СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc11918)

[1.1 Общая характеристика задачи 5](#_Toc2996)

[1.2 Проблемы автоматизации 8](#_Toc27008)

[1.3 Анализ существующих систем 9](#_Toc19404)

[1.4 Анализ существующих программных средств реализации 15](#_Toc24458)

[1.5 Постановка задача 18](#_Toc11103)

[2 АНАЛИЗ ТРЕБОВАНИЙ 19](#_Toc19945)

[3 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 21](#_Toc17055)

[4 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 27](#_Toc17452)

[5 ПРОЕКТИРОВАНИЕ ОБЪЕКТНЫХ МОДУЛЕЙ 38](#_Toc614)

[6 ОПИСАНИЕ ПРОГРАММЫ 44](#_Toc6469)

[6.1 Алгоритмы 44](#_Toc20849)

[6.2 Описание функций 47](#_Toc7751)

[6.3 Описание программных модулей 55](#_Toc21670)

[7 ОПИСАНИЕ ИНТЕРФЕЙСА 62](#_Toc9239)

[8 ТЕСТИРОВАНИЕ 68](#_Toc17033)

[ЗАКЛЮЧЕНИЕ 86](#_Toc21394)

[СПИСОК ЛИТЕРАТУРЫ 89](#_Toc25412)

# ВВЕДЕНИЕ

Ни одна деятельность человека или компании не обходится без общения.

Каждой организации необходима платформа для безопасного общения между сотрудниками. Технологии позволяют оптимизировать, упростить и улучшить этот сложный процесс. Внедрение собственной платформы позволяет свести к минимуму возможность перехвата важных данных.

Преимущества уникального чат-мессенджера: экономия времени, более эффективное использование ресурсов, повышение прозрачности общения в компании, сохранение истории каждого взаимодействия, гибкость.

Концепция чат-мессенджера подразумевает особый способ общения между людьми по всему миру. Данная тема очень актуальна, так как чат решает многие проблемы любого учреждения.

При выполнении данной работы необходимо не только изучить принципы работы существующих систем, но и реализовать "каркас" чата и модули для работы с базой данных. При этом необходимо учитывать универсальность системы, так как данный продукт может использоваться как в некоммерческих целях, так и на предприятиях.1 СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Общая характеристика задачи

Чат-мессенджер – это многопользовательская система для формирования, передачи и хранения сообщений[1].

Задачи чат-мессенджера:

1. **Функциональность чата. О**сновная цель сайта - обеспечить пользователей средствами обмена сообщениями в режиме реального времени. Это включает в себя возможность отправки текстовых сообщений, использование смайликов, возможность отправки файлов (изображений, аудио, видео и документов), а также функции, такие как групповые чаты и приватные сообщения.
2. **Регистрация и аутентификация.** Пользователи должны иметь возможность создания учетной записи и входа в систему. Для безопасности и конфиденциальности информации важно реализовать механизм аутентификации с использованием паролей или других методов (например, двухфакторной аутентификации).
3. **Безопасность.** С учетом важности конфиденциальности переписки, необходимо обеспечить безопасность передаваемых данных, в том числе с использованием шифрования сообщений.
4. **Интерфейс пользователя.** Веб-интерфейс должен быть интуитивно понятным и удобным для использования. Это включает в себя разработку удобного дизайна, легкость навигации и поддержку мобильных устройств.
5. **Хранение сообщений.** Важно иметь эффективную систему хранения сообщений для обеспечения доступности переписки пользователей и возможности просмотра истории сообщений.
6. **Масштабируемость.** Сайт должен быть способен обрабатывать большое количество пользователей и сообщений, поэтому важно учитывать масштабируемость при разработке архитектуры системы.
7. **Мобильная совместимость.** Учитывая популярность мобильных устройств, важно обеспечить полноценную работу чат-мессенджера на мобильных платформах, возможно, с помощью мобильных приложений или адаптивного веб-дизайна.

Основная цель коммуникации – облегченение общения пользователей в сети.

Основные этапы работы с сообщениями:

1) обработка сообщения;

2) сохранение сообщения в БД;

3) отправка сообщения получателю.

Рассмотрим на примере диаграммы состояния жизненный цикл сообщения в системе (см. рис. 1.1).



Рисунок 1.1 – Состояния сообщения на различных этапах работы коммуникации

В рамках дипломного проекта предусмотрена разработка модуля коммуникации. Принцип работы следующий:

Для отправки сообщения требуется:

1. аккаунт отправителя;
2. отправитель(по умолчанию);
3. получатель(можно выбрать из графического интерфейса);
4. текст сообщения;
5. вложение (опционально).

На данном этапе проводится последовательное сохранение сообщения. Получатель может авторизироваться на сайте и получить новое сообщение.

## Проблемы автоматизации

Мессенджер решает множество проблем учреждения, таких как экономии времени на коммуникацию, сохранность данных и т.д.. Достигается всё за счёт автоматизации многих действий пользователя.

Системы коммуникации помогают выстроить простую и безопасную цепочку действий пользователя, приводящую к ожидаемому результату. Автоматизация различных процессов избавляет от недочетов, слабых мест в чате.

Мессенджер повышает производительность работы, так как позволяет:

1. коммуницировать с любого места;
2. надёжно хранить файлы на сервере;
3. фиксировать историю историю диалогов;
4. снизить число ошибок из-за несогласованности действий.

## Анализ существующих систем

1. WhatsApp (2009) на данный момент подразделение Facebook:

"WhatsApp" — это легендарный мессенджер, разработанный Ян Кумом и выпущенный в 2009 году. Эта самый обычный мессенджер в базовом его понимании [2]. Интерфейс мессенджера изображен на рисунке 1.2.

Мессенджер представляет собой мобильное, десктопное приложение для общения между пользователями в записной книжке смартфона. Дизайн приложения создан настолько простым, что с ним разберётся даже не самый продвинутый пользователь интернета.

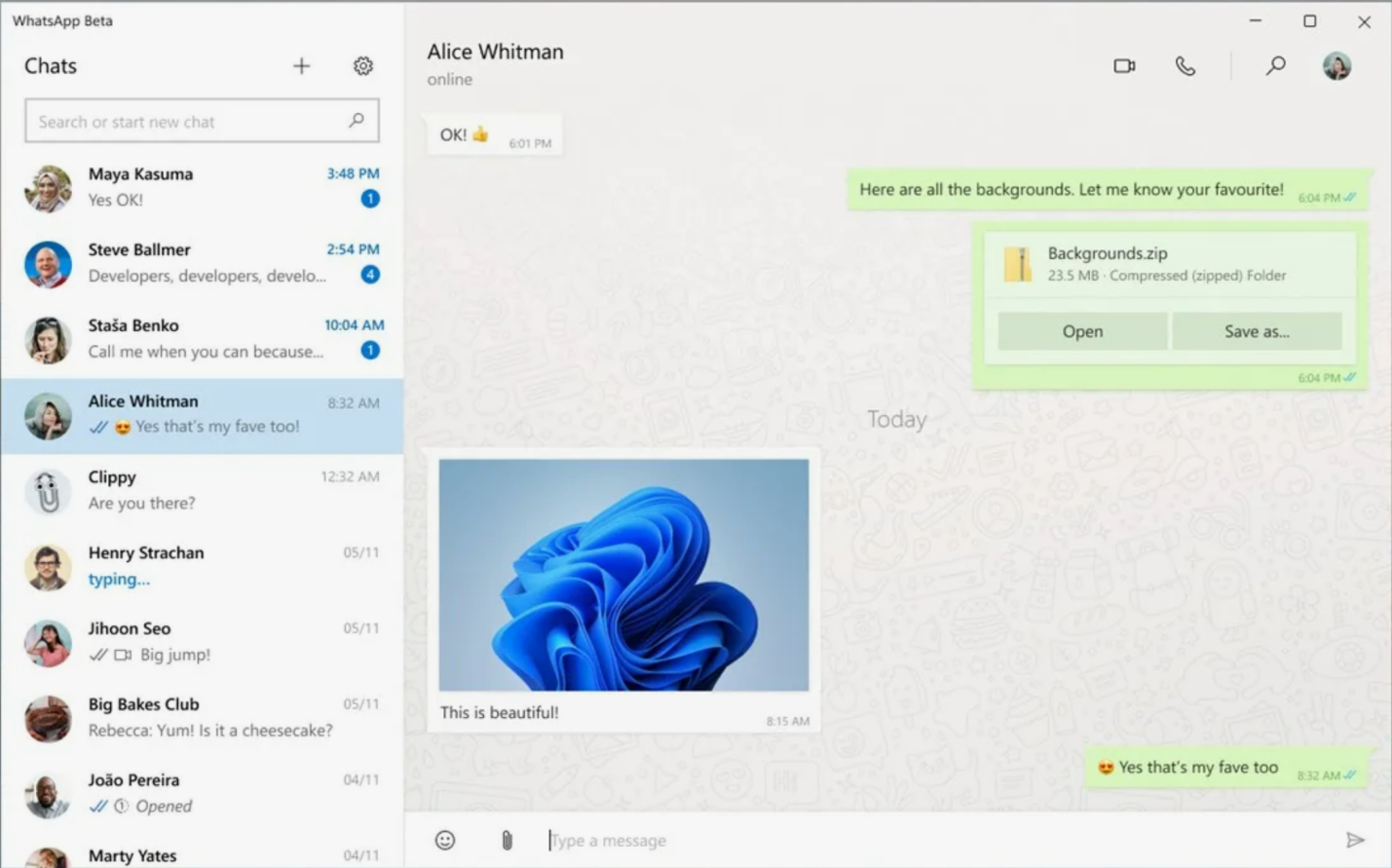


Рисунок 1.2 – Интерфейс WhatsApp

Чат предлагает относительно скудный набор инструментов. Разработчики не торопятся добавлять современные функции для своих пользователей. Так ,например, свет не увидели функции самоудаляющихся сообщений.

В свое время "WhatsApp" впечатляла тем, что был похож на приложение «Сообщения» на Android и IOS и можно было без оплаты оператору сотовой связи связываться с родственниками и друзьями в понятном интерфейсе.

Основатели компании внедрили очень простую и удобную систему регистрации в приложении, только по номеру телефона, без сбора конфиденциальных данных пользователей. Переписка является приватной, никто не сможет ее расшифровать.

Кум и Эктон говорили, что не планировали создавать компанию, а хотели лишь создать продукт, которым будет удобно пользоваться абсолютно каждому человеку. Важно отметить, что основатели WhatsApp были противниками рекламы, их девиз: «Никакой рекламы! Никаких игр! Никаких уловок!». Они считали, что маркетинг пускает пыль в глаза, не давая возможности сфокусироваться на самом продукте. Они не тратили средства на привлечение пользователей, полагаясь на органический рост.

2. WeChat(2010) работа исследовательского центра Tencent:

За каждым великим делом стоит великая личность. В случае с WeChat это программист Аллен Чжан , чье имя на родине при жизни обросло легендами. Чжан родился в 1969 году, в крестьянской семье. Его с детства тянуло к науке и он, помимо занятий в школе, постоянно учился сам, поглощая книгу за книгой. Это позволило мальчику из глухой деревни поступить в Хуачжунский университет науки и технологий (Ухань) и получить степень магистра телекоммуникационной инженерии в 1994-м году. Интерфейс мессенджера изображен на рисунке 1.3.

Изначально над проектом работала команда из семи человек во главе с Чжаном. К ноябрю 2010-го у них был готов прототип, носивший китайское название Weixin. В основном, приложение копировало Kik, мало чем отличалось от конкурентов и было запущено в январе 2011-го без особого шума и рекламы.

В начале 2012-го в мессенджере появилась одна из важнейших составляющих будущей экосистемы – QR-код. Каждый пользователь получал уникальный код и мог, просканировав код на смартфоне другого человека, добавить его в друзья, посмотреть каталог товаров, меню, расписание. Через три месяца (14 после первоначального запуска) количество подписчиков выросло до 100 млн.

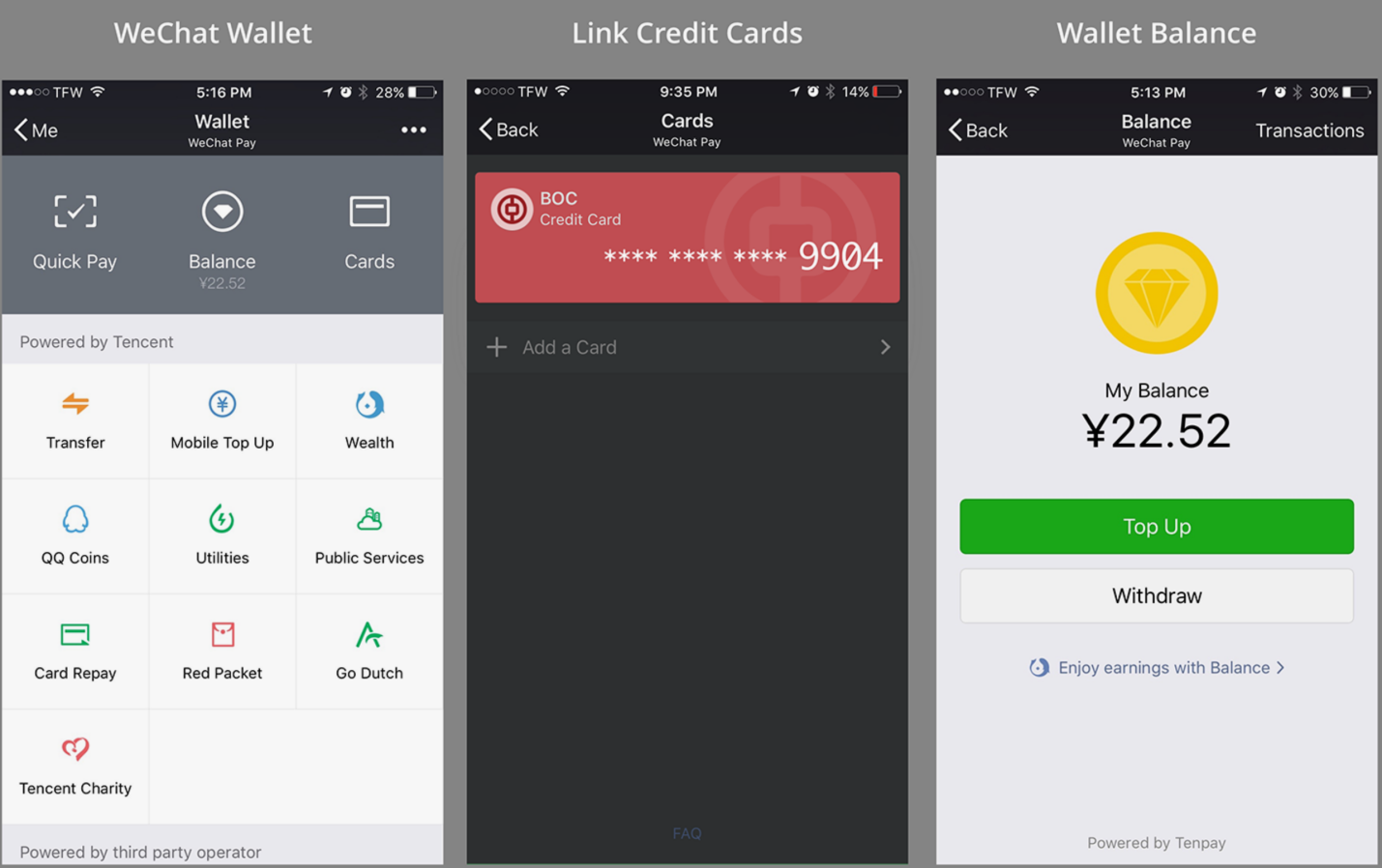


Рисунок 1.3 – Интерфейс WeChat

По данным за 2021-й год WeChat активно пользуются 1,2 миллиарда человек ежемесячно, каждый из которых тратит на него в среднем 77 минут в день, а 17% ежедневно проводит в приложении по 4 часа и больше.

3. "Telegram"(2013) Павел Дуров:

Telegram - это популярный чат-инструмент, доступный на 32 языках. За десять с лишним лет существования Telegram постоянно входит в пятерку самых скачиваемых приложений в мире, а к 2023 году число его пользователей достигнет 800 миллионов в месяц. Интерфейс мессенджера изображен на рисунке 1.4.

Со слов самого Павла Дурова, идея создания мессенджера пришла к нему еще в 2011 году. В тот момент на фоне проблем с правоохранительными органами он осознал, что у него нет безопасного способа коммуникации с близкими. Брат Павла, Николай Дуров, разработал криптографический протокол для шифрования переписки [MTProto](https://ru.wikipedia.org/wiki/MTProto" \t "https://martrending.ru/smm/_blank), который лег в основу Телеграм.

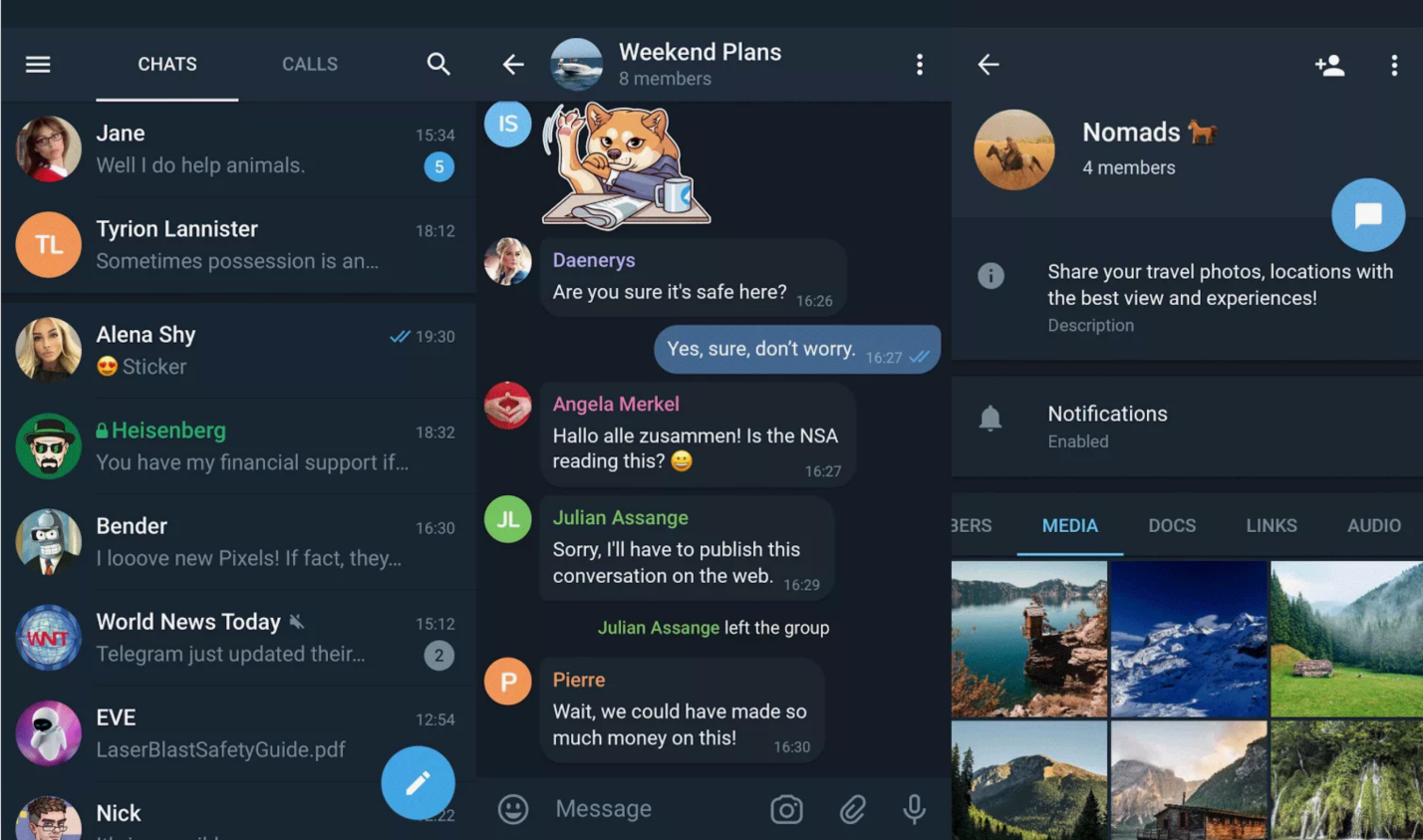


Рисунок 1.4 – Интерфейс Telegram

Одной из ключевых особенностей "Telegram" является защищенность. Разработанный протокол является безопасным способом передавать сообщений. Это и легло в основу незаконной деятельности внутри мессенджера.

16 апреля 2018 года на территории России начался процесс ограничения доступа к мессенджеру Telegram.

Предпосылкой блокировки мессенджера стали поправки в законе , имеющие антитеррористическую направленность.

Это не отменяет факта популярности мессенджера на территории СНГ. Telegram является тем, что представляет собой безопасный мессенджер.

Сравнение основных характеристик аналогов систем по основным функциональным характеристикам представлено в таблице 1.1.

Таблица 1.1 ⸺ Анализ функциональных характеристик программных систем

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики / Игры | WhatsApp | WeChat | Telegram |
| Год выпуска | 2009 | 2010 | 2013 |
| Отправка сообщений | Есть | Есть | Есть |
| Отправка голосовых сообщений | Есть | Есть | Есть |
| Поддержка платформ | Windows, MacOS, Android, iOS, S40, KaiOS, WEB | Windows, MacOS, Android, iOS, S40/J2ME, Windows Phone, Symbian, BlackBerry | Windows, MacOS, Android, iOS, WEB |
| Функция самоуничтожения сообщений | Доступна для отдельных диалогов | Нет | Есть |
| Цифровые звонки и видеозвонки | Есть | Есть | Есть |

Продолжение таблицы 1.1 ⸺ Анализ функциональных характеристик программных систем

|  |  |  |  |
| --- | --- | --- | --- |
| Защита сообщений | Присутствует. Сообщения хранятся у пользователей. | Присутствует | Присутствует |
| Платежная система | Нет | Есть. Самая популярная в Китае платежная система. | Есть. Криптовалюты GRAM,TON. |

На основе проведённого анализа мессенджеров следует отметить, что ни один из них не является идеальным, но у каждого есть свои преимущества, которые выделяют его на фоне других.

Например, WhatsApp, хоть и не хранит письма на сервере и в случае утери доступа к устройству теряются и переписки но является безопасным от взлома серверов WhatsApp. WeChat несмотря на лучшую платёжную систему, не поддерживает базовую на текущее время функцию самоудаляющихся сообщений. Telegram, хотя и имеет криптовалютные наработки но не раз попадались на обмане пользователей и привлекались к ответственности.

## Анализ существующих программных средств реализации

За время существования чатов было создано множество проектов. Как следствие, существует множество вариантов решений, связанных с используемыми инструментами.

В большинстве случаев используется клиент-серверная система - основные модули управления сообщениями и данными располагаются на специальном сервере, а клиентская часть представляет собой интерфейс для взаимодействия пользователя с системой;

Клиент-серверный подход выбран для реализации чатов. Все данные хранятся и обрабатываются на сервере, а пользователь манипулирует ими с помощью клиентского приложения. Однако для создания клиентского приложения мы используем веб-технологии.

Преимущества такого подхода - быстрая обработка данных и надежность системы. Самое главное преимущество веб-технологий - их кроссплатформенность. Доступ к сайтам можно получить практически с любого устройства.

В качестве СУБД для создания серверной части рабочего процесса была выбрана MySQL[3].

MySQL - это объектно-реляционная система управления базами данных. Ключевой особенностью объектно-реляционной базы данных является поддержка определяемых пользователем объектов и их поведения, таких как типы данных, функции, операции, домены и индексы. Это делает выбранную СУБД очень гибкой и надежной.

Клиентская часть реализована с помощью веб-технологий, таких как HTML, CSS и JavaScript(jQuery). Рассмотрим, почему были выбраны именно эти технологии.

HTML - это язык гипертекстовой разметки, который используется для отображения веб-страниц. Все содержимое хранится в статических файлах.

Преимуществами HTML-приложений являются:

1. легкий вес;
2. экономичное потребление ресурсов;
3. высокая стабильность (ошибка на одной странице не влияет на весь сайт);
4. поддерживается широким спектром браузеров [4].

CSS - это язык для обозначения стиля сайта, предназначенный для улучшения процесса взаимодействия с веб-страницами. Каскадные таблицы стилей задают внешний вид страниц.

CSS также обладает рядом преимуществ:

1. простота использования;
2. экономия времени (один и тот же код можно применять к разным страницам);
3. позволяет адаптировать страницы под разные устройства;
4. глобальные веб-стандарты;
5. приложения могут хранить CSS локально в автономном кэше;
6. CSS не зависит от платформы клиента [5].

JavaScript присутствует на странице для обработки действий пользователя, и его положительными качествами являются:

1. он быстр для конечного пользователя (действия выполняются локально, а не на сервере);
2. он использует модель DOM и предоставляет множество функций для различных объектов на странице;
3. универсальность.
4. Надстройка jQuery[6] упрощает написание кода и сокращает общий объем проекта, что положительно сказывается на читаемости кода.

Предполагается, что сервер будет написан на языке программирования Golang[7].

Golang обладает рядом неоспоримых преимуществ:

1. высокая скорость и общая производительность;
2. простой синтаксис (сходство с C++ и Python);
3. многозадачность и широкие возможности разработки;
4. язык разработан как серверный.

Поэтому выбор технологии рационален и дает возможность писать качественные, гибкие системы.

## 1.5 Постановка задача

В результате проведенного анализа существующих мессенджеров, выявленных недостатков и учтённых требований пользователей к подобным системам, принято решение о создании чата-мессенджера для демонстрации основных возможностей при разработке веб сайтов на Golang, с использованием базы данных MySQL.

Цель работы – разработка веб приложения, представляющего собой мессенджер, демонстрирующую основные механики при создании веб сайта, такие как: дизайн вёрстки, вёрстка веб-сайта, разработка БД, интеграция бэкэнд-составляющей.

Задачи работы:

1. разработка архитектуры системы (определение структуры приложения);
2. создание интерфейса пользователя (проектирование простого и интуитивно понятного пользовательского интерфейса, включая страницы входа и страницы чатов);
3. разработка программных модулей бэкэнд (реализация механики обмена сообщениями, хранения сообщений в базе данных);
4. тестирование разработанной системы (проведение тестов, включающих проверку чата на работоспособность, корректность работы интерфейса и обработку сценариев использования);
5. формулировка выводов и областей дальнейшего совершенствования системы.

# 2 АНАЛИЗ ТРЕБОВАНИЙ

Требование к программному обеспечению – это функциональная или нефункциональная потребность, которая должна быть реализована в системе.

Функциональная потребность подразумевает предоставление пользователю определённой услуги. Требование к программному обеспечению также может быть нефункциональным, это может быть требование к производительности и интерфейсу [7].

Рассмотрим бизнес-требования для программного обеспечения для системы обмена сообщениями:

1. система должна обеспечить обмен информацией между пользователями;
2. пользователь должен иметь возможность покинуть свой аккаунт;
3. система должна снизить время на коммуникацию корпоративных пользователей;
4. система должна оптимально использовать серверное время;
5. система должна оптимально;
6. система должна оптимально использовать ресурсы устройств пользователей.

Следующий этап анализа требований – формирование функциональных требований к системе:

1. информация о сообщениях при получении в списке должна содержать информацию: время отправки, вложение (если есть), текст сообщения;
2. должна быть удобная система поиска пользователей;
3. разработать систему отправки файла с базы данных пользователю ( на случай запроса пользователем вложения );
4. должно быть предусмотрено примитивное шифрование всего контента в обязательном автоматическом порядке;
5. в системе должно быть руководство пользователя;
6. необходима возможность создания неограниченного количества сообщений;
7. должна быть возможность регистрации и авторизации уже зарегистрированных пользователей;
8. необходимо обеспечить сохранность данных при потере соединения с сервером;
9. должна быть возможность сохранения активных диалогов на сервере;

Рассмотрим не фукнциональные требования к интерфейсу:

1. система должна иметь простой и понятный интерфейс;
2. форма авторизации должна содержать два текстовых поля для ввода логина и пароля, а также кнопку для входа;
3. сообщение об ошибке выводить в виде отдельной страницы;
4. на странице чата сделать меню в шапке, в котором реализовать выход из аккаунта и переключение цветовой темы.
5. реализовать добавление смайликов к сообщению.

# 3 АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Следующие вопросы оказывают существенное влияние на архитектуру разрабатываемого приложения

Параллелизм, включая вопросы организации процессов, подходы и методы.

Обработка ошибок (особые случаи) и отказоустойчивость. Проблему этой темы можно сформулировать очень просто: как предотвратить сбои или гарантировать дальнейшую функциональность системы в случае сбоя.

Взаимодействие и представление - взаимодействие между пользователем и системой, т.е. представление информации пользователю и реакция системы на действия пользователя [8].

Для создания приложения была выбрана архитектура клиент-сервер, поскольку она имеет множество преимуществ.

При создании приложения очень важно представить разрабатываемую архитектуру графически. Для реализации этого используются язык UML.

Унифицированный язык моделирования (UML) - это графический язык для визуализации, спецификации, построения и документирования систем, в которых основную роль играет программное обеспечение С помощью UML можно создавать детальные планы создаваемой системы, включая системные функции, бизнес-процессы и другие концептуальные элементы. детальные планы создаваемой системы, включая концептуальные элементы, такие как системные функции, бизнес-процессы и т. д. [9].

Для начала построим абстрактную модель разрабатываемого модуля – диаграмму вариантов использования. Основываясь на требованиях к системе, можем создать обзор приложения.

В системе предусмотрено один вида пользователей: собеседник. Собеседник получает весь функционал продукта(кроме конфиденциальной информации других пользователей).

Составим диаграмму вариантов использования чата на примере двух пользователей (см. рис. 3.1).

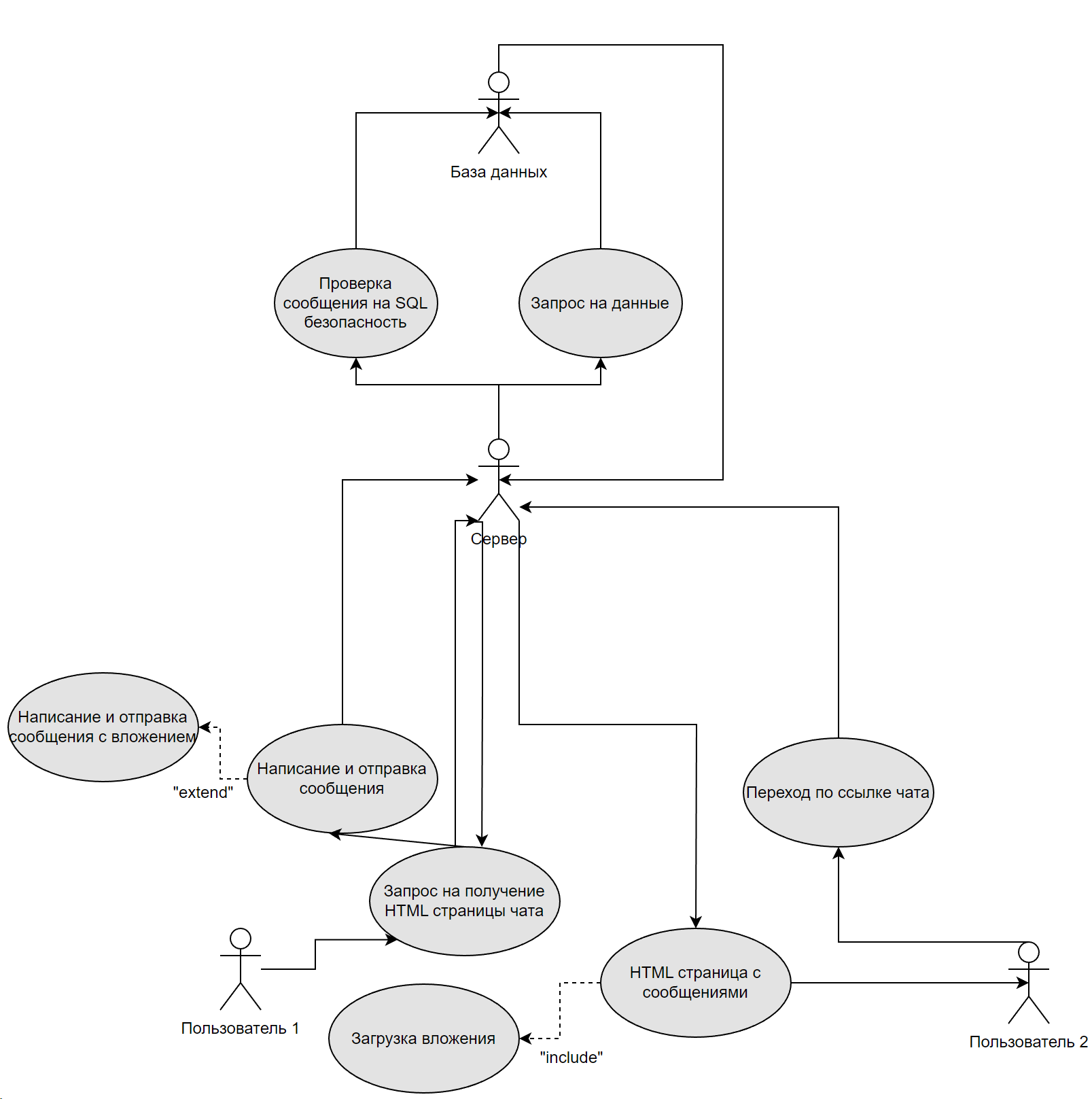


Рисунок 3.1 – Диаграмма прецедентов для двух пользователей чата

Стоит выделить какие участники предусмотрены и что им позволено делать:

1. «Пользователь 1» – отправитель сообщения.
2. «Пользователь 2» – получатель сообщения.

Рассмотрим диаграмму вариантов использования для пользователей (см. рис. 3.2).

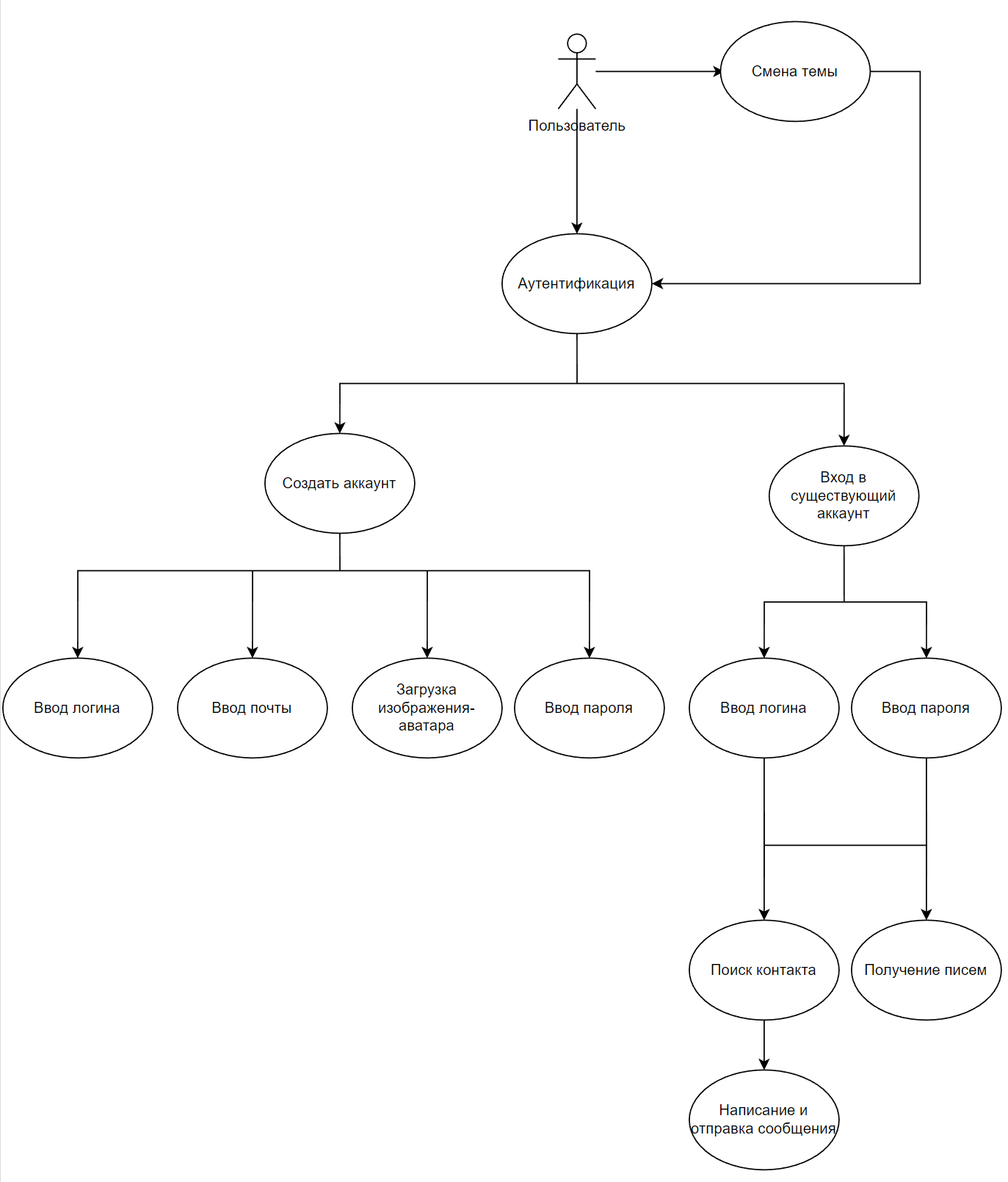


Рисунок 3.2 – Диаграмма вариантов использования для пользователя

Как видно на диаграмме, пользователь может участвовать как в отправке так и в получении сообщения (по сценарию рисунка 3.1).

Ниже представлена диаграмма состояний. На ней отображается жизненный цикл объекта, начиная с момента его создания и заканчивая разрушением. Таким объектом будет выступать сообщение (см. рис. 3.3).

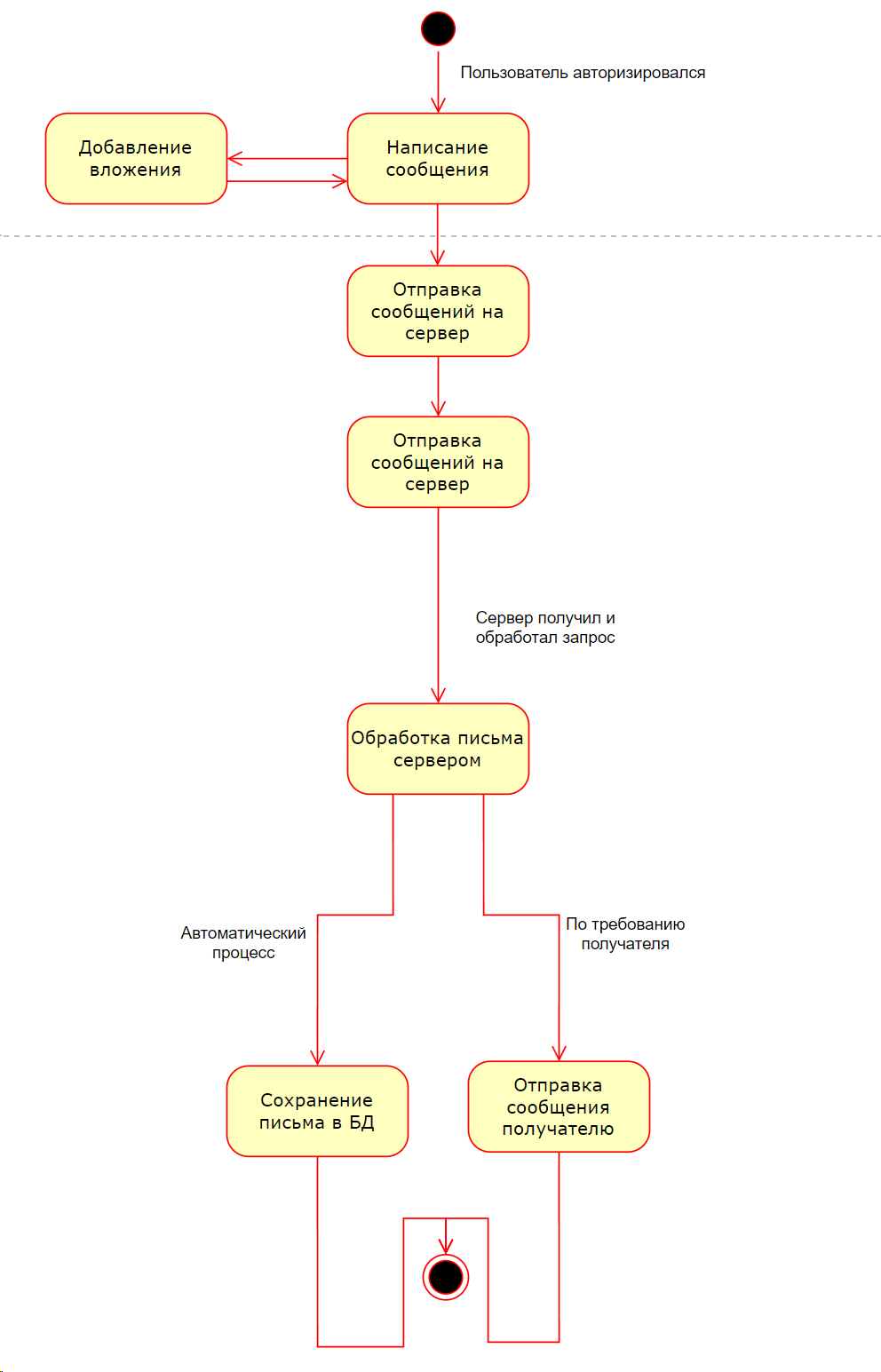


Рисунок 3.3 – Диаграмма состояний

Как и описывалось раннее, автор создаёт сообщение и отправляет сообщение серверу. Сервер отображает сообещние получателю на его странице чата.

Рассмотрим диаграмму последовательности (см. рис. 3.4), которая описывает взаимодействие объектов посредством приёма и передачи сообщений в определённой последовательности во времени.

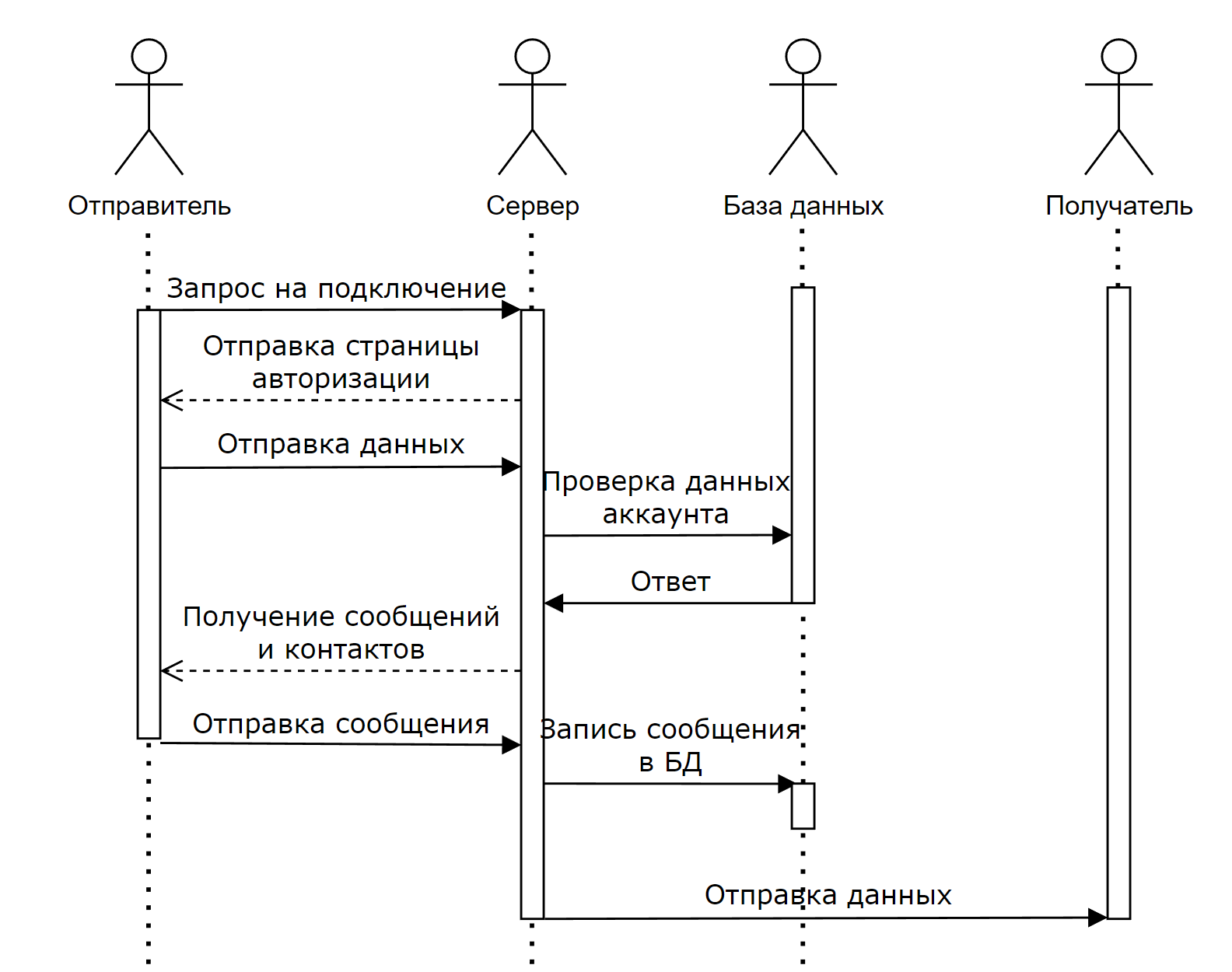


Рисунок 3.4 – Диаграмма последовательности

На данной диаграмме наглядно представлено, как разные участники ПО взаимодействуют друг с другом и чатом. Также показано, когда производится работа с базой данных.

Диаграмма компонентов позволяет создать физическое отражение системы (см. рис. 3.5).

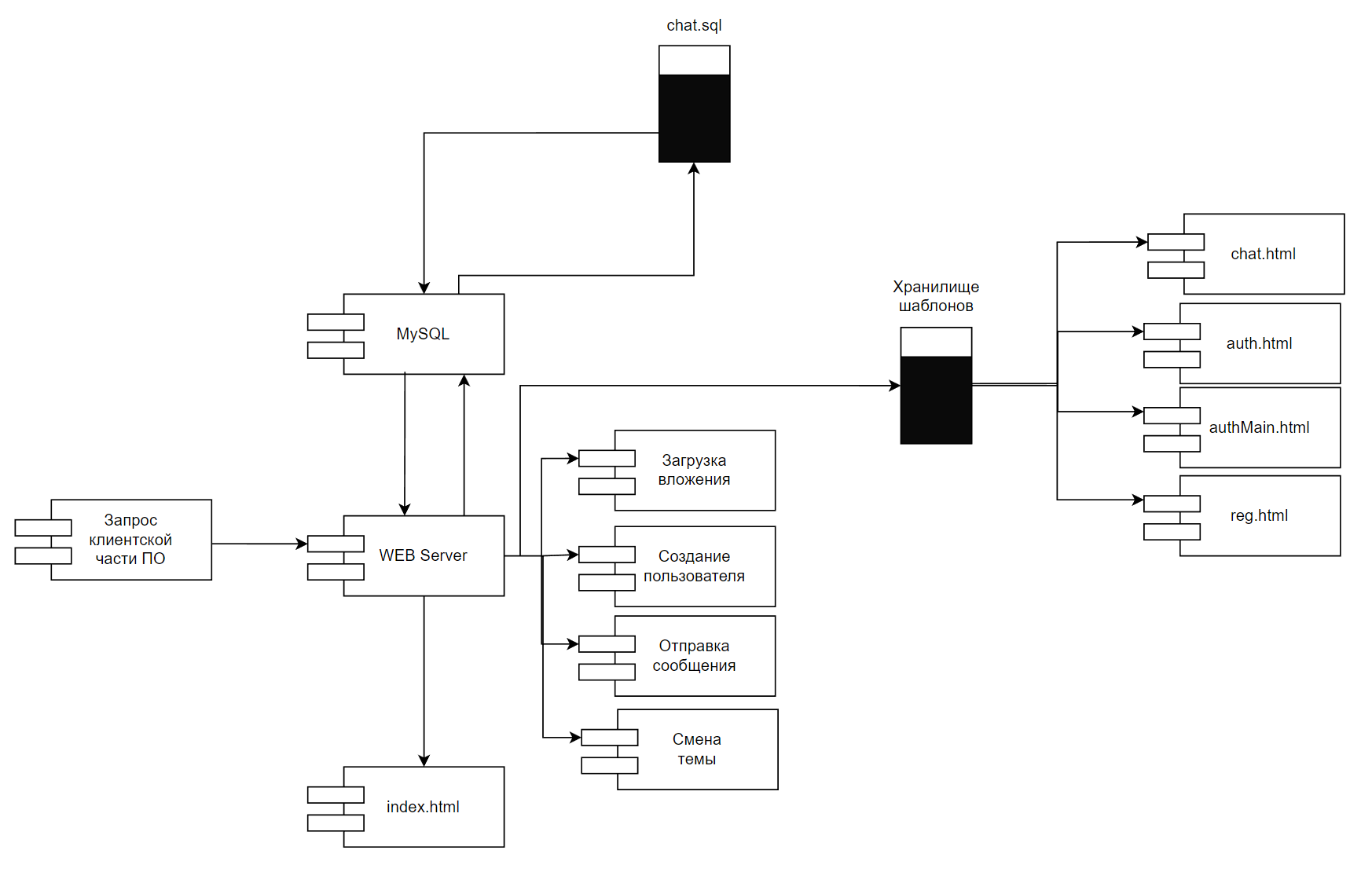


Рисунок 3.5 – Диаграмма компонентов

Продукт состоит из нескольких основных страниц: авторизация, регистрация, главная, чат. К большему числу этих файлов подключены автономные js скрипты для работы локально. Также на схеме видно взаимодействие с базой данных через «MySQL».

На диаграмме не указан css-файл. Он универсален под каждую страницу.

# 4 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Основное назначение базы данных — управление пользовательским контентом и взаимодействиями между пользователями, такими как сообщения и избранные сообщения. Для чат-мессенджера была разработана база данных с учётом ограничений для обеспечения работоспособности логики сайта. Схема данных чата представлена на рисунке 4.1



Рисунок 4.1 – Схема данных

Рассмотрим назначение каждой таблицы и ограничений более подробно:

**Таблица «**content» (см.таб. 4.1)

Таблица 4.1 – «content»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Name | Type | Key | Not null | Comment |
| 1 | contentId | int | Key | Not null | Идентификатор. |
| 2 | text | longtext |  | Not null | Текстовое содержание сообщения. |
| 3 | file | longblob |  | Not null | Содержимое файла в байтовой строке. |
| 4 | fileName | text |  | Null | Название файла. |

Таблица хранит контент, загруженный пользователями, который может включать текст и файлы. Идентификатор на эту таблицу находится в таблице «message». Было предпринято решение вынести тяжёлые данные в эту таблицу, чтобы во время поиска сообщений было ниже. Для лучшей отзывчивости поля таблицы были проиндексированы и применено сжатие данных. Скрипт создания таблицы изображён на рисунке 4.2.



Рисунок 4.2 – Таблица «content»

**Таблица «**user» (см.таб. 4.2)

Таблица 4.2 – «user»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Name | Type | Key | Not null | Comment |
| 1 | **username** | Varchar(30) |  | Not null | Уникальное имя пользователя. |
| 2 | **email** | Varchar(255) |  | Not null | Электронная почта пользователя. |
| 3 | **password** | Binary(16) |  | Not null | Хэш пароля. |
| 4 | **create\_time** | Timestamp |  | Null | Время регистрации. |
| 5 | **userID** | int | Key | Not null | Идентификатор. |
| 6 | **logo** | longblob |  | Null | Логотип пользователя в байтовом представлении. |

Таблица хранит информацию о всех пользователях мессенджера, включая их учётные данные и логотипы в байтовом представлении. Для лучшей отзывчивости поля таблицы были проиндексированы и применено сжатие данных. Скрипт создания таблицы изображён на рисунке 4.2.

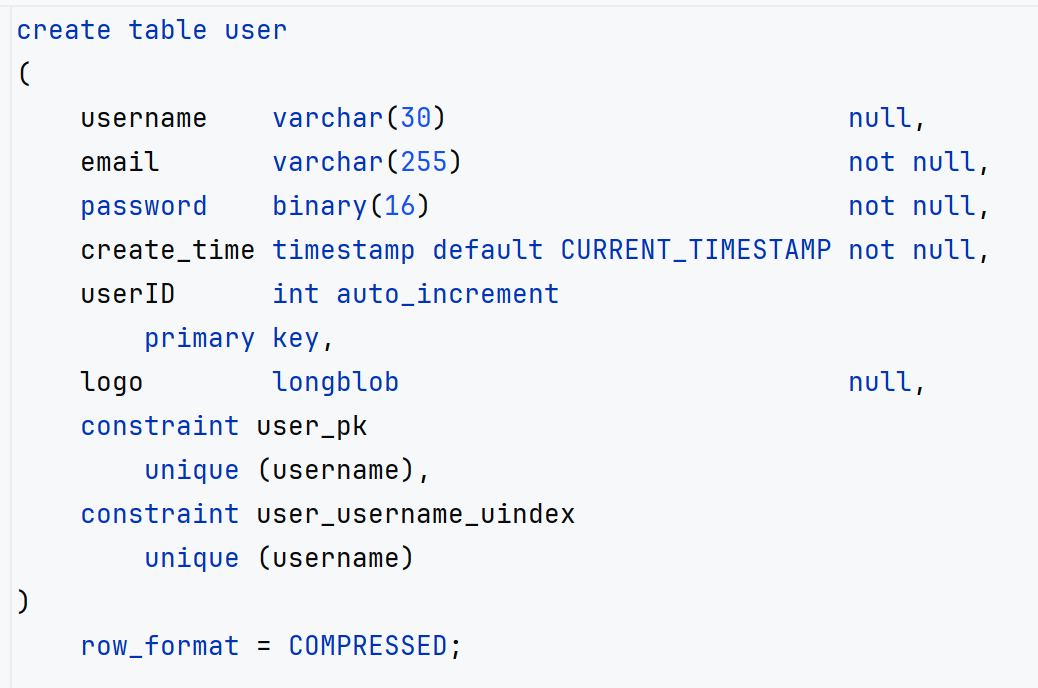


Рисунок 4.2 – Таблица «user»

Для обеспечения корректной работы логики базы данных был разработан триггер **before\_delete\_user. Он** удаляет сообщения, отправленные пользователем, перед удалением записи пользователя, Триггер необходим для поддержки целостности данных. Скрипт создания триггера изображён на рисунке 4.3

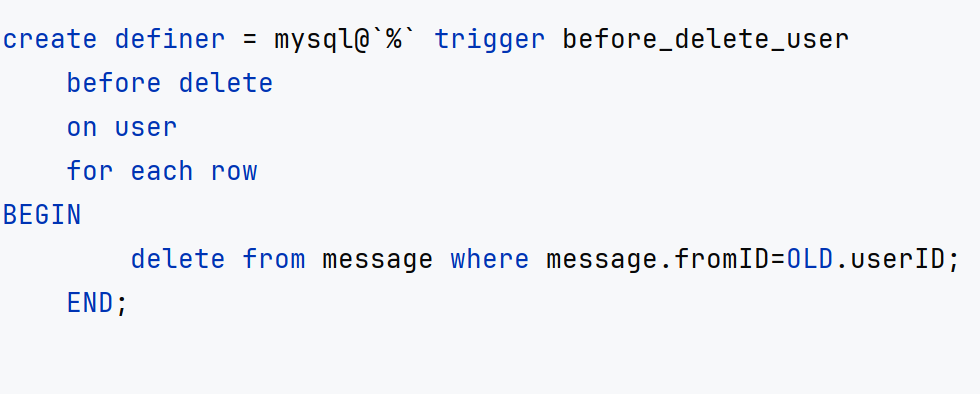


Рисунок 4.3 – Триггер «**before\_delete\_user**»

**Таблица «**connects» (см.таб. 4.3)

Таблица 4.3 – «connects»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Name | Type | Key | Not null | Comment |
| 1 | **userA** | int |  | Null | Идентификатор пользователя. |
| 2 | **userB** | int |  | Null | Идентификатор пользователя. |
| 3 | **id** | int | Key | Not null | Уникальный идентификатор связи. |
| 4 | **timeVisit** | timestamp |  | Not null | Время последнего открытия диалога. |

Таблица управления связями между пользователями. Эта таблица необходима для корректной работы последних диалогов пользователя. Для лучшей отзывчивости поля таблицы были проиндексированы и применено сжатие данных. Скрипт создания таблицы изображён на рисунке 4.4.

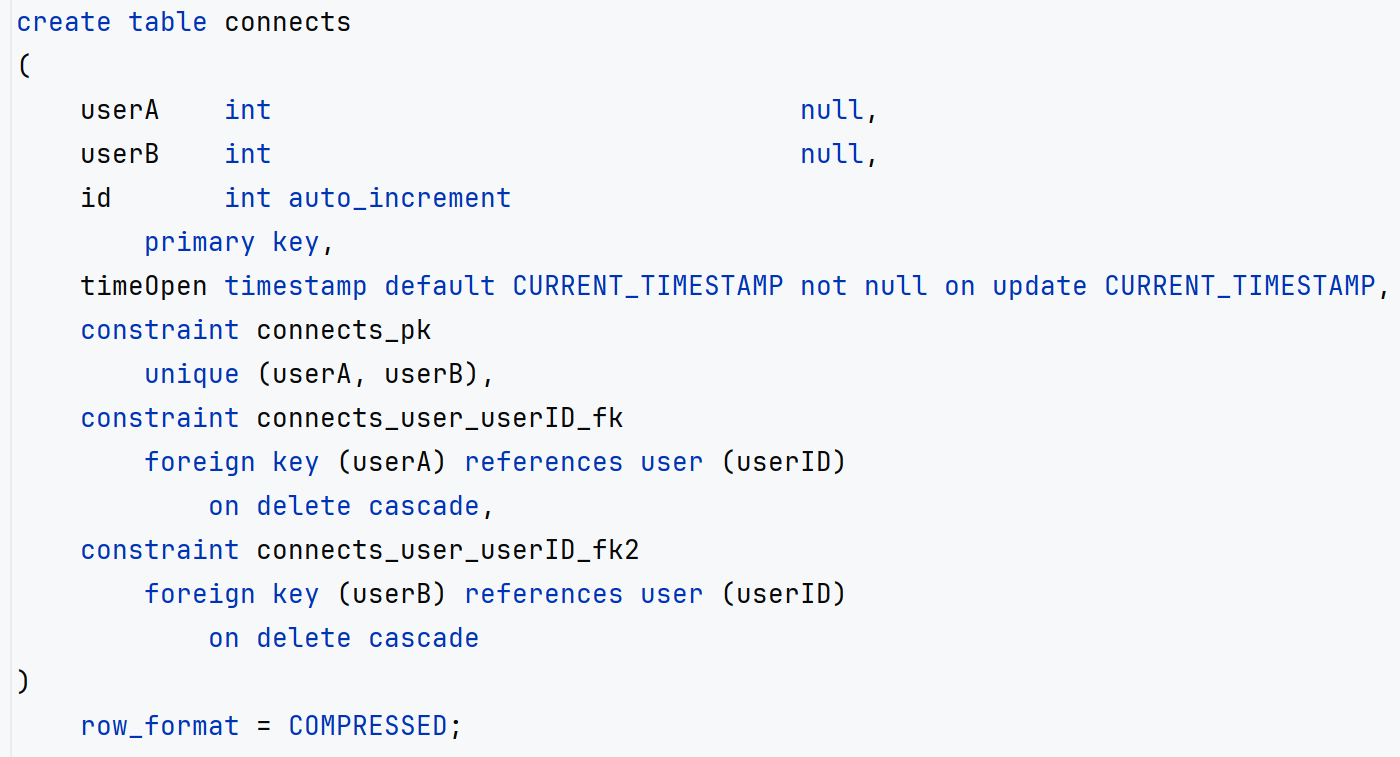


Рисунок 4.4 – Таблица «connects»

**Таблица «**message» (см.таб. 4.4)

Таблица 4.4 – «message»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Name | Type | Key | Not null | Comment |
| 1 | **messageID** | int | Key | Not null | Идентификатор сообщения |
| 2 | **fromID** | int |  | Not null | Идентификатор отправителя сообщения. |
| 3 | **toID** | int |  | Not null | Идентификатор получателясообщения. |
| 4 | **date** | Timestamp |  | Not null | Дата и время отправки сообщения. |
| 5 | **contentID** | int |  | Not null | Идентификатор контента сообщения. |

Таблица хранит информацию о сообщениях. Эта таблица необходима для хранения сообщений пользователей. Весь контент сообщения хранится в таблице «content». Для лучшей отзывчивости поля таблицы были проиндексированы и применено сжатие данных. Скрипт создания таблицы изображён на рисунке 4.5.

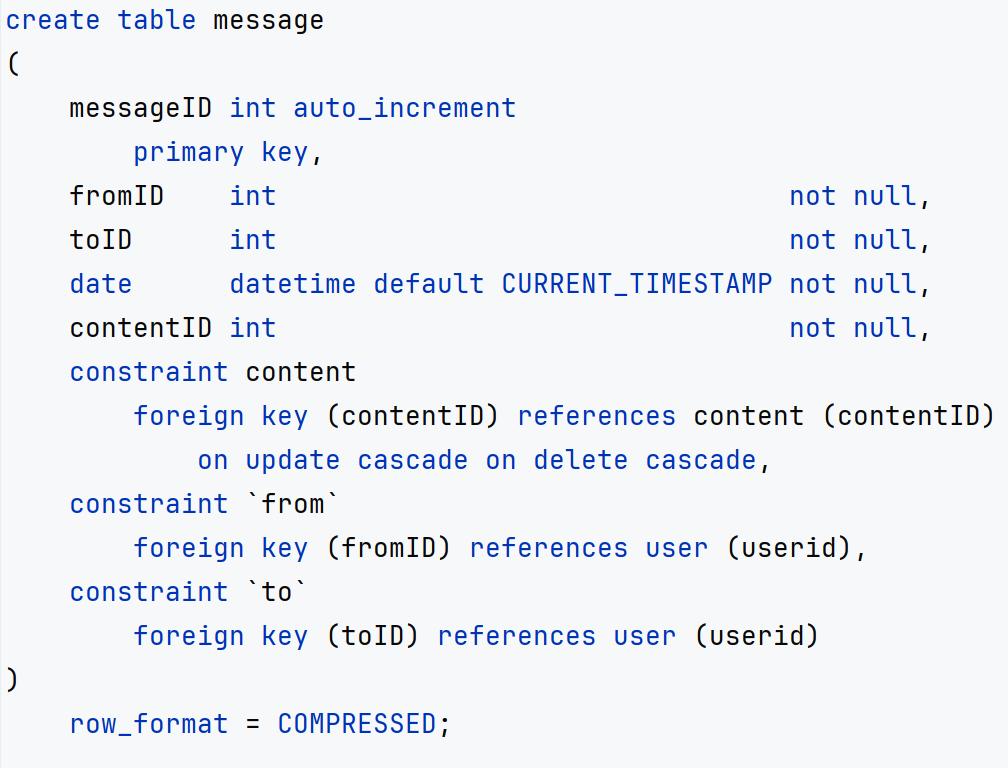


Рисунок 4.5 – Таблица «messages»

Для обеспечения корректной работы логики базы данных был разработан триггер **before\_delete\_message**: удаляет запись из таблицы content перед удалением сообщения, чтобы избежать "висячих" данных**. Он** удаляет контент, отправленный пользователем, перед удалением упоминания сообщения. Скрипт создания триггера изображён на рисунке 4.6.



Рисунок 4.6 – Триггер «**before\_delete\_message**»

**Таблица «**favouritemessage» (см.таб. 4.5)

Таблица 4.5 – «favouritemessage»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Name | Type | Key | Not null | Comment |
| 1 | **id** | int | Key | Null | Идентификатор связи. |
| 2 | **fromID** | int |  | Not null | Идентификатор отправителя сообщения. |
| 3 | **messageID** | int |  | Not null | Идентификатор сообщения. |
| 4 | **favouriteuserid** | int |  | Not null | Идентификатор пользователя. |

Таблица хранит информацию об избранных сообщениях. Эта таблица необходима для хранения избранных сообщений пользователей. Представляет собой связь между пользователем и сообщением, которое он посчитал важным для него. Для лучшей отзывчивости поля таблицы были проиндексированы и применено сжатие данных. Скрипт создания таблицы изображён на рисунке 4.7.

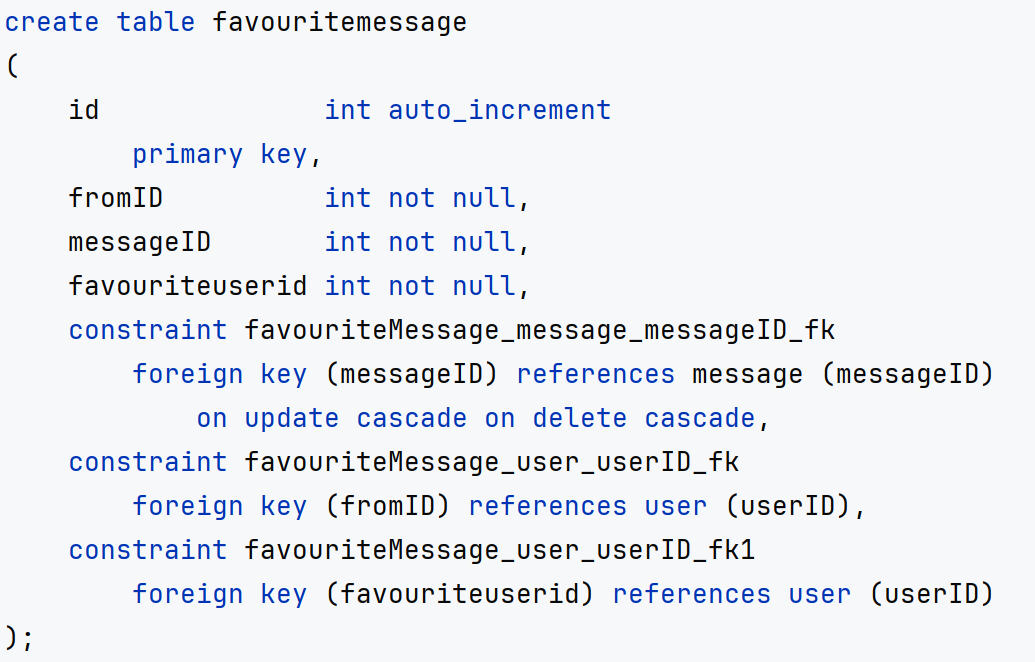


Рисунок 4.7 – Таблица «favouritemessage»

Эта структура базы данных позволяет управлять пользователями, их контентом, взаимодействиями между ними и избранными сообщениями. База данных обеспечивает целостность и поддерживает каскадное обновление и удаление данных для поддержания согласованности.

# 5 ПРОЕКТИРОВАНИЕ ОБЪЕКТНЫХ МОДУЛЕЙ

Для разработки полноценного мессенджера в веб окружении требуется слаженная работа серверной и клиентской частей. Для этого необходимо разработать устойчивые модели данных.

Рассмотрим процесс авторизации. Для авторизации пользователь заполняет форму и отправляет её на сервер. Сервер опрашивает базу данных на предмет наличия такого пользователя. Если такого пользователя не было зарегистрировано то браузер получает cookie настройку с токеном доступа(в токене зашифрован никнейм пользователя) (см. рис. 5.1)

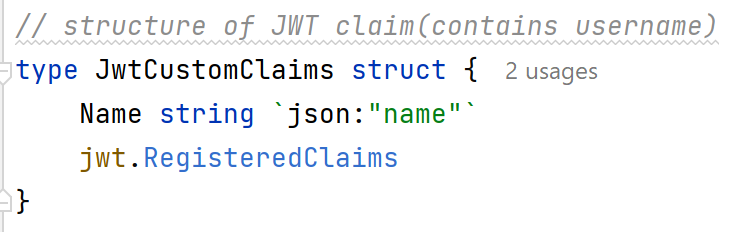


Рисунок 5.1 – Структура токена JWT

В дальнейшем сервер вместе с запросами получает и этот JWT токен. Так происходит аутентификация пользователя в системе.

Рассмотрим процесс загрузки сообщений. Предусмотрено две структуры сообщений ( избранные и обычные). Для загрузки сообщений при открытии страницы чата браузер посылает запрос серверу. Сервер проверяет корректность токена доступа и делает запрос в базу данных. После получения ответа данные добавляются в структуру (см. рис 5.2)

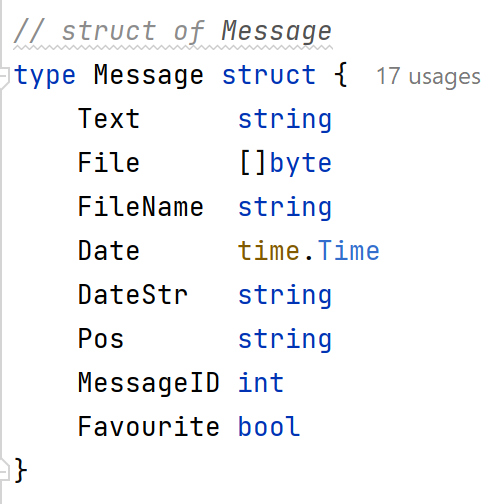


Рисунок 5.2 – Структура сообщения

В структуре сообщения предусмотрены поля для текста сообщения, содержимого файла,названия файла, даты в формате time.Time , строкового представления даты отправки сообщения, позиции сообщения(входящие сообщения отображаются слева а исходящие справа), идентификатор сообщения, индикатор избранного сообщения.

Все эти данные подставляются в шаблон HTML (см. рис. 5.3)

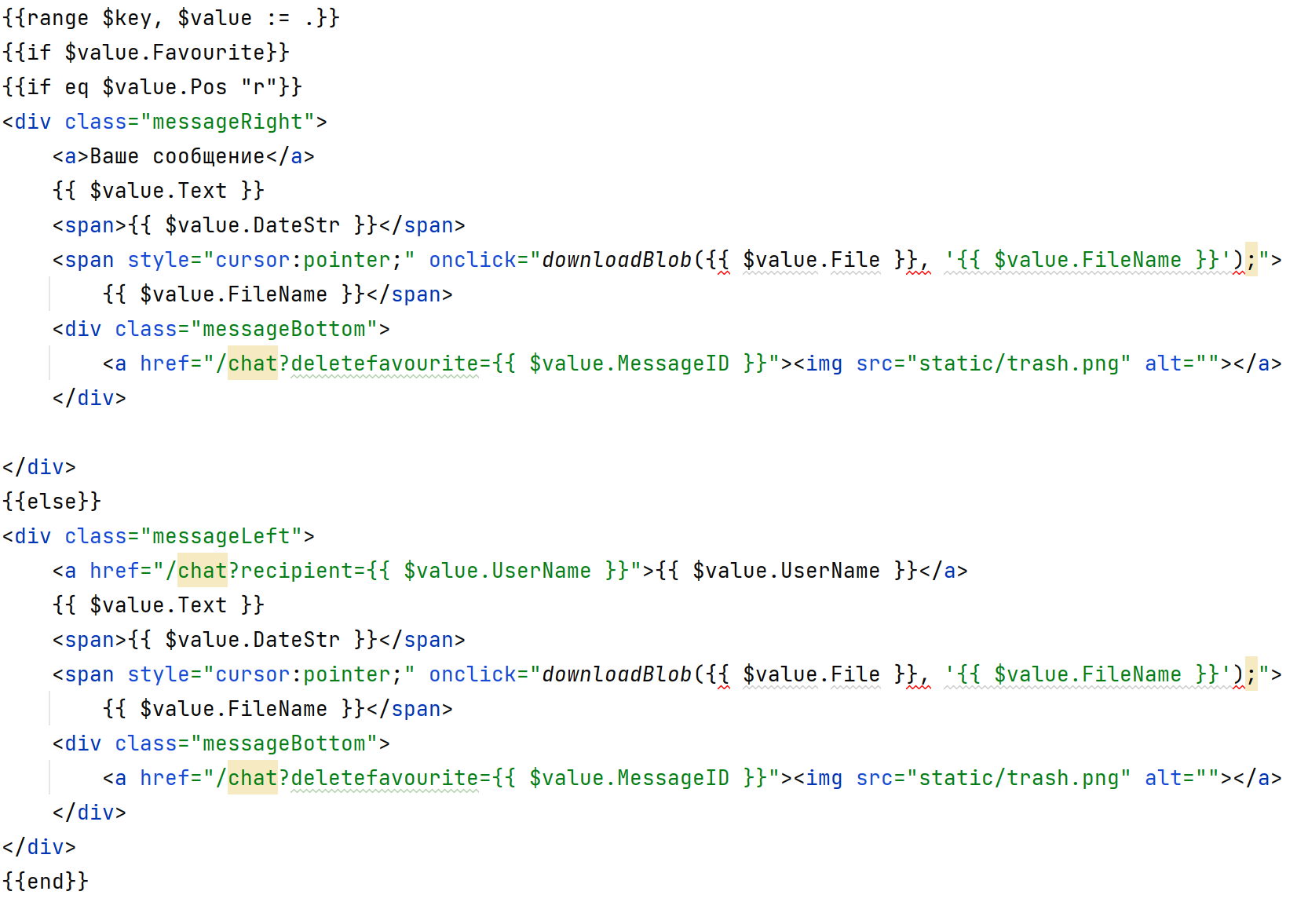


Рисунок 5.3 – Шаблон HTML для отображения исходящих сообщений

Для отображения избранных сообщений выполняется запрос на получение (нажатие на кнопку «Избранные»). Сервер обрабатывает запрос и подставляет данные из структуры (см. рис. 5.4).

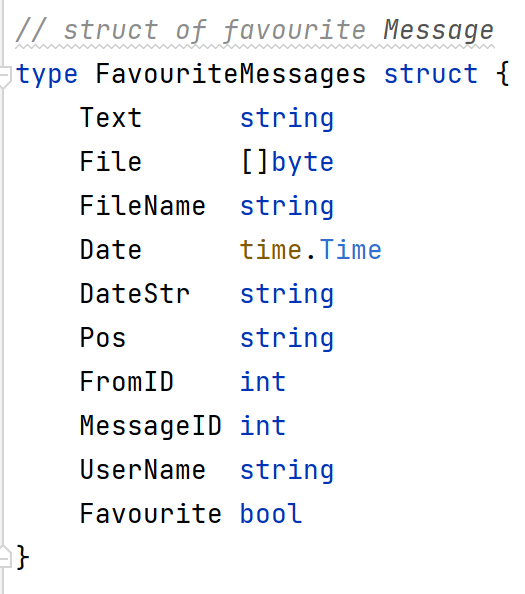


Рисунок 5.4 – Структура избранного сообщения

Как видно на рисунке, структура избранного сообщения отличается только наличием поля идентификатора отправителя.

Для автоматической загрузки сообщений используется AJAX (см. рис. 5.5). Каждые несколько секунд происходит фоновый запрос на сервер. В ответе браузер получает структуру сообщения в формате JSON. Новые сообщения уже отсортированы по дате отправки.



Рисунок 5.5 – AJAX алгоритм по добавлению новых сообщений

Чтобы браузер не получил уже полученные сообщения разработан алгоритм сохранения и обновления времени последнего отправленного сообщения(см. рис. 5.6). В cookie сохраняется самое позднее время из всех загруженных с базы данных сообщений.



Рисунок 5.6 – Сохранение времени последнего отправленного сообщения в cookie

# 6 ОПИСАНИЕ ПРОГРАММЫ

## 6.1 Алгоритмы

Программное обеспечение предполагает наличие сложных разветвлений алгоритмов для обеспечения работы мессенджера в полной мере. Рассмотрим основные алгоритмы внутри функций.

Функция сортировки дат. После преобразования всех сообщений из базы данных в структуры происходит сортировка сообщений по дате(см. рис. 6.1)

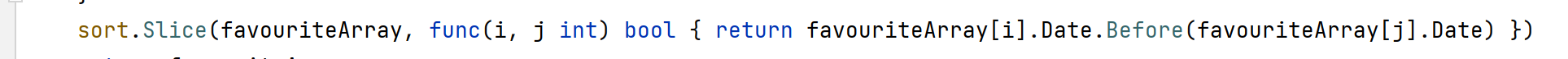


Рисунок 6.1 – Функция сортировки дат

В фунукции сортировке используется функция из пакета «sort». Встроенные функции языка Golang позволяют оптимально отсортировать слайсы и другие типы данных. На вход функция Slice получает interface{} и функцию сортировки(по какому признаку сортировать слайс. В данном случае сравниваются две даты и возвращается логическое значение (больше/меньше).

Функция сжатия изображения. Для уменьшения размера базы данных данные в неё помимо сжатия средствами MySQL также сжимаются средствами Golang. На рисунке 6.2 представлен алгоритм сжатия изображений логотипов пользователей.



Рисунок 6.2 – Функция сжатия изображения

Файл с логотипом извлекается из формы и читается побайтово. Создаётся объект изображения. Изображение масштабируется до размеров 100х100пикселей c использованием сглаживания методом Ланцоша. Такой подход позволяет с минимальными потерями качества самого изображения уменьшить разрешение. Такой алгоритм позволяет сжать размер изображение в размерах более чем в 1000 раз, что явно положительно скажется на скорости работы запросов к базе данных и снизит нагрузку на сервер.

Алгоритм хэширования пароля. Для обеспечения простого и безопасного способа хранения паролей в базе данных было принято решение использовать алгоритм MD5 для хэширования пароля. Преимуществом такого подхода является безопасность при передаче пароля через интернет и фиксированный размер хэш. Функция хэширования на рисунке 6.3



Рисунок 6.2 – Функция хэширования пароля

Для дополнительной безопасности к паролю пользователя добавляется секретная фраза-соль, что значительно усложняет подбор пароля методом Bruteforce. Во время авторизации при расшифровке пароля эта строка также добавляется к паролю пользователя.

Алгоритм кэширования. Для ускорения доступа к серверу данные в нем кэшируются средствами фреймворка ECHO. Функция включения кэширования изображена на рисунке 6.3.

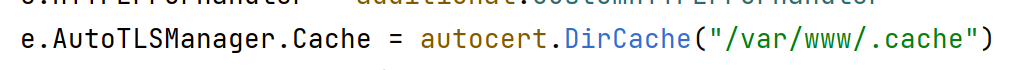


Рисунок 6.3 – Функция кэширования данных

Алгоритм валидации токена JWT. Валидация происходит путем расшифровки токена секретной фразой. Функция валидации токена изображена на рисунке 6.4



Рисунок 6.4 – Функция валидации токена авторизации

После этого происходит извлечение из него структуры данных (см. рис. 5.1). Если в этих процессах происходит ошибка то возвращается строка с ошибкой. Если всё прошло успешно то функция возвращает никнейм пользователя.

## 6.2 Описание функций

Рассмотрим функцию работы страницы чата на рисунке 6.5



Рисунок 6.5 – Функция GET обработчика сайта (часть 1)

В этой функции обработчика присутствуют такие модули:

1. Проверка GET запроса на наличие поля «logout». Если оно присутствует то вызывается функция по выходу из аккаунта. Алгоритм предполагает удаление токена доступа из cookie.
2. Проверка GET запроса на наличие поля «toggle». Если оно присутствует то алгоритм меняет текущую тему на другую (тёмную/светлую).
3. Проверка cookie пользователя с целью определения цветовой темы сайта(она записывается в cookie автоматически при первом входе на сайт).
4. Проверка токена авторизации(если он не действителен то происходит перенаправление на страницу авторизации).
5. Поиск собеседника в cookie(автоматически добавляется при переходе на страницу диалога).
6. Проверка GET запроса на наличие поля «favouritechats». Если поле присутствует значит пользователь желает перейти на страницу избранное. Алгоритм удаляет из cookie идентификатор текущего диалога.
7. Проверка GET запроса на наличие поля «deletefaviourite». Если поле присутствует значит пользователь хочет удалить сообщение. Функция удаления представлен на рисунке 6.6.

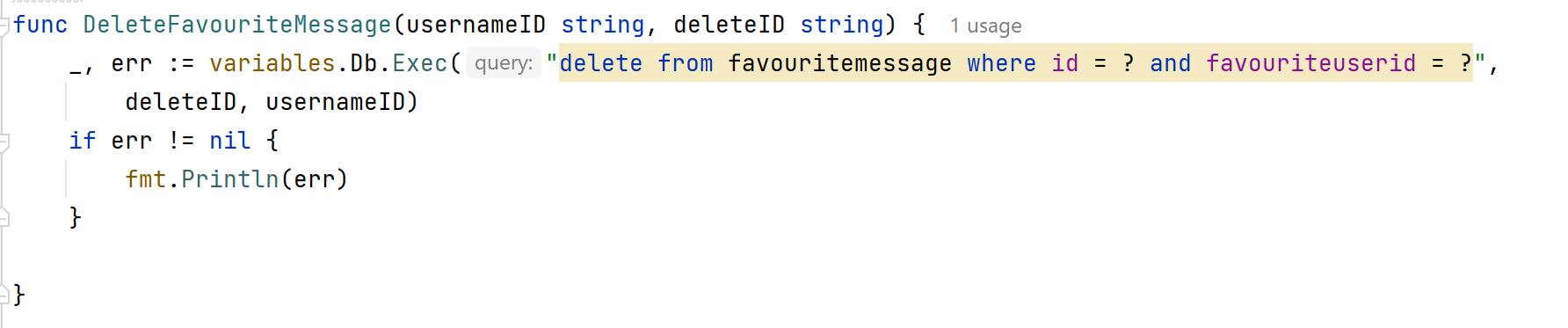


Рисунок 6.6 – Функция удаления избранного сообщения

1. Проверка GET запроса на наличие поля «delete». Алгоритм проверяет принадлежность пользователя к этому сообщению и если пользователь действительно получал или отправлял это сообщение то сообщение будет удалено из базы данных. Функция удаления сообщения представлена на рисунке 6.7.

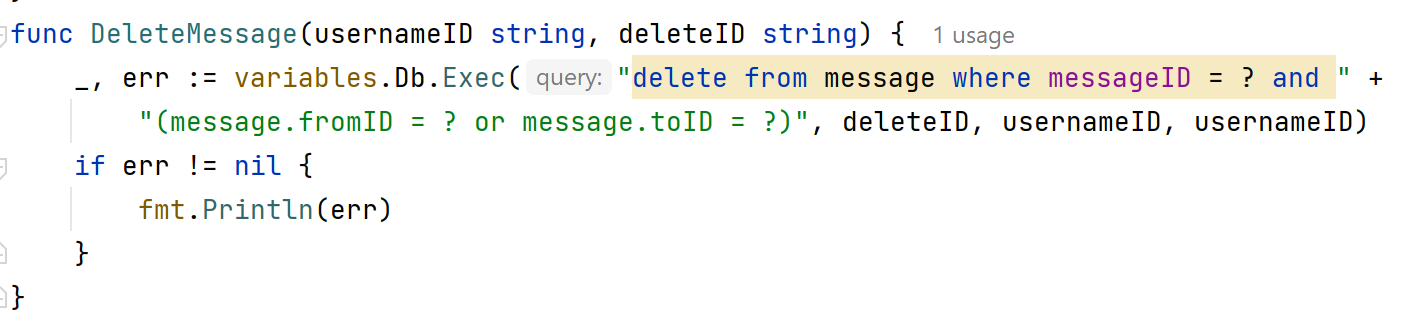


Рисунок 6.7 – Функция удаления сообщения

Вторая часть функции работы обработчика GET запросов представлена на рисунке 6.8.



Рисунок 6.8 – Функция GET обработчика сайта (часть 2)

В этой функции обработчика присутствуют такие модули:

1)Модуль добавления сообщения в избранный список. Если пользователь нажал на кнопку добавления в избранные сообщения то сервер получит GET запрос который обработается и идентификатор добавится в базу данных.

2)Проверка на наличие собеседника в диалоге(если его нет значит пользователь желает попасть на страницу с избранными сообщениями).

3) Если собеседника нет то выполняется загрузка избранных сообщений из базы данных и отправка шаблона с нужной цветовой темой пользователю. Алгоритм загрузки избранных сообщений представлен на рисунке 6.9



Рисунок 6.9 – Функция получения избранных сообщения

В функцию получения избранных сообщений выполняется запрос в базу данных на получение сообщений. После выполняется чтение каждого сообщения отдельно и добавление нового сообщения в слайс favouriteArray. Предусмотрен случай когда пользователь не прикреплял файл. Его сообщение не имеет имени файла и содержимого в структуре. На рисунке 6.4 представлен только алгоритм получения исходящих сообщений. Функция получения входящих сообщений аналогичен. Отличия присутствуют в запросе к базе данных и полем «Pos» в структуре сообщения(l - входящие, r - исходящие).

Получение сообщений из диалогов происходит подобным образом за исключением другого запроса SQL.

Рассмотрим функцию определения собеседника диалога на рисунке 6.10.

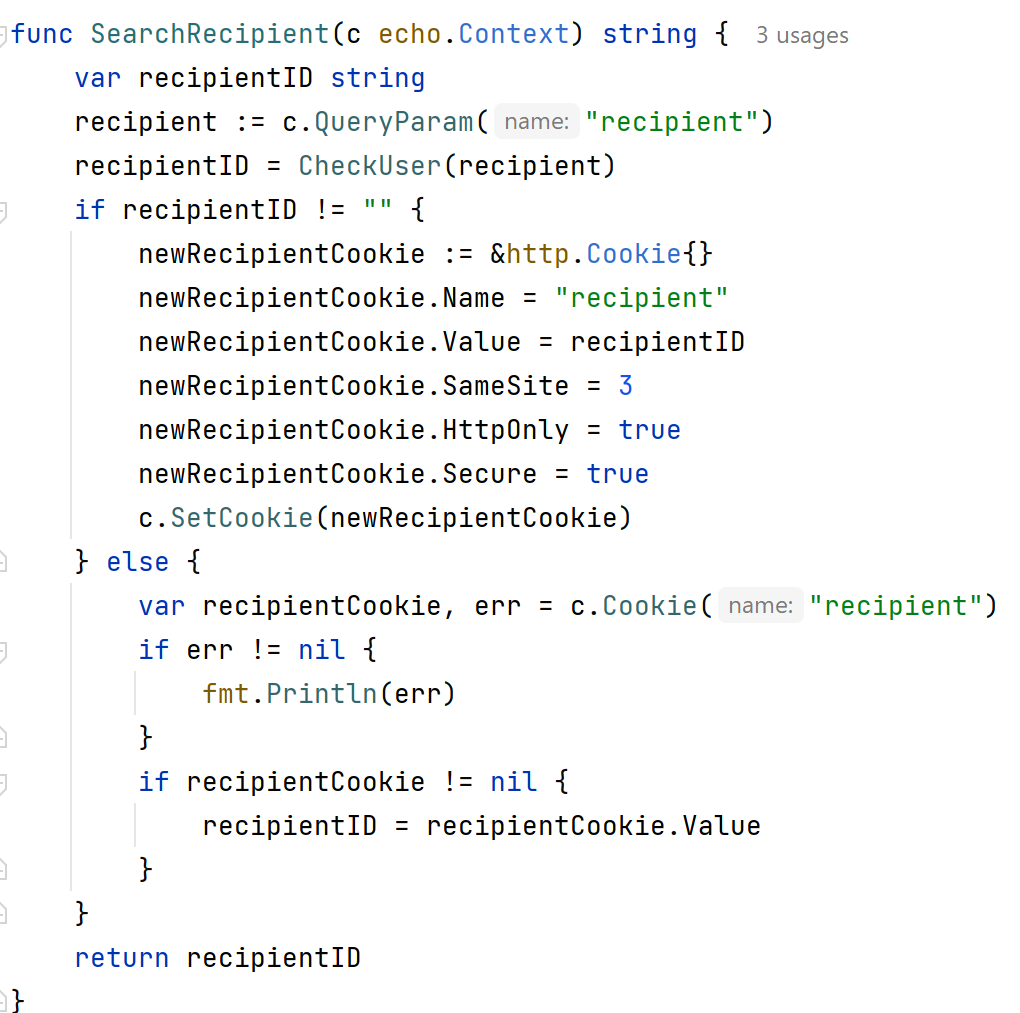


Рисунок 6.10 – Функцию определения идентификатора собеседника

В функции происходит поиск параметра собеседника в GET запросе. Если собеседника нет(не было совершено перехода на страницу с диалогом пользователя) то данных извлекаются из cookie и возвращаются в функцию chatGET с рисунка 6.1. Если собеседника нет и в cookie данных то возвращается пустая строка идентификатора.

Рассмотрим функцию смены цветовой темы сайта на рисунке 6.11.

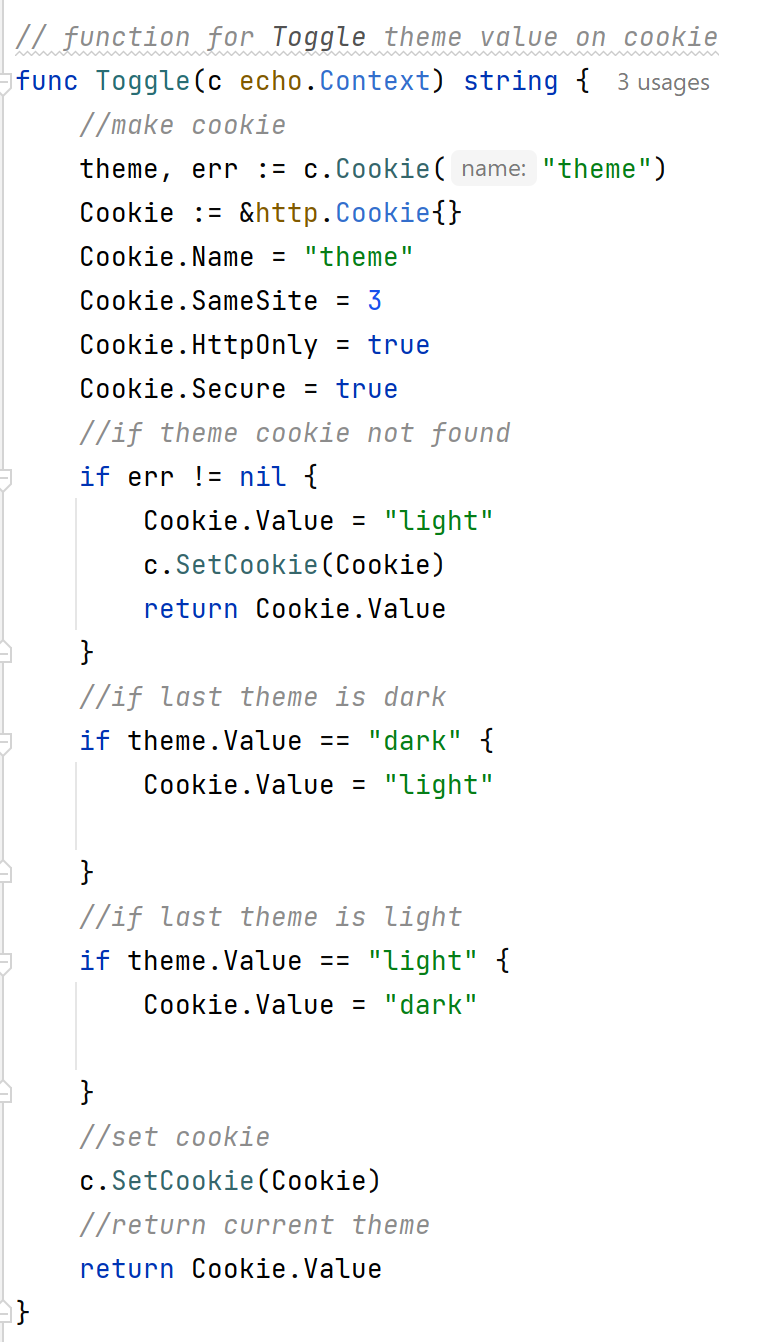


Рисунок 6.11 – Функция изменения цветовой темы сайта

В функции происходит извлечение cookie с именем «theme» и создается новый cookie с противоположным значением. Если cookie не найдено то задаётся светлая тема сайта по умолчанию.

Рассмотрим алгоритм поиска пользователя в базе данных на рисунке 6.12.



Рисунок 6.12 – Функция поиска пользователя в базе данных

Происходит запрос в базу данных для поиска идентификатора пользователя по нику. Возвращаемое значение является идентификатором пользователя. Он необходим для работы остальных запросов SQL связанных с мессенджером.

## 6.3 Описание программных модулей

Для удобства разработки и читабельности кода весь проект был разбит на связанные пакеты. Серверная часть проекта изображена на рисунке 6.13.

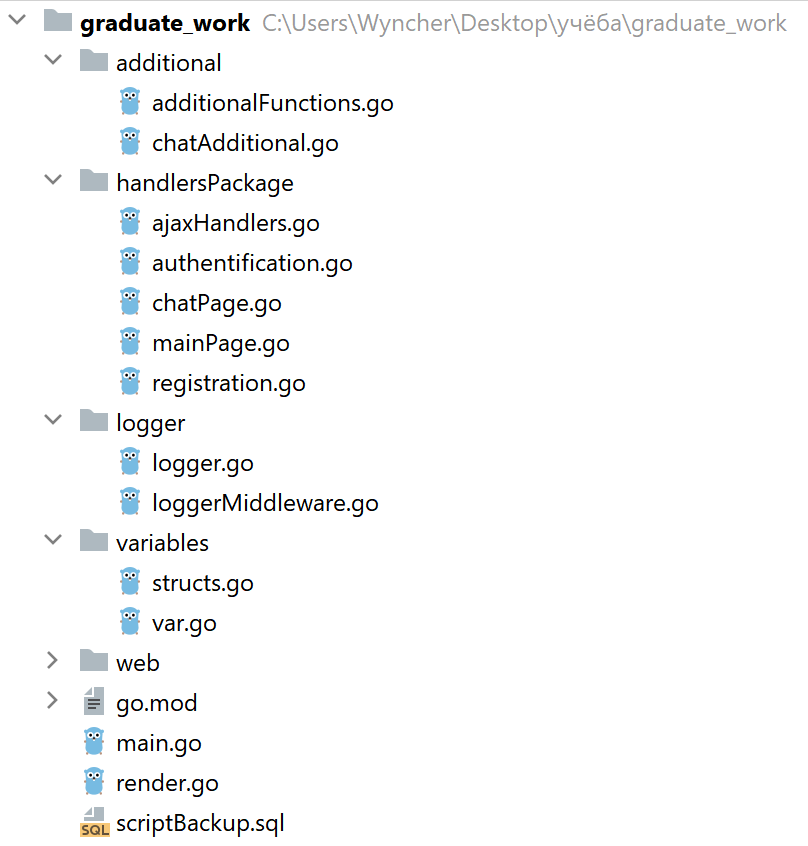


Рисунок 6.11 – Файловая структура серверной части мессенджера

Назначение файлов указано в таблице 6.1

Таблица 6.1 – Файловая структура серверной части

|  |  |  |
| --- | --- | --- |
| Название файла | Назначение файла | Пакет |
| additionalFunctions.go | **Дополнительные функции для обеспечения работы всего сайта в целом** | additional |
| chatAdditonal.go | **Дополнительные функции для обеспечения работы страницы чата** | additional |
| ajaxHandler.go | **Функции обеспечивает ответы на запросы AJAX от клиента.** | handlersPackage |
| authentification.go | Функции обеспечения работы страницы авторизации. | handlersPackage |
| chatPage.go | Функции обеспечения работы страницы чата. | handlersPackage |
| mainPage.go | Функции обеспечения работы главной страницы авторизации. | handlersPackage |
| registration.go | Функции обеспечения работы страницы регистрации. | handlersPackage |
| logger.go | **Общая структура логгера событий.** | logger |
| loggerMiddleware.go | **Реализация отображения событий в консоли сервера.** | logger |
| structs.go | **Файл для всех структур проекта.** | variables |
| var.go | **Важные глобальные переменные сервера.** | variables |
| go.mod | **Метаданные модуля** | - |

Для обеспечения работы клиентской части присутствуют файлы шаблонов и стилей для отображения готовых страниц у пользователя в браузере. Рассмотрим файловую структуру серверной части мессенджера на рисунке 6.12

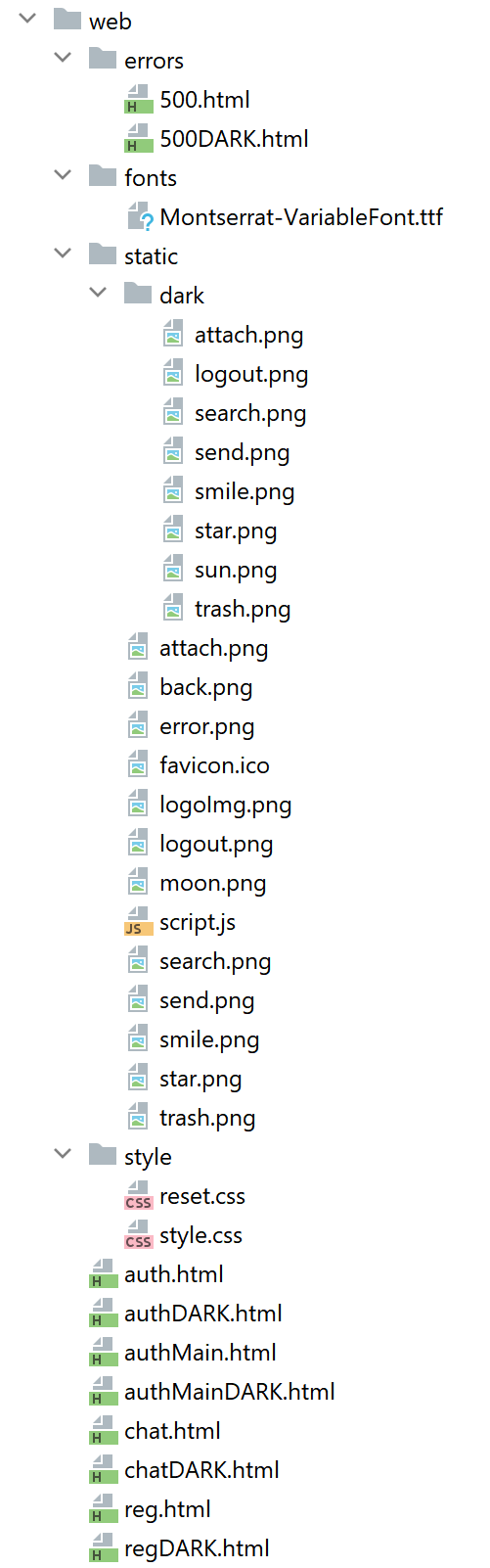


Рисунок 6.11 – Файловая структура клиентской части мессенджера

Назначение файлов указано в таблице 6.1

Таблица 6.1 – Файловая структура серверной части

|  |  |  |
| --- | --- | --- |
| Название файла | Назначение файла | Директория |
| 500.html | **Страница ошибки.** | errors |
| 500DARK.html | **Страница ошибки.** | errors |
| Montserrat-VariableFont.ttf | **Шрифт сайта.** | fonts |
| dark/attach.png | **Кнопка вложения (**тёмная **тема)** | static |
| dark/logout.png | Кнопка выхода (тёмная тема). | static |
| dark/search.png | Кнопка поиска пользователя (тёмная тема). | static |
| dark/smile.png | Кнопка открытия панели смайликов (тёмная тема). | static |
| dark/star.png | Кнопка «избранное» (тёмная тема). | static |
| dark/sun.png | Кнопка смены темы на светлую(тёмная тема). | static |
| dark/trash.png | Кнопка «удаление» (тёмная тема). | static |
| attach.png | **Кнопка вложения (светлая тема)** | static |
| back.png | **Кнопка назад (светлая тема)** | static |
| favicon.ico | **Favicon сайта** | static |
| error.png | Изображение на странице ошибки. | static |
| logoImg.png | Логотип сайта в шапке. | static |
| logout.png | Кнопка выхода (светлая тема). | static |
| moon.png | Кнопка смены темы на тёмную (светлая тема). | static |
| script.js | Файл JavaScript для сайта. | static |
| search.png | Кнопка поиска пользователя (светлая тема). | static |
| send.png | Кнопка отправки сообщения (светлая тема). | static |
| smile.png | Кнопка открытия панели смайликов (светлая тема). | static |
| star.png | Кнопка «избранное» (светлая тема). | static |
| trash.png | Кнопка «удаление» (светлая тема). | static |
| reset.css | Сброс встроенных стилей CSS. | style |
| style.css | Файл стилей CSS. | style |
| auth.html | Шаблон страницы авторизации (светлая тема). | / |
| authDARK.html | Шаблон страницы авторизации (тёмная тема). | / |
| authMain.html | Шаблон главной страницы авторизации (светлая тема). | / |
| authMainDARK.html | Шаблон главной страницы авторизации (тёмная тема). | / |
| chat.html | Шаблон страницы диалога (светлая тема). | / |
| chatDARK.html | Шаблон страницы диалога (тёмная тема). | / |
| reg.html | Шаблон страницы регистрации(светлая тема). | / |
| regDARK.html | Шаблон страницы регистрации(тёмная тема). | / |

Структура клиентской части сайта была разработана с целью предоставления пользователям интуитивно понятного и удобного интерфейса. Все элементы навигации, страницы и функциональные возможности тщательно продуманы, для обеспечения максимально положительного пользовательского опыта.

Структура сервера была спроектирована с учётом требований высокой производительности, безопасности и масштабируемости. Каждый компонент системы был тщательно выбран и настроен для обеспечения надёжной и эффективной работы.

# 7 ОПИСАНИЕ ИНТЕРФЕЙСА

Страница «Главная» (см. рис.7.2) открывается при переходе на любую страницу неавторизированным пользователем.



Рисунок 7.1 – Экранная форма страницы «Главная»

**Элементы интерфейса:**

* **Логотип**;
* **Меню:**

1. **«ВХОД»:** переход на страницу авторизации для зарегистрированных пользователей;
2. «РЕГИСТРАЦИЯ»**:** переход на страницу регистрации для новых пользователей;

* Луна/солнце : кнопка смены цветовой темы;

#### Страница «Авторизация» (см. рис.7.2) открывается по нажатию на кнопку «ВХОД» на странице «Главная».

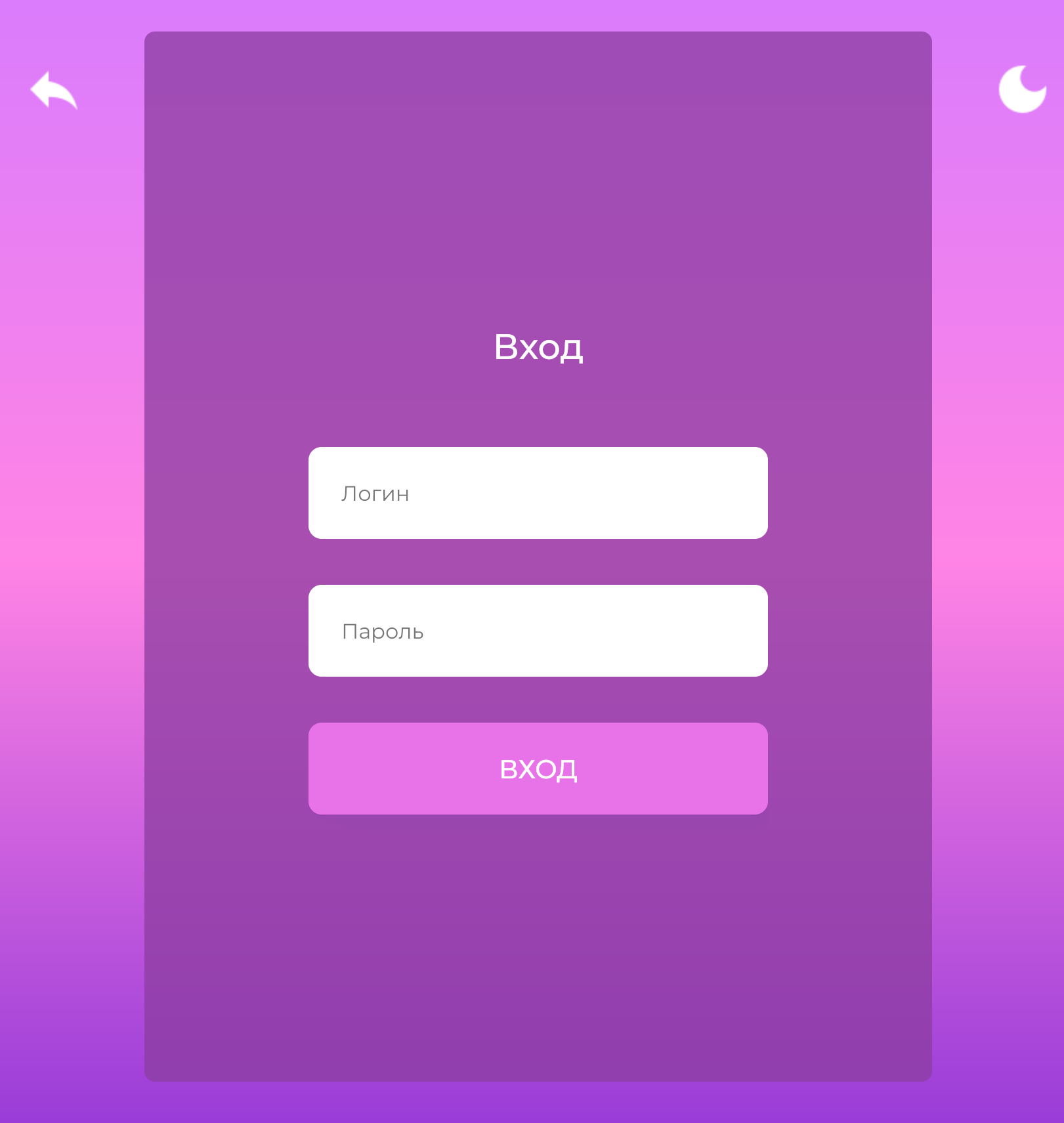


Рисунок 7.2 – Экранная форма страницы «Авторизация»

**Элементы интерфейса:**

* **заголовок**;
* **форма авторизации:**

1. **«Логин»:** поле для ввода логина пользователя;
2. «Пароль»**:** поле для ввода пароля пользователя;
3. «ВХОД»: кнопка отправки формы авторизации;

* Луна/солнце : кнопка смены цветовой темы;
* кнопка возврата на страницу «Главная»;

#### Страница «Регистрация» (см. рис.7.3) открывается по нажатию на кнопку «РЕГИСТРАЦИЯ» на странице «Главная».

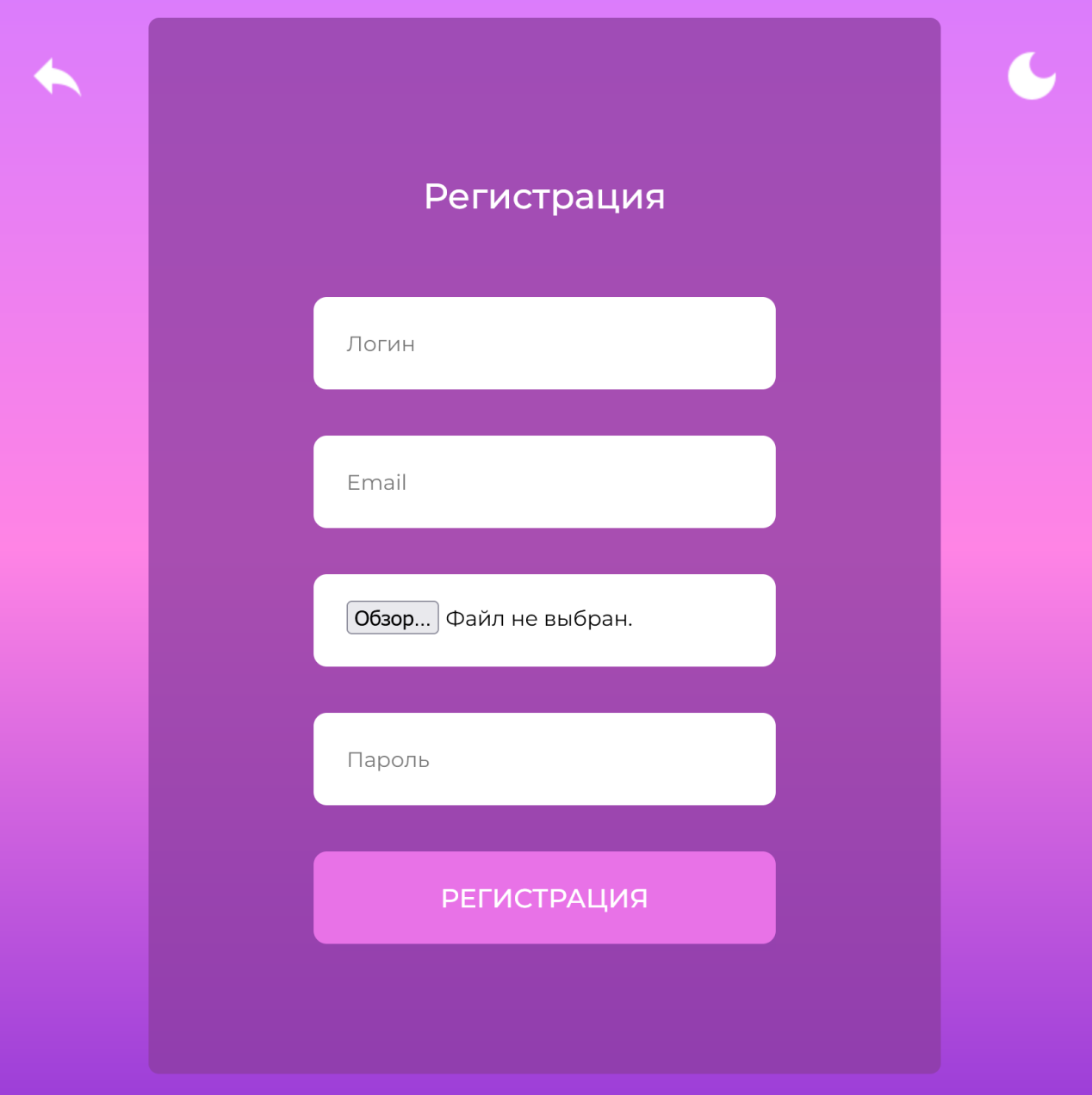


Рисунок 7.3 – Экранная форма страницы «Регистрация»

**Элементы интерфейса:**

* **заголовок**;
* **форма авторизации:**

1. **«Логин»:** поле для ввода логина пользователя;
2. «Email»: поле для ввода электронной почты пользователя;
3. Форма выбора файла логотипа;
4. «Пароль»**:** поле для ввода пароля пользователя;
5. «РЕГИСТРАЦИЯ»: кнопка отправки формы авторизации;

* Луна/солнце : кнопка смены цветовой темы;
* кнопка возврата на страницу «Главная»;

#### Страница «Чат» (см. рис.7.4) открывается после успешной авторизации пользователя в системе.

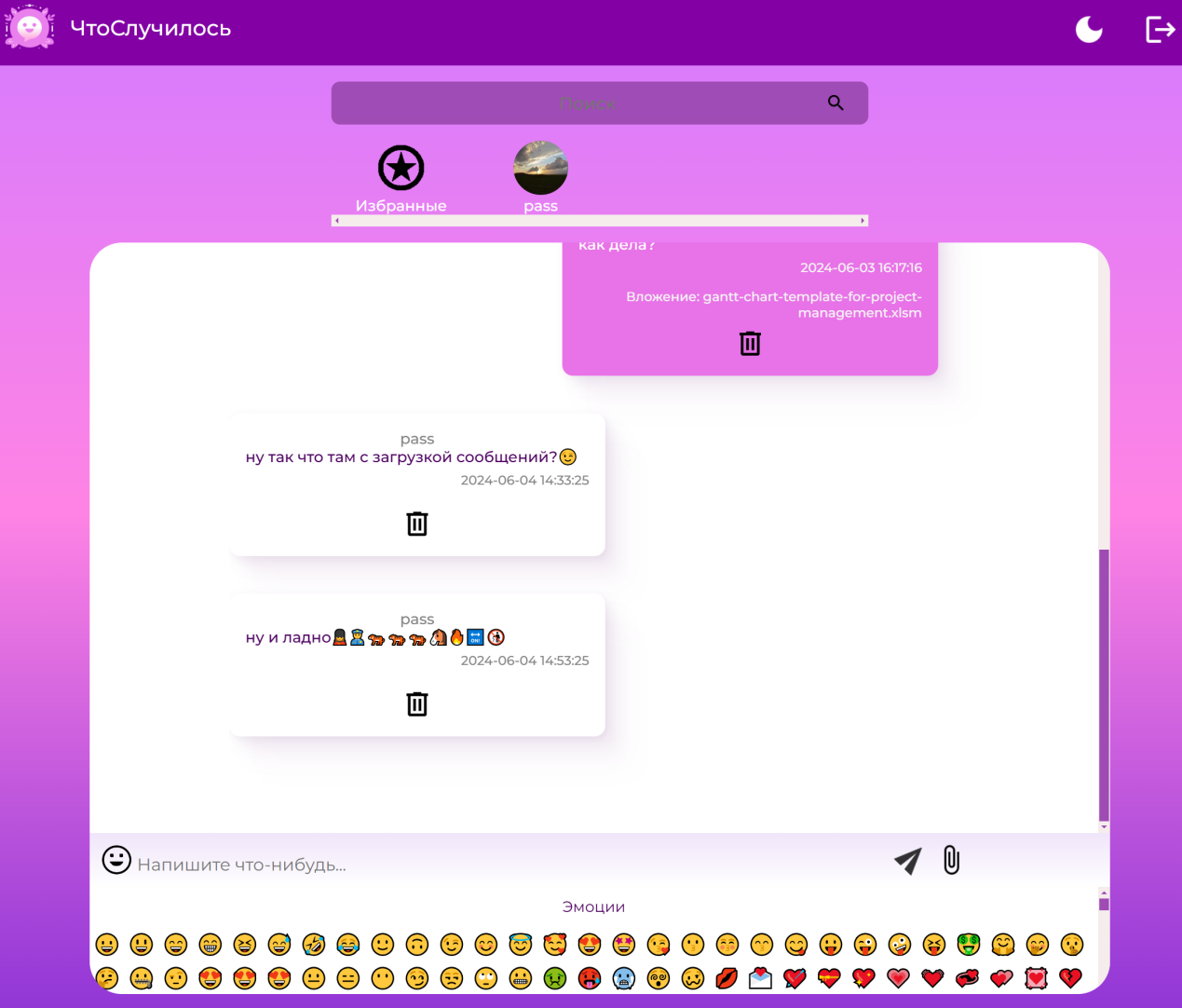


Рисунок 7.4 – Экранная форма страницы «Чат»

**Элементы интерфейса:**

* **логотип**;
* Луна/солнце : кнопка смены темы сайта;
* **кнопка выхода из аккаунта;**
* **форма поиска пользователя:**

1. **«Поиск»:** поле для ввода никнейма пользователя;
2. лупа: поиск пользователей;
3. контейнер с пользователями;
4. звезда: переход в избранные сообщения;
5. логотип пользователя: переход в диалог с пользователем;

* **форма чата:**

1. **Список сообщений диалога;**
2. «Напишите что-нибудь...»: поле для ввода текста сообщения;
3. скрепка: форма выбора файла вложения;
4. самолёт: кнопка отправки сообщения;
5. смайлик: открытие списка смайликов;
6. корзина: удаление сообщения;
7. звезда: добавление в избранное сообщения;

* кнопка смены цветовой темы;
* кнопка возврата на страницу «Главная»;

Описание функциональности**:**

* Авторизация: Пользователь вводит email и пароль, система проверяет данные и предоставляет доступ к чату;
* Регистрация: Новый пользователь вводит необходимые данные, система создает новый аккаунт;
* Чат: Пользователь может выбирать контакты, отправлять и получать сообщения в режиме реального времени, добавлять и просматривать избранные сообщения;

Таким образом этот интерфейс обеспечивает простой и интуитивно понятный пользовательский опыт, предоставляя основные функции мессенджера.

# 8 ТЕСТИРОВАНИЕ

Проводить тестирование предполагается методом чёрного ящика.

Тестирование методом чёрного ящика - это основной метод, используемый специалистами по контролю качества для проверки цифровых продуктов, которые недоступны и внутренняя структура которых неизвестна.

Основная цель этой процедуры - последовательно исследовать, соответствует ли поведение программы текущим требованиям. Работоспособность изделия также проверяется в критических ситуациях, то есть при вводе неверных входных данных.

Тестирование страницы «Авторизация»:

1. Проверка отображения страницы авторизации(см. рис. 8.1):

- Открыть страницу авторизации.

- Убедиться, что все элементы отображаются корректно.

- Предполагаемый результат: отображение страницы авторизации (поле email, поле ввода пароля, кнопка «Войти», ссылки «Назад» и «Тема»).

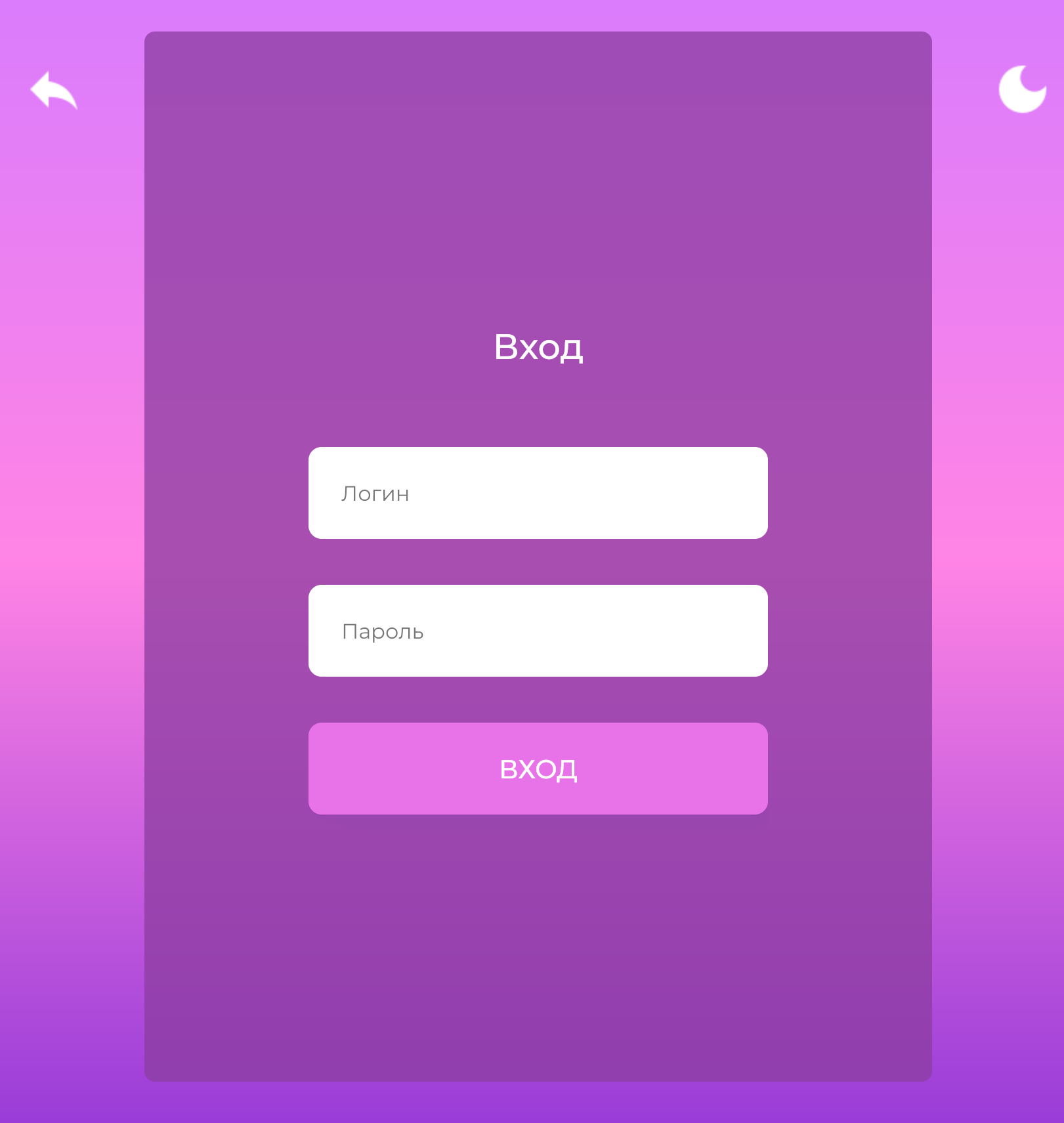


Рисунок 8.1 – Проверка отображения страницы авторизации

- Полученный результат: отображение страницы авторизации со всеми элементами.

2. Валидация поля email (см.рис. 8.2):

- Ввести некорректный email (например, "user" или "user@domain").

- Проверить, что отображается сообщение об ошибке.

- Предполагаемый результат: отображение страницы ошибки или иного уведомления об ошибке или перезагрузка страницы..

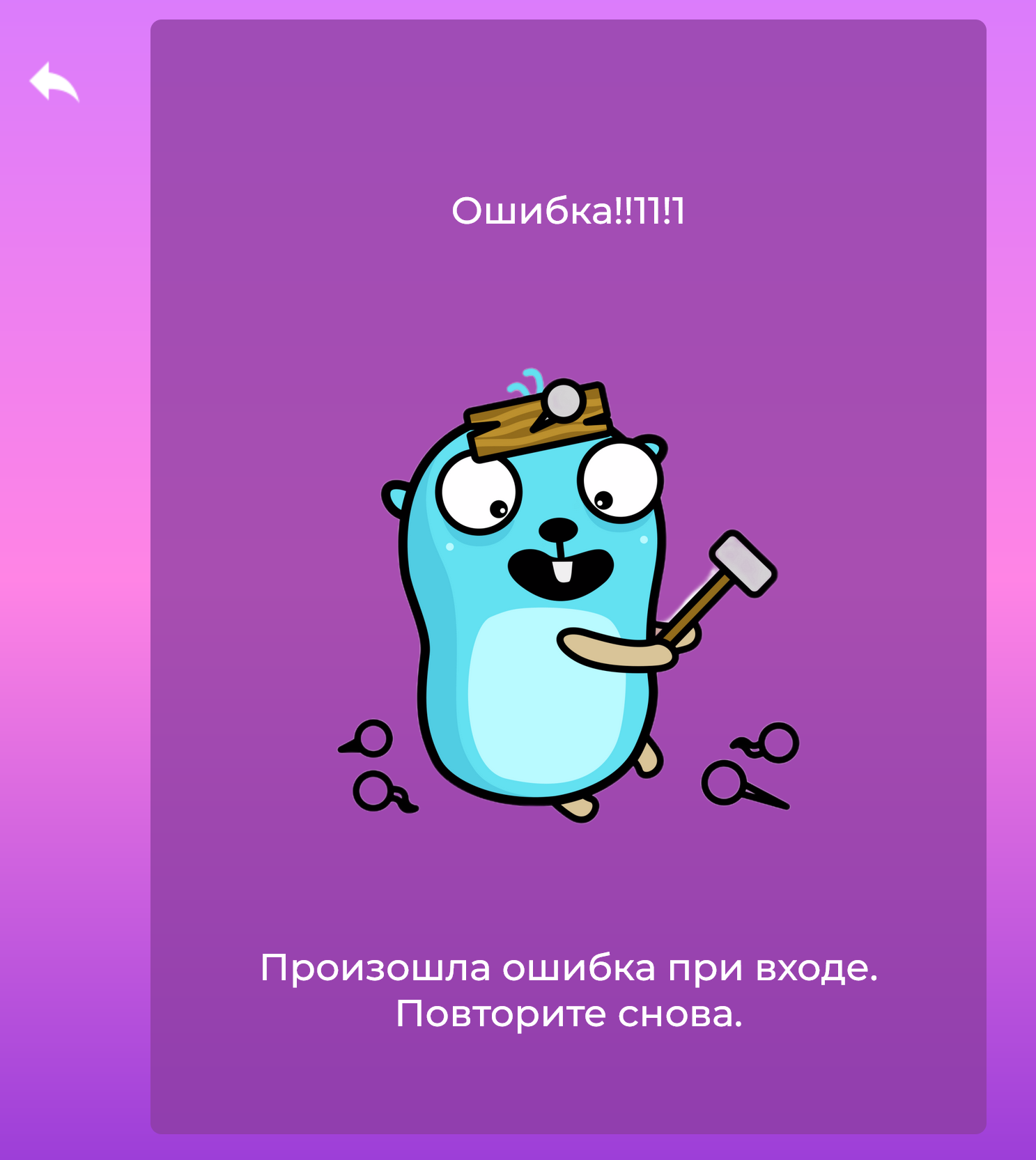


Рисунок 8.2 – Валидация поля email

- Полученный результат: отображение страницы ошибки

3. Валидация поля пароля (см.рис. 8.3):

- Оставить поле пароля пустым.

- Предполагаемый результат: отображение страницы ошибки или иного уведомления об ошибке или перезагрузка страницы..

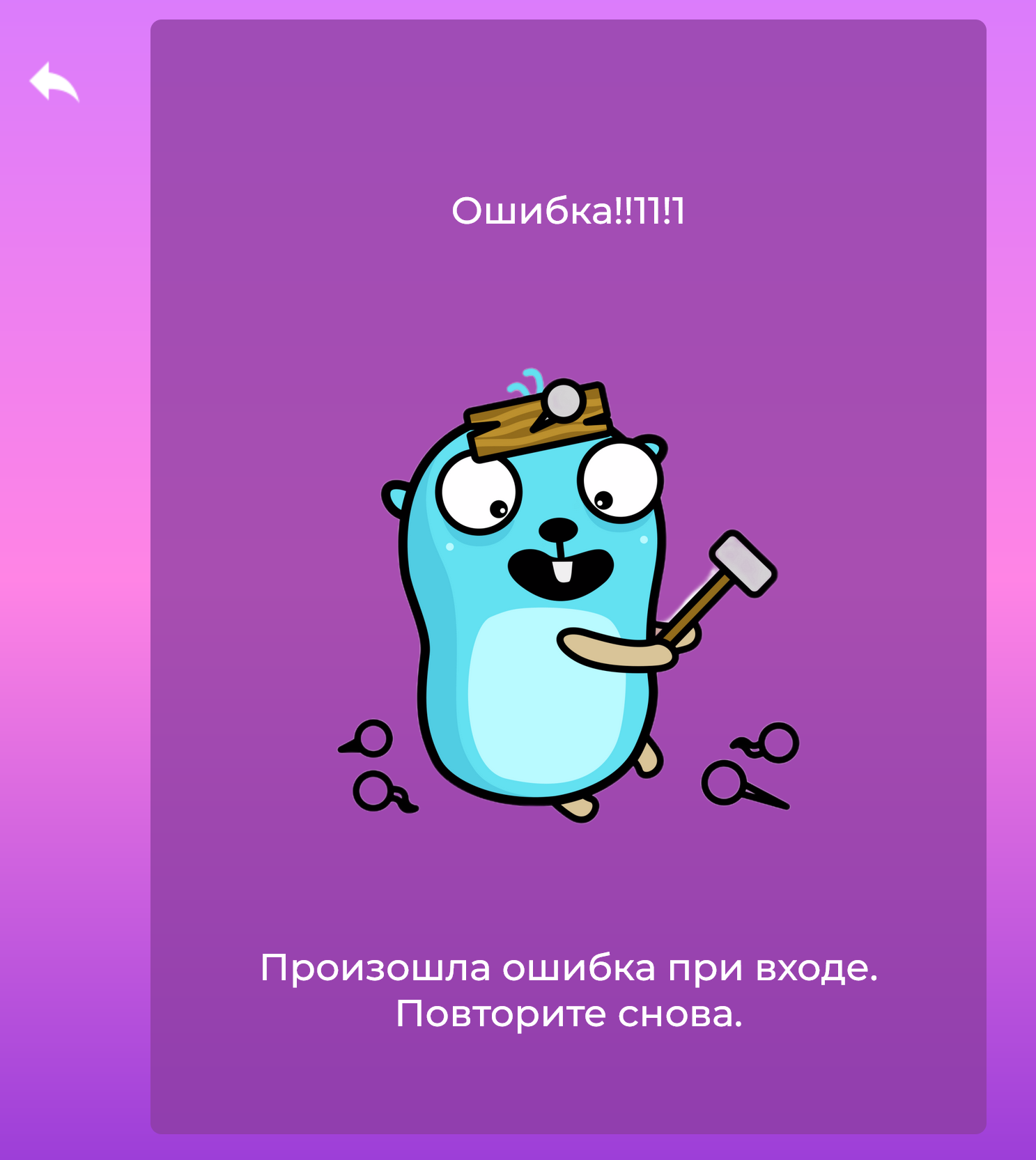


Рисунок 8.3 – Валидация поля пароля

- Полученный результат: отображение страницы ошибки

4. Авторизация с неверными данными (см.рис. 8.4):

- Ввести несуществующий email и любой пароль и нажать на кнопку «ВХОД».

- Предполагаемый результат: отображение страницы ошибки или иного уведомления об ошибке или перезагрузка страницы..

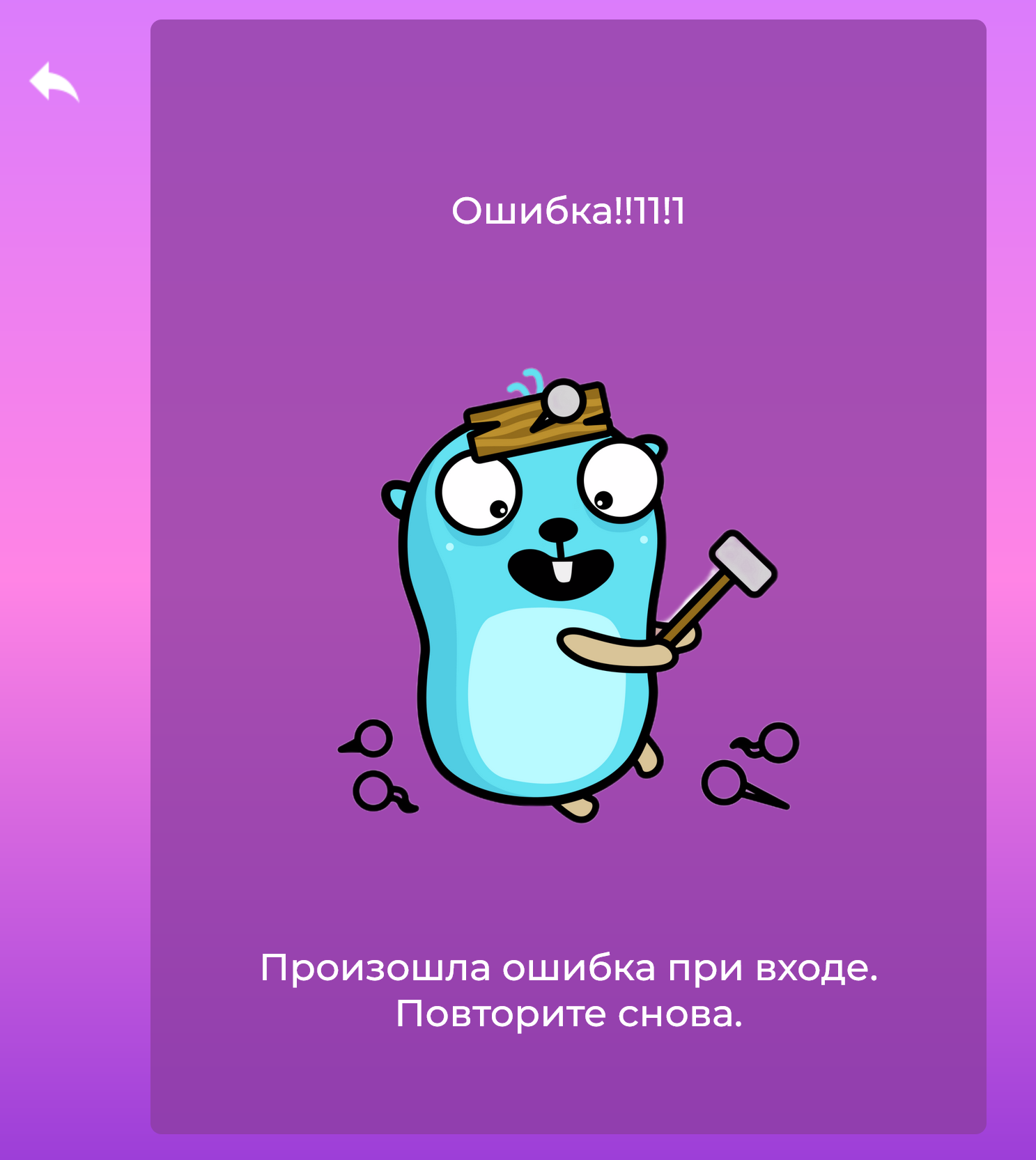


Рисунок 8.4 – Авторизация с неверными данными

- Полученный результат: отображение страницы ошибки

5. Авторизация с корректными данными (см.рис. 8.5):

- Ввести существующий email и правильный пароль и нажать на кнопку «ВХОД».

- Предполагаемый результат: отображение страницы чата с присутствующими элементами(список сообщений,список контактов, кнопка смены темы,кнопка выхода из аккаунта, кнопка открытия списка смайликов, кнопка отправки, кнопка добавления вложения).

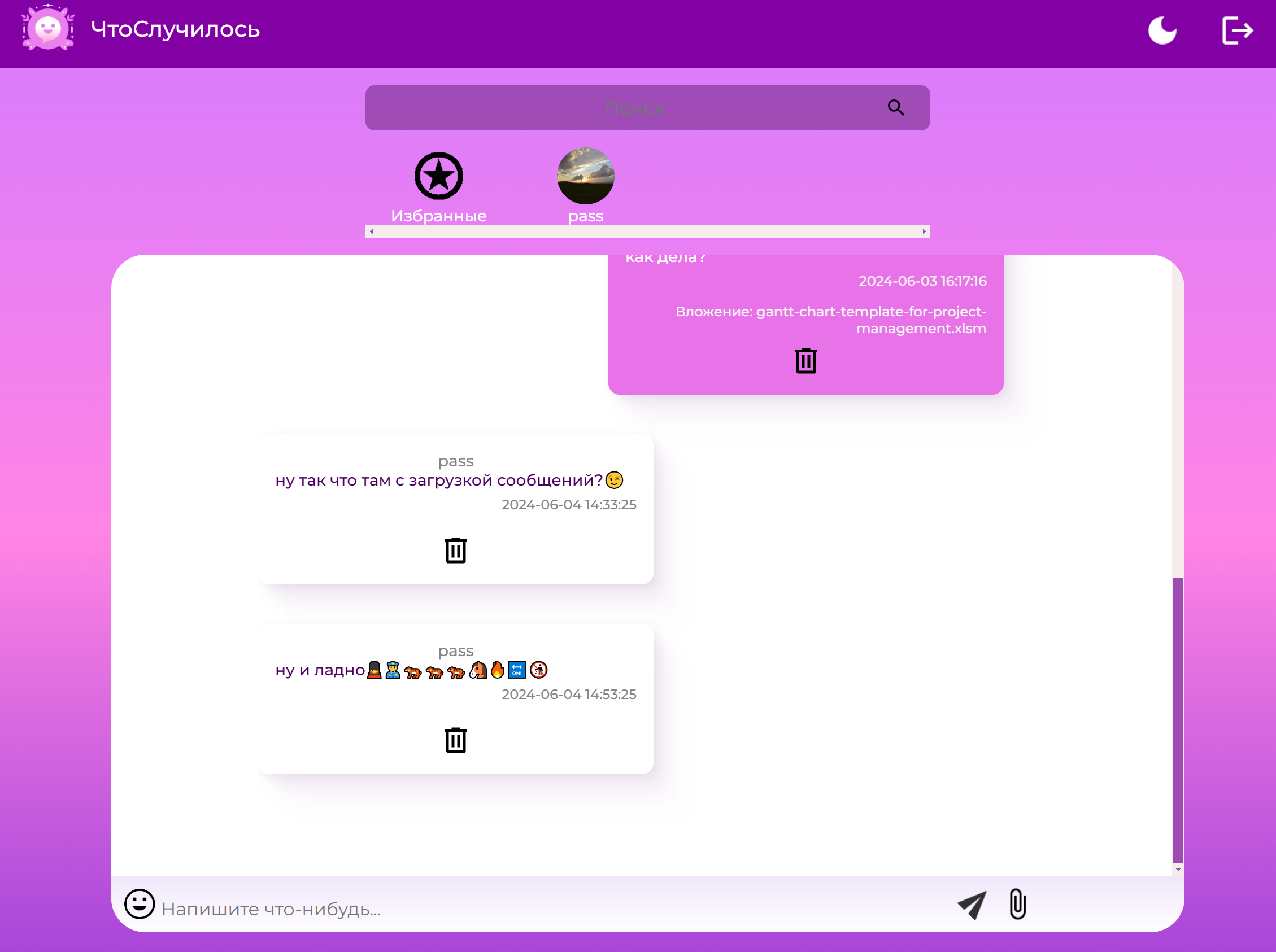


Рисунок 8.5 – Авторизация с корректными данными

- Полученный результат: отображение страницы чата со всеми элементами.

7. Ссылка «Регистрация» (см.рис. 8.6):

- Нажать на ссылку «Регистрация».

- Проверить, что происходит переход на страницу регистрации.

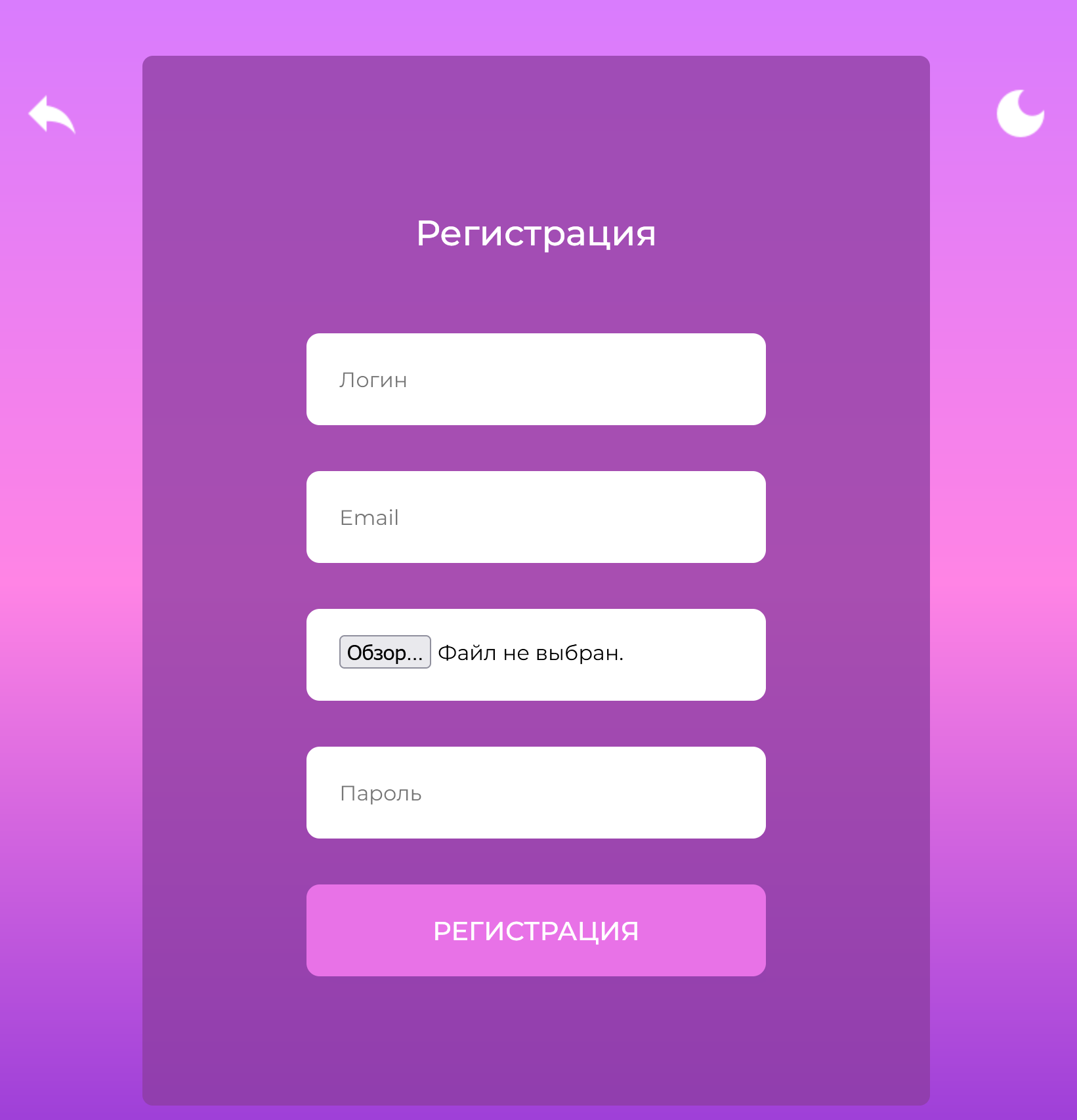


Рисунок 8.6 – Ссылка «Регистрация»

- Полученный результат: отображение страницы регистрации со всеми элементами.

Тестирование страницы «Регистрация»:

1. Проверка отображения страницы регистрации (см.рис. 8.7):
2. - Открыть страницу регистрации.

- Предполагаемый результат: отображение страницы авторизации (поле имени, поле email, поле пароля, кнопка смены темы, кнопка «назад»).

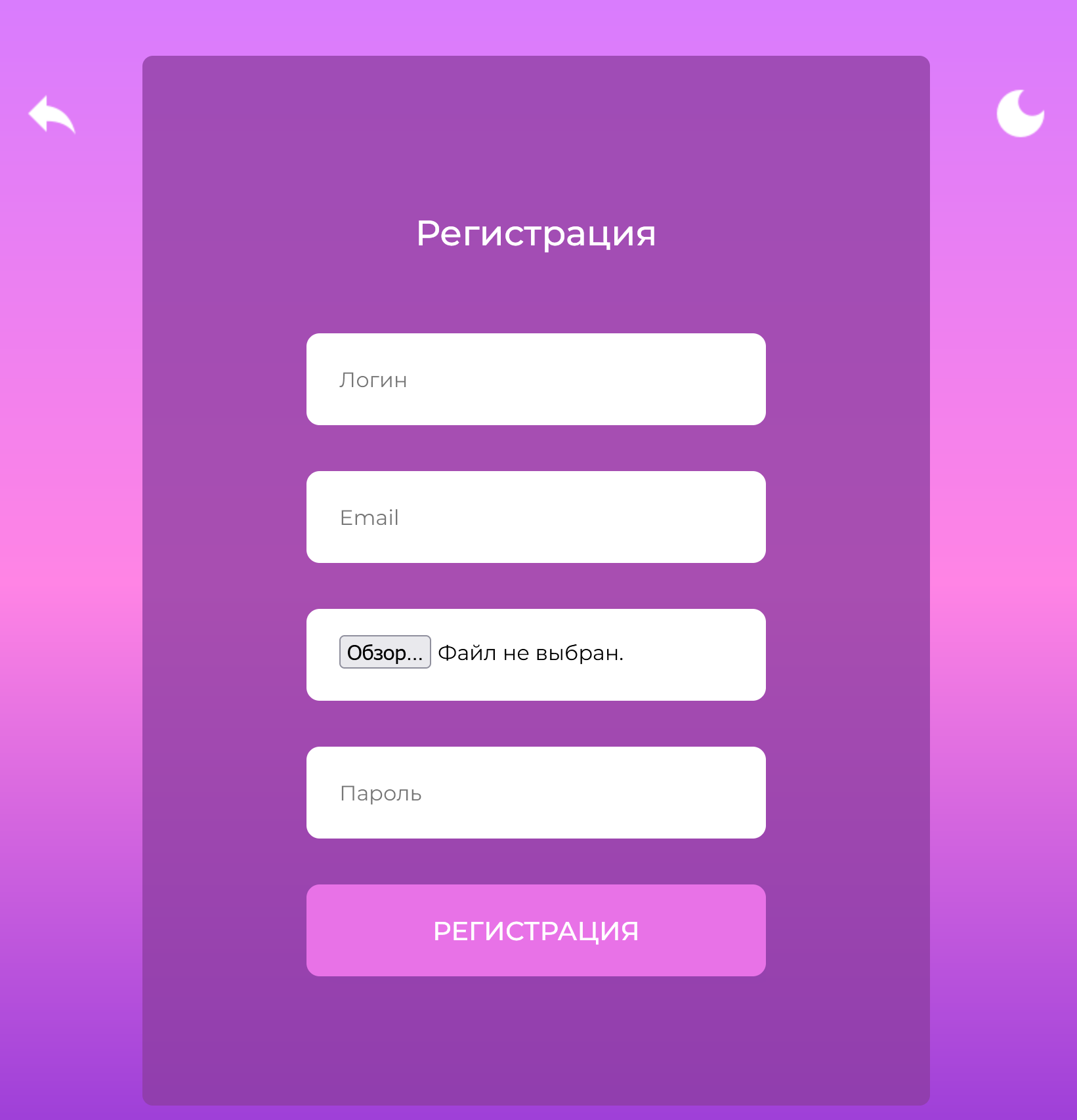


Рисунок 8.7 – Проверка отображения страницы регистрации

- Полученный результат: отображение страницы регистрации со всеми элементами.

2. Валидация поля email (см.рис. 8.8):

- Ввести некорректный email.

- Предполагаемый результат: отображение страницы ошибки или иного уведомления об ошибке или перезагрузка страницы.

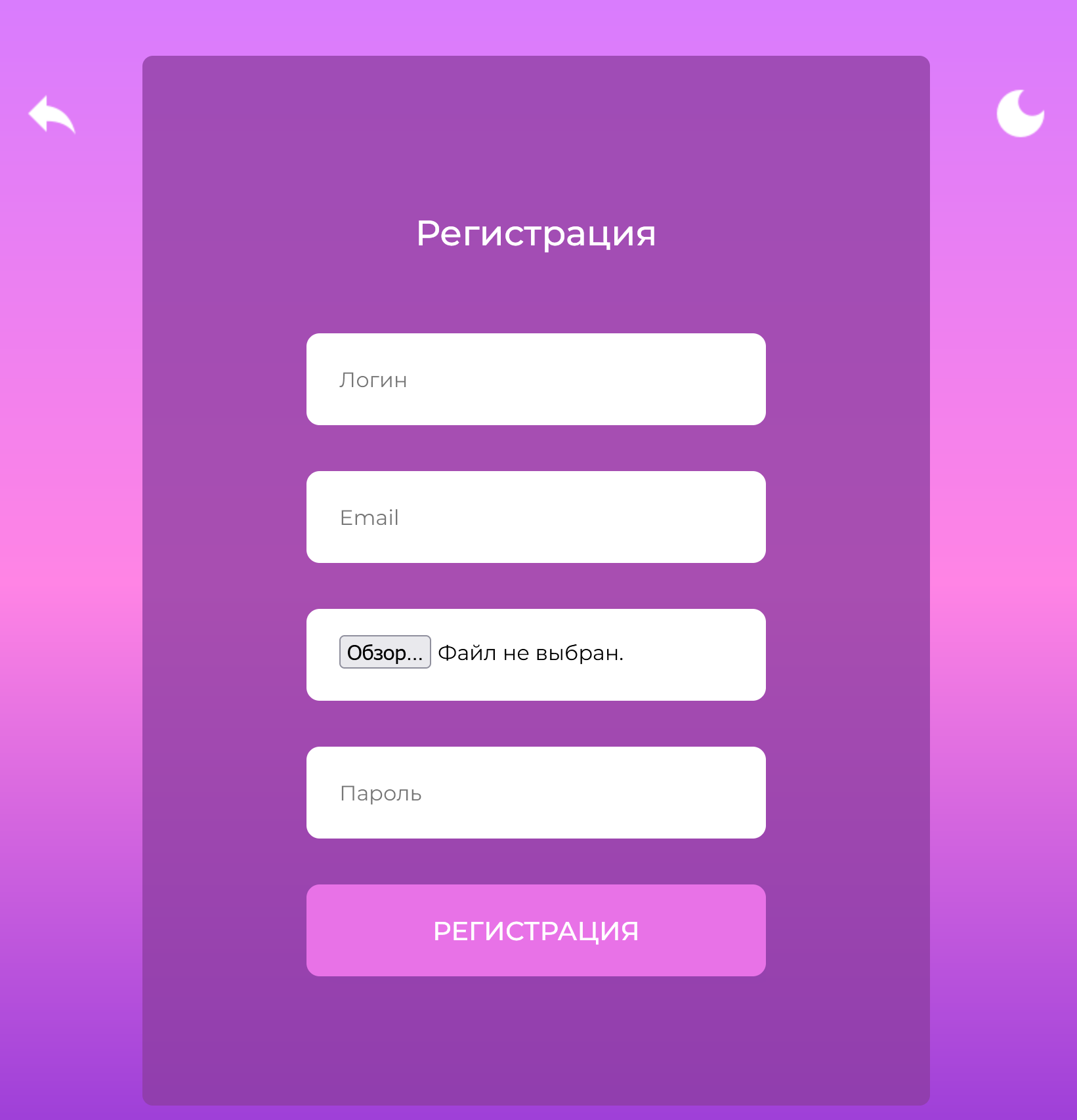


Рисунок 8.8 – Валидация поля email

- Полученный результат: очистка полей для ввода.

3. Валидация поля пароля (см.рис. 8.9):

- Ввести пустой пароль.

- Предполагаемый результат: отображение страницы ошибки или иного уведомления об ошибке или перезагрузка страницы.

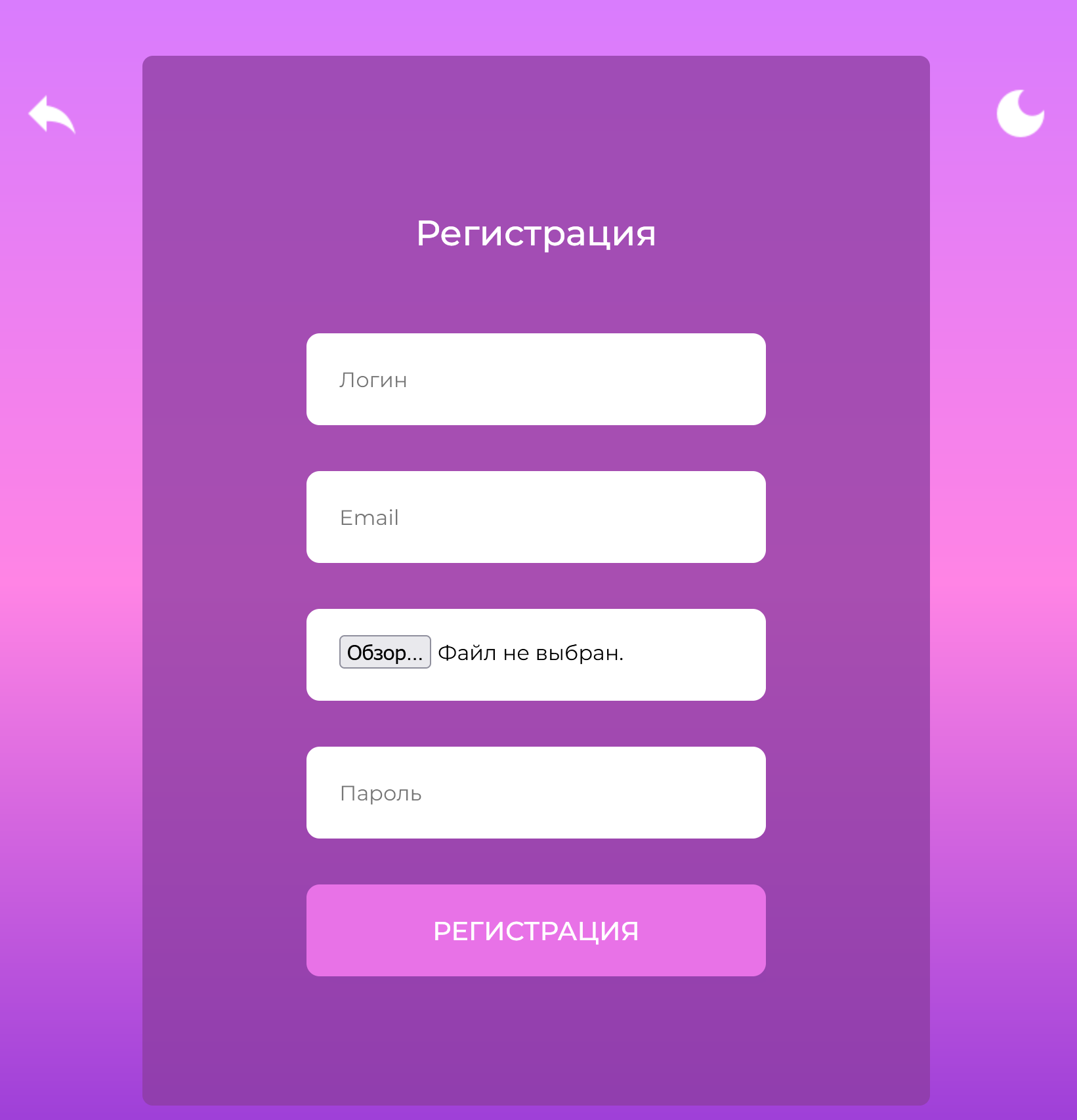


Рисунок 8.3 – Валидация поля пароля

- Полученный результат: очистка полей для ввода.

1. Регистрация без логотипа (см.рис. 8.10):

- Не добавляя файл нажать «РЕГИСТРАЦИЯ».

- Предполагаемый результат: отображение страницы ошибки или иного уведомления об ошибке или перезагрузка страницы.

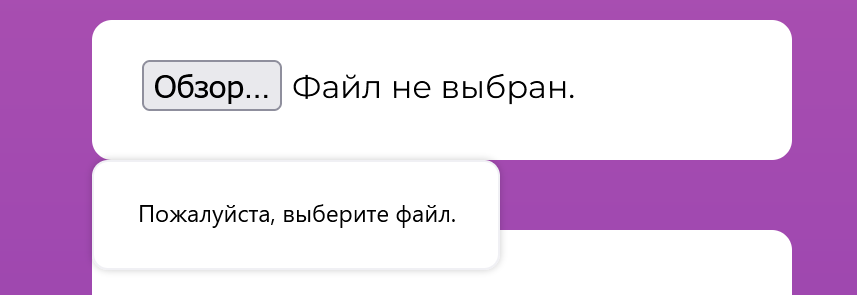


Рисунок 8.10 – Регистрация без логотипа

- Полученный результат: окно с ошибкой.

6. Регистрация с корректными данными (см.рис. 8.11):

- Ввести валидные данные.

- Предполагаемый результат: отображение страницы чата с присутствующими элементами(список сообщений,список контактов, кнопка смены темы,кнопка выхода из аккаунта, кнопка открытия списка смайликов, кнопка отправки, кнопка добавления вложения).

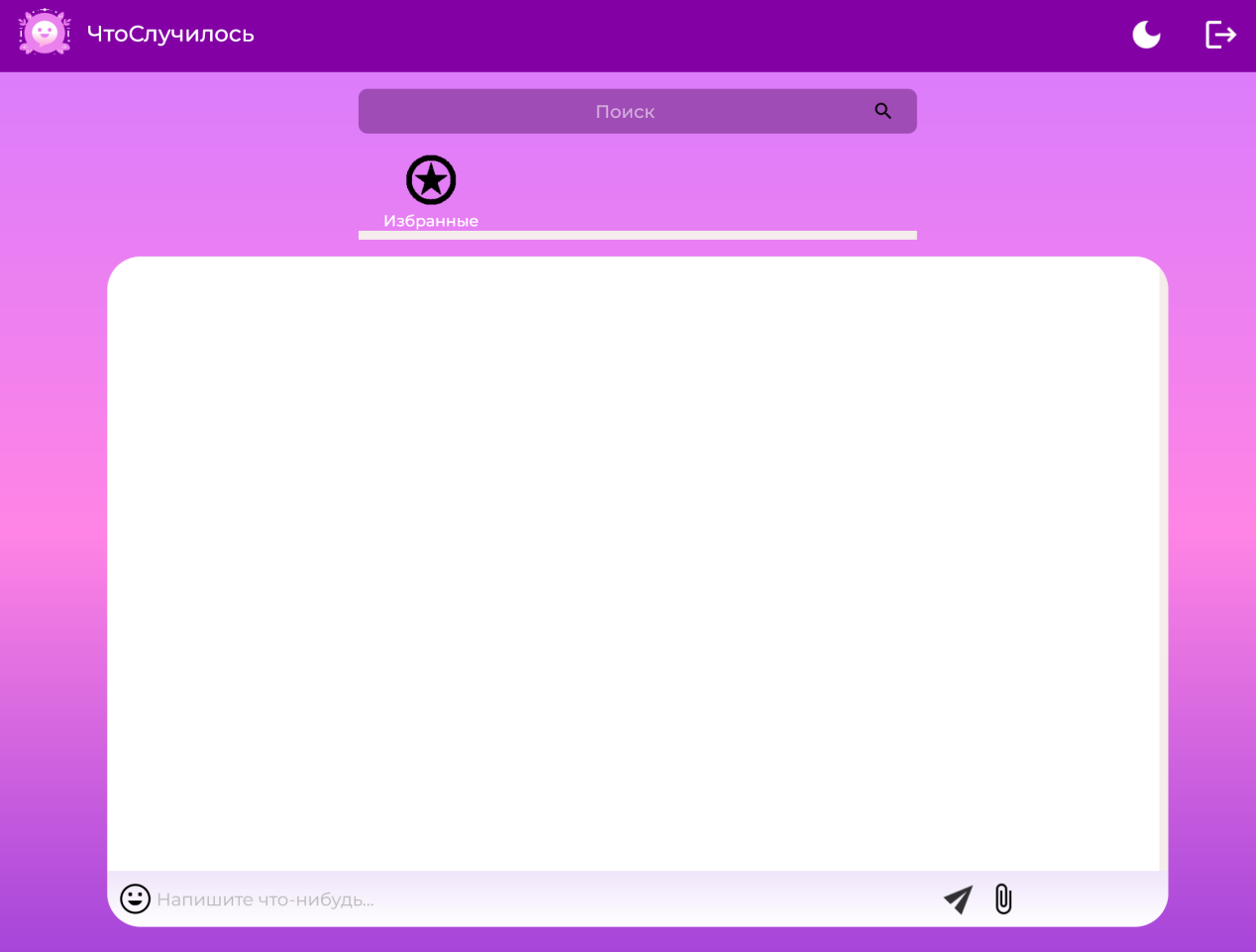


Рисунок 8.11 – Регистрация с корректными данными

- Полученный результат: отображение страницы чата со всеми элементами.

Тестирование страницы «Чат»:

1. Проверка отображения страницы чата (см.рис. 8.12):

- Переход на страницу чата авторизированному пользователю.

- Предполагаемый результат: отображение страницы чата с присутствующими элементами(список сообщений,список контактов, кнопка смены темы,кнопка выхода из аккаунта, кнопка открытия списка смайликов, кнопка отправки, кнопка добавления вложения).

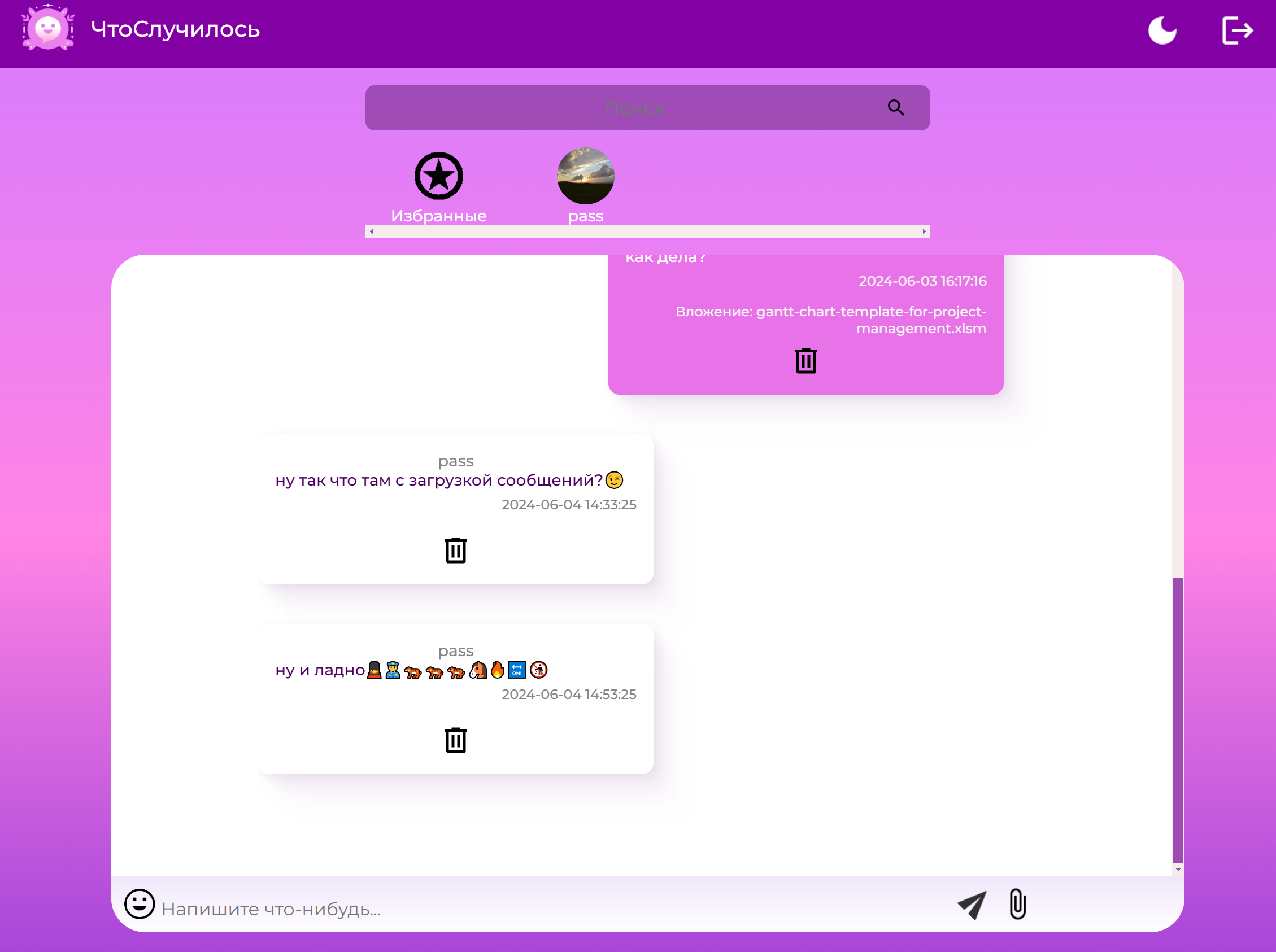


Рисунок 8.12 – Проверка отображения страницы чата

- Полученный результат: отображение страницы чата со всеми элементами.

2. Отправка сообщения (см.рис. 8.14):

- Ввести текст в поле ввода сообщения.

- Нажать кнопку отправки сообщения.

- Предполагаемый результат: отображение страницы чата с присутствующим новым сообщением(временем отправки,кнопок добавления в избранное, удаления сообщения, текста сообщения, ссылкой на вложение).

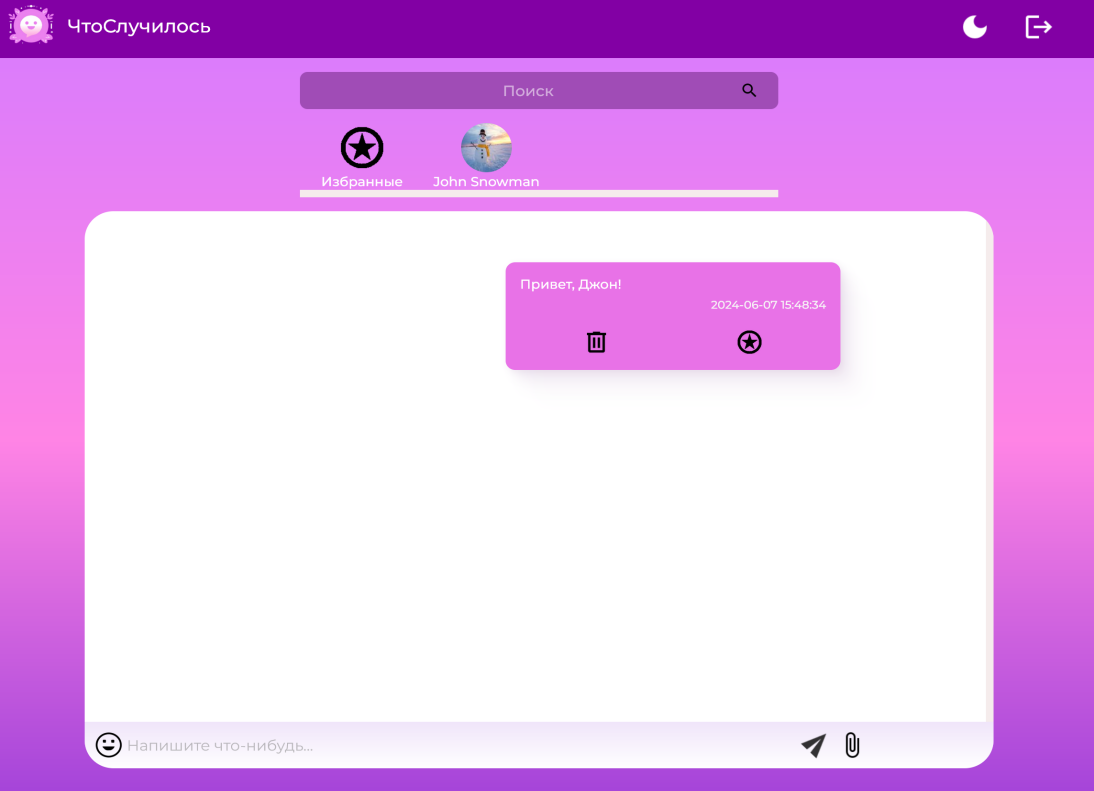


Рисунок 8.14 – Отправка сообщения

- Полученный результат: отображение нового сообщения со всеми элементами.

3. Получение сообщения (см.рис. 8.15):

- Отправить сообщение из другого аккаунта.

- Предполагаемый результат: отображение страницы чата с присутствующим новым сообщением(временем отправки,кнопок добавления в избранное, удаления сообщения, текста сообщения, ссылкой на вложение) без перезагрузки.

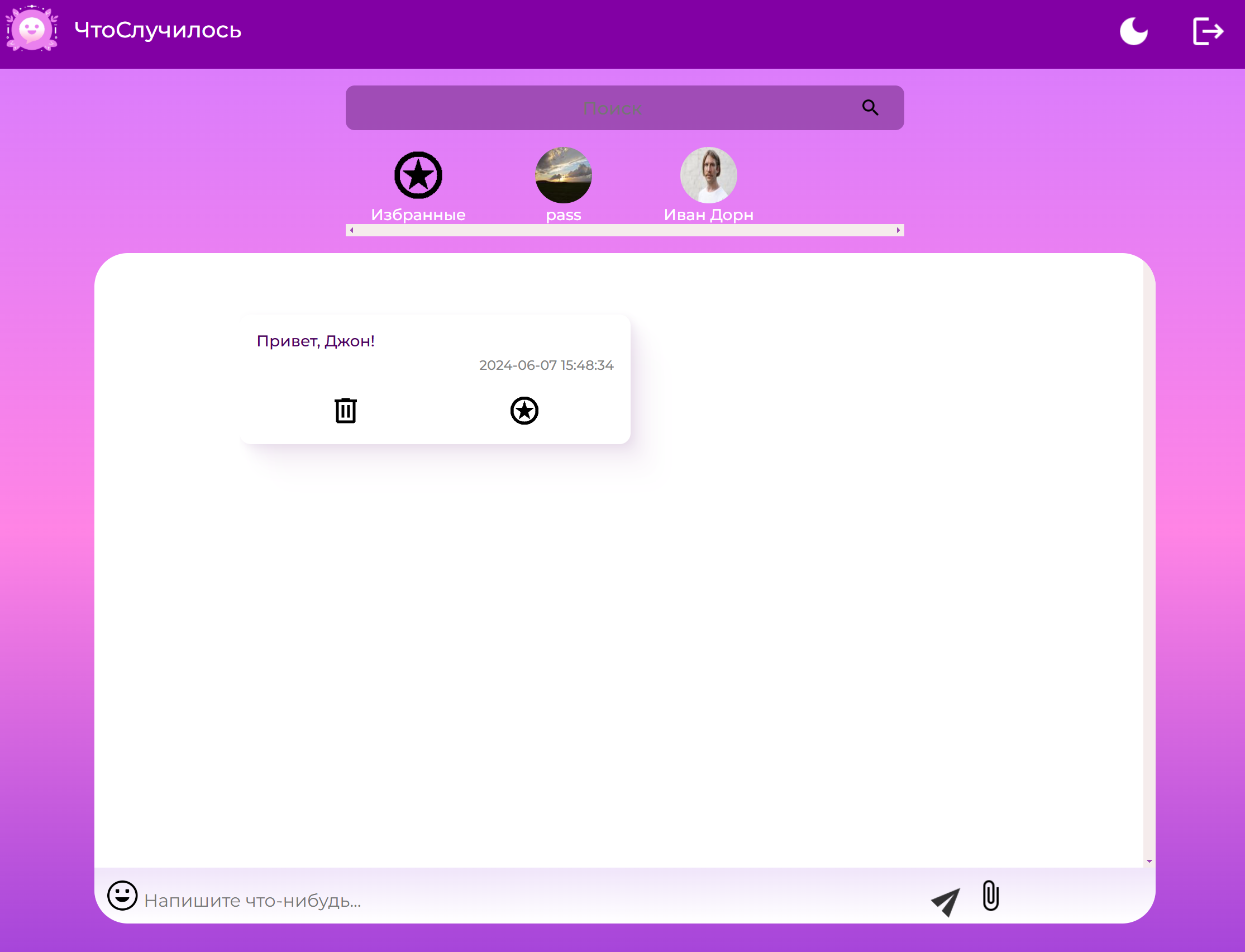


Рисунок 8.15 – Получение сообщения

- Полученный результат: отображение нового сообщения со всеми элементами без перезагрузки.

4. Отправка пустого сообщения (см.рис. 8.16):

- Оставить поле ввода пустым и нажать кнопку отправки.

- Предполагаемый результат: отображение страницы чата без нового сообщения или сообщение об ошибке.

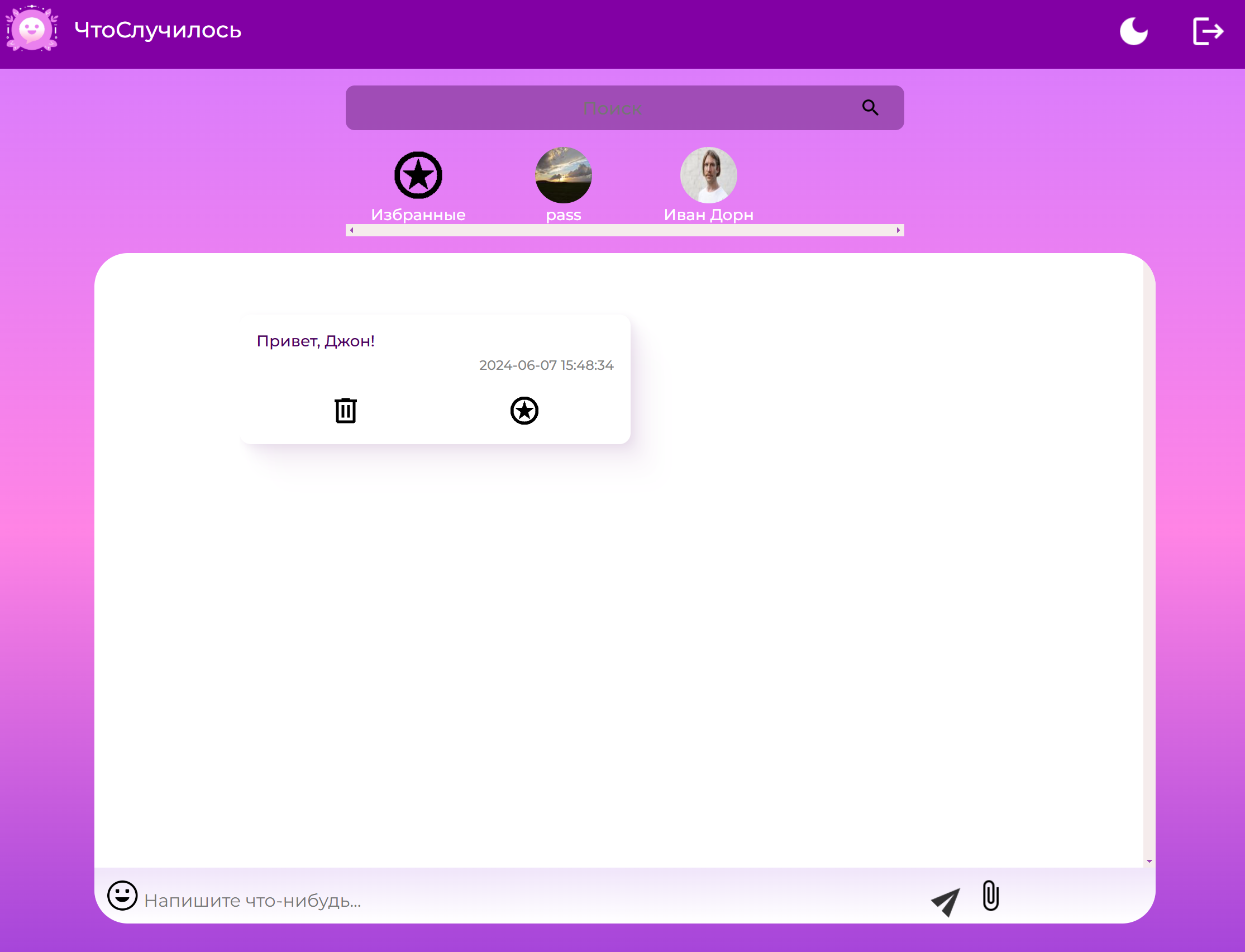


Рисунок 8.16 – Отправка пустого сообщения

- Полученный результат: ничего не изменилось.

5. Переключение между контактами (см.рис. 8.17):

- Нажать на разные контакты в списке.

- Предполагаемый результат: отображение страницы чата с другими сообщениями с диалога выбранных пользователей.



Рисунок 8.17 – Переключение между контактами

- Полученный результат: после перезагрузки страницы сообщения в чате изменились на сообщения из диалога с другим пользователем.

6. Проверка меток времени сообщений (см.рис. 8.18):

- Предполагаемый результат: отображение времени отправки сообщений отображаются в полном объёме ( год, день, час, минута).

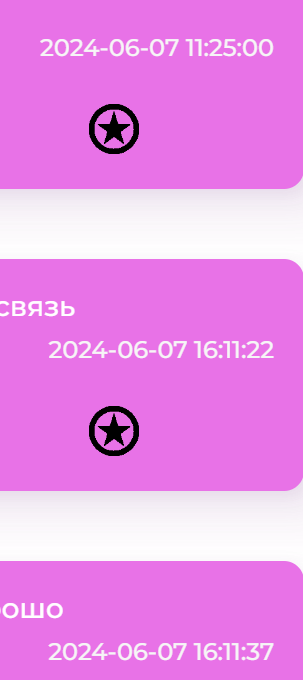


Рисунок 8.18 – Проверка меток времени сообщений

- Полученный результат: Сообщения имеют подпись со временем отправки( год, день, час, минута, секунда ).

1. Проверка отображения непрочитанных сообщения (см.рис. 8.19):

- отправить сообщение из аккаунта А аккаунту Б. У аккаунта Б в диалоге с другим пользователем должно отобразится уведомление о непрочитанном сообщении.

- Предполагаемый результат: отображение уведомления о новом сообщении.



Рисунок 8.19 – Проверка уведомления о новых сообщениях

- Полученный результат: Возле контакта появляется счётчик непрочитанных сообщений.

Тестирование методом чёрного ящика чат-мессенджера включает проверку функциональных элементов на каждой странице, а также валидацию ввода данных и корректность отображения сообщений. По результатам тестирования программное обеспечение обеспечивает должную стабильность, логика отображения экранных форм соблюдена.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была достигнута цель создания чат-мессенджера с использованием языка программирования Go и фреймворка Echo. Разработанное приложение демонстрирует высокую производительность, масштабируемость и надёжность, что подтверждается результатами тестирования.

Основные достижения проекта включают:

1. Разработка серверной части: Создано серверное приложение на Go, обеспечивающее обработку запросов, управление соединениями пользователей и маршрутизацию сообщений. Использование фреймворка Echo позволило существенно ускорить процесс разработки благодаря его простоте и гибкости.
2. Обеспечение безопасности: Реализованы механизмы аутентификации и авторизации пользователей, а также меры по защите данных и предотвращению возможных атак. Использование JWT-токенов и MD5 хеширования обеспечивает высокий уровень безопасности обмена данными.
3. Интеграция с базой данных: В проекте использована база данных MySQL для хранения пользовательской информации и сообщений. Это решение обеспечило надёжное и эффективное управление данными.
4. Тестирование и отладка: Проведены обширные тестирования и отладка приложения, включающие ручное тестирование исключительных ситуаций. Это позволило выявить и устранить возможные ошибки и узкие места в производительности и логике ПО.

Проект "Многопользовательский чат-мессенджер «Что случилось»" является примером успешной реализации современных технологий и практик в области разработки веб-приложений. Полученные результаты подтверждают жизнеспособность выбранных решений и технологий, а также демонстрируют потенциал для дальнейшего развития и расширения функциональности приложения.

# СПИСОК ЛИТЕРАТУРЫ

1. [Мессенджер – что это такое, для чего он нужен](https://elama.ru/glossary/messendzher" \o "elama.ru — Мессенджер – что это такое, для чего он нужен, список популярных мессенджеров | Словарь по маркетингу и интернет-рекламе" \t "https://yandex.ru/_blank) – URL: https://elama.ru/glossary/messendzher

2. WhatsApp: история самого популярного мессенджера в мире – URL: https://dzen.ru/a/ZEDARmuMHklH3UfL

3. MySQL: что это, как устроена, чем хороша и как начать с ней работать – URL: https://skillbox.ru/media/code/mysql-chto-eto-kak-ustroena-chem-khorosha-i-kak-nachat-s-ney-rabotat/

4. HTML или CMS – что лучше для коммерческого сайта [Электронный ресурс] – Режим доступа: https://vzh.ru/article/html-ili-cms-chto-luchshe-dlya-kommercheskogo-sajta/ - Загл. с экрана.

5. CSS — Что такое, история создания и преимущества [Электронный ресурс] – Режим доступа: http://proglang.su/css/introduction - Загл. с экрана.

6. Русская документация по API jQuery [Электронный ресурс] – Режим доступа: https://jquery-docs.ru/ - Загл. с экрана.

7. Руководство по языку Go [Электронный ресурс] – Режим доступа: https://metanit.com/go/tutorial/ - Загл. с экрана.