ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»
ПО ПРОФИЛЮ «ИНЖЕНЕРНОЕ ДЕЛО»

регистрационный номер

Секция: ***Информационная безопасность и цифровая криминалистика (ИУ8)***

название секции

МЕССЕНДЖЕР — NAKI

название работы

Автор: ***Чугреев Никита Сергеевич***

фамилия, имя, отчество

ГБОУ Инженерная школа № 1581

наименование учебного заведения, класс

Научный руководитель: ***Ковынёв Николай Витальевич***

фамилия, имя, отчество

МГТУ им. Н.Э. Баумана

место работы

Старший преподаватель

звание, должность

_____

подпись научного руководителя

Москва – 2026

Аннотация

Название: Приложение для общения — Naki.

Цель работы: создать безопасный мессенджер, в котором сервер не будет видеть зашифрованных сообщений от пользователя, а в базе данных все сообщения будут храниться полностью зашифрованными.

Методы и приёмы: Клиентская часть реализована на HTML, CSS и JavaScript с использованием Web Crypto API. Серверная часть разработана на Python с фреймворком Eel для интеграции веб-интерфейса. База данных - MongoDB для хранения зашифрованных сообщений и пользовательских данных. Система безопасности реализует гибридную архитектуру с использованием ECDH для обмена ключами и AES-GCM для сквозного шифрования сообщений.

Результаты: разработано функционирующее приложение мессенджера, реализующее архитектуру Zero-Knowledge Encryption. Все текстовые сообщения проходят сквозное шифрование на стороне клиента с использованием AES-GCM, что исключает возможность доступа к данным со стороны сервера. Реализована система безопасной аутентификации, управления контактами и обмена зашифрованными сообщениями. Приложение демонстрирует стабильную работу.

Перспективы: планируется полная реализация сквозного шифрования по умолчанию для всех пользователей, интеграция ECDH для автоматического обмена ключами, создание групповых чатов с шифрованием. Также планируется глубокая персонализация интерфейса и функций. В приоритете стоит создание адаптивного приложения для macOS и всех популярных операционных систем, включая мобильные платформы, чтобы обеспечить пользователям максимальную доступность и комфорт работы с мессенджером на любых устройствах.

Содержание

Введение 4
Основная часть 5
1. Анализ конкурентов 5
2. Разработка системы аутентификации 6
3. Разработка графического интерфейса 7
4. Выбор и подключение базы данных 7
5. Разработка системы шифрования и отправки сообщений 9
5.1 Генерация ключевых пар 10
5.2 Шифрование приватного ключа 10
5.3 Создание общего секрета 11
5.4 Шифрование на стороне клиента 12
5.5 Передача зашифрованных сообщений на сервер 12
5.6 Получение и дешифрование сообщения 13
6 Система синхронизация времени 14
7 Система отслеживания активности 14
8 Система ответов на сообщения 14
9 Система последнего сообщения 15
10 Система статусов прочтения сообщений 16
11 Система удаления сообщения 16
12 Система обновления сообщений в реальном времени 17
13 Измерение производительности 17
Заключение 19
Используемые библиотеки 20
Список литературы 23
Приложение А. Файлы проекта 25
Приложение Б. Вид пользовательского интерфейса 26

Введение

В современном цифровом мире мессенджеры стали неотъемлемой частью повседневной коммуникации. Но рост их популярности сопровождается увеличением рисков, связанных с конфиденциальностью и безопасностью личных данных. Пользователи ежедневно доверяют мессенджерам огромные объемы важной информации — от личных переписок и фотографий до конфиденциальных данных. Существующие решения часто предлагают либо полное сквозное шифрование как платную опцию, либо подразумевают полный доступ разработчика приложения к сообщениям. В этих условиях необходим сбалансированный подход, обеспечивающий как высокую безопасность, так и удобство использования. Представленный в данной работе мессенджер Naki предлагает инновационное гибридное решение данной проблемы. Ключевой целью проекта является реализация системы безопасности на основе Zero-Knowledge[1)] аутентификации и клиентского шифрования с использованием алгоритмов AES-GCM. Архитектура приложения включает разработку эффективного механизма хранения зашифрованных данных в MongoDB для работы с сообщениями. Параллельно ставится задача создания адаптивного пользовательского интерфейса с поддержкой отправки текстовых сообщений и системы ответов. Важным аспектом является обеспечение стабильного клиент-серверного взаимодействия через фреймворк Eel для совместимости всех компонентов. Завершающим этапом работы станет всестороннее тестирование и оценка устойчивости системы к потенциальным угрозам.

[1)] Zero-Knowledge — криптографический протокол, позволяющей стороне подтвердить истинность какого-либо утверждения другой стороне, не раскрывая дополнительной информации о самом утверждении.

Основная часть

Работа выполнялась в рамках подготовки проекта для конференции школьников «Шаг в будущее» в 2025–2026 учебном году. Основная часть разработки велась в домашних условиях с использованием компьютера.

1. Анализ конкурентов

Данный этап проходил с июня по июль 2025 года. Первый мессенджер появился в 1996 году. За это время множество функций были доведены до совершенства. В наше время существует всего 3 действительно больших компаний, которые могут предоставить хорошие услуги по общению с другим человеком. Это Max, Telegram и WhatsApp. В ходе анализа были выделены основные пункты, связанные с системой шифрования у этих мессенджеров. Результаты представлены в Таблица 1 — Анализ конкурентов. Они позволяют наглядно оценить конкурентоспособность разрабатываемого приложения.

Таблица 1 — Анализ конкурентов

Параметр	Max	Telegram	WhatsApp
Тип шифрования	TLS[2)]	End-to-End Encryption[3)] только в секретных чатах	End-to-End Encryption
Шифрование данных	TLS	MTProto[4)] (собственный)	Signal Protocol
Шифрование по умолчанию	Чат хранится на сервере	Нет (только в секретных чатах)	Да

[2)] TLS (Transport Layer Security) — криптографический протокол, который защищает соединение между пользователем и сервером. Сервер видит расшифрованные сообщения.
[3)] End-to-End Encryption — сквозное шифрование, при котором информация шифруется на стороне отправителя и расшифровывается на стороне получателя. Передает информацию от отправителя к получателю.
[4)] MTProto (Mobile Telegram Protocol) — тоже самое, что и TLS, только разработан Telegram.

Ключи шифрования	На сервере	На устройстве пользователя и на сервере	На сервере
Доступ к данным	Сервер имеет полный доступ	Сервер имеет доступ к данным	Сервер имеет доступ к мета данным

2. Разработка системы аутентификации

Этап проходил с июля по август 2025 года. Процесс аутентификации включает валидацию введенных данных на клиентской стороне с последующей проверкой на сервере. При входе в систему введенный пароль хешируется алгоритмом SHA-256[5)] и сравнивается с хешем[6)], хранящимся в базе данных. При совпадении хешей осуществляется успешная аутентификация пользователя. На Рисунок 1 — Схема работы регистрации нового пользователя представлен алгоритм регистрации нового пользователя. Вся работа с хешами была сделана благодаря библиотеки Hashlib, которая позволяет преобразовывать пароли в хеши, а также хешировать email. Email-адреса пользователей также защищаются путем хеширования с использованием SHA-256 перед сохранением в базе данных. Такой подход обеспечивает конфиденциальность персональных данных пользователей даже в случае компрометации базы данных. Обмен данными между клиентской и серверной частями приложения осуществляется через фреймворк Eel[7)], обеспечивающий интеграцию Python с JavaScript.

[5)] SHA-256 (Secure Hash Algorithm) — это криптографическая функция, которая преобразует любые данные в хеш длинной 256 бит.

[6)] Хеш — уникальная строка фиксированной длины.

[7)] Фреймворк Eel — это Python библиотека, позволяющая создавать веб-приложения с использованием python.

Рисунок 1 — Схема работы регистрации нового пользователя

3. Разработка графического интерфейса

Данный этап создавался попутно с системой аутентификации. Работа началась с июля 2025 года и дополнялся на протяжении всего проекта. Графический интерфейс разделен на 2 блока. Первый блок — это меню для авторизации, а второй блок — это главный экран мессенджера, а также сами чаты. Для разработки графического интерфейса использовались HTML5 для разметки и структуры, CSS3 для дизайна и анимации, JavaScript ES6+ для логики и обработки событий. В Приложение Б**.** показан весь интерфейс приложения.

4. Выбор и подключение базы данных

Этап был реализован в конце июля 2025 года. MongoDB использовалась в проекте как основная база данных. Она подключалась через библиотеку pymongo, так как ее архитектура идеально подходит для мессенджеров с

быстрым хранением и обработкой сообщений. Пример производительности базы данных изображен на

Рисунок 2 — Производительность базы данных MongoDB. В отличие от классических баз данных, MongoDB сохраняет данные в виде коллекций документов, а не таблиц, что делает структуру гибкой, расширяемой и особенно удобной для динамических приложений. Пример того, как выглядит структура хранения данных о пользователе представлена на Рисунок 3 — Вид хранения данных о пользователей. База данных может с легкостью обрабатывать многочисленный поток новых сообщений.

Главные плюсы MongoDB:

1. Гибкая схема: не требует четко определенных структур (таблиц и типов), удобно хранить разные типы данных в одном месте.
2. Высокая производительность: оптимизирована для больших объемов информации и быстрых операций чтения-записи.
3. Простота интеграции: хорошо работает с Python, JavaScript, современными веб-фреймворками, поддерживает множество индексов.
4. Быстрая обработка операций чтения и записи, что важно для реального времени и современной передачи сообщений.

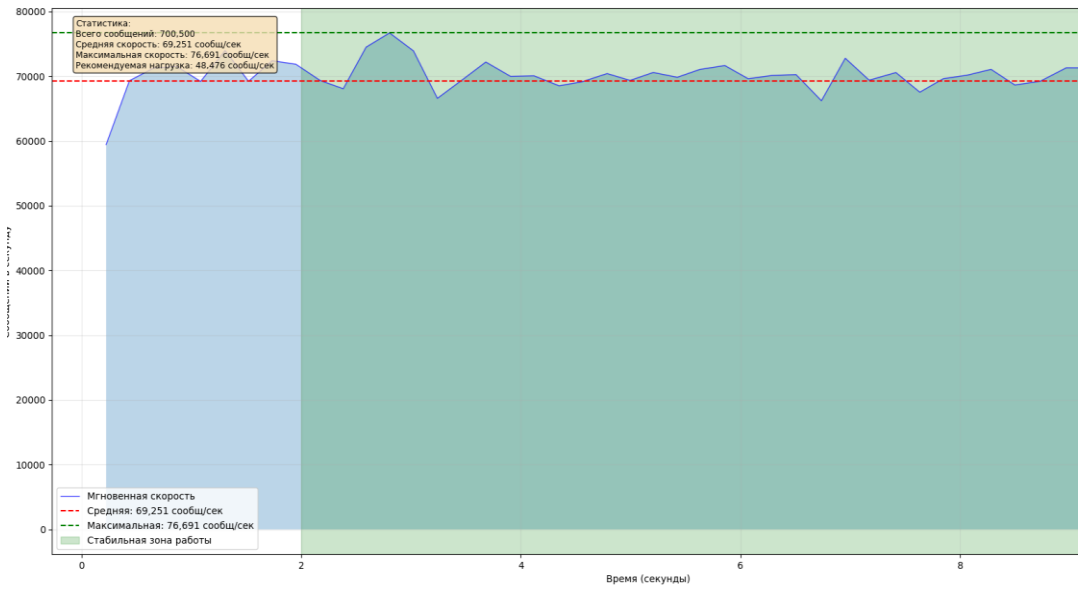

Рисунок 2 — Производительность базы данных MongoDB

Рисунок 3 — Вид хранения данных о пользователей

5. Разработка системы шифрования и отправки сообщений

Данный этап разрабатывался с августа по декабрь 2025 года. В проекте была реализована криптографическая система на основе взаимодействия двух алгоритмов: ECDH и AES-GCM. ECDH позволяет устанавливать общий секретный ключ по незащищенному каналу связи. Данный протокол обеспечивает безопасный обмен ключами даже при наличии пассивного прослушивания канала. В Таблица 2 — Сравнение алгоритма ECDH с аналогами показано сравнение аналогов ECDH. Для полного создания этого алгоритма у каждого пользователя должен быть свой публичный ключ и приватный. Публичный ключ будет виден в базе данных и хранится в не зашифрованном виде, а приватный ключ будет тоже хранится в базе данных, но будет зашифрован.

Таблица 2 — Сравнение алгоритма ECDH с аналогами

Параметр	ECDH	DH	X3DH
Стойкость	256-bit EC[8)]	2048-bit требуется	256-bit EC
Производительность	Очень высокая	Средняя	Высокая
Размер ключа	256 бит	2048+ бит	256 бит
Forward Secrecy	Встроенная	Да	Да

[8)] EC (Elliptic Curve) — обозначает, что протокол использует криптографию на эллиптических кривых

5.1 Генерация ключевых пар

Каждый пользователь системы обладает парой криптографических ключей: приватным и публичным. Приватный ключ представляет собой криптографически стойкое случайное число размерностью 256 бит, генерируемое в пространстве от 1 до n-1, где n — порядок базовой точки эллиптической кривой secp256r1 (приблизительно 2^{256}). Выбор такого диапазона обеспечивает равномерное распределение ключей и максимальную энтропию. Публичный ключ вычисляется:

$$Q = d \times G, \quad (1)$$

где d — скалярное значение (приватный ключ);

G — генераторная точка кривой, являющаяся фиксированным параметром системы.

Точка G, которая берется из формулы *(1)***,** является точкой эллиптической кривой. Криптографическая стойкость системы основывается на вычислительной сложности задачи дискретного логарифмирования в группе точек эллиптической кривой (ECDLP). Это позволяет оставлять публичный ключ открытым, так как раскрытие приватного ключа через публичный невозможна.

5.2 Шифрование приватного ключа

В случае компрометации базы данных, при открытых приватных ключах можно с легкостью расшифровать любое сообщение. Поэтому приватные ключи нужно хранить в зашифрованном виде. Приватный ключ будет шифроваться на клиенте и передаваться на сервер. Как только пользователь создает аккаунт, то у него автоматически генерируется безопасная соль длиной в 16 байт, которая будет обеспечивать уникальность производных ключей даже при одинаковых паролях. После этого происходит объединение пароля пользователя с солью и с помощью функции PBKDF2 происходит 100,000 итераций с применением

HMAC-SHA-256[9], что значительно уменьшает риск от перебора паролей. Генерируемый пароль имеет длину 32 байта. Полученный производный ключ используется в криптографической схеме Fernet, которая представляет собой симметричное аутентифицированное шифрование. На выходе получаем зашифрованный приватный ключ. На Рисунок 4 — Вид приватных и публичными ключей в базе данных представлен пример хранения приватного и публичного ключа в базе данных.

Рисунок 4 — Вид приватных и публичными ключей в базе данных

5.3 Создание общего секрета

При установлении защищенного канала связи между двумя пользователями происходит обмен публичными ключами через сервер. Каждая сторона независимо вычисляет общий секрет путем скалярного умножения собственного приватного ключа на публичный ключ собеседника. Полученная общая точка кривой подвергается дальнейшей обработке с применением функции формирования ключа HKDF-SHA256, так как длинна общего секрета не фиксированная. Данная функция делает ключ длинной в 32 байта. Это позволяет получить симметричный ключ для алгоритма AES-GCM, используемого для сквозного шифрования сообщений. Данный способ гарантирует, что при перехвате трафика или

[9] HMAC-SHA-256 — алгоритм, который обеспечивает целостность данных, используя секретный ключ для генерации уникального хеша.

взломе базы данных, все сообщения невозможно расшифровать и злоумышленник увидит просто набор разных букв с цифрами.

5.4 Шифрование на стороне клиента

Перед отправкой каждое текстовое сообщение проходит локальное шифрование на устройстве пользователя. Для этого был выбран алгоритм AES-256-GCM. В Таблица 3 — Сравнение алгоритмов шифрования представлено сравнение алгоритмов шифрования. После анализа было принято решение выбрать AES-256-GCM для шифрования сообщений.

Таблица 3 — Сравнение алгоритмов шифрования

Характеристика	AES-256-GCM	ChaCha20-Poly1305	AES-SIV (RFC 5297)
Тип шифра	Блочный (128 бит блок)	Потоковый	Блочный
Квантовая стойкость	Высокая	Высокая	Высокая
Пропускная способность	Высокая	Средняя	Низкая
Сфера применения	Гос. Стандарты, VPN[10)]	Телефоны, интернет	Финансы

5.5 Передача зашифрованных сообщений на сервер

Зашифрованное сообщение передается на сервер через фреймворк Eel, который обеспечивает взаимодействие между JavaScript-кодом интерфейса и Python-кодом серверной части:

1. Структура передачи данных: Фронтенд[11)] вызывает функцию, передавая идентификатор отправителя, идентификатор получателя, зашифрованный текст и, при необходимости, идентификатор сообщения, на которое дается ответ.

[10)] VPN (Virtual Private Network) — технология, создающее зашифрованное соединение между устройством пользователя и сетью/интернетом.
[11)] Фронтенд — пользовательский интерфейс.

2. Обработка на сервере: Сервер принимает данные и сохраняет их в базе MongoDB без какой-либо попытки дешифрования. В структуру сообщения добавляются метаданные: метка времени отправки, статус прочтения и флаг is_encrypted, указывающий на то, что содержимое зашифровано.

3. Хранение в базе данных: Зашифрованные сообщения сохраняются в коллекции messages с соответствующей структурой, показанной на Рисунок 5 — Вид зашифрованных сообщений, хранящихся в базе данных.

Рисунок 5 — Вид зашифрованных сообщений, хранящихся в базе данных

5.6 Получение и дешифрование сообщения

Когда получатель открывает чат, происходит обратный процесс:

1. Загрузка зашифрованных сообщений: Клиентская часть получателя запрашивает историю чата через функцию, которая возвращает все сообщения в зашифрованном виде.

2. Локальное дешифрование: Используя свой пароль и сохраненную соль, клиентская часть создает ключ и последовательно дешифрует каждое сообщение с помощью Web Crypto API[12)].

3. Отображение в интерфейсе: Расшифрованные сообщения отображаются в интерфейсе чата с указанием времени отправки и статуса прочтения.

[12)] Web Crypto API — стандартизированный набор правил, функций для выполнения криптографических операций непосредственно в браузере, без использования сторонних библиотек.

6 Система синхронизация времени

Этап был начат и завершен в конце декабря 2025 года. Когда пользователь отправляет сообщение, то время фиксируется на сервере в момент сохранения сообщения в базу данных. Этот способ помогает предотвращать проблемы со временем у пользователей. Само время сохраняется в формате UTC[13)]. Когда пользователь заходит вновь в приложение, то функция автоматически отображает текущее серверное время в интерфейсе. Если пользователь получает сообщение от сервера, то время автоматически переводится браузером в локальный часовой пояс пользователя. Также были добавлены плашки, разделяющие чат по дням.

7 Система отслеживания активности

Этап создавался одновременно с системой синхронизации времени. Когда пользователь входит в свой аккаунт, он отправляет сигнал на сервер о своем присутствие. Сервер принимает этот сигнал и сохраняет время в базу данных. После этого введется отсчёт сессии активность. Каждые тридцать секунд клиент автоматически отправляет на сервер фоновый запрос на обновление активности. Как только сервер принимает запрос, то изменяется last_online в базе данных. В этот момент у других пользователей, данный человек подсвечивается в зеленном кругляшке, а также при открытии чата в верхнем правом углу будет писаться, что человек в сети. В этом помогает клиент, который сравнивает last_online с нынешним временем. Если прошло больше пяти минут с последней активности пользователя, то отметка ‘Человек в сети’ пропадает.

8 Система ответов на сообщения

Этап разрабатывался одновременно с системой шифрования. Работа завершилась в конце декабря 2025 года. Система ответов позволяет людям быстро цитировать предыдущие сообщения, тем самым улучшая общение в мессенджере. При нажатии на сообщение, содержащее ответ, может отправлять

[13)] UTC (Coordinated Universal Time) — всемирно координированное время

пользователя к тому сообщению, на которое он отвечал. Теперь у каждого сообщения, хранящегося в базе данных, будет содержать пункт: reply_to_message_id. В нем будет написан id[14)] сообщения, на который пользователь отвечал. Если сообщение не является ответом, то этого пункта не будет. На Рисунок 6 — Вид отправленного ответа на сообщение приведен вид отправленного ответа на сообщение.

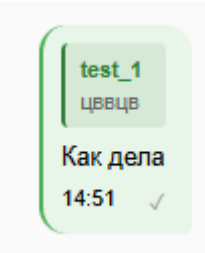

Рисунок 6 — Вид отправленного ответа на сообщение

9 Система последнего сообщения

Этап разрабатывался одновременно с системой шифрования. Работа завершилась в конце декабря 2025 года. В мессенджере реализована система последнего сообщения. Она позволяет видеть ответил ли вам собеседник или нет, а также замечать контекст общения без входа в чат. Как только вы входите в приложение, то на сервер отправляется запрос: определить последнее сообщение. Сервер начинает поиск последних сообщений в базе данных. Как только это удается сделать, то сервер для каждого пользователя отправляет последнее сообщение. Как только все последние сообщения пришли, то приложение дешифрует их автоматически. Последним этапом будет отображение их рядом с карточками пользователей. Если писал пользователь, то рядом с сообщением будет писаться “Вы”. Если же писали вам, то будет текст самого сообщения. Вид отображения последнего сообщения представлена на Рисунок 7 — Вид отображения последнего сообщения.

[14)] Id — это уникальный код, номер объекта.

Рисунок 7 — Вид отображения последнего сообщения

10 Система статусов прочтения сообщений

Этап продлился две недели и закончился к концу января 2026 года. Система отслеживания статусов прочтения позволяет пользователям видеть, прочитаны ли их сообщения собеседником. Как только пользователь отправляет сообщение, то на стороне отправителя сразу отображается одна галочка. Само сообщение сохраняется в базу данных с отдельным полем read, в котором пишется false. Когда получатель сообщения заходит в чат с отправителем, то запускается функция, которая находит все непрочитанные сообщения из базы данных от текущего собеседника и помечает их в базе данных вместо false на true. Как только получатель прочитал сообщение, то статус должен обновиться у отправителя. У него запускается интервальная проверка каждые 2 секунды. Фронтенд собирает id своих сообщений в этом чате и отправляет запрос на сервер для проверки их статуса прочтения. После этого сервер проверяет статус каждого запрошенного сообщения и возвращает информацию. Фронтенд получает ответ от сервера и меняет визуальное отображение. На Рисунок 8 — Вид отображения последнего сообщения показано, как выглядит прочитанное сообщение.

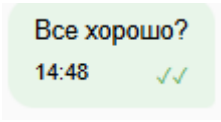

Рисунок 8 — Вид отображения последнего сообщения

11 Система удаления сообщения

Этап продлился несколько дней и закончился в конце января 2026 года. Когда человек нажимает правой клавишей мыши по сообщению и выбирает пункт ‘Удалить сообщение’, то срабатывается обработчик удаления сообщения. Он определяет уникальный идентификатор(id), а также идентификатор

пользователя. После этого запрашивается подтверждение у человека. Если тот согласился, то формулируется запрос к серверу. Этот запрос содержит идентификатор сообщения, который нужно удалить. Как только сервер получил запрос, он находит сообщение по идентификатору в базе данных, а также проверяет является ли текущий пользователь отправителем сообщения. Затем сообщение удаляется у обоих пользователей. Как только это происходит, появляется еще один запрос в базу данных. Сервер ищет последнее сообщение. Далее сервер формирует ответ для отправителя, в котором пишется, что сообщение успешно удалено. Параллельно с этим сервер подает запрос собеседнику на удаление сообщения. После этого меняется последнее сообщение в карточке пользователя.

12 Система обновления сообщений в реальном времени

Этот этап занял половину февраля 2026 года. Когда пользователь в сети и находится в чате с пользователем, то у него в фоновом режиме работает механизм опроса сервера. Раз в две секунды браузер отправляет запрос с вопросом, нет ли новых сообщений в текущем открытом чате. В этом запросе передается идентификатор собеседника и идентификатор последнего сообщения, который есть у пользователя. Сервер, получив запрос, обращается к базе данных и ищет все сообщения между двумя пользователями. Если новые сообщения найдены, то сервер возвращает все новые сообщения сразу. Когда клиент получает список с данными, то он начинает их проверять на дубликаты, затем расшифровывает и отображает у пользователя в чате.

13 Измерение производительности

Этот этап занял весь январь 2026 года. При завершении работы над проектом была проведена зависимость скорости шифрования и дешифрования от длины сообщений. Результат приведен на Рисунок 9 . Также проводилось множество тестов, связанных с безопасностью.

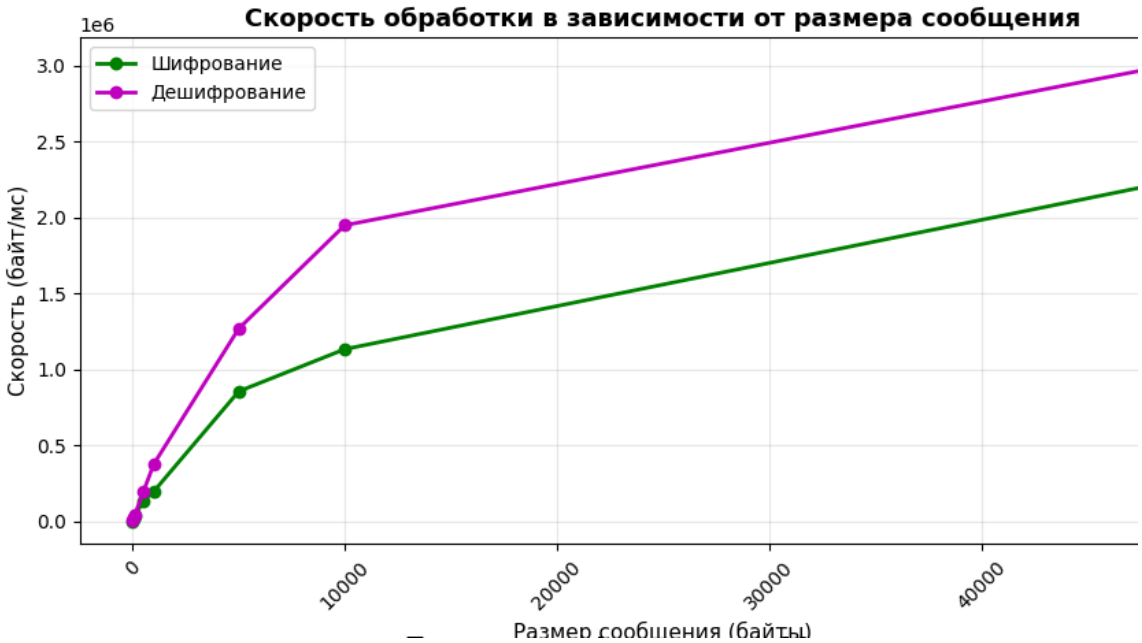

Рисунок 9 — График зависимости скорости шифрования/дешифрования от размера сообщения

Заключение

В ходе выполнения работы автором создано полностью функционирующее приложение-мессенджер Naki, обеспечивающее защищённое общение пользователей. Ключевым достижением является реализация архитектуры, при которой сервер не имеет доступа к содержанию сообщений, поскольку их шифрование осуществляется исключительно на стороне клиента. Даже в случае компрометации базы данных информация остаётся защищённой — злоумышленник получит лишь зашифрованные данные.

Разработано приложение с современным интерфейсом, включающее все основные функции мессенджера: регистрацию пользователей, систему друзей, обмен текстовыми сообщениями с поддержкой ответов, отображение статусов онлайн и истории переписки.

Практическая значимость работы заключается в создании доступного инструмента для безопасного общения, уровень защиты которого превышает аналогичные показатели многих популярных мессенджеров за счёт обязательного сквозного шифрования для всех пользователей.

Перспективы развития проекта включают внедрение групповых чатов для коллективного общения, реализацию видеозвонков для индивидуальных и групповых бесед, создание мобильных версий для iOS и Android с публикацией в официальных магазинах приложений. Также планируется обеспечить кроссплатформенность для работы на различных операционных системах и добавить функции обмена медиафайлами при сохранении высокого уровня безопасности.

Используемые библиотеки

Таблица 4 — используемые библиотеки python (серверная часть)

Библиотека	Версия	Назначение в проекте
Ell	0.14.0	Создание гибридного приложения с веб-интерфейсом
pymongo	4.5.0	Работа с базой данных MongoDB
cryptography	41.0.7	Криптографические операции. Шифрование, дешифрование. Управление публичным/приватным ключем.
hashlib	Встроенная	Хеширование данных. Использование для хеширования пароля и почты при входе
logging	Встроенная	Запись ошибок, ложных реагирований
datatime	Встроенная	Работа с датой и временем. Фиксация и управление временными метками
os	Встроенная	Генерация случайных значений. Управление солью для шифрования

json	Встроенная	Преобразование данных из одной формы в другую, чтобы сохранить или передать их.
secrets	Встроенная	Генерация разных криптографически безопасных чисел. Создание случайных чисел для ключа шифрования
base64	Встроенная	Кодирование/декодирование ключей и сообщений
pathlib	Встроенная	Управление путями к файлам проекта

Таблица 5 — используемые библиотеки JavaScript/Web API (клиентская часть)

Библиотека/API	Тип	Назначение в проекте
Web Crypto API	Строенный API	PBKDF2, AES-GCM, шифрование/дешифрование на клиенте
localStorage	Web Storage API	Постоянное хранение данных (настройки пользователя)
sessionStorage	Web Storage API	Временное хранение пароля для дешифрования сообщений во время сессии
Fetch API	API	Вызов python функций. Разные запросы

MutationObserver	Web API	Отслеживание изменений в чате для обновления скроллбара

Список литературы

1. Qt: Cross-platform software development framework [Электронный ресурс] // Qt: [сайт]. - URL: https://doc.qt.io/ (дата обращения: 10.01.2025).
2. Создание мессенджера с нуля: как это сделать [Электронный ресурс] // Purrweb: [сайт]. – URL: https://www.purrweb.com/ru/blog/kak-sozdat-svoj-messendzher-otvety-na-populyarnye-voprosy/ (дата обращения: 03.06.2025).
3. Как создать свой мессенджер: ответы на популярные вопросы [Электронный ресурс] // Purrweb: [сайт]. – URL: https://www.purrweb.com/ru/blog/kak-sozdat-svoj-messendzher-otvety-na-populyarnye-voprosy/ (дата обращения: 03.06.2025).
4. Защищённый мессенджер на Python: реализация E2EE шифрования [Электронный ресурс] // Habr: [сайт]. – URL: https://habr.com/ru/articles/701488/ (дата обращения: 03.06.2025).
5. DeepSeek AI assistant [Электронный ресурс] // DeepSeek: [сайт]. – URL: https://www.deepseek.com/ (дата обращения: 03.06.2025).
6. Google Gemini AI [Электронный ресурс] // Google AI: [сайт]. – URL: https://gemini.google.com/ (дата обращения: 03.06.2025).
7. Web Cryptography API [Электронный ресурс] // W3C: [сайт]. – URL: https://www.w3.org/TR/WebCryptoAPI/ (дата обращения: 11.06.2025).
8. MongoDB Documentation [Электронный ресурс] // MongoDB: [сайт]. – URL: https://www.mongodb.com/docs/ (дата обращения: 11.06.2025).
9. Python Documentation [Электронный ресурс] // Python Software Foundation: [сайт]. – URL: https://docs.python.org/3/ (дата обращения: 11.06.2025).
10. Eel library documentation [Электронный ресурс] // GitHub: [сайт]. – URL: https://github.com/ChrisKnott/Eel (дата обращения: 12.06.2025).
11. PyMongo documentation [Электронный ресурс] // Read the Docs: [сайт]. – URL: https://pymongo.readthedocs.io/ (дата обращения: 12.06.2025).
12. Рекомендации по выбору криптографических алгоритмов [Электронный ресурс] // NIST Special Publication 800-57 Part 1, Revision 5. –

URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf (дата обращения: 01.08.2025).

13. Cryptography library documentation [Электронный ресурс] // PyPI: [сайт]. – URL: https://pypi.org/project/cryptography/ (дата обращения: 01.08.2025).
14. AES (Advanced Encryption Standard) [Электронный ресурс] // NIST FIPS PUB 197. – URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf (дата обращения: 01.08.2025).
15. PBKDF2 (Password-Based Key Derivation Function 2) [Электронный ресурс] // RFC 8018. – URL: https://www.rfc-editor.org/rfc/rfc8018 (дата обращения: 02.08.2025).
16. Zero-Knowledge Proofs [Электронный ресурс] // Signal Documentation: [сайт]. – URL: https://signal.org/docs/ (дата обращения: 02.08.2025).
17. MDN Web Docs: Веб-технологии для разработчиков [Электронный ресурс] // MDN Web Docs: [сайт]. – URL: https://developer.mozilla.org/ru/ (дата обращения: 11.09.2025).
18. ISO 8601:2004. Representation of dates and times [Электронный ресурс] // International Organization for Standardization: [сайт]. – URL: https://www.iso.org/standard/40874.html (дата обращения: 22.10.2025).

Приложение А. Файлы проекта

Файлы проекта: link

Приложение Б. Вид пользовательского интерфейса

Рисунок 10 — Вид окна для входа в аккаунт

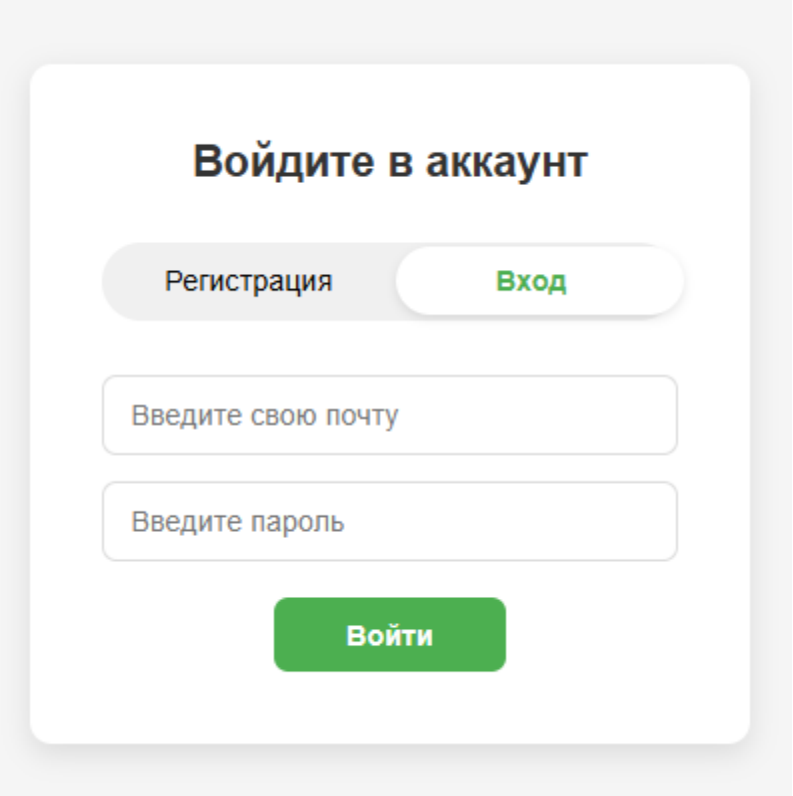

Рисунок 11 — Вид окна для регистрации

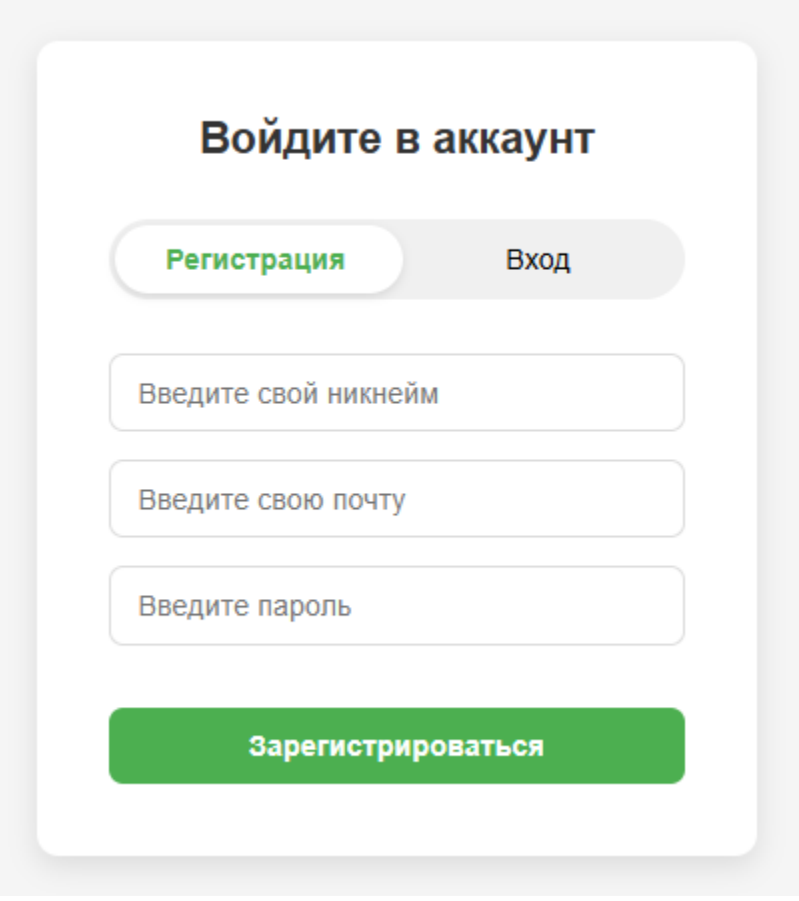

Рисунок 12 — Вид главного экрана после входа

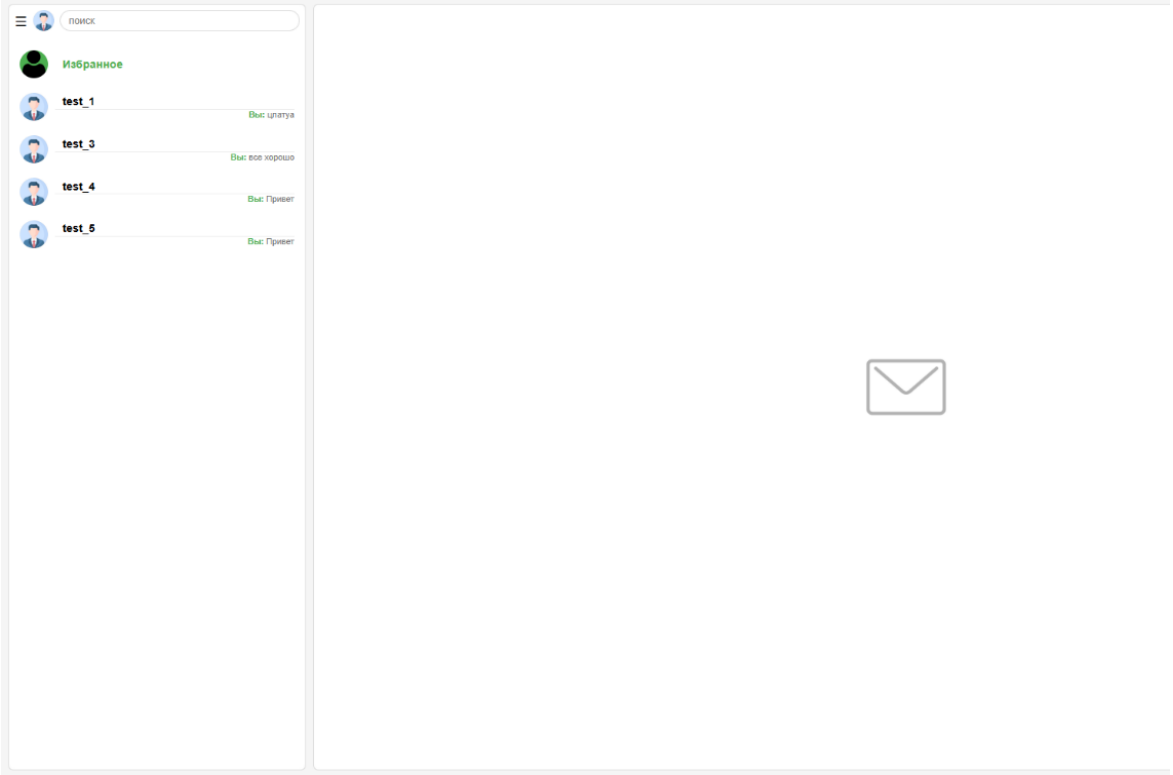

Рисунок 13 — Вид диалогового окна с пользователем