1. **Код API**

from flask import Flask, request, render\_template, redirect, url\_for, session, flash

from flask\_sqlalchemy import SQLAlchemy

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from datetime import datetime

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_super\_secret\_key'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://user:password@localhost:5432/messenger'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

class User(db.Model):

    \_\_tablename\_\_ = 'users'

    id = db.Column(db.Integer, primary\_key=True)

    username = db.Column(db.String(80), unique=True, nullable=False)

    email = db.Column(db.String(120), unique=True, nullable=False)

    password\_hash = db.Column(db.String(128))

class Chat(db.Model):

    \_\_tablename\_\_ = 'chats'

    id = db.Column(db.Integer, primary\_key=True)

    name = db.Column(db.String, nullable=False)

class Message(db.Model):

    \_\_tablename\_\_ = 'messages'

    id = db.Column(db.Integer, primary\_key=True)

    chat\_id = db.Column(db.Integer, db.ForeignKey('chats.id'))

    sender\_id = db.Column(db.Integer, db.ForeignKey('users.id'))

    content = db.Column(db.String, nullable=False)

    timestamp = db.Column(db.DateTime, default=datetime.utcnow)

class ChatMember(db.Model):

    \_\_tablename\_\_ = 'chat\_members'

    id = db.Column(db.Integer, primary\_key=True)

    chat\_id = db.Column(db.Integer, db.ForeignKey('chats.id'), nullable=False)

    user\_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

@app.route('/')

def home():

    return redirect(url\_for('login\_page'))

@app.route('/register-page', methods=['GET', 'POST'])

def register\_page():

    if request.method == 'POST':

        username = request.form.get('username')

        email = request.form.get('email')

        password = request.form.get('password')

        if not username or not email or not password:

            flash("Все поля обязательны для заполнения.")

            return render\_template('register.html')

        if User.query.filter((User.username == username) | (User.email == email)).first():

            flash("Пользователь с таким логином или email уже существует.")

            return render\_template('register.html')

        hashed\_password = generate\_password\_hash(password)

        new\_user = User(username=username, email=email, password\_hash=hashed\_password)

        db.session.add(new\_user)

        db.session.commit()

        flash("Успешная регистрация. Войдите в систему.")

        return redirect(url\_for('login\_page'))

    return render\_template('register.html')

@app.route('/login-page', methods=['GET', 'POST'])

def login\_page():

    if request.method == 'POST':

        username = request.form.get('username')

        password = request.form.get('password')

        user = User.query.filter\_by(username=username).first()

        if not user or not check\_password\_hash(user.password\_hash, password):

            flash("Неправильный логин или пароль.")

            return render\_template('login.html')

        session['user\_id'] = user.id

        return redirect(url\_for('chats\_page'))

    return render\_template('login.html')

@app.route('/logout-page')

def logout\_page():

    session.clear()

    return redirect(url\_for('login\_page'))

@app.route('/chats-page', methods=['GET', 'POST'])

def chats\_page():

    user\_id = session.get('user\_id')

    if not user\_id:

        return redirect(url\_for('login\_page'))

    if request.method == 'POST':

        chat\_name = request.form.get('name')

        if chat\_name:

            new\_chat = Chat(name=chat\_name)

            db.session.add(new\_chat)

            db.session.commit()

            chat\_member = ChatMember(chat\_id=new\_chat.id, user\_id=user\_id)

            db.session.add(chat\_member)

            db.session.commit()

            initial\_message = Message(chat\_id=new\_chat.id, sender\_id=user\_id, content="Чат создан")

            db.session.add(initial\_message)

            db.session.commit()

            flash("Чат создан.")

        else:

            flash("Имя чата не задано.")

        return redirect(url\_for('chats\_page'))

    chats = (

        db.session.query(Chat)

        .join(ChatMember, Chat.id == ChatMember.chat\_id)

        .filter(ChatMember.user\_id == user\_id)

        .all()

    )

    return render\_template('chats.html', chats=chats)

@app.route('/chats/<int:chat\_id>/leave', methods=['POST'])

def leave\_chat(chat\_id):

    user\_id = session.get('user\_id')

    if not user\_id:

        return redirect(url\_for('login\_page'))

    # Найдём пользователя

    user = User.query.get(user\_id)

    if not user:

        flash("Пользователь не найден.")

        return redirect(url\_for('chats\_page'))

    # Удалим участника из чата

    ChatMember.query.filter\_by(chat\_id=chat\_id, user\_id=user\_id).delete()

    db.session.commit()

    # Добавим сообщение о выходе пользователя

    leave\_message = Message(chat\_id=chat\_id, sender\_id=user\_id, content=f"{user.username} покинул чат")

    db.session.add(leave\_message)

    db.session.commit()

    # Проверим, остались ли участники

    remaining\_members = ChatMember.query.filter\_by(chat\_id=chat\_id).count()

    if remaining\_members == 0:

        # Удалим все сообщения

        Message.query.filter\_by(chat\_id=chat\_id).delete()

        # Удалим сам чат

        Chat.query.filter\_by(id=chat\_id).delete()

        db.session.commit()

        flash("Чат удалён, так как в нём не осталось участников.")

    else:

        flash("Вы покинули чат.")

    return redirect(url\_for('chats\_page'))

@app.route('/chats/<int:chat\_id>/messages-page', methods=['GET', 'POST'])

def view\_chat(chat\_id):

    user\_id = session.get('user\_id')

    if not user\_id:

        return redirect(url\_for('login\_page'))

    if request.method == 'POST':

        action = request.form.get('action')

        if action == 'send\_as\_self':

            content = request.form.get('content')

            if content:

                new\_message = Message(chat\_id=chat\_id, sender\_id=user\_id, content=content)

                db.session.add(new\_message)

                db.session.commit()

        elif action == 'add\_user':

            new\_user\_id\_str = request.form.get('new\_user\_id')

            try:

                new\_user\_id = int(new\_user\_id\_str)

                user = User.query.get(new\_user\_id)

                if user:

                    exists = ChatMember.query.filter\_by(chat\_id=chat\_id, user\_id=new\_user\_id).first()

                    if not exists:

                        db.session.add(ChatMember(chat\_id=chat\_id, user\_id=new\_user\_id))

                        db.session.commit()

                        db.session.add(Message(chat\_id=chat\_id, sender\_id=user\_id, content=f"{user.username} добавлен"))

                        db.session.commit()

                        flash("Пользователь добавлен.")

                else:

                    flash("Пользователь не найден.")

            except:

                flash("Неверный ID пользователя.")

        return redirect(url\_for('view\_chat', chat\_id=chat\_id))

    messages = (

        db.session.query(Message, User)

        .join(User, Message.sender\_id == User.id)

        .filter(Message.chat\_id == chat\_id)

        .order\_by(Message.timestamp.asc())

        .all()

    )

    formatted = [

        {'timestamp': m.timestamp.strftime('%Y-%m-%d %H:%M:%S'), 'sender': u.username, 'content': m.content}

        for m, u in messages

    ]

    return render\_template('messages.html', messages=formatted, chat\_id=chat\_id)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

1. **Код интерфейса**

**style.css**

body {

    font-family: Arial, sans-serif;

    padding: 20px;

    background-color: #f4f4f4;

}

header {

    background-color: #333;

    color: #fff;

    padding: 10px 20px;

}

header h1 {

    display: inline;

}

nav a {

    color: #fff;

    margin-left: 15px;

    text-decoration: none;

}

nav a:hover {

    text-decoration: underline;

}

main {

    background-color: #fff;

    padding: 20px;

    margin-top: 10px;

    border-radius: 5px;

}

form input, form button {

    display: block;

    margin: 10px 0;

    padding: 10px;

    width: 100%;

    box-sizing: border-box;

}

ul {

    list-style-type: none;

    padding: 0;

}

li {

    margin-bottom: 10px;

}

.flashes {

    background-color: #ffdddd;

    border-left: 6px solid #f44336;

    padding: 10px;

    margin-bottom: 15px;

}

**base.html**

<!DOCTYPE html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <title>Messenger</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

    <header>

        <h1>Messenger</h1>

        <nav>

            {% if session.get('token') %}

                <a href="{{ url\_for('chats\_page') }}">Чаты</a>

                <a href="{{ url\_for('logout\_page') }}">Выйти</a>

            {% else %}

                <a href="{{ url\_for('login\_page') }}">Войти</a>

                <a href="{{ url\_for('register\_page') }}">Регистрация</a>

            {% endif %}

        </nav>

    </header>

    <main>

        {% with messages = get\_flashed\_messages() %}

          {% if messages %}

            <ul class="flashes">

              {% for message in messages %}

                <li>{{ message }}</li>

              {% endfor %}

            </ul>

          {% endif %}

        {% endwith %}

        {% block content %}{% endblock %}

    </main>

</body>

</html>

**chats.html**

{% extends "base.html" %}

{% block content %}

<h2>Ваши чаты</h2>

<ul>

    {% for chat in chats %}

        <li>

            <a href="{{ url\_for('view\_chat', chat\_id=chat['id']) }}">{{ chat['name'] }}</a>

            <!-- Кнопка "Покинуть чат" -->

            <form method="POST" action="{{ url\_for('leave\_chat', chat\_id=chat['id']) }}" style="display:inline;">

                <button type="submit">Покинуть</button>

            </form>

        </li>

    {% endfor %}

</ul>

<form method="POST">

    <input type="text" name="name" placeholder="Название нового чата" required>

    <button type="submit">Создать чат</button>

</form>

{% endblock %}

**login.html**

{% extends "base.html" %}

{% block content %}

<h2>Вход</h2>

<form method="POST">

    <input type="text" name="username" placeholder="Имя пользователя" required>

    <input type="password" name="password" placeholder="Пароль" required>

    <button type="submit">Войти</button>

</form>

{% endblock %}

**messages. html**

{% extends "base.html" %}

{% block content %}

<h2>Сообщения</h2>

<ul>

    {% for message in messages %}

        <li>

            {{ message['timestamp'] }} | <b>{{ message['sender'] }}</b>: {{ message['content'] }}

        </li>

    {% endfor %}

</ul>

<form method="POST">

    <input type="text" name="content" placeholder="Ваше сообщение" required>

    <button type="submit" name="action" value="send\_as\_self">Отправить</button>

</form>

<form method="POST">

    <input type="text" name="new\_user\_id" placeholder="ID пользователя для добавления" required>

    <button type="submit" name="action" value="add\_user">Добавить пользователя</button>

</form>

<a href="{{ url\_for('chats\_page') }}">Назад к чатам</a>

{% endblock %}

**register.html**

{% extends "base.html" %}

{% block content %}

<h2>Регистрация</h2>

<form method="POST">

    <input type="text" name="username" placeholder="Имя пользователя" required>

    <input type="email" name="email" placeholder="Email" required>

    <input type="password" name="password" placeholder="Пароль" required>

    <button type="submit">Зарегистрироваться</button>

</form>

{% endblock %}

1. **Описание структуры интерфейса и логики работы с API**

Приложение построено на базе Flask и использует SQLAlchemy для взаимодействия с базой данных PostgreSQL. Интерфейс организован через HTML-шаблоны с использованием render\_template, а данные передаются между сервером и клиентом через формы и URL-параметры.

Пользователь начинает с регистрации или входа в систему. После успешной аутентификации ему открывается доступ к интерфейсу чатов. Чаты создаются пользователями вручную, и каждый новый чат автоматически включает создателя в список участников. Добавление новых пользователей в чат возможно по их ID, что реализовано через POST-запрос на ту же страницу сообщений чата.

Логика API включает следующие действия: регистрация пользователя, вход, выход, создание чатов, просмотр списка чатов, переход к сообщениям, отправка сообщений, добавление участников и выход из чата. Если пользователь выходит, и он был последним участником, чат полностью удаляется вместе с сообщениями.

1. **Примеры рендеринга данных и передачи параметров**

Пример передачи чатов в шаблон происходит при отображении страницы /chats-page. Сначала в базе данных выбираются чаты, в которых состоит пользователь, после чего результат передаётся в HTML-шаблон chats.html:

return render\_template('chats.html', chats=chats)

Сообщения отображаются аналогично: при заходе в конкретный чат сервер получает список всех сообщений и информацию об отправителях, форматирует их (дата, имя, текст), а затем отправляет в messages.html:

return render\_template('messages.html', messages=formatted, chat\_id=chat\_id)

Отправка данных происходит через HTML-формы с методом POST. На странице сообщений можно выбрать действие — отправить сообщение или добавить пользователя, что различается по значению скрытого поля action. Сервер обрабатывает это значение и выполняет соответствующее действие.

1. **Презентация сценария работы приложения**

После запуска пользователь попадает на страницу входа. Если аккаунт ещё не создан, он переходит на форму регистрации, вводит данные, и они сохраняются в базе. Далее он входит в систему и видит список чатов. Новый чат можно создать, указав имя — он сразу появится в списке.

При открытии чата пользователь видит все сообщения, отсортированные по времени. Он может отправить своё сообщение, а также добавить участника по его ID. В чате отображаются имя отправителя, дата и содержание каждого сообщения.

Если пользователь решает выйти из чата, система удаляет его из участников. Если после этого участников не остаётся, удаляются и все связанные данные — сам чат и сообщения. Это поддерживает чистоту данных и автоматическое завершение "пустых" чатов.