Авторы: Ахметов Канат, Новосёлов Андрей, Торопов Александр

Оглавление

https://neo-inst.vercel.app/

Введение в мир NEONinst

Добро пожаловать в NEONinst, где магия общения встречается с волнующим миром новостей! Мы не просто мессенджер, мы – твой личный гид в мире актуальной информации и удивительных общений. Готов погрузиться в это приключение?

Как пользоваться этой книгой?

Каждая глава книги построена таким образом, чтобы вы могли постепенно наращивать свои знания и навыки. В каждой главе вы найдете подробные инструкции, примеры кода и пояснения к ним.

Рекомендуется читать книгу последовательно, не пропуская главы, так как каждая следующая глава опирается на материал предыдущих. Однако, если вы уже обладаете определенными знаниями в некоторых областях, вы можете переходить к интересующим вас разделам.

Требования к окружению

Для работы над проектом вам потребуется установленный Python и текстовый редактор или IDE, такие как Visual Studio Code или PyCharm. Также понадобится браузер для тестирования фронтенда и установленный менеджер пакетов pip для установки необходимых библиотек.

Целевая аудитория

Мессенджер NEONinst имеет функционал, который удовлетворяет потребностям множеству групп пользователей, а именно:

- студенты и образовательные сообщества: для обмена информацией, обсуждения учебных материалов и получения новостей об актуальных событиях в образовании;
- профессиональные группы и команды: для коммуникации внутри организаций, обсуждения проектов и оперативного информирования о новостях в отрасли;
- любители новостей и общественно-политических дискуссий: для получения свежих новостей, обмена мнениями и участия в дискуссиях на различные темы;

- семейные группы и друзья: для обмена личными сообщениями, планирования мероприятий и получения обновлений от близких.

Путешествие по функционалу

Регистрация и аутентификация

Для начала своего путешествия в NEONinst просто создай свой аккаунт и окунись в удивительный мир общения и новостей. Аутентификация здесь так же легка, как покачивание пальцами!

- Профиль

Твой профиль — это твоя визитная карточка в NEONinst. Добавь фото, расскажи немного о себе и готово! Подготовься к тому, чтобы все вокруг узнали о твоих интересах и талантах.

- Чаты

Пришло время веселиться в чатах! Общайся с друзьями, семьей или коллегами - в любое время и в любом месте. Здесь каждое обсуждение превращается в невероятное приключение!

- Новостная лента

И, конечно же, наша новостная лента — это твой персональный информационный портал. Будь в курсе всех событий, которые тебя интересуют, и делай свои выводы.

Основные понятия: все, что ты должен знать

Хочешь понять, как всё работает? Вот небольшой справочник, который поможет тебе сориентироваться в мире NEONinst:

- регистрация и аутентификация пользователей: здесь каждый новый пользователь — это как герой, который только начинает свое путешествие;
- профиль пользователя: твой паспорт в мир NEONinst. Здесь ты можешь показать всю свою индивидуальность и неповторимость;
- чаты: Место, где слова становятся магией общения. Здесь можно узнать много интересного и поделиться своими мыслями;

- новостная лента: твой информационный космос, где ты всегда будешь в курсе всех горячих событий;
- фронтенд (англ. front end, frontend) — это презентационная часть web-приложений, информационной или программной системы, её пользовательский интерфейс и связанные с ним компоненты.

Теперь, когда ты знаешь все основные понятия, давай начнем это удивительное путешествие вместе!

Фронтенд

В данном разделе мы рассмотрим, как создавался интерфейс приложения с использованием HTML и CSS.

Профиль

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Профиль</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <div class="profile-box">
            <img src="menu.png" class="menu-icon">
            <img src="setting.png" class="setting-icon">
            <img src="ava.jpg" class="ava-icon">
            <h3>Имя пользователя</h3>
            <p>Студент ОмГПУ</p>
        </div>
    </div>
</body>
</html>
```

CSS:

```css
/*profile*/
*{
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "poppins", sans-serif;
}

.container{
    width: 100vw;
    height: 100vh;
    background-color: #0b1114;
```

```
    background-size: cover;
    background-position: center;
    display: flex;
    align-items: center;
    justify-content: center;
}

.profile-box{
    background: #9508CD;
    text-align: center;
    padding: 40px 90px;
    color: #F8F4FA;
    position: relative;
    border-radius: 20px;
}

.menu-icon{
    width: 25px;
    position: absolute;
    left: 40px;
    top: 40px;
    transition: all 0.3s ease-in-out;
}

.menu-icon:hover{
    transform: scale(1.2);
    color: #F8F4FA;
}

.setting-icon{
    width: 25px;
    position: absolute;
    right: 40px;
    top: 40px;
    transition: all 0.3s ease-in-out;
}

.setting-icon:hover{
    transform: scale(1.2);
    color: #F8F4FA;
}

.ava-icon{
    height: 200px;
    width: 200px;
    border-radius: 50%;
    background: #F8F4FA;
    padding: 6px;
    margin-top: 40px;
}

.icons{
    display: flex;
    justify-content: space-around;
    align-items: center;
}

.image{
    margin: 15px 10px;
```

```
    transition: all 0.3s ease-in-out;
}

.image:hover{
    transform: scale(1.2);
    color: #F8F4FA;
}

.profile-box h3 {
    font-size: 22px;
    margin-top: 20px;
    font-weight: 500;
}

.a {
    display: block;
    width: 100%;
    height: 100%;
}
```

В результате, данная страница создает простой профиль пользователя с аватаром, иконками меню и настроек, а также текстовой информацией о пользователе. Все стили применяются из внешнего файла style.css, который не представлен в данном фрагменте, но предполагается, что он содержит стили для классов menu-icon, setting-icon, ava-icon, и других элементов.

Авторизация

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Авторизация</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <form action="#">
            <h1>Регистрация</h1>

            <div class="input-box">
                <input type="text" placeholder="Логин"
                required/>
            </div>

            <div class="input-box">
                <input type="text" placeholder="Пароль"
                required/>
            </div>

            <div class="remember-forgot">
                <label><input type="checkbox"/>Запомнить меня</label>
                <a href="#">Забыли пароль?</a>
```

```
            </div>

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

            <div class="register-link">
                <p>Нет аккаунта? <a href="#">Зарегестрируйся здесь!</a></p>
            </div>
        </form>
    </div>

</body>
</html>
```

```
@import
url('https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300..800;1,300..800&display=swap');

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "Poppins";
}

body {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background: url(./bck1.jpg) no-repeat;
    background-size: cover;
}

.container {
    /*background: green;*/
    width: 420px;
    border: 2px solid (255, 255, 255, 0.1);
    box-shadow: 0px 0px 10px (0, 0, 0, 0.1);
    color: #fff;
    border-radius: 10px;
    padding: 30px 40px;
    backdrop-filter: blur(18px);
}

.container h1 {
    font-size: 36px;
    text-align: center;
}

.container .input-box{
    position: relative;
    width: 100%;
    height: 50px;
    margin: 30px 0px
}

.input-box input {
    width: 100%;
```

```
    height: 100%;
    background: transparent;
    border: none;
    outline: none;
    border: 2px solid (255, 255, 255, 0.2);
    border-radius: 5px;
    font-size: 16px;
    color: #fff;
    padding: 20px 45px 20px 20px;
}

.input-box input::placeholder {
    color: #fff;
}

.container .remember-forgot {
    display: flex;
    justify-content: space-between;
    font-size: 14.5px;
    margin: -15px 0px 15px;
}

.remember-forgot label input {
    color: #fff;
    margin-right: 3px;
}

.remember-forgot a{
    color: #fff;
    text-decoration: none;
}

.remember-forgot a:hover {
    text-decoration: underline;
}

.container .btn {
    width: 100%;
    height: 45px;
    background: #fff;
    border: none;
    outline: none;
    border-radius: 5px;
    box-decoration-break: 0px 0px 10px rgba(0, 0, 0, 0.1);
    cursor: pointer;
    font-size: 16px;
    font-weight: 400;
    color: #333;
}

.container .register-link {
    font-size: 14.5px;
    text-align: center;
    margin: 20px 0px 20px;
}

.register-link p a {
    color: #fff;
    text-decoration: none;
```

```
}

.register-link p a:hover {
    text-decoration: underline;
}
```

В результате, данная страница создает форму для авторизации пользователя с полями для ввода логина и пароля, опцией "Запомнить меня", ссылкой на восстановление пароля и ссылкой на регистрацию для новых пользователей. Все стили применяются из внешнего файла style.css, который не представлен в данном фрагменте, но предполагается, что он содержит стили для классов input-box, remember-forgot, btn, register-link, и других элементов.

Настройка профиля:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Профиль</title>
    <link rel="stylesheet" href="style.css">
    <script src="https://kit.fontawesome.com/31b0575cc7.js"
crossorigin="anonymous" referrerpolicy="no-referrer"></script>

</head>
<body>
    <div class="container">
        <div class="profile">
            <div class="profile-header">
                <img src="neoDark.png" alt="profile" class="profile-img">
                <div class="profile-text-container">
                    <h1 class="profile-title">Профиль</h1>
                    <p class="profile-email">
                        alextorpv@gmail.com
                    </p>
                </div>
            </div>

            <div class="menu">
                <a href="#" class="menu-link"><i class="fa-solid fa-user-tie
menu-icon"></i>Основные</a>
                <a href="#" class="menu-link"><i class="fa-solid fa-key menu-
icon"></i>Пароль</a>
                <a href="#" class="menu-link"><i class="fa-solid fa-address-card
menu-icon"></i>Профиль</a>
                <a href="#" class="menu-link"><i class="fa-solid fa-right-from-
bracket menu-icon"></i>Выйти</a>
            </div>
        </div>
```

```
        <form class="account">
            <div class="account-header">
                <h1 class="account-title">Основное</h1>
                <div class="btn-container">
                    <button class="btn-ok">Сохранить</button>
                    <button class="btn-cancel">Отмена</button>
                </div>
            </div>

            <div class="account-edit">
                <div class="input-container">
                    <label>Фамилия</label>
                    <input type="text" placeholder="Фамилия">
                </div>
                <div class="input-container">
                    <label>Имя</label>
                    <input type="text" placeholder="Имя">
                </div>
            </div>

            <div class="account-edit">
                <div class="input-container">
                    <label>Email</label>
                    <input type="text" placeholder="Email">
                </div>
                <div class="input-container">
                    <label>Номер телефона</label>
                    <input type="text" placeholder="Номер телефона">
                </div>
            </div>

            <div class="account-edit">
                <div class="input-container">
                    <label>О себе</label>
                    <input type="text" placeholder="О себе">
                </div>
            </div>

            <div class="account-edit">
                <div class="input-container2">
                    <label>Страна</label>
                    <select class="custom-select">
                        <option>Казахстан</option>
                        <option selected>Россия</option>
                        <option>Республика Беларусь</option>
                        <option>Китай</option>
                    </select>
                </div>
            </div>
        </form>
    </div>
</body>
</html>
```

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300
```

```
;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,
800;1,900&display=swap');

body {
    background-color: #0b1114;;
}

.container {
    display: flex;
    margin: 50px 10px;
}

.profile {
    flex: 1;
    margin-right: 10px;
    background-color: #262626;
    box-shadow: 0px 1px 4px rgba(0, 0, 0, 0.25);
    border-radius: 5px;
    height: max-content;
}

.profile-header {
    display: flex;
    margin-left: 15px;

}

.profile img {
    width: 60px;
    height: 60px;
    margin: 10px;
}

.profile-text-container {
    line-height: 0.5;
}

.profile-title {
    font-family: "Poppins", sans-serif;
    font-size: 20px;
    color: aliceblue;
}

.profile-email {
    font-family: "Poppins", sans-serif;
    font-size: 14px;
    color: #fff;
}

.menu {
    margin: 0 20px;
}

.menu-link {
    display: block;
    text-decoration: none;
    color: aliceblue;
    font-family: "Poppins", sans-serif;
    padding: 10px;
```

```
    margin: 10px;
    border-radius: 10px;
}

.menu-icon {
    margin-right: 10px;
}

.menu-link:first-child {
    color: aliceblue;
    background-color: #9508CD;
}

.menu-link:hover {
    background-color: #5d6061;
    color: #fff;
}

.menu-link:first-child:hover {
    background-color: #9508CD;
    color: #fff;
}

.account {
    flex: 2;
    margin-left: 10px;
    background-color: #262626;
    box-shadow: 0px 1px 4px rgba(0, 0, 0, 0.25);
    border-radius: 5px;
    height: max-content;
}

.account-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
    border-bottom: 1px solid #fff;
}

.account-title {
    font-family: "Poppins", sans-serif;
    font-size: 20px;
    color: aliceblue;
    font-weight: 500;
    margin-left: 10px;
}

.btn-container {
    margin-right: 5px;
}

.btn-ok {
    width: 8em;
    height: 3em;
    cursor: pointer;
    border: 1px solid #0b1114;
    border-radius: 5px;
    color: #fff;
```

```
    font-weight: bold;
    margin-right: 6px;
    background-color: #9508CD;
    transition-duration: 0.3s;
    box-shadow: rgba(0, 0, 0, 0.25);
}

.btn-ok:hover {
    background-color: #5d6061;
}

.btn-cancel {
    width: 8em;
    height: 3em;
    cursor: pointer;
    border: 1px solid #0b1114;
    border-radius: 5px;
    color: #fff;
    font-weight: bold;
    margin-right: 6px;
    background-color: #768499;
    transition-duration: 0.3s;
    box-shadow: rgba(0, 0, 0, 0.25);
}

.btn-cancel:hover {
    background-color: #5d6061;
}

.account-edit {
    display: flex;
    justify-content: space-between;
    margin: 15px 0;
}

.input-container {
    width: 100%;
    display: flex;
    flex-direction: column;
    margin: 0 20px;
}

.input-container label {
    font-family: "Poppins", sans-serif;
    color: #fff;
    font-size: 14px;
}

.input-container input {
    height: 25px;
    border: 1.5px solid #5d6061;
    border-radius: 5px;
    padding: 5px;
    color: #5d6061;
    font-family: "Poppins", sans-serif;
}

.input-container input:focus {
    outline: none;
```

```
    border: 1.5px solid #9508CD;
}

.input-container textarea {
    display: block;
    width: 100%;
    height: 100px;
    padding: 5px;
    border: 1.5px solid #5d6061;
    border-radius: 5px;
    font-family: "Poppins", sans-serif;
    color: #5d6061;
}

.input-container textarea:focus {
    outline: none;
    border: 1.5px solid #9508CD;
}

.input-container2 label {
    font-family: "Poppins", sans-serif;
    color: #fff;
    font-size: 14px;
}

.input-container2 {
    width: 100%;
    display: flex;
    flex-direction: column;
    margin: 0 20px;
}

.input-container2 input {
    height: 25px;
    border: 1.5px solid #5d6061;
    border-radius: 5px;
    padding: 5px;
    color: #5d6061;
    font-family: "Poppins", sans-serif;
}

.input-container2 input:focus {
    outline: none;
    border: 1.5px solid #9508CD;
}

.input-container2 textarea {
    display: block;
    width: 100%;
    height: 100px;
    padding: 5px;
    border: 1.5px solid #5d6061;
    border-radius: 5px;
    font-family: "Poppins", sans-serif;
    color: #5d6061;
}

.input-container2 textarea:focus {
    outline: none;
```

```
    border: 1.5px solid #9508CD;
}


@media screen and (max-width: 768px) {
    .container {
        flex-direction: column;
    }

    .profile {
        margin-bottom: 20px;
        margin-right: 0;
    }

    .account {
        margin: 0;
    }

    .account-header {
        flex-direction: column;
    }

    .account-edit {
        flex-direction: column;
        margin: 10px;
    }

    .input-container {
        margin: 10px;
    }

    .input-container input {
        margin-right: 10px;
    }

    .input-container textarea {
        display: block;
        width: auto;
        margin-right: 10px;
    }

    .input-container2 {
        margin: 10px;
    }

    .input-container2 input {
        margin-right: 10px;
    }

    .input-container2 textarea {
        display: block;
        width: auto;
        margin-right: 10px;
    }
}
```

Страница включает информацию о профиле пользователя, меню навигации и форму для редактирования данных аккаунта. Форма содержит

поля для фамилии, имени, email, номера телефона, информации о себе и выбора страны. Подключается внешний файл стилей и библиотека иконок Font Awesome для дополнительной стилизации.

Бэкенд

В предыдущем разделе мы рассмотрели, как мы создали интерфейс приложения, теперь же нужно разобрать как это всё работает внутри. Для реализации бэканда использовался язык программирования Python.

Авторизация и регистрация, простая аутентификация (auth.py):

```python
from passlib.context import CryptContext
from jose import JWTError, jwt
from datetime import datetime, timedelta
from fastapi import Depends, HTTPException, status
from fastapi.security import OAuth2PasswordBearer

SECRET_KEY = "your_secret_key"
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

def verify_password(plain_password, hashed_password):
    return pwd_context.verify(plain_password, hashed_password)

def get_password_hash(password):
    return pwd_context.hash(password)
```

Авторизация и регистрация, простая аутентификация (database.py):

```python
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

# Параметры подключения к базе данных PostgreSQL
SQLALCHEMY_DATABASE_URL =
"postgresql://pan_user:pan_password@172.17.0.2:5432/pan_db"

# Создание движка базы данных
engine = create_engine(SQLALCHEMY_DATABASE_URL)
# Создание фабрики сессий
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

# Базовый класс для всех моделей
Base = declarative_base()

# Функция для получения сессии базы данных
def get_db():
    db = SessionLocal()
```

```
    try:
        yield db
    finally:
        db.close()
```

Код, осуществляющий работу мессенджера (main.py):

```
from fastapi import FastAPI, Request, Depends, HTTPException, status, Form,
Cookie
from fastapi.responses import HTMLResponse, RedirectResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from sqlalchemy.orm import Session
from sqlalchemy import or_, and_
import models, schemas, auth, database
import requests  # Импортируем requests для выполнения HTTP запросов
from fastapi.security import OAuth2PasswordRequestForm
from datetime import datetime, timedelta
from typing import Union

# Нужно для сохранения токеты
TOKEN = None
USER = None

# Создание таблиц в базе данных

models.Base.metadata.create_all(bind=database.engine)

app = FastAPI()

app.mount("/static", StaticFiles(directory="./static"), name="static")
app.mount("/images", StaticFiles(directory="./images"), name="images")

templates = Jinja2Templates(directory="./pages")

# Дальше роуты

# Главная страница
@app.get("/", response_class=HTMLResponse)
def read_item(request: Request, usr_id: Union[str, None] = Cookie(default=None)):
    global USER

    if usr_id is not None:
        template = "index.html"
        context = {"request": request, 'user': USER}

        return templates.TemplateResponse(template, context)

    return RedirectResponse("/auth")  # Перенаправление на страницу входа

# Страница сообщений
@app.get("/messanger/", response_class=HTMLResponse)
def read_item(request: Request, db: Session = Depends(database.get_db), usr_id:
Union[str, None] = Cookie(default=None)):
    if usr_id is not None:
        user = db.query(models.User).filter(models.User.id == usr_id).first()
        users = db.query(models.User).filter(models.User.id != usr_id).all()
```

```
        template = "messanger.html"
        context = {"request": request, 'users': users,'user': user, 'iid': None}
        return templates.TemplateResponse(template, context)

    return RedirectResponse("/auth")  # Перенаправление на страницу входа

@app.get("/messanger/{id}", response_class=HTMLResponse)
def read_item(request: Request, id: int, db: Session = Depends(database.get_db),
usr_id: Union[str, None] = Cookie(default=None)):
    if usr_id is not None:
        user = db.query(models.User).filter(models.User.id == usr_id).first()
        users = db.query(models.User).filter(models.User.id != usr_id).all()

        # Модель собеседника
        iid = db.query(models.User).filter(models.User.id == id).first()

        # Все сообщения
        messages = db.query(models.Message).filter(
            or_(
                and_(models.Message.sender_id == usr_id,
models.Message.recipient_id == iid.id),
                and_(models.Message.sender_id == iid.id,
models.Message.recipient_id == usr_id)
            )
        ).all()


        template = "messanger.html"
        context = {"request": request, 'users': users,'user': user, 'iid': iid,
'messages': messages}
        return templates.TemplateResponse(template, context)

    return RedirectResponse("/auth")  # Перенаправление на страницу входа

# Страница о проекте
@app.get("/about", response_class=HTMLResponse)
def read_item(request: Request):
    template = "about.html"
    context = {"request": request}

    return templates.TemplateResponse(template, context)

@app.get("/profile", response_class=HTMLResponse)
def profile(request: Request, db: Session = Depends(database.get_db), usr_id:
Union[str, None] = Cookie(default=None)):
    if usr_id:
        user = db.query(models.User).filter(models.User.id == usr_id).first()
        template = "profile.html"
        context = {"request": request, 'user': user}

        return templates.TemplateResponse(template, context)

    return RedirectResponse("/auth")  # Перенаправление на страницу входа

# Страница настройки
@app.get("/settings", response_class=HTMLResponse)
def read_item(request: Request, usr_id: Union[str, None] = Cookie(default=None)):
    global USER
```

```
    if usr_id is not None:
        template = "settings.html"
        context = {"request": request, 'user': USER}

        return templates.TemplateResponse(template, context)

    return RedirectResponse("/auth")  # Перенаправление на страницу входа

# Страница авторизации
@app.get("/auth", response_class=HTMLResponse)
def read_item(request: Request, usr_id: Union[str, None] = Cookie(default=None)):
    global USER

    if usr_id is None:
        template = "auth.html"
        context = {"request": request}

        return templates.TemplateResponse(template, context)

    return RedirectResponse("/profile")

# Страница регистрации
@app.get("/registration", response_class=HTMLResponse)
def read_item(request: Request, usr_id: Union[str, None] = Cookie(default=None)):
    if usr_id is None:
        template = "registration.html"
        context = {"request": request}

        return templates.TemplateResponse(template, context)

    return RedirectResponse("/profile")

# Регистрация пользователя
@app.post("/register") # response_model=schemas.User
def read_item(username: str = Form(...), email: str = Form(...), password: str =
Form(...), db: Session = Depends(database.get_db), usr_id: Union[str, None] =
Cookie(default=None)):
    if usr_id is None:
        db_user = db.query(models.User).filter(models.User.username ==
username).first()
        if db_user:
            raise HTTPException(status_code=400, detail="Username already
registered")
        hashed_password = auth.get_password_hash(password)
        db_user = models.User(username=username, email=email,
hashed_password=hashed_password)
        db.add(db_user)
        db.commit()
        db.refresh(db_user)

        # Получаем модель нашего пользователя
        user = db.query(models.User).filter(models.User.username ==
username).first()

        if user:
            usr_id = user.id

            resp = RedirectResponse("/profile", status_code=302)
            resp.set_cookie(key="usr_id", value=usr_id)
```

```
            return resp

    else:
        return RedirectResponse("/")
        # return db_user

# Новый маршрут для аутентификации и сохранения токена
@app.post("/login")
def read_item(username: str = Form(...), password: str = Form(...), db: Session =
Depends(database.get_db), usr_id: Union[str, None] = Cookie(default=None)):
    global TOKEN
    global USER

    if usr_id is None:
        user = db.query(models.User).filter(models.User.username ==
username).first()

        if auth.verify_password(password, user.hashed_password):
            if user:
                # USER = user
                usr_id = user.id

                resp = RedirectResponse("/profile", status_code=302)
                resp.set_cookie(key="usr_id", value=usr_id)

                return resp

    else:
        return RedirectResponse("/")

# Функа выхода
@app.get("/logout")
def logout(request: Request, usr_id: Union[str, None] = Cookie(default=None)):
    global USER

    if USER is not None:
        USER = None

    resp = RedirectResponse("/")
    resp.delete_cookie(key="usr_id")
    return resp

# Отправляем сообщение
@app.post("/sendmessage")
def send_message(from_user_id: str = Form(...), text_message: str = Form(...),
db: Session = Depends(database.get_db), usr_id: Union[str, None] =
Cookie(default=None)):
    if usr_id is not None:
        new_message = models.Message(sender_id=usr_id, recipient_id=from_user_id,
content=text_message)
        db.add(new_message)
        db.commit()
        db.refresh(new_message)

        return RedirectResponse("/messanger/" + from_user_id, status_code=302)

    return RedirectResponse("/auth")  # Перенаправление на страницу входа
```

Код, осуществляющий работу мессенджера (models.py):

```
from sqlalchemy import Column, Integer, String, ForeignKey
from sqlalchemy.orm import relationship
from database import Base

# Тут описываются модели данных (на их основе создаются таблицы)

class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True)
    email = Column(String, unique=True, index=True)
    hashed_password = Column(String)

    # sent_messages = relationship("Message", foreign_keys="[Message.sender_id]",
back_populates="sender")
    # received_messages = relationship("Message",
foreign_keys="[Message.recipient_id]", back_populates="recipient")

# Модель сообщений
class Message(Base):
    __tablename__ = "messages"

    id = Column(Integer, primary_key=True, index=True)
    sender_id = Column(Integer, )#ForeignKey("users.id"))
    recipient_id = Column(Integer,)# ForeignKey("users.id"))
    content = Column(String)

    # sender = relationship("User", foreign_keys=[sender_id],
back_populates="sent_messages")
    # recipient = relationship("User", foreign_keys=[recipient_id],
back_populates="received_messages")
```

Код, осуществляющий работу мессенджера (schemas.py):

```
from pydantic import BaseModel

# Схемы данных

class UserCreate(BaseModel):
    username: str
    email: str
    password: str

class User(BaseModel):
    id: int
    username: str
    email: str

    class Config:
        orm_mode = True


class Message(BaseModel):
    sender_id: int
    recipient_id: int
    content: str
```

```
    class Config:
        orm_mode = True
```

Двигайся в ритме новостей и общения!

Пока мы подходим к концу этого волшебного путешествия по миру Pulse, не забудь, что здесь всегда найдется место для новых открытий и приключений! От рассказов в чатах до свежих новостей в ленте, каждый день здесь наполнен возможностями узнать что-то новое и встретить интересных людей.

Так что держи руку на пульсе времени, открывай новые горизонты и наслаждайся каждым мгновением, проведенным в мире NEONinst. Помни, что ты - часть этого удивительного сообщества, и твои возможности здесь бесконечны!

Погружайся во все, что предлагает NEONinst, и создавай свою собственную историю в этом захватывающем мире общения и новостей. Встречай новых друзей, делай открытия и оставайся всегда на волне высоких эмоций!

До встречи в следующем приключении, где NEONinst продолжит вдохновлять и удивлять тебя! Всегда будь в центре событий и пульсируй в ритме новостей и общения!

С любовью и энтузиазмом,

Команда NEONinst