МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра системного анализа и автоматического управления

РАЗРАБОТКА КОРПОРАТИВНОГО МЕССЕНДЖЕРА НА НЕСКОЛЬКИХ ПЛАТФОРМАХ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Ладяева Александра Алексеевича

Научный руководитель
к. ф.-м. н., доцент _____ И. Е. Тананко

Заведующий кафедрой
к. ф.-м. н., доцент _____ И. Е. Тананко

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. Информационные технологии и средства коммуникации стремительно развиваются и стали неотъемлемой частью жизни общества. Каждая организация нуждается в бесперебойном обмене информацией между сотрудниками независимо от того, какими устройствами и операционными системами они пользуются. В течение последних десятилетий мессенджеры получили широкое распространение в корпоративной среде. Компании нередко используют доступные, простые и недорогие технологии, чтобы повысить продуктивность и наладить внутреннюю коммуникацию. Вместе с этим использование сторонних решений создает риск утечки конфиденциальных данных. Готовые продукты не всегда подходят под конкретные требования организации, а в долгосрочной перспективе могут оказаться дорогостоящими из-за необходимости приобретать лицензии. Собственный корпоративный мессенджер с разграничением доступа помогает защитить данные компании, сократить расходы и учесть все особенности корпоративной культуры и процессов организации. Зарубежные решения либо недоступны на территории Российской Федерации, либо не отвечают требованиям безопасности при передаче конфиденциальных данных, а отечественные продукты сложно адаптировать под специфику конкретных бизнес-процессов. Данная проблематика и обусловила актуальность разработки собственного корпоративного мессенджера.

Цель бакалаврской работы — разработка базового функционала корпоративного мессенджера, поддерживающего распространенные платформы.

Поставленная цель определила следующие **задачи**:

1. провести анализ существующих решений в области корпоративных коммуникаций и выявить их основные преимущества и недостатки;
2. сформировать требования к новому программному продукту;
3. разработать архитектуру системы с учетом требований масштабируемости и безопасности;
4. выбрать оптимальный стек технологий и инструментарий;
5. подготовить дизайн настольного и мобильного клиента мессенджера;
6. реализовать версию приложения с полноценным функционалом;
7. провести полноценное тестирование разработанного прототипа на наличие критических ошибок и проблем с производительностью.

Методологические основы разработки корпоративного мессенджера представлены в работах Е. Яровой по принципам построения архитектуры программного обеспечения [1], Н. Лимановой и И. Селезнева по анализу эффективности клиент-серверной архитектуры [2], Р. Мартина по чистой архитектуре [3], С. А. Чернышева, Ю. М. Петрова, С. П. Ильина и П. А. Гершевича по фреймворку Flutter [4], Э. Троелсена и Ф. Джепикса по платформе .NET и языку программирования C# [5], Е. Игнатьева по защите информации и криптографическим алгоритмам хеширования [6].

Теоретическая значимость бакалаврской работы заключается в систематизации подходов к разработке корпоративных коммуникационных решений и формировании методологической базы для создания кроссплатформенных систем обмена сообщениями.

Практическая значимость бакалаврской работы заключается в разработке корпоративного средства коммуникации для использования внутри организации и интеграции с ее сервисами.

Структура и объём работы. Бакалаврская работа состоит из введения, 6 разделов, заключения, списка использованных источников и 9 приложений. Общий объём работы — 90 страниц, из них 56 страниц — основное содержание, включая 2 рисунка, цифровой носитель в качестве приложения, список использованных источников информации — 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор существующих аналогов» посвящен анализу современного рынка средств корпоративных коммуникаций. Рынок условно разделен на две группы: универсальные мессенджеры (VK Messenger, Max, ОК Сообщения), ориентированные прежде всего на личное общение, и специализированные бизнес-платформы, изначально создаваемые для систематизации внутренних коммуникаций и организации совместной работы сотрудников. Универсальные решения предоставляют привычные инструменты для обмена сообщениями, файлами и проведения видеозвонков, однако их возможности, как правило, не выходят за рамки повседневных потребностей пользователя.

В разделе были изучены распространенные отечественные корпоративные платформы. Яндекс.Мессенджер тесно интегрирован с почтой, календарем и облачным хранилищем, умеет автоматизировать рутину и расшифровывать

голосовые сообщения. Compass делает ставку на безопасность и порядок в процессах: гибко управляет правами доступа, открывает гостевой доступ для клиентов и подрядчиков, предлагает расширенный поиск и аналитику активности. Контур.Толк выделяется открытым API, который позволяет разрабатывать собственных ботов и адаптировать функциональность под конкретный бизнес-процесс. У каждого из этих решений имеются и сильные стороны, и ограничения. При этом зарубежные продукты либо недоступны в России, либо не удовлетворяют требованиям по защите конфиденциальных данных, а отечественные нередко ограничены функционалом готового решения, которое бывает тяжело интегрировать со специфическими бизнес-процессами. Собственный мессенджер лишен подобных ограничений — он учитывает нюансы работы организации, закрывает все ее потребности и в долгосрочной перспективе экономит средства, избавляя от постоянных затрат на лицензии. Именно это и делает разработку собственного корпоративного мессенджера актуальной.

Второй раздел «Сбор требований к программному продукту» посвящен функциональным и техническим требованиям, которые закладывают фундамент будущего интерфейса и архитектуры. Функциональные требования удобно рассматривать по компонентам, каждый из которых отвечает за конкретную потребность пользователя. Вход в систему складывается из регистрации, аутентификации и авторизации; учетная запись защищается двухфакторной аутентификацией по телефону или электронной почте, а при желании можно установить дополнительный облачный пароль. Коммуникационный модуль должен обладать стандартными возможностями — отправлять, получать, удалять, редактировать и искать сообщения, а также выполнять специфические функции: разделять чаты на личные и групповые, при необходимости разделять их на каналы, ограничивать отправку сообщений в определенное время, закреплять сообщения, разграничивать роли участников и интегрироваться с бизнес-системами. Продукт включает в себя настольное приложение для Windows 10, мобильное приложение для Android (версия 9 и выше) и iOS (версия 11 и выше) и серверную часть, причем интерфейсы API на сервере и клиентах должны развиваться синхронно, чтобы не возникало проблем несовместимости.

К надежности требования достаточно строгие: доступность от 99,9% времени (то есть простой не более 9 часов в год), работа в режиме 24/7/365 с минимальными перерывами на обслуживание, горячее резервирование критиче-

ских компонентов и отсутствие единых точек отказа. Время отклика не должно выходить за пороговые значения даже под нагрузкой: текстовые сообщения доставляются за 1–2 секунды, история чата загружается за 2–3 секунды, а передача файлов сопровождается индикацией прогресса. Система обязана обрабатывать ошибки, сохранять атомарность операций и целостность данных и защищать их от несанкционированного доступа и атак типа «отказ в обслуживании». Отдельно выделены требования к технике для настольного и мобильного клиента — процессор, объем оперативной памяти, свободное место на диске и параметры сетевого подключения.

Третий раздел «Выбор архитектуры и средств реализации» посвящен обоснованию архитектуры и технологического стека: именно эти решения определяют, насколько хорошо система будет масштабироваться и выдерживать нагрузку [1]. Проектирование начинается с UML-диаграмм, которые очерчивают границы системы и роли ее участников. Из структурных и поведенческих диаграмм основной функционал нагляднее всего передает диаграмма прецедентов, где акторами выступают пользователи, а прецедентами — доступные им функции мессенджера.

Клиент и сервер взаимодействуют согласно распределенной клиент-серверной модели с внешним интерфейсом в виде REST API [2]: задачи и ресурсы разделены между сторонами, и сервер не хранит состояния прошлых обращений клиента. Внутреннюю структуру компонентов задает чистая архитектура Р. Мартина с делением на доменный слой (сущности, репозитории, сценарии использования), слой данных и слой представления [3]. На этой основе строится общая многослойная схема приложения: повторно используемые компоненты вынесены в отдельный core-модуль, а зависимости направлены строго вовнутрь через принцип инверсии зависимостей — благодаря этому слои изолированы друг от друга, а бизнес-логика не зависит от деталей хранения и интерфейса. Структуру доменного слоя дополняет диаграмма классов основных сущностей. Для каждого уровня системы подобран собственный набор инструментов: Flutter и язык Dart — для мобильного клиента [4], фреймворк WPF и язык C# — для настольного [5], ASP.NET Core, СУБД MS SQL Server, Redis для кеширования и S3-совместимое хранилище — для серверной части; в качестве вспомогательных инструментов выступают система контроля версий Git и система контроля задач Yandex Tracker.

Четвертый раздел «Проектирование программного продукта» является ключевым: в нем сосредоточена основная работа над системой. Для хранения данных на сервере выбрана реляционная СУБД Microsoft SQL Server, доступ к которой идет через ORM Entity Framework Core, а отображение настраивается средствами Fluent API. Схема данных охватывает основные сущности (User, Chat, Message, ChatUser, ChatTopic, Document и другие). Чтобы удаление не приводило к потере информации и нарушению связей, записи не стираются физически, а помечаются флагом IsDeleted, и фильтрация по нему выполняется автоматически. Типовые CRUD-операции собраны в обобщенном классе доступа к данным BaseDal, построенном по паттерну «шаблонный метод».

Самый частый запрос в мессенджере — список чатов пользователя, поэтому такие данные как превью последних сообщений и счетчики непрочитанных кешируются в Redis. Эту логику инкапсулирует класс ChatCacheService, а согласованность данных при параллельном доступе обеспечивают атомарные Lua-скрипты. Модель данных кеша формализована, и для каждой операции выведена оценка временной сложности: чтение и запись превью последнего сообщения — $O(1)$, работа с отсортированным множеством чатов — $O(\log N)$, постраничное извлечение k чатов — $O(\log N + k)$. Чтобы не писать вручную шаблонный код для каждой новой сущности, в проект встроен модуль автоматической генерации на базе библиотеки Scriban: он анализирует модели базы данных и формирует классы всех слоев архитектуры, то есть достаточно обновить модель и перезапустить генерацию.

Отдельное внимание было уделено взаимодействию по HTTP и REST API и тому, как пользователь входит в систему. Сервер распознает конкретного пользователя по JWT-токену, при этом работают сразу два токена — короткоживущий токен доступа и долгоживущий токен обновления; для них была описана механика вычисления подписи и выведены соотношения между сроками жизни, которые обеспечивают баланс между безопасностью и удобством. Пароли на сервере не хранятся в открытом виде — вместо них сохраняется результат криптографической хеш-функции Argon2 [6], устойчивой к перебору; чтобы ее вычислительная стоимость не превратилась в уязвимость, частота тяжелых операций ограничивается механизмом rate limiting. События реального времени (новые сообщения, смена статусов) доставляет модуль RTC поверх веб-сокетов с полнодуплексным обменом, а для частых операций чте-

ния, чувствительных к объему трафика, дополнительно задействована технология gRPC на базе Protobuf и HTTP/2. Сессиями пользователя управляет класс UserStatusService: он хранит состояние соединений в Redis и вычисляет агрегированный статус, который могут видеть другие участники системы.

Настольный и мобильный клиенты опираются на одну и ту же многослойную чистую архитектуру и отличаются лишь способом управления состоянием: в настольном клиенте используется паттерн MVVM (с библиотеками CommunityToolkit.Mvvm и System.Reactive), а в мобильном — BLoC. При разработке клиентов был решен ряд практических задач интерфейса. Отправленное сообщение появляется в списке сразу, еще до ответа сервера (оптимистичное обновление). При открытии чата список автоматически прокручивается к первому непрочитанному сообщению, а недостающие сообщения подгружаются пакетами растущего размера, поэтому число обращений к серверу остается логарифмическим. Данные хранятся локально в SQLite согласно стратегии «сначала кеш, затем сервер» и синхронизируются в реальном времени. Аватары пользователей и чатов обновляются по номеру версии, а токен доступа продлевается автоматически — при этом секция обновления защищена блокировкой, чтобы несколько запросов не запустили продление одновременно.

Пятый раздел «Тестирование разрабатываемого программного обеспечения» проверяет, насколько продукт соответствует заявленным требованиям. Стандарт ISO/IEC 25010 связывает качество программного обеспечения с набором характеристик — функциональной пригодностью, производительностью, надежностью, удобством использования; тестирование показывает, насколько продукт им отвечает. Виды тестирования различают по степени автоматизации (ручное и автоматизированное) и по целям (функциональное и нефункциональное). Для проверки выбраны два подхода: ручное функциональное тестирование по тест-кейсам и автоматизированное нагрузочное тестирование серверной части. Нагрузку моделирует фреймворк NBomber на платформе .NET, который позволяет задавать разные профили нагрузки и описывать сложные пользовательские сценарии. Сами сценарии охватывают полный цикл аутентификации, получение списка чатов, чтение сообщений конкретного чата, получение списка пользователей, полнотекстовый поиск и отправку сообщений через WebSocket.

Нагрузка была задана интенсивностью 3 запроса в секунду на протяже-

нии 15 секунд. Все сценарии прошли без единой ошибки, доля отказов — 0%. Быстрее всего отвечает полнотекстовый поиск (менее 40 мс), что свидетельствует об эффективности полнотекстового индекса; чтение и отправка сообщений работают примерно за 60 мс. Однако аутентификация выполняется заметно медленнее — около 100 мс, так как здесь проявляет себя тяжелая вычислительная стоимость алгоритма Argon2. Но так как сама операция выполняется лишь однажды, при входе, это не является критичным местом. В итоге серверная часть уверенно обслуживает параллельные запросы пользователей с приемлемой производительностью и без потерь данных.

Шестой раздел «Обзор программного продукта» содержит описание рабочей версии приложения. Вход осуществляется по номеру телефона с подтверждением одноразового кода верификации и облачного пароля, если он был установлен в профиле. Новые пользователи проходят регистрацию (имя, фамилия, почтовый адрес, дата рождения, пол и опциональная фотография профиля) с последующим подтверждением со стороны администратора, до получения которого использование системы недоступно. Пользователи разделяются по ролям: сотрудник, менеджер и администратор. Администратору доступна панель управления пользователями с возможностью их подтверждения и блокировки.

Главный экран состоит из трех вкладок — «Настройки», «Чаты» и «Коллеги». В настройках можно обновить профиль, настроить уведомления (как для всех чатов сразу, так и для каждого по отдельности), задать облачный пароль и выйти из системы. Вкладка «Коллеги» показывает список активных сотрудников: можно открыть их профили и написать напрямую в личную переписку. Во вкладке «Чаты» находится список личных и групповых бесед с поиском и группировкой по папкам; роли в чате совпадают с глобальными: права администратора, менеджера и сотрудника разграничены. Групповые чаты при необходимости делятся на тематические каналы. Внутри чата доступны вложения любого формата, поиск сообщений по тексту и по дате, пересылка, редактирование и удаление, а каждое сообщение наглядно отмечено статусом — *отправляется*, *не отправлено*, *отправлено* или *прочитано*. Сообщения, чаты, контакты и аватары кешируются локально, поэтому приложение остается работоспособным даже при нестабильном соединении. На данный момент реализован основной функционал мессенджера, он заложен с расчетом на дальнейшее расширение и интеграцию с другими сервисами организации.

ЗАКЛЮЧЕНИЕ

Итогом работы стала рабочая версия мессенджера для внутрикорпоративной коммуникации. Отправной точкой послужил анализ существующих решений: их недостатки подсказали, каким должен быть новый инструмент и какие требования к нему предъявить.

Такая система не сводится к одной задаче — за ней стоят анализ бизнес-требований, проектирование отказоустойчивой архитектуры, осознанный выбор технологий и поэтапная реализация компонентов. В результате получилась архитектура, которая масштабируется вертикально и горизонтально и обеспечивает стабильную работу мессенджера.

Технологический стек подбирался таким образом, чтобы клиентские приложения оставались кроссплатформенными. Это позволило сократить цикл разработки и предоставить одинаковый набор возможностей на разных операционных системах.

Разработанный прототип прошел функциональное и нагрузочное тестирование, подтвердившее его работоспособность и устойчивость к параллельным запросам.

Все поставленные задачи решены: проанализированы существующие решения, сформированы требования, обоснована архитектура, выбран технологический стек, спроектированы и реализованы настольный и мобильный клиенты вместе с серверной частью, проведено тестирование. В дальнейшем систему планируется развивать, расширяя ее возможности и интегрируя со сторонними корпоративными сервисами.

ОСНОВНЫЕ ИСТОЧНИКИ ИНФОРМАЦИИ

1. Яровая, Е. Принципы построения архитектуры программного обеспечения / Е. Яровая // E-Scio. — 2022. — № 8. — С. 1–5.
2. Лиманова, Н. Анализ эффективности клиент-серверной архитектуры / Н. Лиманова, И. Селезнев // Бюллетень науки и практики. — 2022. — № 7. — С. 392–395.
3. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — Санкт-Петербург: Питер, 2024. — 352 с.
4. Чернышев, С. А. Основы Flutter / С. А. Чернышев, Ю. М. Петров, С. П. Ильин, П. А. Гершевич. — Санкт-Петербург: Питер, 2026. — 688 с.

5. Троелсен, Э. Язык программирования C# 9 и платформа .NET 5: основные принципы и практики программирования / Э. Троелсен, Ф. Джепикс. — Санкт-Петербург: Диалектика, 2022. — 1392 с.
6. Игнатьев, Е. Защита информации: криптоалгоритмы хеширования: учебное пособие для вузов / Е. Игнатьев. — Санкт-Петербург: Лань, 2024. — 264 с.