МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01 Информационная безопасность**

**Образовательная программа: "Информационная безопасность / Information security"**

**Дисциплина:**

**«Web программирование»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

**«Разработка Веб сервиса»**

**Выполнил студент(ы):**

Ахаров Али Рустамович

**Проверил:**

\_\_Менщиков Александр Алексеевич\_\_/\_\_\_\_\_\_\_\_\_\_

*ФИО Подпись*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно, зачтено)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата*

Санкт-Петербург

2024г.

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc182235189)

[ВВЕДЕНИЕ 4](#_Toc182235190)

[1. Описание архитектуры ПО 5](#_Toc182235191)

[2. Описание структуры базы данных 6](#_Toc182235192)

[3. Описание протокола и форматов передачи данных 7](#_Toc182235193)

[4. Описание API 9](#_Toc182235194)

[5. Тестирование функционала ПО 10](#_Toc182235195)

[ЗАКЛЮЧЕНИЕ 15](#_Toc182235196)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc182235197)

[ПРИЛОЖЕНИЕ 17](#_Toc182235198)

ВВЕДЕНИЕ

Цель работы – разработать онлайн чат для пользователей, состоящий из двух микросервисов, написанных на языке Python с использованием фреймворка FastAPI:

* Первый должен отвечать за регистрацию, аутентификацию, создание и поиск чат-комнат.
* Второй - за функционал Websocket чата.

# Описание протокола и форматов передачи данных

Взаимодействие между клиентом и сервером происходит через WebSocket, что позволяет поддерживать постоянное соединение и обмен сообщениями в реальном времени. Процесс передачи данных включает несколько этапов:

1. **Запрос на установку соединения**:
   * Клиент инициирует запрос на установление WebSocket-соединения с сервером, передавая JWT-токен для авторизации. Этот токен используется для подтверждения прав пользователя на доступ к чат-комнате.
2. **Подтверждение соединения сервером**:
   * Сервер проверяет правильность переданного JWT-токена. Если токен действителен, сервер аутентифицирует клиента и открывает WebSocket-соединение для обмена сообщениями.
3. **Двусторонний обмен сообщениями**:
   * После установления соединения осуществляется двусторонний обмен данными. Сообщения от клиента передаются в сервис чата, который распределяет их между всеми пользователями, подключёнными к той же чат-комнате. Это позволяет всем участникам чата получать сообщения в реальном времени.
4. **Закрытие соединения**:
   * Когда обмен сообщениями завершён, одна из сторон (клиент или сервер) инициирует закрытие WebSocket-сессии, и соединение закрывается. Все пользователи получают уведомление о завершении сеанса.

**Формат передачи данных**:

При передаче сообщений используется формат «почта\_пользователя: сообщение\_пользователя», где "почта\_пользователя" — это идентификатор пользователя, а "сообщение\_пользователя" — текст сообщения.

Взаимодействие через WebSocket и использование JWT-токенов обеспечивает безопасный и эффективный обмен данными между пользователями в чате, а также простоту масштабирования за счёт использования микросервисной архитектуры.

# Описание протокола и форматов передачи данных

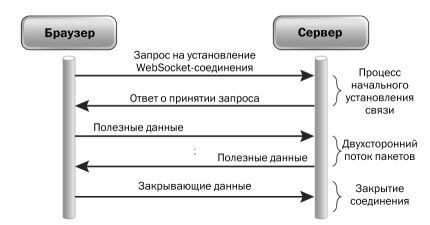
На рисунке 3 представлен процесс взаимодействия между клиентом и сервером через WebSocket и включает следующие этапы:

**Запрос на установку соединения**: Клиент отправляет запрос на установление WebSocket-соединения с сервером. Этот запрос включает проверку на авторизацию с использованием JWT-токена для подтверждения прав доступа.

**Подтверждение соединения сервером**: Сервер проверяет запрос, аутентифицирует клиента (если JWT-токен валиден) и, при успешной проверке, подтверждает запрос, открывая постоянное соединение.

**Двусторонний обмен сообщениями**: После установления соединения происходит обмен сообщениями между клиентом и сервером. Сообщения от клиента поступают в чат-сервис, который доставляет их всем пользователям, подключенным к той же чат-комнате. Сервер может также отправлять обновления клиенту по мере необходимости.

**Закрытие соединения**: По завершении обмена одна из сторон (клиент или сервер) инициирует завершение WebSocket-сессии, и соединение закрывается



1. Схема передачи данных по Websocket

В данной работе используется тактовый формат при передачи данных в виде: «почта\_пользователя: сообщение\_пользователя»

# Описание API

#Изображение выглядит как текст, снимок экрана, число, Шрифт

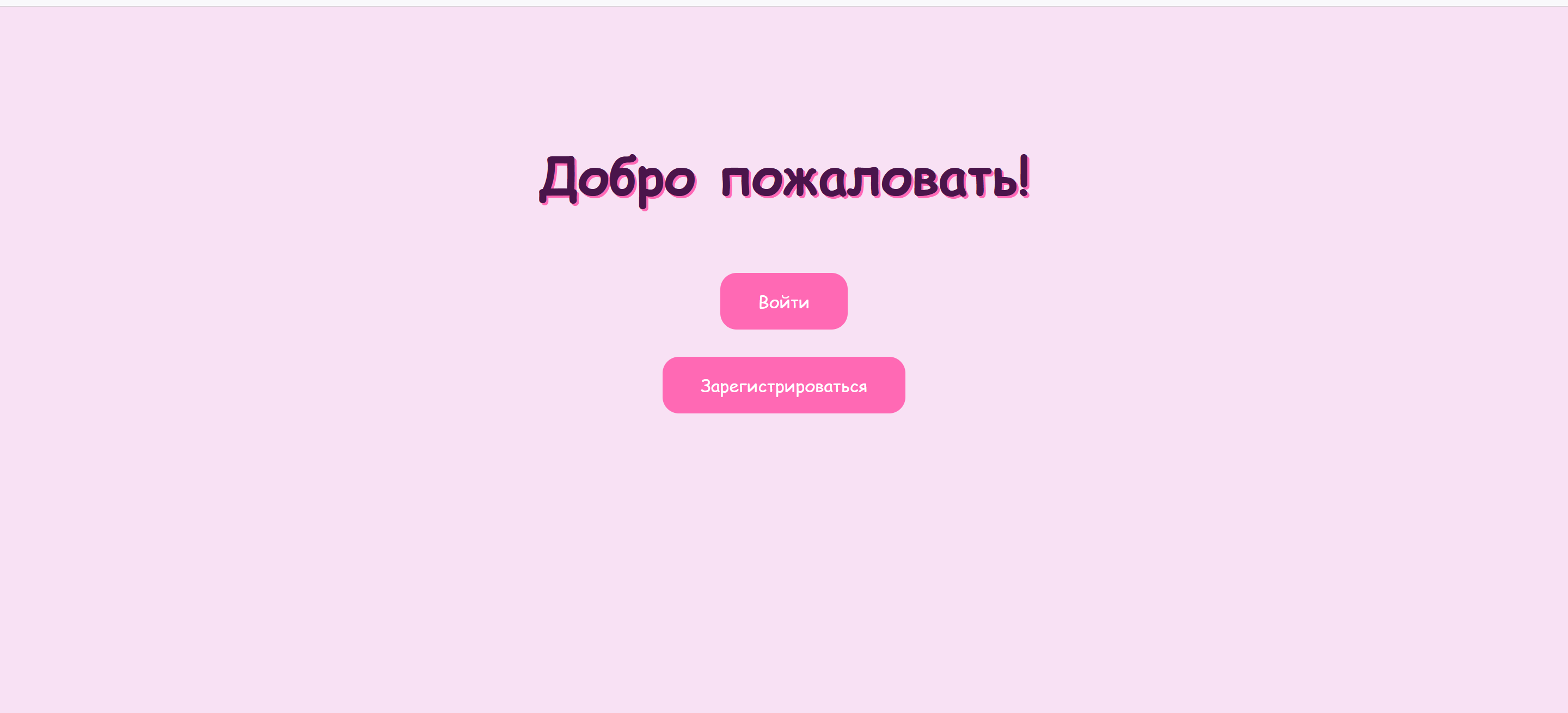
Автоматически созданное описание

1. API веб сервиса

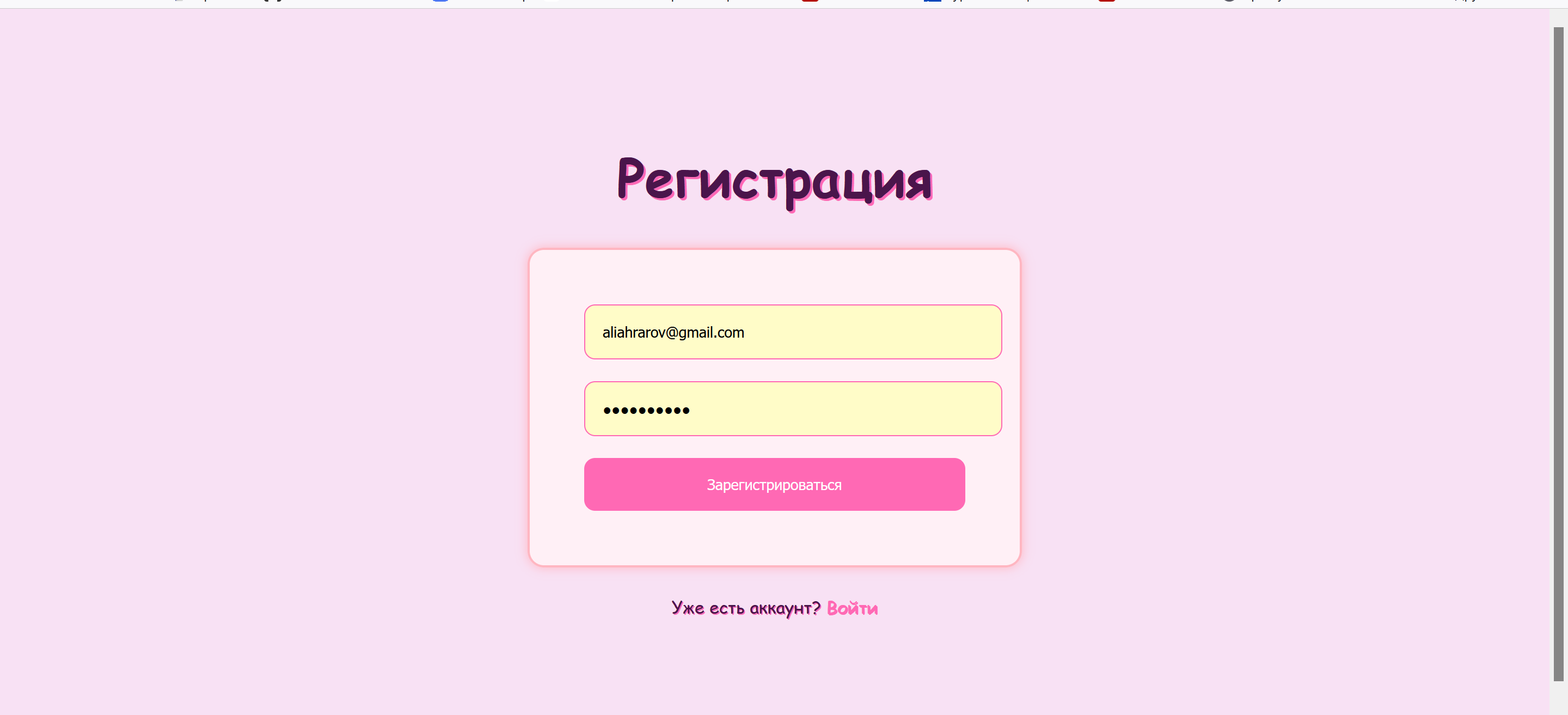
# Тестирование функционала ПО



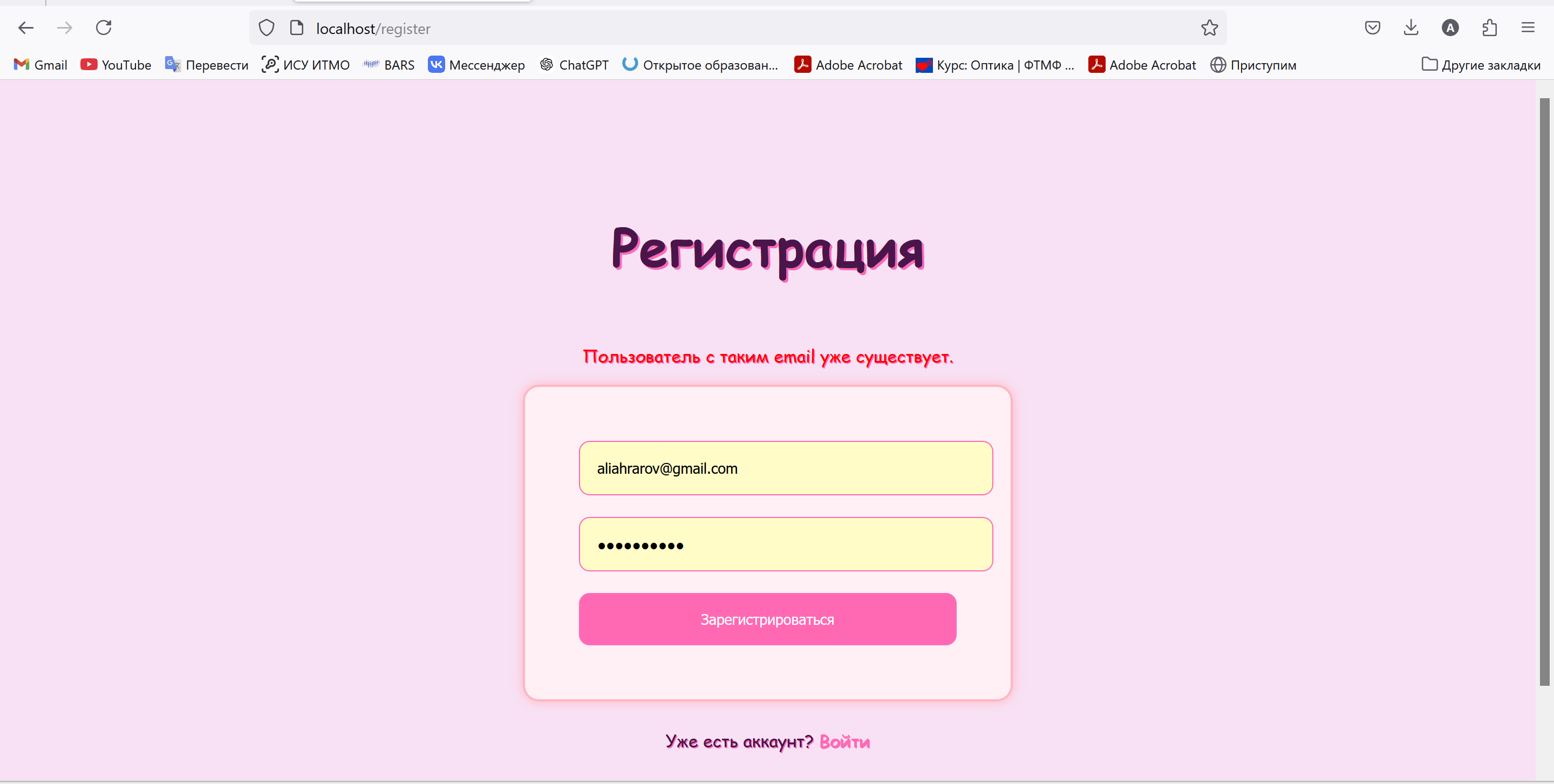
1. Запущенный docker-контейнер



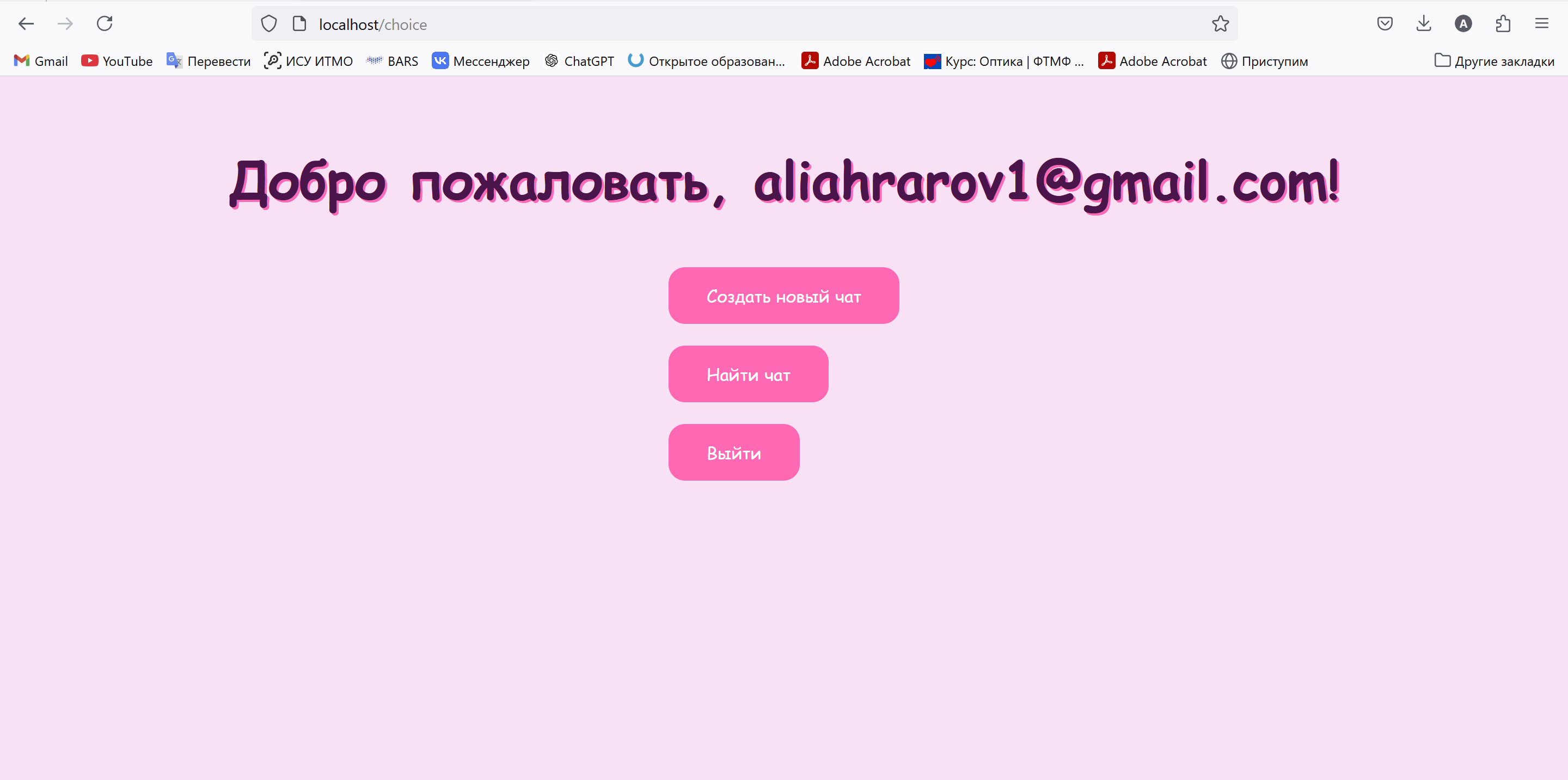
1. Гламурная главная страница



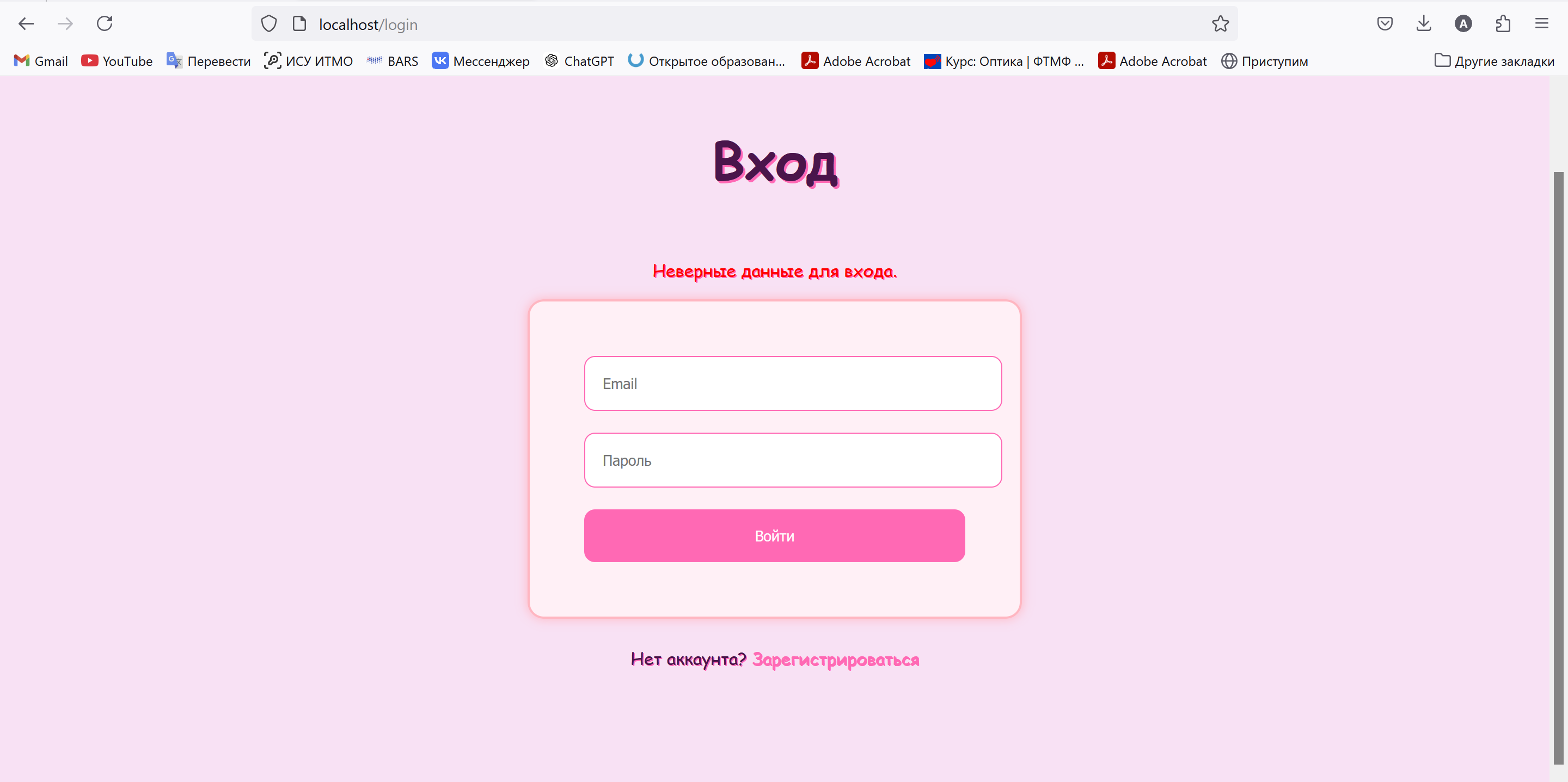
1. страница регистрации



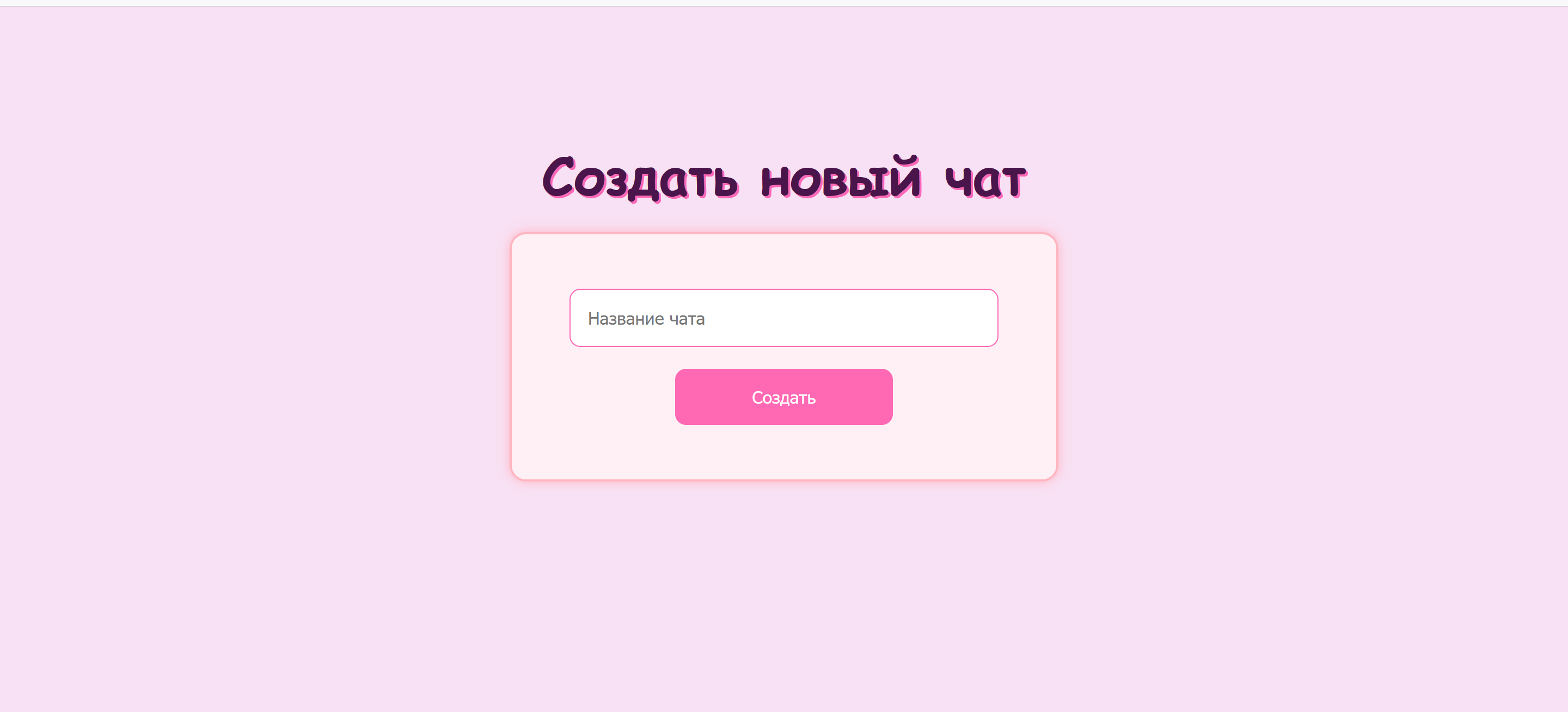
1. Ошибка при попытке зарегистрироваться под уже существующем email-ом



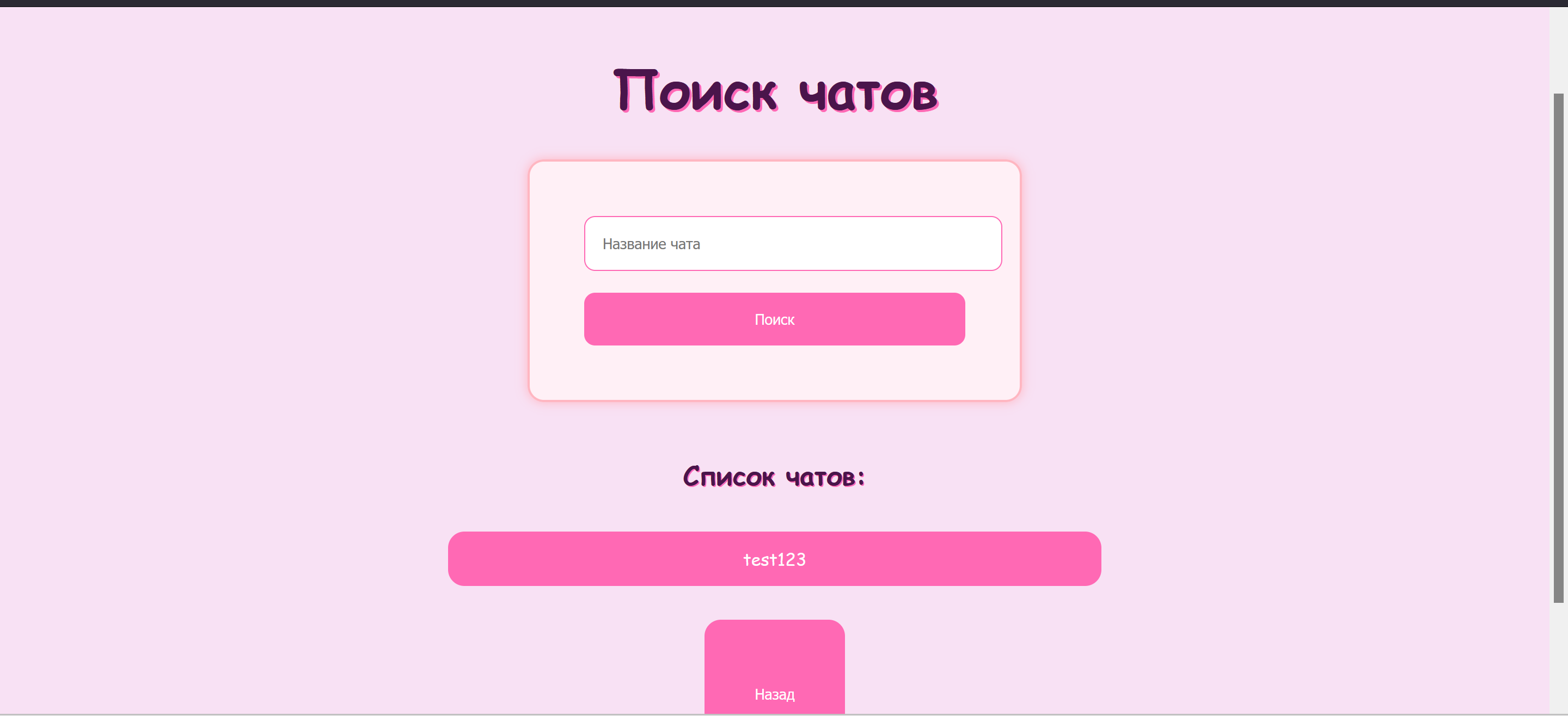
1. Вход пользователя



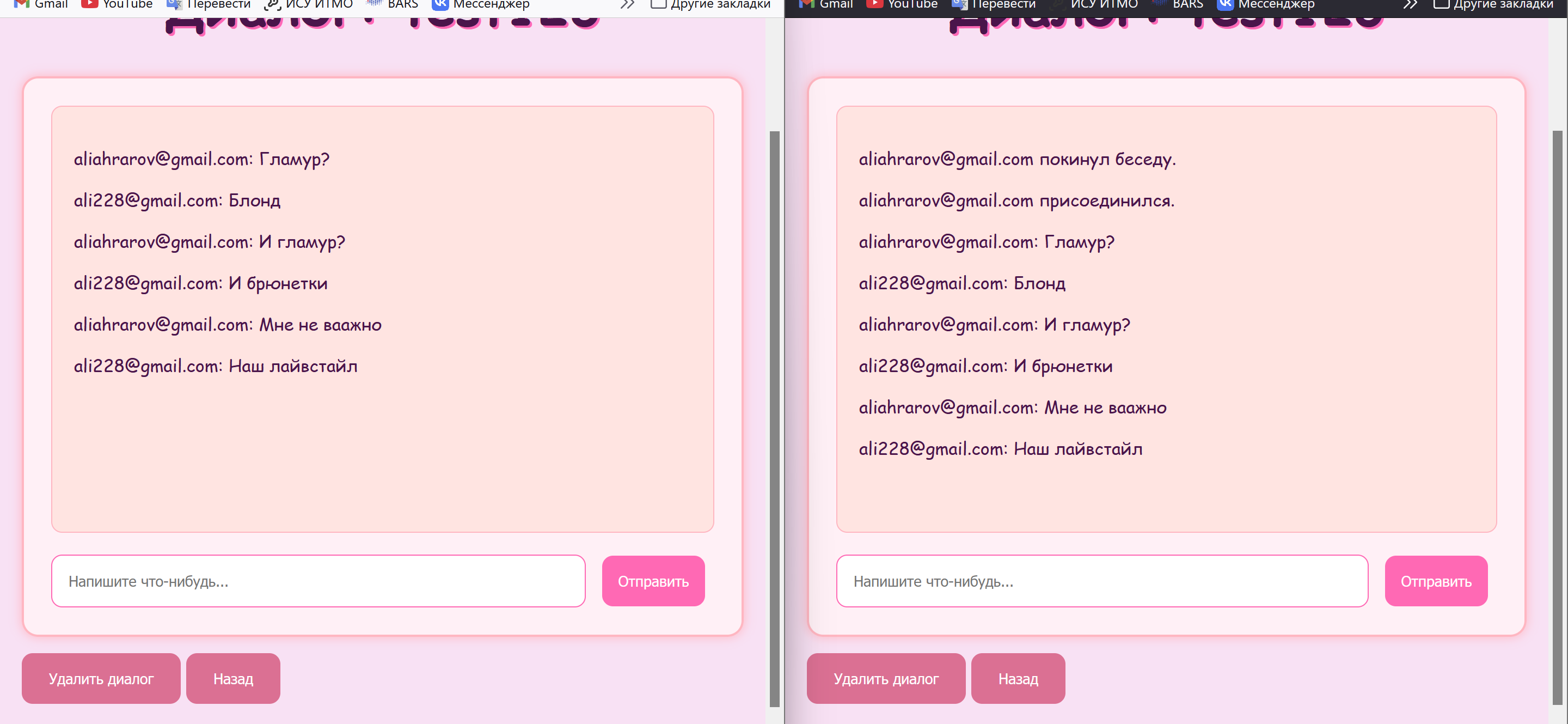
1. Ошибка идентификации или аутентификации



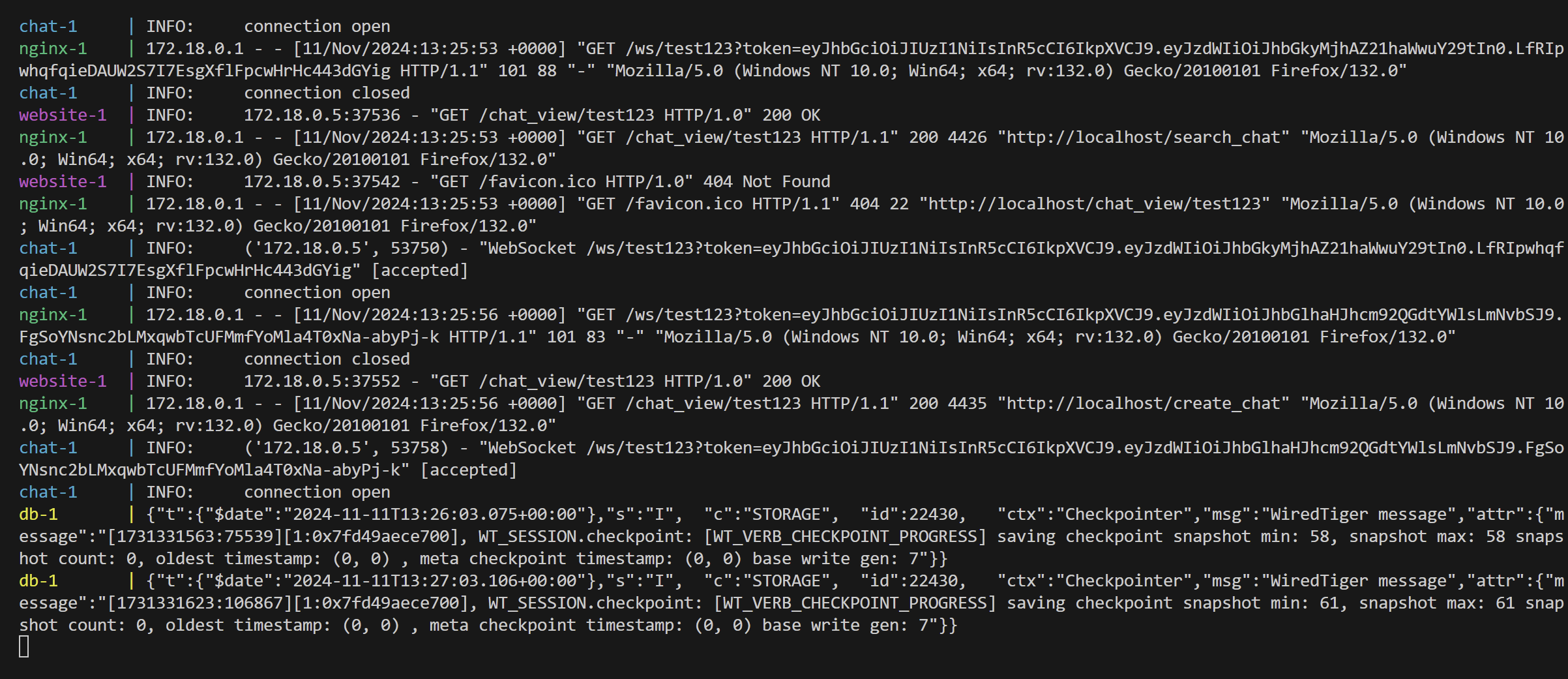
1. Создание нового чата



1. Поиск нового чата



1. Общение в чате пользователей



1. Файл с логами после работы сервера

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был разработан веб-сервис онлайн-чата, состоящий из двух основных микросервисов — **website** и **chat**, реализованных с использованием Python и фреймворка FastAPI. Основные характеристики и компоненты системы:

1. **Микросервис website** отвечает за функциональность веб-приложения, включая регистрацию, аутентификацию пользователей, а также создание и управление чатами. Аутентификацию и хранение данных поддерживает база данных PostgreSQL, с которой взаимодействуют через ORM-библиотеку, упрощающую операции с базой данных.
2. **Микросервис chat** управляет обменом сообщениями в чатах в режиме реального времени через WebSocket-протокол, обеспечивая многопользовательскую отправку и получение сообщений. Пользователи аутентифицируются с использованием JWT-токенов, что позволяет поддерживать защищённый доступ.

Для фронтенда используются шаблоны **Jinja2**, обеспечивающие рендеринг страниц и интуитивное взаимодействие пользователя с веб-интерфейсом.

**Развёртывание и конфигурация**

Все компоненты системы были развернуты с использованием **Docker**, что облегчает управление инфраструктурой и развертывание. Конфигурация собрана в **docker-compose.yml**, который включает следующие сервисы:

* **website** и **chat** микросервисы
* **Сервер Nginx** для балансировки нагрузки и проксирования запросов
* **PostgreSQL** как база данных

**Nginx** настроен на маршрутизацию HTTP-запросов к микросервису website и WebSocket-запросов к микросервису chat, что обеспечивает эффективную и масштабируемую работу системы.

Таким образом, был успешно реализован веб-сервис, который поддерживает аутентификацию, управление чатами и обмен сообщениями в реальном времени с использованием WebSocket.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. FastAPI Documentation. [Электронный ресурс] // URL: https://fastapi.tiangolo.com.
2. PostgreSQL Documentation. [Электронный ресурс] // URL: <https://www.postgresql.org/docs/13/index.html>.
3. Docker Documentation. [Электронный ресурс] // URL: https://docs.docker.com/get-started/overview/.
4. JWT.IO. JSON Web Tokens (JWT): What is JSON Web Token?. [Электронный ресурс] // URL: <https://jwt.io>
5. Nginx Documentation. [Электронный ресурс] // URL: https://nginx.org/en/docs/.

ПРИЛОЖЕНИЕ

<https://github.com/Leha23112004/web> - ссылка на исходный код