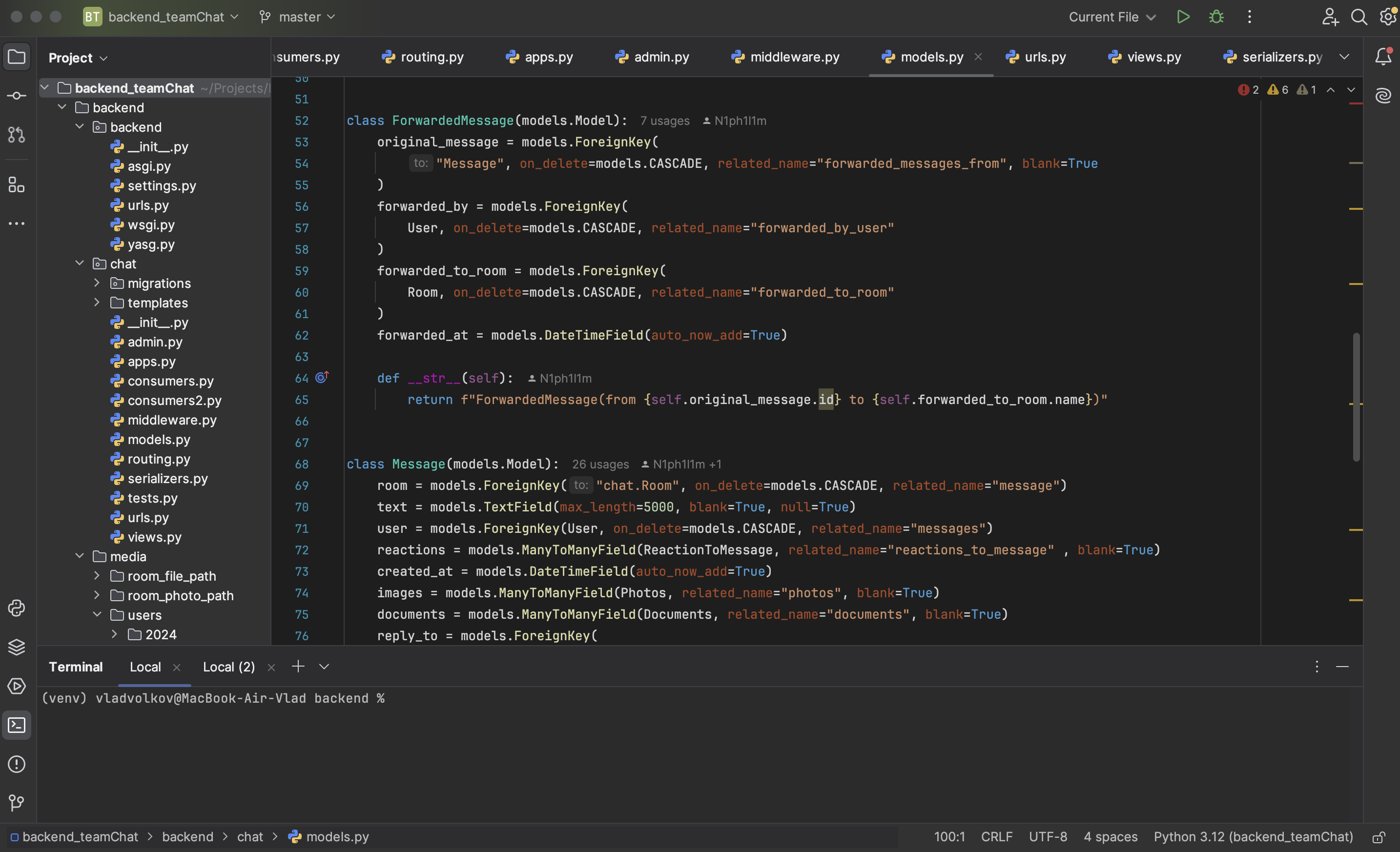


Рисунок 2.2– Среда разработки *PyCharm*

Среда разработки *PyCharm* идеально подходит для работы с проектами на Python, так как предоставляет специализированный функционал для разработки серверной логики. Основные преимущества PyCharm включают:

* Удобство работы с Django: наличие инструментов для работы с моделями, миграциями и шаблонами.
* Отладка: мощный дебаггер позволил эффективно отслеживать и устранять ошибки в коде.
* Интеллектуальные функции: автодополнение, проверка синтаксиса и статический анализ помогли повысить качество кода.
* Интеграция с базами данных: возможность управлять базами данных прямо из среды разработки.

PyCharm сыграл ключевую роль в обеспечении стабильной и надёжной серверной части приложения, упростив выполнение сложных задач и ускорив процессы разработки.

Таким образом, использование Visual Studio Code и PyCharm в рамках данного проекта позволило максимально эффективно реализовать задачи клиентской и серверной частей, учитывая их особенности и требования.

2.2 Структура, архитектура программного продукта

Эффективная организация системы и согласованное взаимодействие между её компонентами являются основой разработки современного программного обеспечения. Архитектура приложения определяет, как модули системы работают вместе, обеспечивая её масштабируемость, надёжность и удобство использования. В данном разделе рассматриваются ключевые аспекты архитектуры проекта, включая выбранный подход, распределение задач между клиентской и серверной частью, а также используемые технологии.

Архитектура программного обеспечения (ПО) — это план или структура, описывающая, как система организована, как разные ее части (компоненты) взаимодействуют между собой и с внешними системами. При построении архитектуры используются архитектурные шаблоны, позволяющие использовать наилучшие практики решения архитектурных проблем. Это как план дома, где есть стенки, комнаты, двери и окна, но заместо строй материалов мы работаем с кодом, базами данных и серверами.

Архитектуры информационных систем бывают разными и выбираются в зависимости от требований проекта. Вот основные типы архитектур:

**Монолитная архитектура** означает, что вся программа состоит из единого блока кода, где все части взаимодействуют прямо друг с другом. Если это большое приложение, в нем есть все возможности и функции в одном месте. Пример монолитной архитектуры отображен на рисунке 2.3.

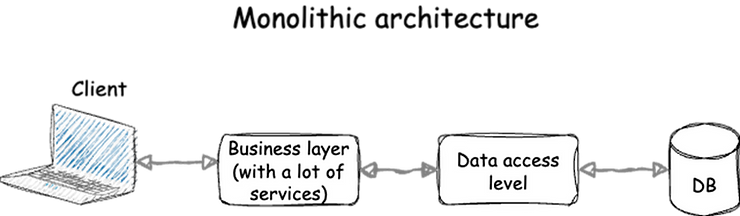


Рисунок 2.3 – Монолитная архитектура

**Пример:** бухгалтерское приложение для малого бизнеса, которое включает функции для ведения финансовых записей, создания отчетов, управления счетами, учета расходов и доходов. Монолитная архитектура в этом случае означает, что вся функциональность системы находится в одном кодовом блоке, развертывающемся как единое целое. Это означает, что все функции – от пользовательского интерфейса до базы данных – интегрированы в одну программу.

**Плюсы:**

* Простота разработки, особенно в начале.
* Легче тестировать и запускать программу, поскольку она представляет из себя единое целое.
* Четкое представление обо всех взаимосвязях в программе.

**Минусы:**

* С ростом системы код становится громоздким и трудным для поддержки.
* Изменение одной части может повлиять на другие части, что может привести к ошибкам.
* Сложно масштабировать отдельные функции системы.

**Микросервисная архитектура** разделяет систему на набор мелких автономных частей (микросервисы), где каждая из них выполняет отдельную функцию. Это практически мини-программы. Эти части взаимодействуют между собой через специальные протоколы (например, через *API*). Пример микросервисной архитектуры отображен на рисунке 2.4

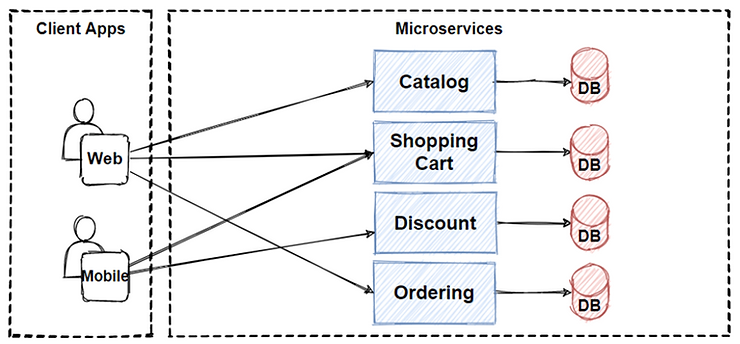


Рисунок 2.4 – Микросервисная архитектура

**Пример**: Интернет–магазин в такой архитектуре может иметь отдельный микросервис для управления товарами, другой для обработки заказов и еще один для работы с пользователями. Все они работают независимо друг от друга, но вместе обеспечивают работу магазина..

**Плюсы:**

* Легче масштабировать отдельные части системы.
* Можно обновлять или изменять один микросервис, не касаясь других.
* Простота в поддержке больших команд разработчиков, любая из которых может работать над своим микросервисом.

**Минусы:**

* Сложность в настройке взаимодействия между микросервисами.
* Необходимо хорошо продумывать тестирование и безопасность, поскольку система состоит из многих частей.
* Усложнено управление инфраструктурой, поскольку каждый микросервис нуждается в отдельных ресурсах.

**Клиент–серверная архитектура** состоит из двух частей: клиент (обычно это то, что видит пользователь, например браузер) и сервер (место, где хранятся данные и выполняется основная логика). Пример клиент-серверной архитектуры отображен на рисунке 2.5.

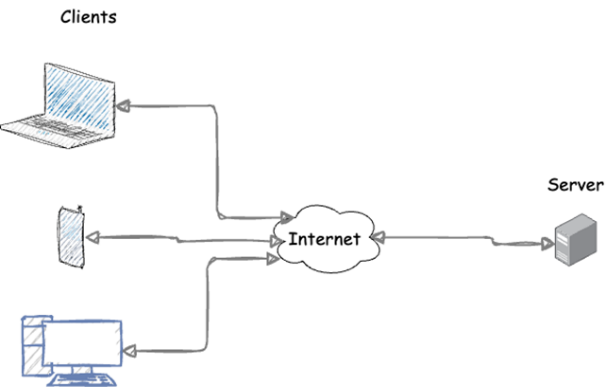


Рисунок 2.5 – Клиент-серверная архитектура

**Пример**: Банковская программа. Клиент – это мобильное приложение на телефоне, а сервер – это компьютеры банка, где хранятся счета и обрабатываются транзакции. Пример микросервисной архитектуры отображен на рисунке 2.3.

**Плюсы:**

* Четкое разделение на две части: клиент для взаимодействия с пользователем, сервер для обработки данных.
* Сервер можно легко обновлять и изменять, не затрагивая клиент.

**Минусы:**

* Если сервер не работает, клиент не может работать.
* Клиент и сервер должны постоянно взаимодействовать, что может создавать проблемы с производительностью или соединением.

Для реализации корпоративного чата была выбрана клиент-серверная архитектура, поскольку она обеспечивает четкое разделение между клиентской частью, ответственной за пользовательский интерфейс и взаимодействие, и серверной частью, где осуществляется обработка данных, хранение сообщений и управление пользователями. Этот подход позволяет:

* обеспечить гибкость в разработке, обновлении и масштабировании серверной и клиентской частей независимо друг от друга;
* легко интегрировать сервер с другими системами, такими как базы данных и модули безопасности, без необходимости внесения изменений в клиент;
* реализовать централизованное управление данными и их синхронизацию между несколькими пользователями в реальном времени;
* клиент–серверная архитектура идеально подходит для корпоративных систем, где важны высокая надежность, возможность масштабирования, а также безопасность и контроль над передачей данных;

Система корпоративного чата включает несколько ключевых компонентов, каждый из которых выполняет свою роль в обеспечении функциональности и стабильности приложения. Взаимодействие между клиентской частью, сервером и базой данных позволяет реализовать такие возможности, как обмен сообщениями в реальном времени, хранение данных пользователей и файлов, а также безопасную авторизацию. В данном разделе описаны основные составляющие системы, их задачи, а также применяемые технологии, которые обеспечивают эффективную работу приложения.

**Клиентская часть** отвечает за взаимодействие пользователя с системой. Она предоставляет удобный интерфейс для обмена сообщениями, просмотра данных и управления настройками.

**Основные задачи:**

* Отображение интерфейса чата с группами и личными сообщениями.
* Обработка пользовательского ввода (ввод текста, отправка сообщений, добавление вложений).
* Реализация маршрутизации между различными страницами приложения (например, список чатов, профиль пользователя).
* Обеспечение взаимодействия с серверной частью через *API* и *WebSocket*.

**Используемые технологии:**

* *React*: для создания динамичного пользовательского интерфейса.
* React Router: для управления маршрутизацией и навигацией.
* WebSocket: для поддержки обмена сообщениями в реальном времени.

**Серверная часть** реализует бизнес-логику, обрабатывает запросы клиентов и управляет передачей данных.

**Основные задачи:**

* Обработка *HTTP*-запросов для взаимодействия с базой данных (REST API).
* Поддержка соединений в реальном времени для доставки сообщений с минимальной задержкой (*WebSocket*).
* Управление авторизацией и проверкой подлинности пользователей.
* Обеспечение сохранности данных и их безопасности.

**Используемые технологии:**

* *Django*: для реализации серверной логики, работы с базой данных и создания *REST API*.
* *Django Channels*: для обработки *WebSocket*-соединений и реализации функциональности реального времени.
* DRF (Django REST Framework): для создания API.

**База данных** является центральным хранилищем всей информации, необходимой для работы приложения.

**Основные задачи:**

* Хранение данных пользователей (профили, аватары).
* Хранение сообщений, включая текст, вложения (изображения, документы) и метаданные (время отправки, статус прочтения).
* Хранение информации о комнатах (личных и групповых чатах).

**Используемая технология:**

* *SQLite* : для обеспечения надежного и масштабируемого хранения данных.

Такая структура системы обеспечивает гибкость, масштабируемость и удобство в поддержке, что особенно важно для корпоративного чата с высокой активностью пользователей.

* 1. Проектирование базы данных

Проектирование базы данных является одним из ключевых этапов разработки программных систем, так как от его качества зависит эффективность хранения, обработки и извлечения данных. Этот процесс включает определение структуры данных, их взаимосвязей, а также методов взаимодействия между приложением и базой данных. В данном разделе рассматриваются основные подходы к созданию базы данных для корпоративного чата, разрабатываемого на фреймворке *Django*.

Для реализации хранения и обработки данных в проекте была выбрана система управления базами данных *SQLite3*, встроенная в *Django*. Данный выбор обусловлен её лёгкостью использования, отсутствием необходимости в дополнительной настройке и полной интеграцией с инструментами фреймворка.

Модель в *Django* является единственным источником информации о данных. Она содержит основные поля и поведение данных, которые хранятся в приложении . Как правило, каждая модель отображается в одну таблицу базы данных.

Основные принципы работы с моделями:

1. Структура модели – каждая модель в *Django* представлена в виде класса Python, который является подклассом *django.db.models.Model*. Это наследование обеспечивает доступ ко всем инструментам, необходимым для работы с объектами базы данных.
2. Поля модели – атрибуты класса модели соответствуют полям базы данных. Они определяют тип данных (например, строка, число, дата) и параметры валидации, такие как длина строки, уникальность, обязательность заполнения и т. д. *Django* предоставляет широкий набор типов полей, включая CharField(тектовый формат), *IntegerField*(числовой формат), *DateTimeField*(формат даты и времени) и многие другие, что позволяет гибко проектировать структуру данных.
3. Автоматический *API* для работы с данными – *django* автоматически генерирует API для выполнения операций с базой данных. Это включает создание, чтение, обновление и удаление записей, что значительно упрощает взаимодействие с базой. Например, вызовы методов, таких как *objects.create(), objects.filter(), objects.update(),* позволяют разработчику быстро и эффективно работать с данными.
4. Дополнительные возможности:

* Модели поддерживают установку отношений между таблицами через такие поля, как *ForeignKey, ManyToManyField и OneToOneField*. Это позволяет создавать сложные взаимосвязанные структуры данных.
* Метаданные модели, определяемые в классе *Meta*, позволяют задавать дополнительные параметры, такие как имя таблицы, порядок сортировки записей и другие настройки.
* Модели могут содержать собственные методы, которые описывают специфическое поведение данных или дополнительные вычисления, упрощая их обработку в логике приложения.

Пример реализации модели Room (Комната) приложения корпоративный чат отображен на листинге 1.ё

|  |
| --- |
| from django.db import models  class Room(models.Model): name = models.CharField(max\_length=255, null=False, blank=True, unique = True)  host = models.ForeignKey(User, on\_delete=models.CASCADE,  related\_name= "rooms")  current\_users = models.ManyToManyField(User,related\_name="current\_rooms", blank = True)  photo\_room = models.ImageField(upload\_to="users/%Y/%m/%d", blank=True, null=True, verbose\_name="Фотография", default="users/default\_groupChat.png")  def \_\_str\_\_(self):  return f"Room({self.name}{self.host})" |

Листинг 2.1 – Модель Room

Модель *Room* представляет собой таблицу базы данных, которая используется для хранения информации о комнатах (чатах) в приложении. Она содержит поля и методы, определяющие основные атрибуты комнаты и её поведение.

После написания модели в *Django* необходимо создать миграцию для внесения изменений в структуру базы данных. Миграция — это механизм, который позволяет автоматически преобразовывать описание моделей в таблицы базы данных. Процесс создания миграции состоит из нескольких шагов.

Сначала необходимо убедиться, что файл модели корректно написан и импортированы все необходимые модули. После этого в терминале используется команда *python* manage.py *makemigrations*. Эта команда сканирует все изменения в моделях приложения и создаёт файл миграции, содержащий инструкции для внесения соответствующих изменений в базу данных.

После создания файла миграции применяется команда *python manage.py migrate*. Она выполняет ранее сгенерированный файл миграции, преобразуя описания моделей в реальные таблицы базы данных. Если всё выполнено правильно, в базе данных появятся новые таблицы или будут обновлены существующие в соответствии с последними изменениями.

Например, если была добавлена модель *Room*, содержащая поля *name, host, current\_users и photo\_room, Django* автоматически создаст таблицу с этими полями и связями. Этот процесс позволяет эффективно синхронизировать код и структуру базы данных без необходимости ручного написания *SQL*-запросов. Миграции должны применяться каждый раз, когда вносятся изменения в модели. Это гарантирует, что структура базы данных всегда соответствует описанию в коде.

Для реализации функциональности приложения было созданы следующие модели данных:

* *Room* – комната чата.
* *Photos* – изображения .
* *Documents* – документами.
* *ReactionToMessage* – с реакциями для сообщения.
* *ForwardedMessage* – пересланные сообщения.
* *Message* – сообщения.
* *Users* – пользователи.

Каждая модель представлена как таблица в базе данных, и каждая из них играет важную роль в обеспечении работы приложения. Ниже приведено подробное описание каждой модели, её полей и связей с другими моделями.

Модель ***Room****,* представленная в таблице 2.3, включает информацию о комнате чата, а именно: кто является её создателем (хозяином), какие пользователи относятся к комнате, а также изображение, ассоциированное с этой комнатой.

Таблица 2.1 – Модель Room

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| *id* | *Integer* | Уникальный идентификатор |
| *name* | *Varchar(255)* | Название |
| *host* | *Bigint* | Создатель комнаты |
| *current\_users* | *Integer* | Пользователе комнаты |
| *photo\_room* | *Varchar(255)* | Фото комнаты |

Модель **Photos**, представленная в таблице 2., хранит иформацию о пути к изображению и дату добавления.

Таблица 2.1 – Модель Room

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| *id* | *Integer* | Уникальный идентификатор |
| *image* | *Varchar(255)* | Пусть к изображению |
| *upload\_ad* | *Datetime* | Время добавления |

Модель **Documents**, представленная в таблице 2., хранит иформацию о названии файла, его пути а так же дату добавления.

Таблица 2.1 – Модель Room

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| *id* | *Integer* | Уникальный идентификатор |
| *name* | *Varchar(255)* | Название файла |
| *documents* | *varchar(255)* | Путь к файлу |
| *upload\_at* | *Datetime* | Время добавления |

Модель *ReactionToMessage* , представленная в таблице 2., хранит информацию о реакциях к сообщениям.

Таблица 2.1 – Модель Room

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| *id* | *Integer* | Уникальный идентификатор |
| *Emoji* | *Varchar(255)* | Выбранный эмодзи |
| *id\_user\_id* | *varchar(255)* | Уникальный идентификатор пользователя, добавивший реакцию |

Модель *ForwardedMessage* , представленная в таблице 2., хранит информацию о пересланных сообщений пользователей.

Таблица 2.1 – Модель *Room*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| *id* | *Integer* | Уникальный идентификатор |
| *Forwarded\_by\_id* | *varchar(255)* | Уникальный идентификатор пользователя, пересылающий сообщение |
| *Forwarded\_to\_room\_id* | *Bigint* | Уникальный идентификатор комнаты |
| *Original\_message\_id* | *Bigint* | Уникальный идентификатор сообщения |
| *Forwarded\_at* | *Datetime* | Выбранный эмодзи |

Модель User, представленная в таблице 2. , используется для хранения информации о пользователях корпоративного чата.

Таблица 2.1 – Модель *Use*r

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| *id* | *Integer* | Уникальный идентификатор |
| *username* | *Varchar(255)* | Логин пользователя |
| *first\_name* | *Varchar(255)* | Фамилия пользователя |
| *last\_name* | *Varchar(255)* | Имя пользователя |
| *email* | *Varchar(255)* | Выбранный эмодзи |
| *data\_birth* | *Datetime* | Дата рождения |
| *Photo* | *Varchar(255)* | Фото профиля |
| *Password* | *Varchar(128)* | Пароль |
| *Last\_login* | *Datetime* | Последняя авторизация |
| *Is\_superuser* | *Bool* | Статус суперпользователя |
| *is\_staff* | *Bool* | Доступ к админ панели |
| *is\_activate* | *Bool* | Активен ли пользователя в системе |
| *data\_joined* | *Bool* | Дата регистрации пользователя |

Модель Message, представленная в таблице, используется для хранения информации о сообщениях в чате и включает несколько важных полей и связей, которые обеспечивают функциональность чата.

модели

Таблица 2.1 – Модель *Room*

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| *id* | *Integer* | Уникальный идентификатор |
| *room* | *Varchar(255)* | Внешний ключ к модели *Room*, который указывает, в какой комнате было отправлено сообщение |
| *text* | *Text* | Текст сообщения |
| *user* | *Integer* | Внешний ключ к модели User, который указывает, кто отправил сообщение |
| *reactions* | *Bigint* | Связь многие ко многим с моделью *ReactionToMessage*, которая хранит реакции на сообщение |
| *images* | *Bigint* | Связь многие ко многим с моделью *Photos*, которая хранит изображения, прикрепленные к сообщению |
| *documents* | *Bigint* | Связь многие ко многим с моделью *Documents*, которая хранит документы, прикрепленные к сообщению. |
| *reply\_to* | *Bigint* | Внешний ключ к самой модели Message, который указывает на сообщение, на которое данное сообщение является ответом |
| *forwarded\_messages* | *Bigint* | Связь многие ко многим с моделью *ForwardedMessage*, которая хранит сообщения, пересланные в другие комнаты. |

Проектирование базы данных является важнейшим этапом разработки корпоративного чата, так как именно оно определяет основу для хранения и управления данными. Разработанные модели обеспечивают эффективную реализацию всех необходимых функций приложения, включая хранение информации о пользователях, комнатах, сообщениях и их связях. Благодаря применению возможностей *Django*, таких как автоматическое создание API для работы с данными и гибкость настройки моделей, процесс разработки базы данных был значительно упрощен.

2.4 Описание логической структуры программного продукта

Программный продукт основан на клиент-серверной архитектуре, где клиентская часть разработана с использованием *React*, что обеспечивает создание динамичных и интерактивных пользовательских интерфейсов, а серверная часть реализована на *Django*, который отвечает за обработку запросов, реализацию бизнес-логики и управление данными. Взаимодействие между клиентом и сервером осуществляется через API, использующее протокол HTTP(S), а для обмена сообщениями в реальном времени применяются *WebSocket*-соединения.

Клиентское приложение представляет собой интерфейс пользователя, через который осуществляется взаимодействие с системой. Оно отвечает за следующие задачи:

* Обработка пользовательских событий: Интерфейс обрабатывает действия пользователя, такие как ввод текста, отправка сообщений, загрузка файлов и навигация между страницами.
* Отображение данных: Полученные от сервера данные отображаются в удобной и наглядной форме. Используются компоненты *React*, которые обеспечивают обновление интерфейса в реальном времени.
* Управление состоянием: Для управления состоянием приложения используются встроенные хуки *React*, такие как *useState* и *useReducer*. Эти инструменты позволяют эффективно обрабатывать изменения данных и синхронизировать их между различными компонентами. Хуки предоставляют гибкий подход к работе со состоянием, упрощая управление как локальными состояниями компонентов, так и более сложными сценариями, такими как обмен данными между компонентами или обработка асинхронных операций.
* Работа с *API*: Клиент отправляет запросы к серверу через *REST* *API,* обрабатывая полученные ответы. Для отправки запросов и обработки ответов используется библиотека *axios*, которая обеспечивает удобный и мощный инструмент для выполнения HTTP-запросов.
* Обмен сообщениями в реальном времени: Для взаимодействия пользователей в режиме реального времени используются *WebSocket*-соединения. Это позволяет мгновенно передавать сообщения, обновления статусов и другую информацию.

Серверное приложение реализовано с использованием *Django* и *Django REST Framework* (DRF), что обеспечивает гибкость и модульность системы. Сервер отвечает за следующие задачи:

* Обработка запросов *API*: Сервер предоставляет *REST API* для обмена данными с клиентом. Это позволяет клиенту выполнять операции, такие как отправка и получение сообщений, аутентификация, управление пользователями и работа с медиа-файлами.
* Обмен данными в реальном времени: *Django Channels* используется для реализации *WebSocket*-соединений. Это позволяет обрабатывать события в реальном времени, такие как доставка сообщений или уведомлений о прочтении.
* Управление данными: Сервер управляет базой данных, хранящей информацию о пользователях, чатах, сообщениях, медиа-файлах и прочих данных. *Django ORM* обеспечивает удобный доступ к данным и их обработку.
* Аутентификация и авторизация: Для обеспечения безопасности используется система аутентификации на основе токенов, позволяющая защитить данные и ограничить доступ к ресурсам.
* Обработка медиа-файлов: Сервер обрабатывает загрузку и хранение файлов, включая изображения и документы, а также предоставляет ссылки для их скачивания клиентом.

Связь между клиентом и сервером реализована следующим образом:

* *API: REST API* основан на протоколе *HTTP(S).* Клиент отправляет запросы на сервер (*GET, POST, PUT, DELETE*), получает ответы в формате JSON и обрабатывает их. На рисунке , отображена схема работы REST API взаимодействия.

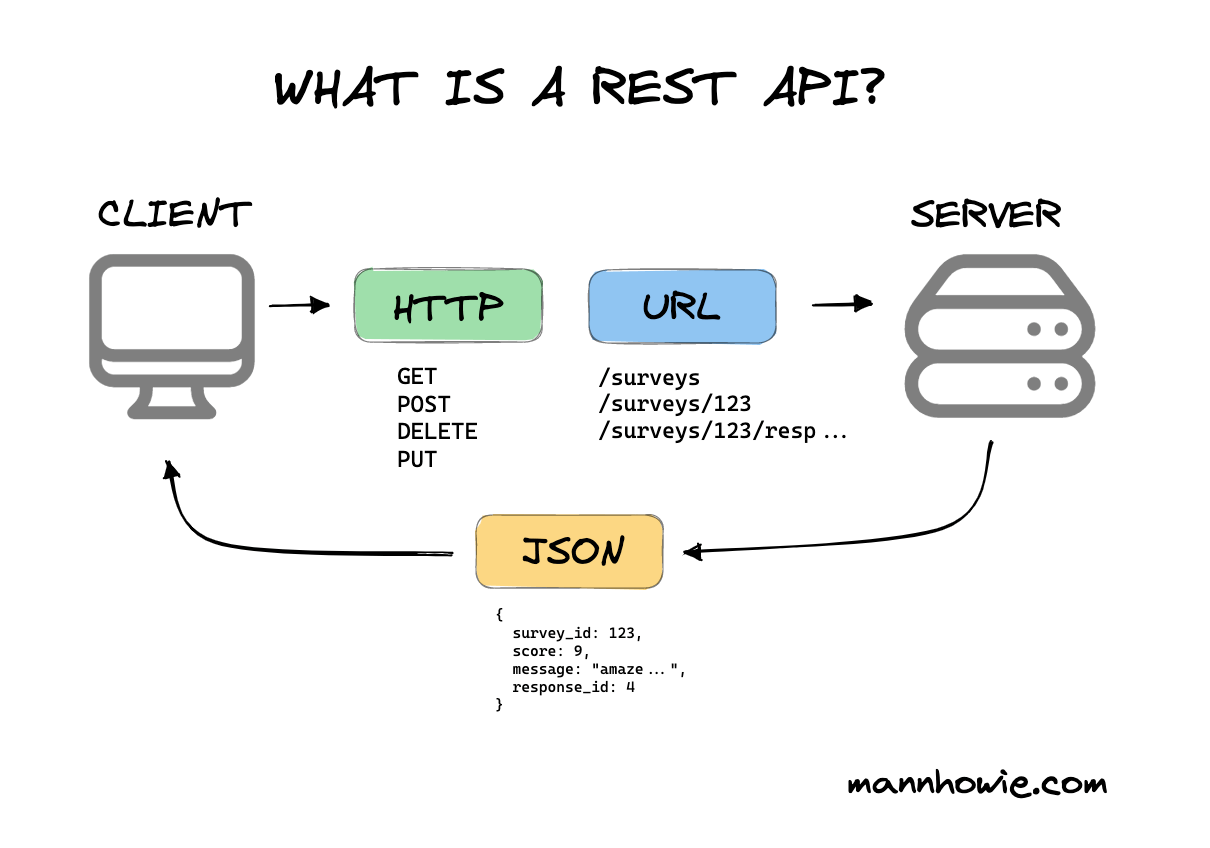


Рисунок 2.5 – REST API

* WebSocket-соединения: *WebSocket* используется для передачи данных в реальном времени. Клиент инициирует соединение с сервером через специальный эндпоинт, после чего сервер поддерживает двухсторонний канал связи для мгновенной передачи данных. Это позволяет как клиенту, так и серверу обмениваться информацией без необходимости повторно устанавливать соединение, что существенно снижает задержку и экономит ресурсы. *WebSocket* идеально подходит для приложений, где требуется передача данных в реальном времени, таких как чаты, игры или финансовые приложения. На рисунке 2.6 отображена схема работы *WebSocket* и *HTTP*.

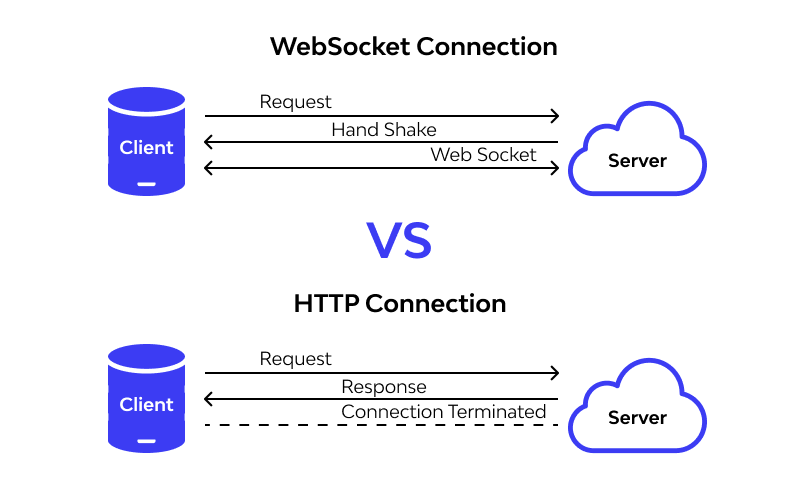


Рисунок 2.6 – Схема соединений *WebSocket* и *HTTP*

В отличие от *WebSocket*, *HTTP*-запросы предполагают однонаправленную связь, где клиент отправляет запрос серверу, а сервер отвечает, после чего соединение закрывается. Каждый новый запрос требует открытия нового соединения. Это делает *HTTP*-запросы менее эффективными для приложений, где требуется постоянный обмен данными, так как каждый запрос и ответ требует времени на установку соединения. *HTTP*-запросы лучше всего подходят для операций, не требующих немедленного обновления данных, таких как загрузка страниц или отправка форм.

* Организация соединений: Соединение устанавливается с использованием стандартных библиотек *Django* и *React*. Для обработки ошибок и повторного подключения в случае разрыва соединения используются дополнительные механизмы, встроенные в клиентскую часть.
* Закрытие соединений: *WebSocket*-соединения закрываются либо при разрыве связи, либо по запросу клиента, чтобы оптимизировать использование ресурсов.

Клиентская и серверная части взаимодействуют посредством API и WebSocket, обеспечивая слаженную работу всей системы. Основные компоненты включают:

* Классы на стороне клиента: реализуют функциональность отправки и получения данных, работу с состоянием и пользовательским интерфейсом.
* Классы на стороне сервера: отвечают за маршрутизацию запросов, обработку данных и управление соединениями.
* Таким образом, взаимодействие между клиентом и сервером, основанное на REST API и WebSocket, обеспечивает гибкость, производительность и удобство использования программного продукта.

2.5 Функциональная схема, функциональное назначение программного

продукта

Функциональная схема  схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств. Для изображения функциональных схем используют специальные обозначения, установленные стандартом. Функциональные схемы более информативны, чем структурные. Все компоненты функциональных схем должны быть описаны. Следует тщательно прорабатывать спецификации межпрограммных интерфейсов, так как от качества их описания зависит количество самых дорогостоящих ошибок, к которым относятся ошибки, обнаруживаемые при комплексном тестировании.

Функциональная схема корпоративного чата включает несколько ключевых модулей, авторизация, главная страница, отправка сообщения. Каждый из этих модулей состоит из последовательных этапов и процессов, описывающих взаимодействие пользователя с интерфейсом приложения, управление состоянием с помощью хуков *React* и работу с базой данных *Django*.

Блок «Авторизации» включает процесс входа пользователя в систему. Функциональная схема авторизации отображена на рисунке 2.7.



Рисунок 2.6 – Блок «Авторизации»

Пользователь вводит логин , пароль нажимает на кнопку «*Login*». Введенные данные отправляются на сервер и проверяет их на корректность. Если данные корректные, создается токен пользователя и отправляется ответ клиенту. В клиентской части после получения ответа от сервера, имя пользователя и токен сохраняется в локальное хранилище для дальнейшей логики бизнес процессов. Следующим этапом на клиенте появляется сообщение об успешной авторизации и происходит переадресация на главную страницу приложения.

В блоке «Главная страница» отображены основные компоненты для взаимодействия с другими пользователя. Функциональная схема главной страницы отображена на рисунке 2.8.



Рисунок 2.8 – Блок «Главная страница»

На главной странице имеются 3 основных компонента:

1. Список чатов
2. Список групповых чатов
3. Контакты

Компоненты реализованы посредством выпадающего списка, при активации которого осуществляется запрос к серверу от авторизованного пользователя для получения данных, связанных с ним. После получения ответа от сервера, данные отображаются в соответствующем выпадающем списке.

Блок «Чата» в данном блоке отображены отправленные и полученные сообщения пользователей. Функциональная схема чата отображена на рисунке 2.9.

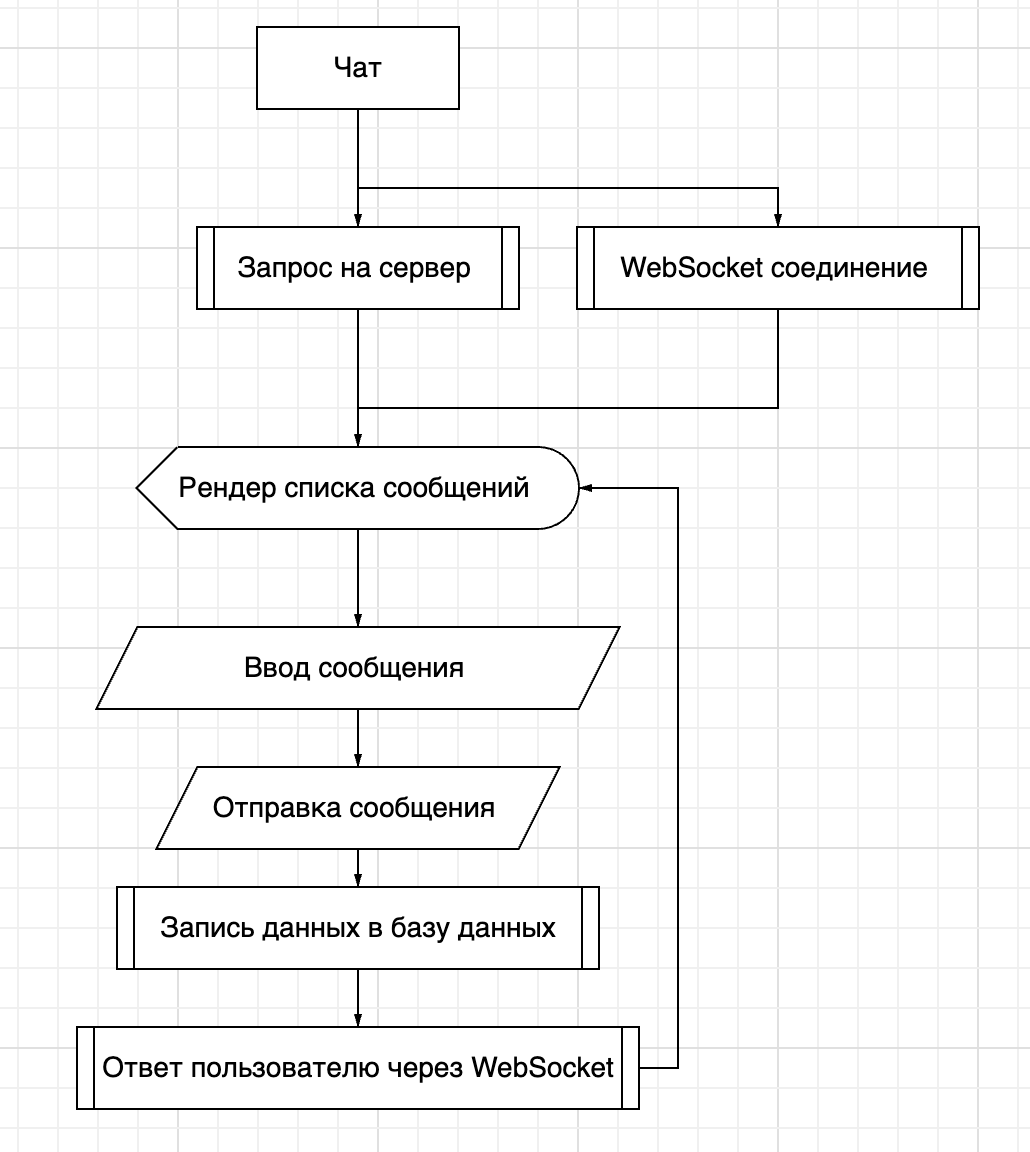


Рисунок 2.9 – Блок «Чата»

При входе пользователя в чат с другим пользователем осуществляется запрос к серверу для получения списка сообщений для соответствующей комнаты чата. После получения данных от сервера, список сообщений добавляется в *React state*, который затем используется для отображения сообщений в соответствующем блоке. Далее устанавливается *WebSocket*-соединение для получения сообщений в реальном времени. Когда пользователь вводит сообщение в текстовом поле и отправляет его, нажимая соответствующую кнопку или клавишу «Enter», данные отправляются на сервер. На сервере полученное сообщение сохраняется в базе данных, после чего, после успешного сохранения, сервер отправляет ответ клиенту через *WebSocket*. Когда клиент получает ответ от сервера, *React state* обновляется, и происходит перерисовка блока сообщений. Благодаря *WebSocket*-соединению список сообщений обновляется в реальном времени.

# **ЗАКЛЮЧЕНИЕ**

# **ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ, ЕДИНИЦ**

# **И ТЕРМИНОВ**

# **СПИСОК ЛИТЕРАТУРЫ**