Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Базовая кафедра «Интеллектуальные системы управления»

**КУРСОВАЯ РАБОТЕ**

создание приложения для переадресации сообщений в мессенджерах  
тема

Руководитель \_\_\_\_\_\_\_\_ А.А. Даничев

подпись, дата

Студент КИ23-03Б, 032326992 \_\_\_\_\_\_\_\_ М.С. Бекетов

номер группы, зачетной книжки подпись, дата

Студент КИ23-03Б, 032322121 \_\_\_\_\_\_\_\_ Д.А. Зверьков

номер группы, зачетной книжки подпись, дата

Студент КИ23-03Б, 032322222 \_\_\_\_\_\_\_\_ Е.А. Баннов

номер группы, зачетной книжки подпись, дата

Студент КИ23-03Б, 032320725 \_\_\_\_\_\_\_\_ М.А. Дедусь

номер группы, зачетной книжки подпись, дата

Красноярск 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc200143761)

[1 Выбор и обоснование выбора структуры приложения 4](#_Toc200143762)

[2 Создание серверного приложения 6](#_Toc200143763)

[2.1 Дублирование сообщений в любимый мессенджер получателя 6](#_Toc200143764)

[2.2 Обработка мультимедийных файлов (фото, голосовые) 7](#_Toc200143765)

[2.3 Создание логера ошибок 8](#_Toc200143766)

[2.4 Вывод 8](#_Toc200143767)

[3 Создание сайта 9](#_Toc200143768)

[3.1 Создание общих объектов для страниц 9](#_Toc200143769)

[3.2 Написание основных разделов для сайта 10](#_Toc200143770)

[3.3 Создание стилей для сайта 12](#_Toc200143771)

[3.4 Вывод 12](#_Toc200143772)

[4 Шифрование данных пользователя 14](#_Toc200143773)

[4.1 Написание функций для шифрования и дешифрования 14](#_Toc200143774)

[5 Реализация аутентификации и системы безопасности 16](#_Toc200143775)

[5.1 Хранение и проверка паролей 16](#_Toc200143776)

[5.2 Генерация JWT и OAuth2-аутентификация 16](#_Toc200143777)

[5.3 Безопасность токенов VK 16](#_Toc200143778)

[5.4 Управление настройками через веб-интерфейс 17](#_Toc200143779)

[5.5 Вывод 17](#_Toc200143780)

[ЗАКЛЮЧЕНИЕ 18](#_Toc200143781)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc200143782)

[ПРИЛОЖЕНИЕ А 20](#_Toc200143783)

[ПРИЛОЖЕНИЕ Б 25](#_Toc200143784)

# ВВЕДЕНИЕ

В современном мире существует огромное множество мессенджеров, и каждый человек использует не тот мессенджер, который ему удобен, а тот что популярный среди его окружения. Мы выбираем Telegram, потому что там сидят друзья, WhatsApp — потому что на нём настаивает семья, а Viber — потому что его до сих пор используют коллеги. Даже если интерфейс неудобен, а функционал ограничен, мы миримся с этим ради возможности оставаться на связи. Так технологии, призванные объединять, становятся ещё одним поводом для раздражения: бесконечные уведомления из десятка приложений, путаница в чатах и вечная необходимость проверять несколько мессенджеров одновременно. Но выбора нет — ведь цифровая коммуникация давно перестала быть вопросом личного удобства, а превратилась в социальное обязательство. Но почему бы это не исправить? [1]

Цель работы: создать приложения для переадресации сообщений в мессенджерах.

Задачи:

- выбрать и обосновать выбор структуры приложения;

- создать серверное приложения для;

- создать сайта;

- создать базу данных;

- обеспечить надежное шифрование данных пользователей.

# 

# Выбор и обоснование выбора структуры приложения

Пользователь нашего приложения должен писать в свой любимый мессенджер, а другой пользователь должен получать его в уже своем любимом мессенджере. Желательно чтобы юзер не устанавливал на устройство сторонних программ, так как это существенно упростит поддержку кроссплатформенности продукта. Приложение должно быть реализовано на сервере что позволит обеспечивать его бесперебойную работу. Схема работы приложения представлена на рисунке 1.

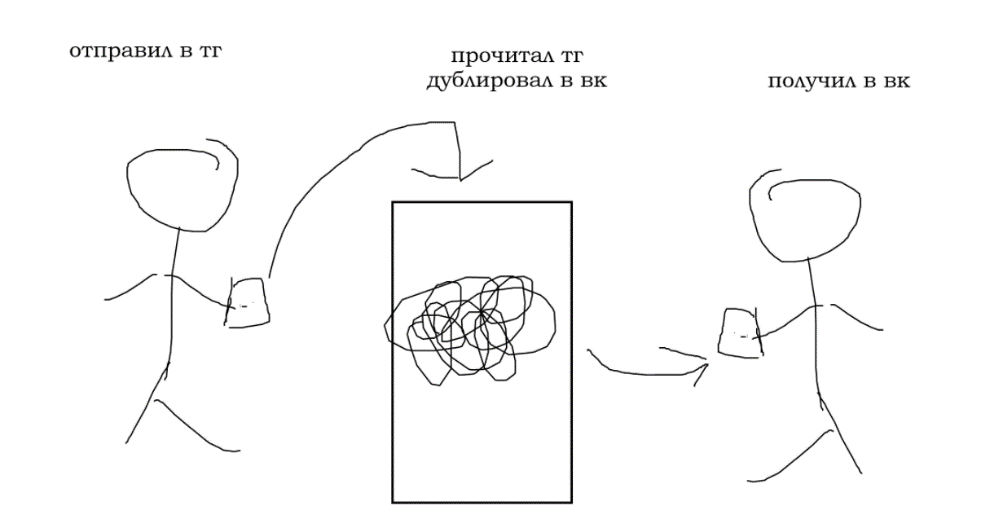


Рисунок 1 ‒ Cхема работы приложения

Для достижения данного результата можно использовать userbot-ов. Их поддерживают большинство современных мессенджеров, и они позволяют писать сообщения от лица человека, не прибегая к чрезмерному усложнению.

Для хранения токенов и данных пользователей необходимо создать 2 соединённых БД. Хранение информации будет производиться в MySQL, так как это СУБД с высокой надёжностью, производительностью и поддержкой современных технологий. Примерная структура приведена ниже на рисунке 2.

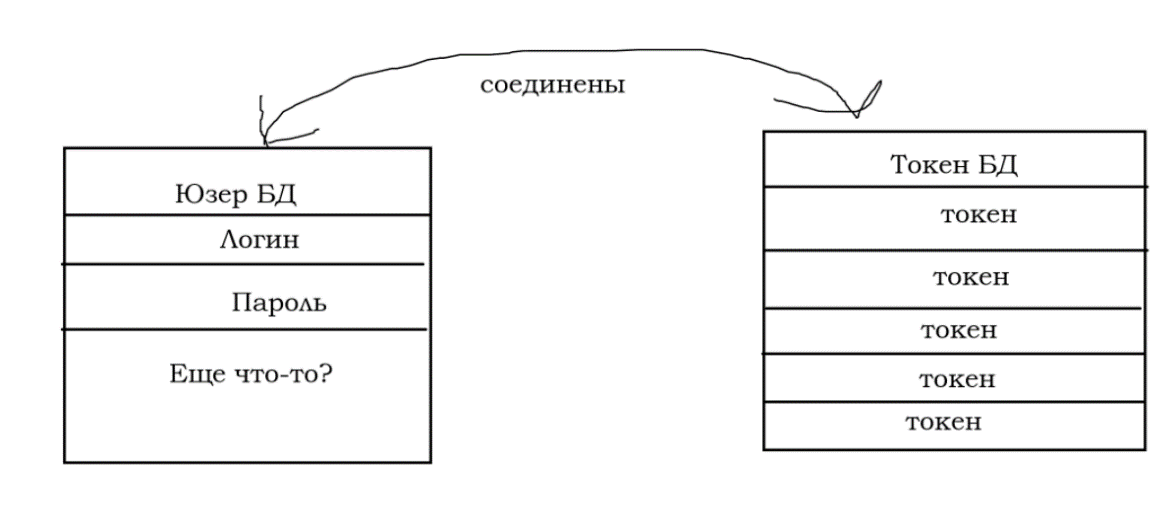


Рисунок 2 ‒ Cхема БД

Для первого знакомства пользователя с продуктом и его дальнейшей регистрации в нем хорошо подойдет вебсайт. Это быстрый, доступный и удобный способ для первого взаимодействия с потребителем, который не требует излишних действий с обеих сторон. Предполагаемый макет сайта приведен на рисунке 3.



Рисунок 3 ‒ Макет сайта

# 2 Создание серверного приложения

Для обеспечения основной функциональности приложения — двусторонней переадресации сообщений между мессенджерами — необходимо реализовать серверное приложение, работающее в реальном времени. Оно должно принимать, обрабатывать и пересылать сообщения пользователей, а также поддерживать работу с мультимедиа и устойчивость к ошибкам.

Исходя из поставленных задач, серверное приложение реализовано с использованием языка Python и библиотек aiogram, vk\_api, SQLAlchemy, а также PostgreSQL для хранения данных. Архитектура включает обработчики сообщений, базу данных пользователей и фоновый процесс опроса входящих сообщений из ВКонтакте. [2] [3] [4] [5] [6]

Для достижения целей были реализованы следующие ключевые блоки:

- дублирование сообщений между мессенджерами;

- передача мультимедийных файлов (фото, голосовые сообщения);

- логирование ошибок и отказоустойчивость.

## 2.1 Дублирование сообщений в любимый мессенджер получателя

Передача сообщений реализована с обеих сторон: пользователь пишет сообщение в Telegram, оно дублируется в ВКонтакте; аналогично, сообщения из ВКонтакте передаются в Telegram. При этом пользователю не требуется покидать привычный мессенджер. [1] [2]

Алгоритм работы показан на рисунке 4.

Процесс работы включает следующие шаги:

Пользователь отправляет команду /start, после чего происходит регистрация и выбор получателя сообщений.

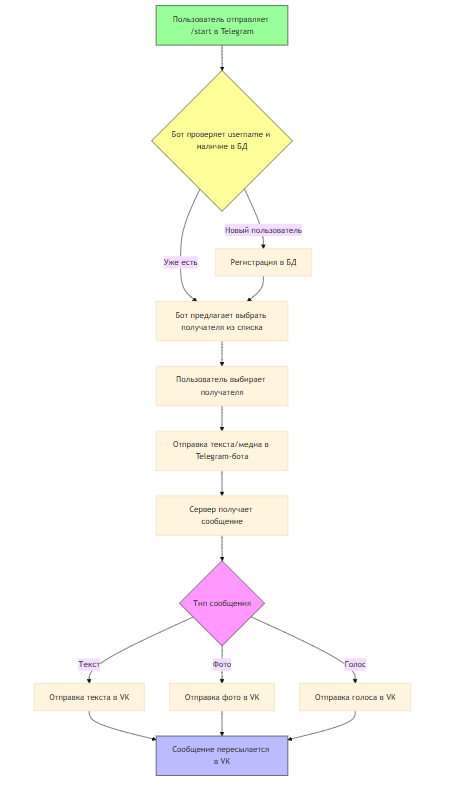
Все текстовые сообщения, отправленные после выбора, автоматически дублируются во ВКонтакте, используя API vk\_api. [2] [4]

Параллельно работает фоновая задача (polling), которая отслеживает входящие сообщения в ВКонтакте и пересылает их в Telegram с указанием имени отправителя. [1] [2]

Поддерживается пересылка не только текста, но и фото и голосовых сообщений.

Все данные (ID чатов, токены и целевые пользователи) хранятся в базе данных PostgreSQL. [6]

Для взаимодействия используется асинхронный подход, что позволяет обрабатывать множество событий одновременно без блокировки.



**Рисунок 4 — Блок-схема работы серверного приложения**

## 2.2 Обработка мультимедийных файлов (фото, голосовые)

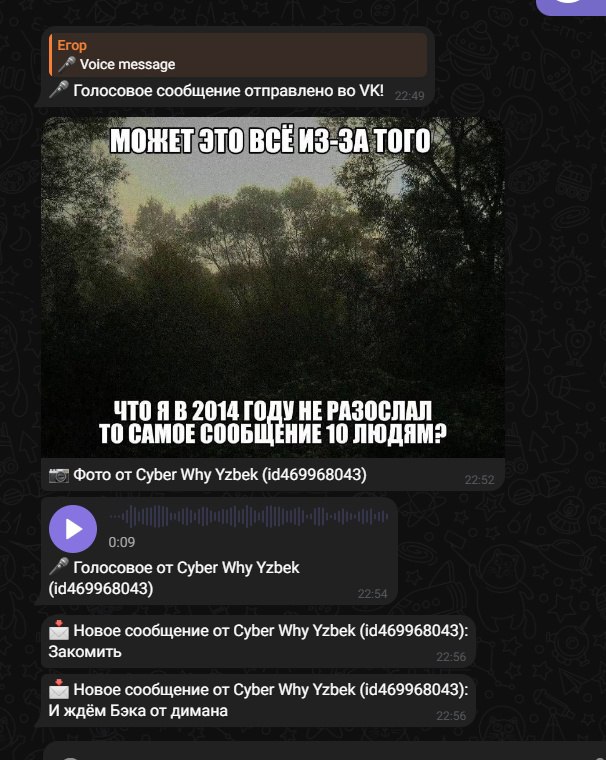
Мессенджеры имеют различия в поддержке форматов, что потребовало реализации трансляции мультимедиа:

- фото скачиваются из Telegram, сохраняются временно на сервере, загружаются в ВКонтакте как photo\_message; [1] [2]

- голосовые сообщения скачиваются как .ogg, отправляются через метод audio\_message API ВКонтакте; [2]

- входящие медиа из ВКонтакте пересылаются обратно в Telegram с прикреплёнными файлами. [1] [2]

Таким образом, пользователь может отправлять и получать медиафайлы без потери информации или необходимости дополнительных действий (рис. 5).



**Рисунок 5 — Пример пересылки изображения и голосового сообщения**

### 2.3 Создание логера ошибок

Для отслеживания состояния приложения и поиска ошибок реализован механизм логирования. В нём:

- записываются все критические ошибки, возникающие при работе с API;

- логируются сбои при отправке сообщений;

- ошибки не приводят к остановке сервера — только к уведомлению в консоли разработчика.

Это повышает стабильность приложения и облегчает его поддержку и сопровождение. Пример лога ошибок показан на рисунке 6.

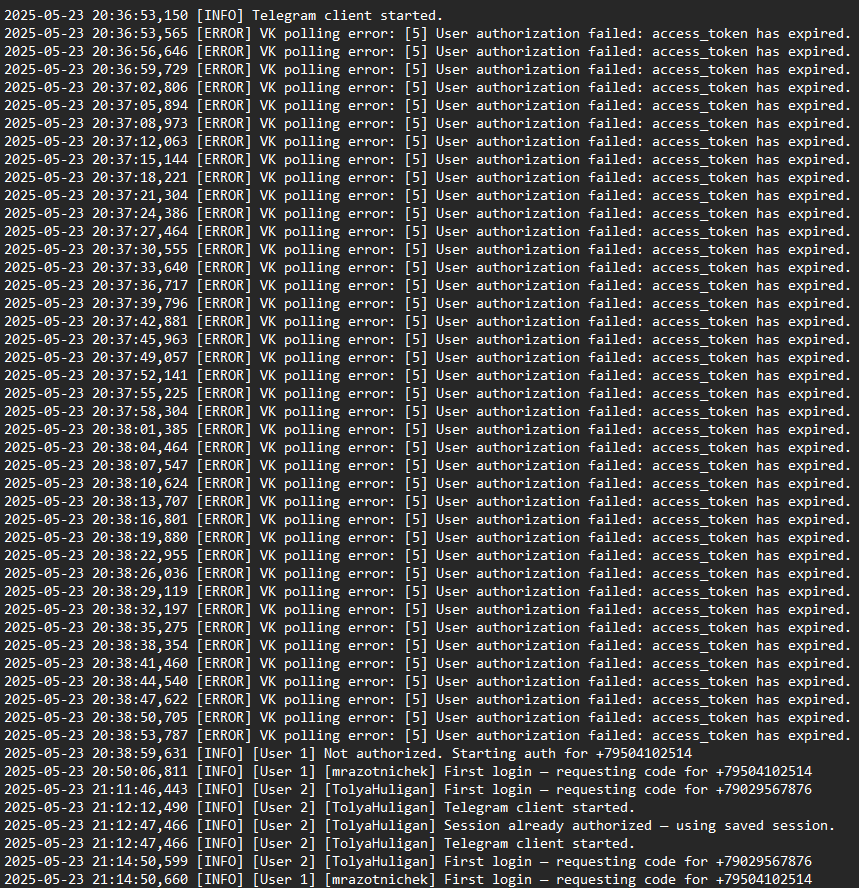


Рисунок 6 **— Пример логируемых ошибок при отправке сообщения**

## ****2.4 Вывод****

В данном разделе была подробно описана разработка серверного приложения. Были успешно реализованы ключевые функции по передаче сообщений и мультимедиа. Также было реализовано логирование.

# 3 Создание сайта

Так как сайт — это место, где пользователь знакомится с продуктом на нем должна располагаться вся интересующая информация. Через сайт удобно будет регистрироваться, добавлять токены, а также выбирать свой любимый мессенджер. Исходя из вышесказанного обозначим задачи:

- создание раздела для информирования пользователей о продукте;

- регистрация пользователя;

- создание личного кабинета;

- выбор любимого мессенджера пользователем, подключение и отключение токенов.

## 3.1 Создание общих объектов для страниц

Для каждого сайта в интернете характерны 3 основных раздела header, footer и main. На нашем сайте мы будет использовать одинаковый header и footer для всех страниц. В header будут помещены: навигационное меню, название и логотип нашего сайта. В footer поместим копирайт и кнопку для смены темы. Реализация объектов представленна на рисунках 7 и 8 соответственно.

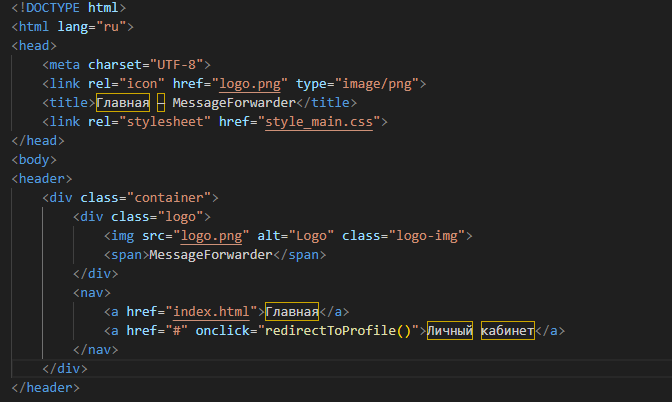


Рисунок 7 ‒ Код header-а сайта

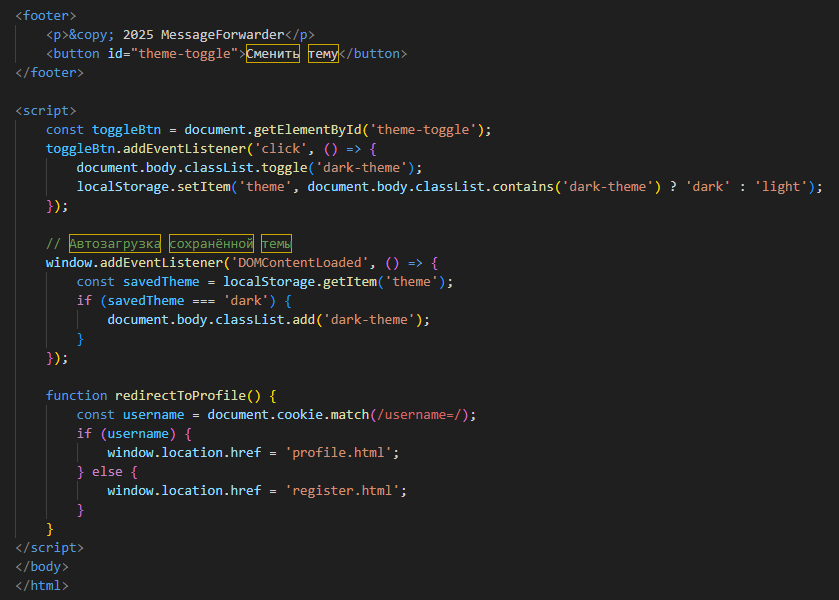


Рисунок 8 ‒ Код footer-а сайта

## 3.2 Написание основных разделов для сайта

На главной странице сайта разместим ключевые преимущества нашего приложения и добавим кнопку для перехода к регистрации/логину (рис. 9).

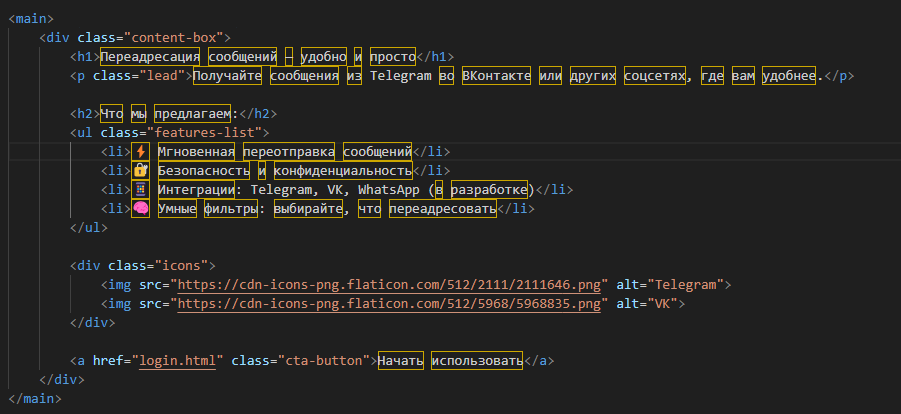


Рисунок 9 ‒ Код основного раздела заглавной страницы сайта

В разделе с логином расположим плейсхолдеры для логина и пароля, а также кнопки для регистрации нового пользователя и восстановления пароля (рис. 10).

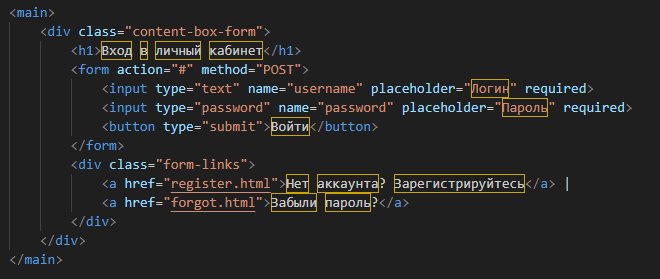


Рисунок 10 ‒ Код основного раздела страницы с логином

В разделе с регистрацией расположим плейсхолдеры для логина, пароля и email пользователя, также необходимо добавить кнопку для входа уже зарегистрированных пользователей (рис. 11).

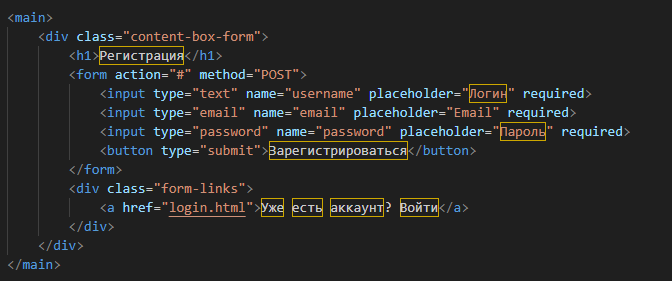


Рисунок 11 ‒ Код основного раздела страницы с регистрацией

В разделе для восстановления пароля создадим форму для запроса почты пользователя (рис. 12).

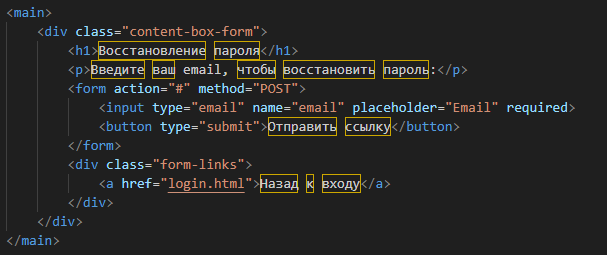


Рисунок 12 ‒ Код основного раздела страницы с восстановлением пароля

В личном кабинете пользователя будет выводится его имя и добавленные токены (рис. 13).

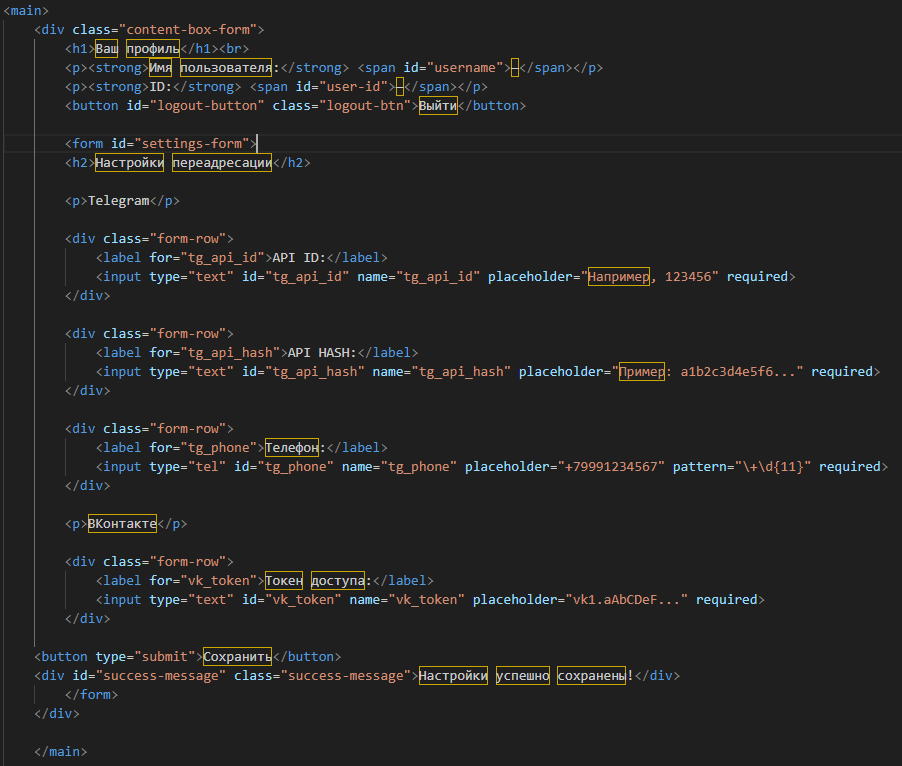


Рисунок 13 ‒ Код основного раздела личного кабинета

## 3.3 Создание стилей для сайта

В CSS-файле реализована стилизация для светлой и тёмной темы интерфейса с переключаемыми переменными (CSS Custom Properties). [8]

Были созданы:

- цветовые темы;

- основная структура;

- формы;

- визуальные элементы;

- адаптивность;

- дополнительные компоненты;

Стили представлены в приложении A.

## 3.4 Вывод

В данном разделе была подробно описана разработка веб-сайта, включающая создание структуры, основных разделов и стилизацию интерфейса. Были успешно реализованы ключевые функциональные блоки: общая структура сайта, основные разделы, стилизация и адаптивность. Скриншоты сайта представлены в приложении Б.

# 4 Шифрование данных пользователя

Шифрование данных пользователя является неотъемлемой частью любого проекта. В данной работе лучшим решением является использования алгоритма AES-256. Это современный, быстрый и надежный стандарт шифрования данных. [7]

## 4.1 Написание функций для шифрования и дешифрования

В python существует множество криптографических библиотек. В данной работе будет использоваться cryptography. Данная библиотека поддерживает выбранный ранее алгоритм шифрования. Реализуем две функции для шифрования и дешифрования данных с использованием алгоритма AES-256 в режиме GCM (Galois/Counter Mode). Этот режим обеспечивает не только конфиденциальность данных, но и их аутентификацию, что является важным аспектом информационной безопасности. Функция encrypt\_aes\_256\_gcm выполняет следующие операции: [7]

- преобразует входную строку в байтовое представление с использованием UTF-8 кодировки;

- при необходимости генерирует криптографически стойкий ключ длиной 32 байта (256 бит) и вектор инициализации (IV) длиной 12 байт;

- создает шифр с использованием алгоритма AES-256 в режиме GCM; [7]

- шифрует данные и формирует тег аутентификации;

- возвращает кортеж, содержащий зашифрованные данные, ключ, вектор инициализации и тег аутентификации.

Функция decrypt\_aes\_256\_gcm выполняет обратную операцию: [7]

- создает объект дешифратора с использованием переданных ключа, вектора инициализации и тега аутентификации;

- дешифрует данные;

- преобразует байтовое представление обратно в строку;

- возвращает исходные данные в виде строки.

Для облегчения интеграций написанных нами функций сделаем их пакетами. Код функций представлен на рисунке 14.



Рисунок 14 ‒ Функции шифрования данных

# 5 Реализация аутентификации и системы безопасности

## 5.1 Хранение и проверка паролей

Для защиты паролей пользователей в проекте используется хеширование с применением алгоритма bcrypt. Это надежный способ хранения паролей, устойчивый к атакам перебора и утечкам базы данных. При регистрации пользовательский пароль хешируется и сохраняется в базе данных, а при входе сравнивается хеш текущего ввода с сохраненным значением.

Реализация основана на библиотеке passlib, которая предоставляет удобный интерфейс для генерации и проверки хешей.

Алгоритм работы:

- пользователь вводит логин и пароль на веб-сайте;

- сервер ищет пользователя в базе по логину;

- проверяется соответствие пароля при помощи pwd\_context.verify(...);

- в случае успеха создается access-токен и отправляется пользователю в виде HTTP-only cookie.

## 5.2 Генерация JWT и OAuth2-аутентификация

Аутентификация пользователя и последующее управление сессиями реализуются с использованием JSON Web Token (JWT). Каждый токен содержит в себе закодированную информацию о пользователе (логин), срок действия и цифровую подпись.

Технологии и шаги:

- используется библиотека jose для генерации и валидации токенов;

- токен формируется с указанием срока действия (по умолчанию 30 минут);

- все защищенные маршруты FastAPI используют зависимость Depends(get\_current\_user), проверяющую валидность токена;

- пользователи без токена или с истекшим сроком получают ошибку авторизации.

Это обеспечивает защиту API и доступ только для авторизованных пользователей.

## 5.3 Безопасность токенов VK

Особое внимание уделено защите access-токенов пользователей ВКонтакте. Так как эти токены позволяют отправлять сообщения от имени пользователя, их утечка недопустима. [2]

В проекте реализовано шифрование токенов с использованием AES-256-GCM (режим Galois/Counter Mode), обеспечивающего как конфиденциальность, так и аутентичность данных. Для шифрования используется библиотека cryptography. [7]

Токены хранятся в базе данных в виде:

- vk\_token\_encrypted — зашифрованный токен;

- vk\_token\_nonce — вектор инициализации (IV);

- vk\_token\_tag — тег аутентификации.

Расшифровка токена происходит только в момент использования, перед отправкой сообщения через VK API. [2]

## 5.4 Управление настройками через веб-интерфейс

Пользователь может изменить свои данные (логин, токены и т.п.) через личный кабинет на сайте. Для этого реализован защищённый API /api/settings, который:

- принимает JSON-данные с настройками;

- расшифровывает и валидирует их;

- шифрует новые токены перед сохранением;

- обновляет данные в базе.

Реализация защищена авторизацией через токен и обработкой исключений, предотвращающей повреждение базы при ошибках.

## 5.5 Вывод

Реализованная система аутентификации и безопасности данных обеспечивает надежную защиту пользователей:

- все пароли хранятся в зашифрованном виде;

- для идентификации используется современный и безопасный механизм JWT;

- VK токены защищены симметричным шифрованием AES-256; [7]

- весь обмен данными происходит через защищенные маршруты, доступные только после авторизации.

Данный подход обеспечивает как безопасность данных, так и масштабируемость системы, позволяя в будущем добавлять новые уровни защиты или интеграции с другими сервисами.

# ЗАКЛЮЧЕНИЕ

При прохождении курсовой работы была выполнена успешная разработка и реализация механизма переадресации сообщений между различными мессенджерами. Для реализации было необходимо создать серверное приложение на Python, использовать Telegram Bot API и VK API для взаимодействия с мессенджерами, а также использовать СУБД PostgreSQL для хранения информации о пользователях и их токенах. В довершение всей работы был написан полноценный веб-интерфейс с использованием языка разметки HTML, и позволяющий работать с системой прямо из интернет-обозревателя, не требуя установки дополнительных программ. Итак, все поставленные задачи и цели выполнены. Система позволяет обмениваться сообщениями различных мессенджеров, сохранять безопасность и криптографическую невзламываемость данных алгоритмом AES-256 GCM обеспечивать журналирование ошибок для удобства контроля и реагирования. Таким образом, данная работа является описанием об успешной реализации цели и поставленной задачи. Разработанное решение действительно позволяет добиться всех целей нельзя не отметить отлично заложенные основы для дальнейшего развития - расширение списка обрабатываемых мессенджеров, внедрение механизма возврата отчетов о статусе доставки сообщения, улучшение администраторской панели управления.  
 [1] [2] [6] [7] [8]

### **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Telegram Bot API. Официальная документация. URL: https://core.telegram.org/bots/api (дата обращения: 29.05.2025).

[2] VK API. Документация для разработчиков. URL: https://dev.vk.com/api (дата обращения: 29.05.2025).

[3] Aiogram — Асинхронный Telegram Bot Framework. URL: https://docs.aiogram.dev (дата обращения: 29.05.2025). [1]

[4] Vk\_api — Python wrapper for VK API. URL: (дата обращения: 29.05.2025). [2]

[5] SQLAlchemy — The Python SQL Toolkit and ORM. URL: https://docs.sqlalchemy.org (дата обращения: 29.05.2025).

[6] PostgreSQL. Официальная документация. URL: <https://www.postgresql.org/docs> (дата обращения: 29.05.2025).

[7] Cryptography — Python cryptography library. URL: https://cryptography.io/en/latest/ (дата обращения: 29.05.2025).

[8] MDN Web Docs. HTML, CSS и JavaScript справочник. URL: <https://developer.mozilla.org> (дата обращения: 29.05.2025).

[9] Bootstrap. Документация CSS-фреймворка. URL: (дата обращения: 29.05.2025).

# ПРИЛОЖЕНИЕ А

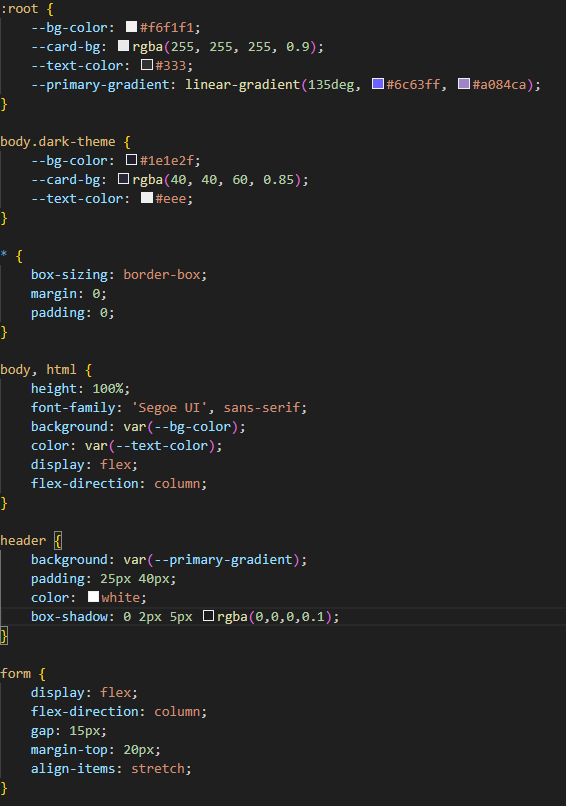


Рисунок А.1 – Стили сайта

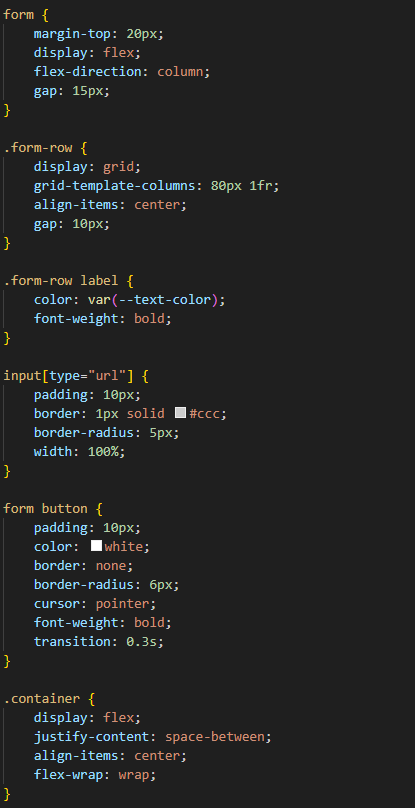


Рисунок А.2 – Стили сайта

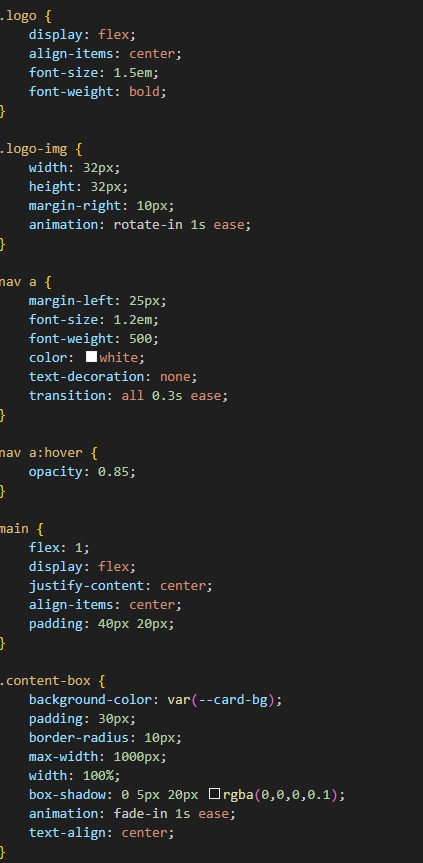


Рисунок А.3 – Стили сайта

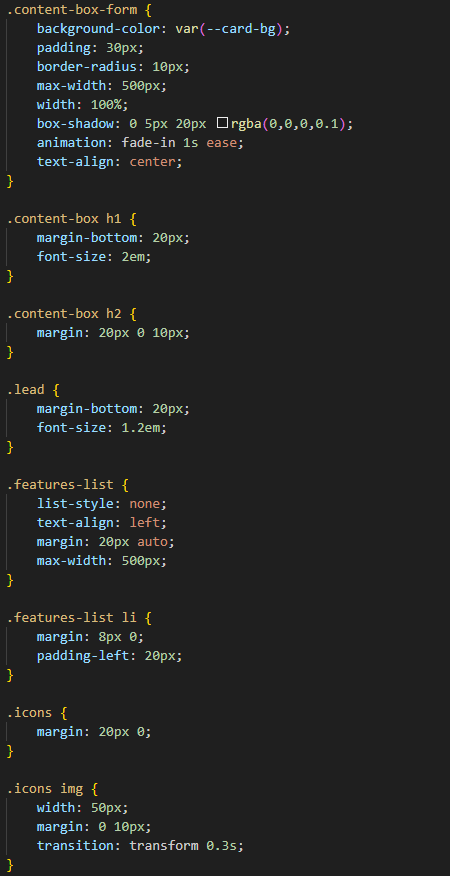


Рисунок А.4 – Стили сайта

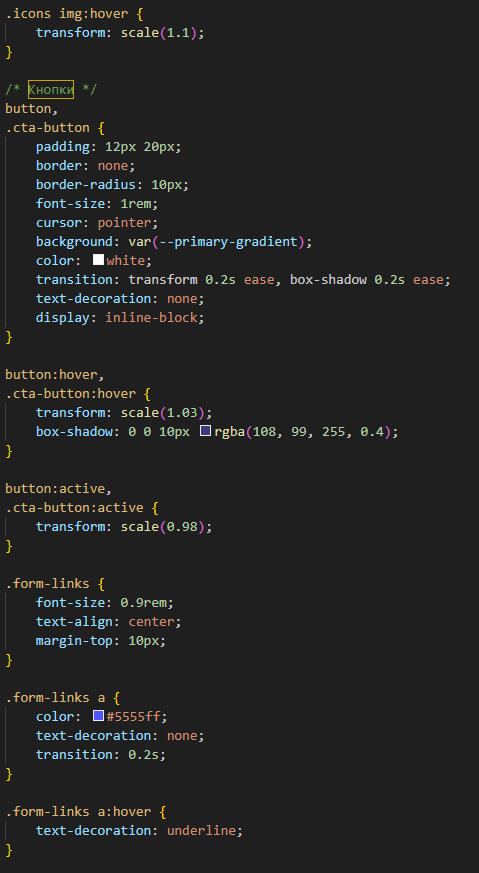


Рисунок А.5 – Стили сайта



Рисунок А.6 – Стили сайта



Рисунок А.7 – Стили сайта

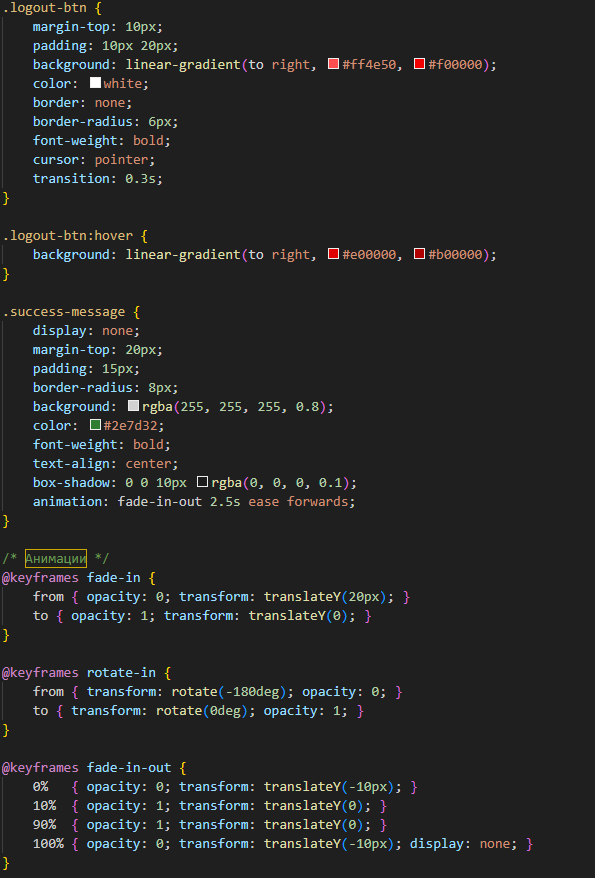


Рисунок А.8 – Стили сайта

# ПРИЛОЖЕНИЕ Б

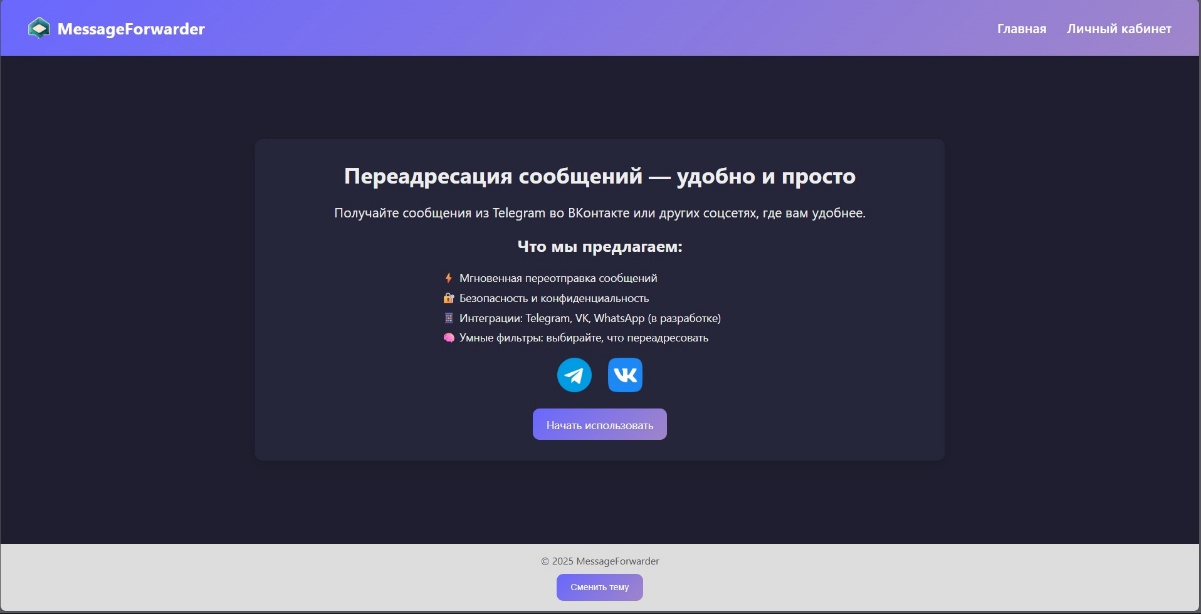


Рисунок Б.1 – Скриншот сайта

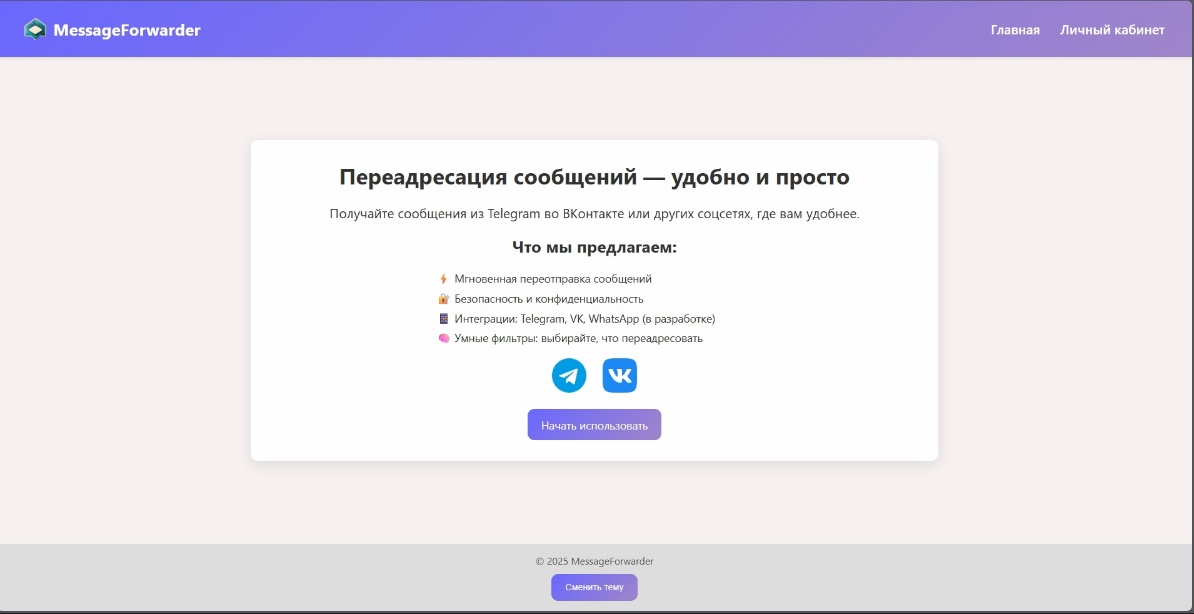


Рисунок Б.2 – Скриншот сайта

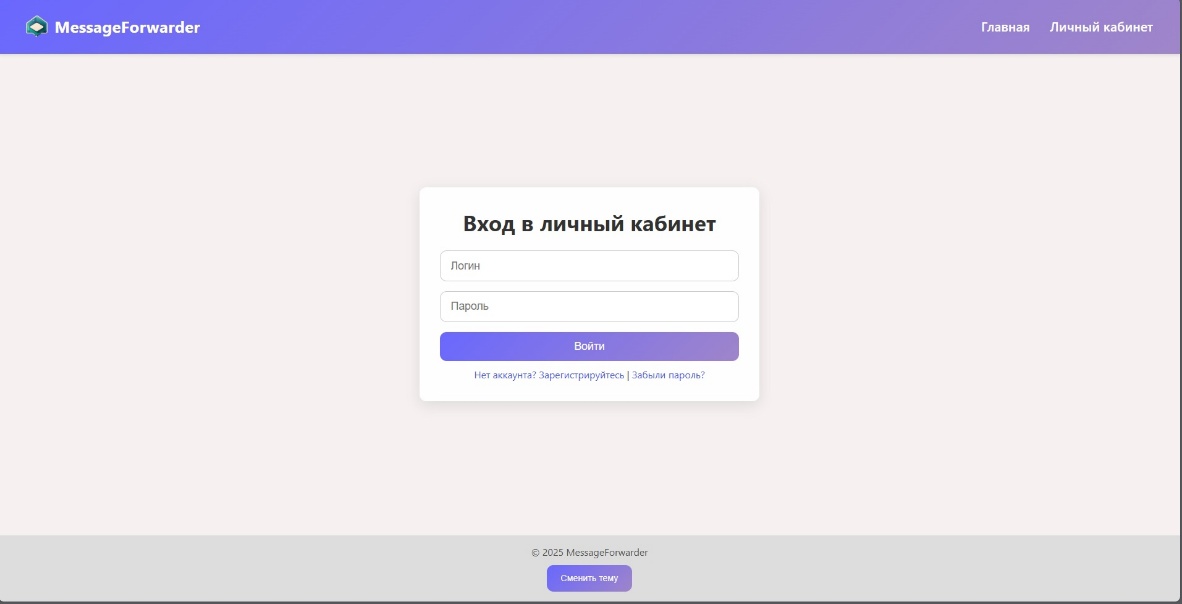


Рисунок Б.3 – Скриншот сайта

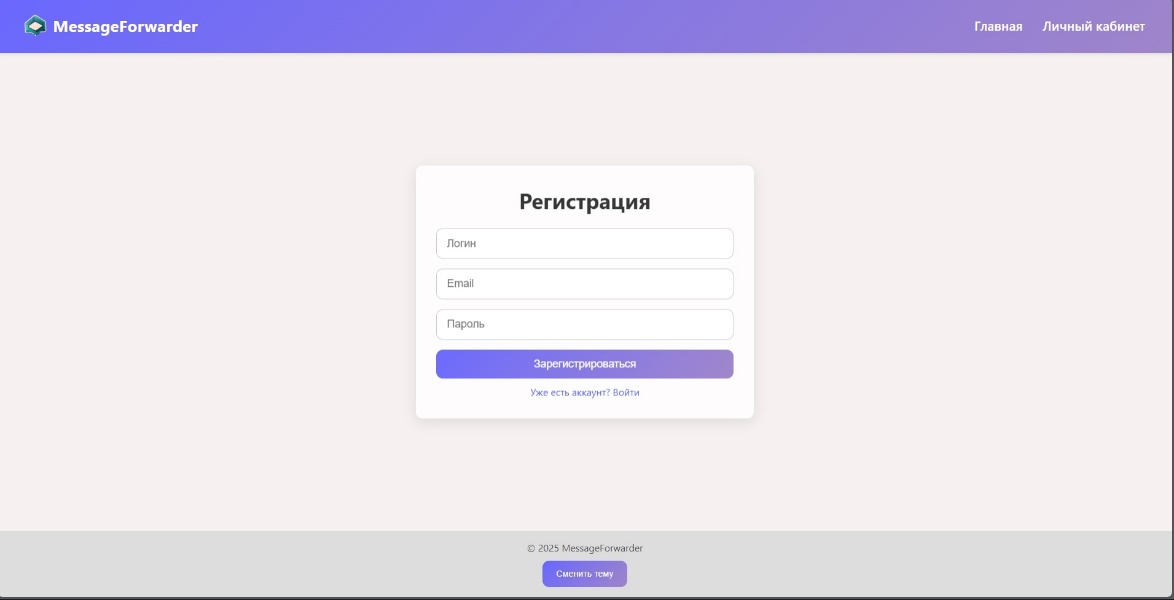


Рисунок Б.4 – Скриншот сайта

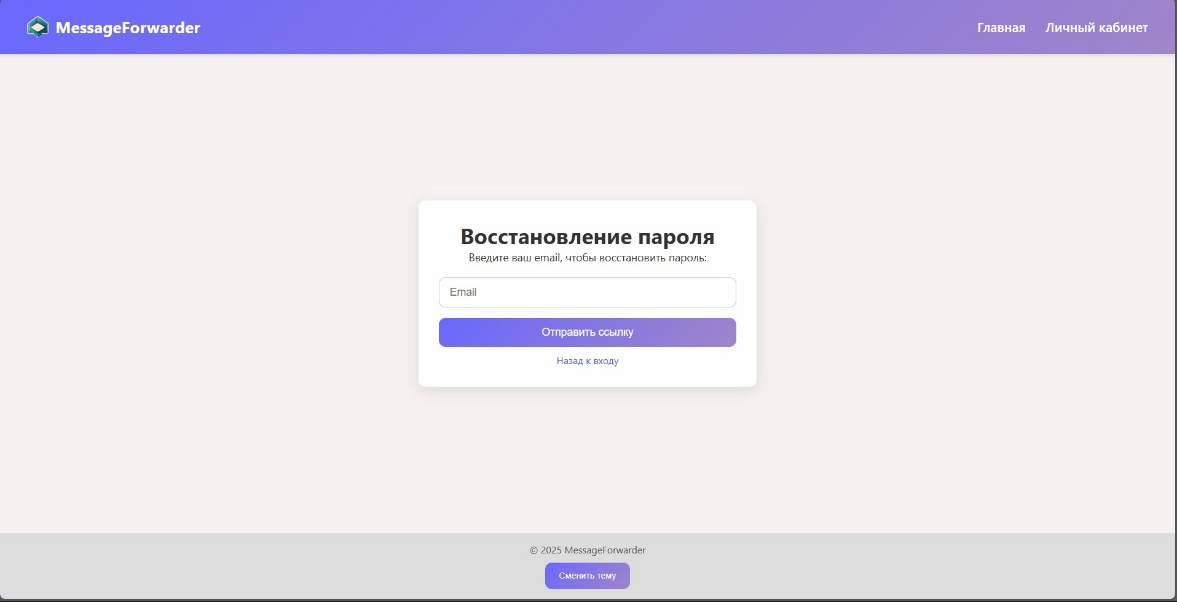


Рисунок Б.5 – Скриншот сайта