**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформаційних систем та технологій**

|  |  |
| --- | --- |
|  |  |

**Курсова робота**

**з дисципліни «Програмування»**

**на тему: «Оренда квартир»**

Виконав:

студент 1 курсу, групи ІА-33

Ничик Олександр Олегович \_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник:

асистент кафедри ІСТ

Мягкий Михайло Юрійович \_\_\_\_\_\_\_\_\_\_

(підпис)

Засвідчую, що у цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_

(підпис)

Зміст

[ВСТУП 2](#_Toc168336176)

[1 ВИМОГИ ДО СИСТЕМИ 4](#_Toc168336177)

[1.1 Функціональні вимоги до системи 4](#_Toc168336178)

[1.2 Нефункціональні вимоги до системи 5](#_Toc168336179)

[2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ 6](#_Toc168336180)

[2.1 Діаграма прецедентів 6](#_Toc168336181)

[2.2 Опис сценаріїв використання системи 7](#_Toc168336182)

[3 АРХІТЕКТУРА СИСТЕМИ 12](#_Toc168336183)

[4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ 14](#_Toc168336184)

[4.1 Загальна структура проекту 15](#_Toc168336185)

[4.2 Компоненти рівня доступу до даних 16](#_Toc168336186)

[4.3 Компоненти рівня бізнес-логіки 16](#_Toc168336187)

[4.4 Компоненти рівня інтерфейсу користувача 17](#_Toc168336188)

[ВИСНОВОК 18](#_Toc168336189)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 19](#_Toc168336190)

[ДОДАТОК А Лістінг програми 20](#_Toc168336191)

ВСТУП

Стрімкий розвиток технологій так само стрімко змінював та змінює наші життя. Почали розвиватись зовсім невідомі раніше напрямки та значно покращуватись вже відомі. В ту саму мить ті, що застаріли, поступово зникали та продовжують зникати з нашого життя, поступаючись своїм місцем більш сучасним та ефективним рішенням. Одним із найбільш помітних прикладів даного розвитку є цифровізація величезної частини нашого життя, наприклад ринку нерухомості.

Останнім часом світ часто стикається з величезною кількістю викликів. Зовсім нещодавно була пандемія COVID-19. Ця подія спричинила багато змін і зрушила багато ідей, речей та проектів з місця. Вона змусила нас адаптуватись під абсолютно нові для нас умови та переглянути всі аспекти нашого життя. Різноманітні обмеження підштовхнули розвиток дистанційних роботи та навчання, а також збільшили попит на оренду квартир, особливо тих, що відповідають умовам для роботи з дому.

Війна в Україні також внесла величезні корективи у наші життя. У нас з’явилась необхідність вирішити проблему з проживанням сотень тисяч людей, що були змушені покинути свої домівки через загрози життю. Через необхідність в цьому, ми стали також потребувати перегляд технологій, що використовувались до цього, їхню модернізацію та створення нових. Люди, що тікали якомога далі від небезпечних територій, потребували швидкого та ефективного способу знайти нове житло. Через це, попит на різного виду інтернет-платформи стрімко зріс.

З технологіями, що ми маємо зараз, набагато легше створювати зручні та ефективні рішення для оренди житла. Інтернет-ресурси, що існують для оренди квартир пропонують дуже зручні та швидкі підходи для користувача, що може вільно шукати все, що йому смакує.

Саме тому, можемо зробити висновок, що створення сайту для оренди квартир у сучасних умовах є актуальним і необхідним завданням. Ця робота спрямована на розробку платформи, що буде дозволяти користувачу зручно та ефективно користуватись перевагами сучасних технологій у вигляді сайту, який буде пропонувати йому швидкий та безпечний спосіб пошуку житла, що відповідає їхнім потребам.

# 1 ВИМОГИ ДО СИСТЕМИ

## Функціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:  
–незареєстрований користувач не може переглядати інформацію про квартири

–зареєстрований користувач повинен мати можливість переглядати інформацію про квартири

–зареєстрований користувач може шукати квартиру за параметрами

–зареєстрований користувач може здати в оренду власну квартиру

–хазяїн квартири має такі ж можливості, як і зареєстрований користувач

## 1.2 Нефункціональні вимоги до системи

Система має відповідати наступним функціональним вимогам:

– система повинна мати відкриту архітектуру;

– система повинна мати веб-інтерфейс;

– інтерфейс користувача має бути зручним та інтуїтивно-зрозумілим;

– система повинна бути крос-платформною.

# 2 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

## 2.1 Діаграма прецедентів

Діаграма прецедентів зображена на рисунку 2.1. Актором є користувач, який одночасно може бути як клієнтом, так і хазяїном. Нижче, в пункті 2.2, будуть наведені всі використовувані сценарії.

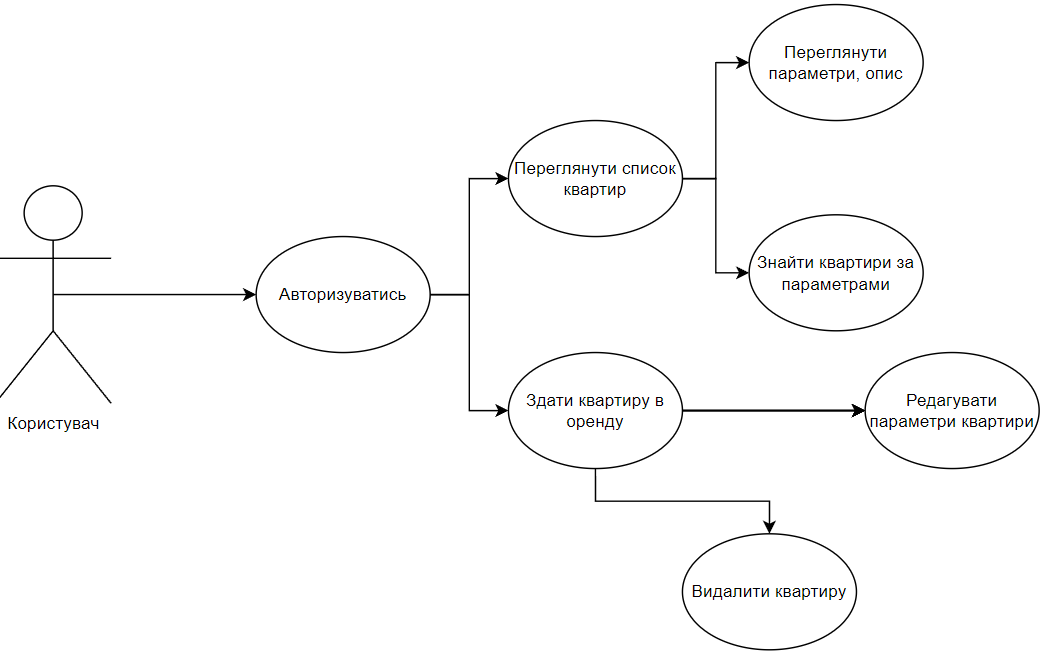


Рисунок 2.1 – Діаграма прецедентів

## 2.2 Опис сценаріїв використання системи

Детальні описи сценаріїв використання наведено у таблицях 2.1 – 2.5.

Таблиця 2.1 – Сценарій використання «Переглянути список квартир»

|  |  |
| --- | --- |
| Назва | Переглянути список квартир |
| ID | 1 |
| Опис | Користувач натискає на посилання в шапці сайту, що є доступним на будь-якій сторінці та переходить до сторінки з списком всіх квартир |
| Актори | Користувач |
| Вигоди компанії | Без можливості переглядати квартири, проект втрачає свій сенс і головну ціль |
| Частота користування | Постійно |
| Тригери | Клієнт переходить на відповідну сторінку |
| Передумови | Посилання на сторінку зі списком всіх квартир, що є доступним у будь–якому вікні |
| Постумови | Клієнт потрапляє на сторінку зі списком всіх квартир |
| Основний розвиток | Клієнт натискає на посилання на сторінку зі всіма квартирами, відбувається перехід на сторінку з квартирами, де клієнт може бачити список всіх квартир |
| Альтернативні розвитки | У випадку якщо користувач не є авторизованим, його буде перенаправляти на сторінку авторизації |
| Виняткові ситуації | – |

Таблиця 2.2 – Сценарій використання «Пошук квартири за параметрами»

|  |  |
| --- | --- |
| Назва | Пошук квартири за параметрами |
| ID | 2 |
| Опис | Користувач вводить значення параметрів, що його цікавлять, в форму та отримує список з квартир, що підходять за цими параметрами |
| Актори | Користувач |
| Вигоди компанії | Можливість шукати квартири за параметрами є необхідною для комфортного використання проекту |
| Частота користування | Часто |
| Тригери | Клієнт переходить на відповідну сторінку |
| Передумови | На сторінці з усіма квартирами маємо форми, що отримують параметри для фільтрації, що застосовуються після натискання на відповідну кнопку форми |
| Постумови | Клієнт потрапляє на сторінку зі списком всіх квартир |
| Основний розвиток | Клієнт натискає на посилання на сторінку зі всіма квартирами, відбувається перехід на сторінку з квартирами, де клієнт може бачити список всіх квартир |
| Альтернативні розвитки | – |
| Виняткові ситуації | Якщо будуть введені некоректні дані, користувача змусять заповнити форму ще раз |

Таблиця 2.3 – Сценарій використання «Створення квартири»

|  |  |
| --- | --- |
| Назва | Створення квартири |
| ID | 3 |
| Опис | Користувач створює квартиру для оренди, заповнюючи поля та натискаючи кнопку. |
| Актори | Користувач |
| Вигоди компанії | Можливість шукати квартири за параметрами є необхідною для комфортного використання проекту |
| Частота користування | Часто |
| Тригери | Клієнт переходить на відповідну сторінку |
| Передумови | На сторінці з усіма квартирами маємо форми, що отримують параметри для фільтрації, що застосовуються після натискання на відповідну кнопку форми |
| Постумови | Клієнт потрапляє на сторінку зі списком всіх квартир |
| Основний розвиток | Клієнт натискає на посилання на сторінку зі всіма квартирами, відбувається перехід на сторінку з квартирами, де клієнт може бачити список всіх квартир |
| Альтернативні розвитки | – |
| Виняткові ситуації | Якщо будуть введені некоректні дані, користувача змусять заповнити форму ще раз |

Таблиця 2.4 – Сценарій використання «Логін»

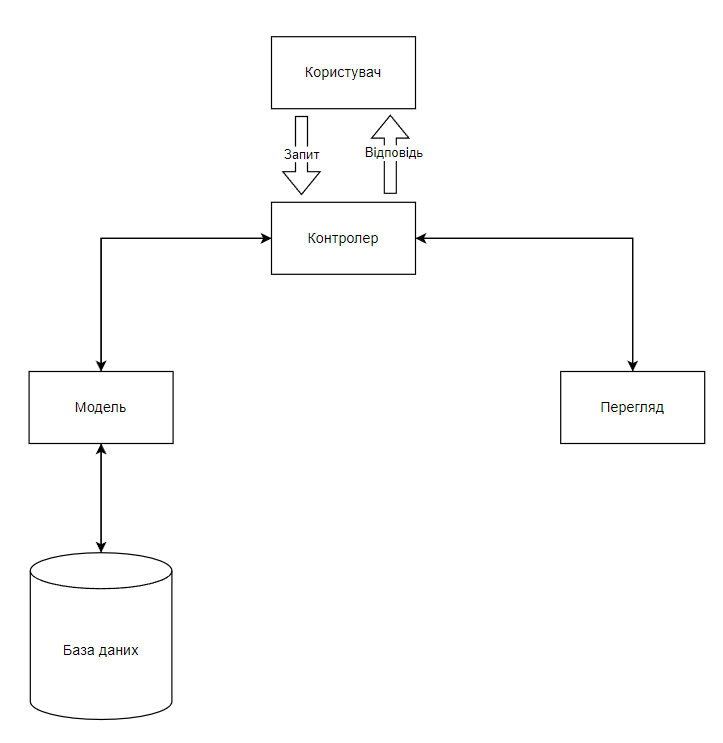
|  |  |
| --- | --- |
| Назва | Логін |
| ID | 4 |
| Опис | Користувач входить у свій існуючий аккаунт |
| Актори | Користувач |
| Вигоди компанії | Авторизація важлива для того, аби мати якомога більше інформації про користувачів. Також, можна застосувати для певної розсилки реклами проекту на електронну пошту користувача |
| Частота користування | Постійно |
| Тригери | Користувач переходить на відповідну сторінку, що спрацьовує тільки тоді, коли користувач не авторизований |
| Передумови | У користувача відсутній токен авторизації, що призводить до того, що користувач не є авторизованим і не може використовувати функціонал проекту |
| Постумови | Користувач потрапляє на сторінку з авторизацією |
| Основний розвиток | Користувач заповнює форму і відправляє її на обробку |
| Альтернативні розвитки | Якщо користувач не має власного аккаунту, він має можливість перейти на сторінку реєстрації під формою |
| Виняткові ситуації | Якщо користувач неправильно введе дані, його змусять заповнити форму ще раз |

Таблиця 2.5 – Сценарій використання «Реєстрація»

|  |  |
| --- | --- |
| Назва | Реєстрація |
| ID | 5 |
| Опис | Користувач створює новий аккаунт |
| Актори | Користувач |
| Вигоди компанії | Реєстрація необхідна для авторизації, а авторизація має вигоди, перелічені пунктом вище |
| Частота користування | Постійно |
| Тригери | Користувач переходить на відповідну сторінку |
| Передумови | Користувач не має власного аккаунту, тому він переходить по посиланню під формою авторизації |
| Постумови | Користувач потрапляє на сторінку з реєстрацією |
| Основний розвиток | Користувач заповнює форму і відправляє її на обробку. |
| Альтернативні розвитки | Якщо користувач не має власного аккаунту, він має можливість перейти на сторінку реєстрації під формою. |
| Виняткові ситуації | Якщо будуть введені некоректні дані, користувача змусять заповнити форму ще раз |

# 3 АРХІТЕКТУРА СИСТЕМИ

Загальна архітектура системи наведена на рис. 3.1



Таблиця 3.1 – Загальна архітектура системи

Система складається з наступних елементів:

– графічний інтерфейс;

– серверна частина;

– база даних.

Графічний інтерфейс необхідний для взаємодії з користувачем. HTTP запит надходить до серверної частини, де оброблюється і повертається відповідь. На серверній частині виконується основна логіка системи. Дані, отриманні з графічного інтерфейсу валідуються, конвертуються. Також, серверна частина формує запит до бази даних та оброблює відповідь і передає її до графічного інтерфейсу. База даних зберігає дані, які були сформовані на серверній частині та повертає їх у разі запиту.

До серверної частини належать наступні елементи:

– контролер;

– модель та вигляд;

– сервіс;

– репозиторій.

На контролер надходять дані з графічного інтерфейсу. З контролеру, дані формуються в сервісі для запиту в репозиторій. З репозиторію дані надсилаються до бази даних і зберігаються. Також в контролері формується вид, тобто об’єкт і його ім’я для відображення на графічному інтерфейсі.

# 4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ СИСТЕМИ

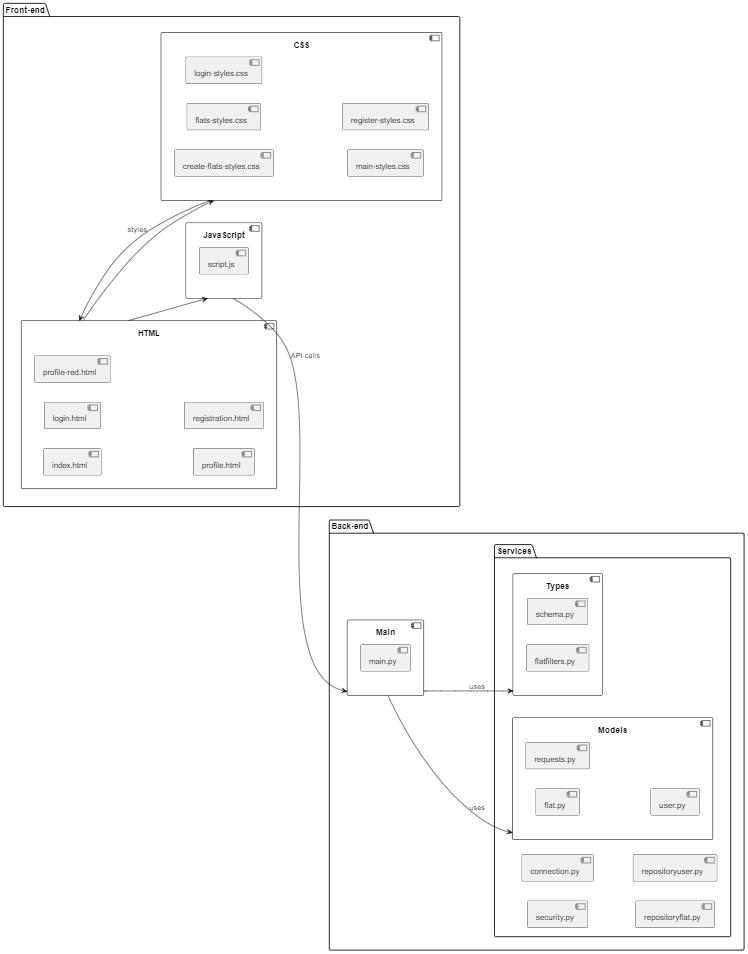


Рисунок 4.0 – Загальна реалізація всіх компонентів системи

## 4.1 Загальна структура проекту

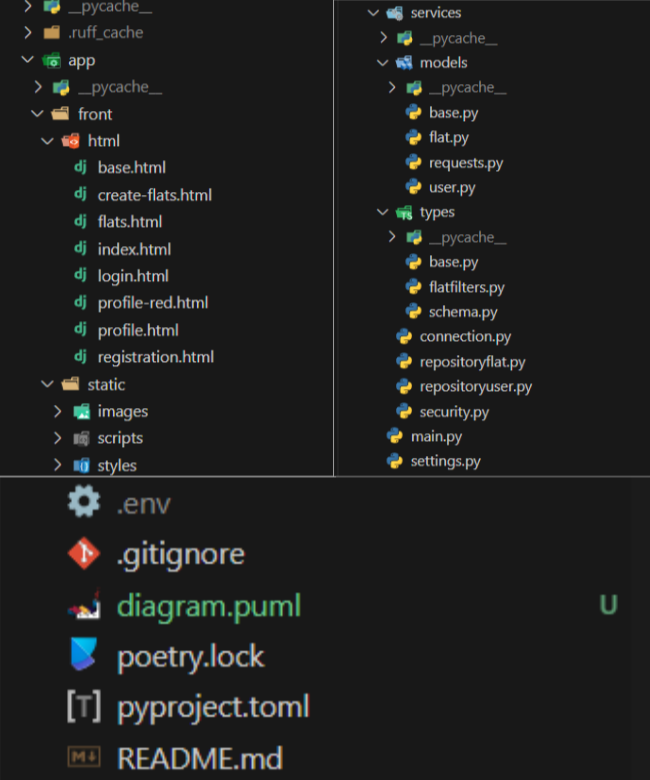


Рисунок 4.1 – Загальна структура проекту

Проект складається з веб-ресурсів, бібліотек та вихідного коду, який в свою чергу можна поділити на компоненти рівня доступу до даних, компоненти бізнес-логіки та веб-компоненти

Веб-ресурси включають HTML файли, які забезпечують структуру веб-сторінок (index.html, login.html, profile.html, registration.html), CSS файли, які забезпечують стилізацію веб-сторінок (create-flats-styles.css, flats-styles.css, login-styles.css, main-styles.css, register-styles.css), та JavaScript файли, які забезпечують інтерактивність та динамічну поведінку веб-сторінок (script.js).

Бібліотеки включають сторонні бібліотеки та фреймворки, необхідні для роботи проекту. Це Python бібліотеки на бекенді.

## 4.2 Компоненти рівня доступу до даних

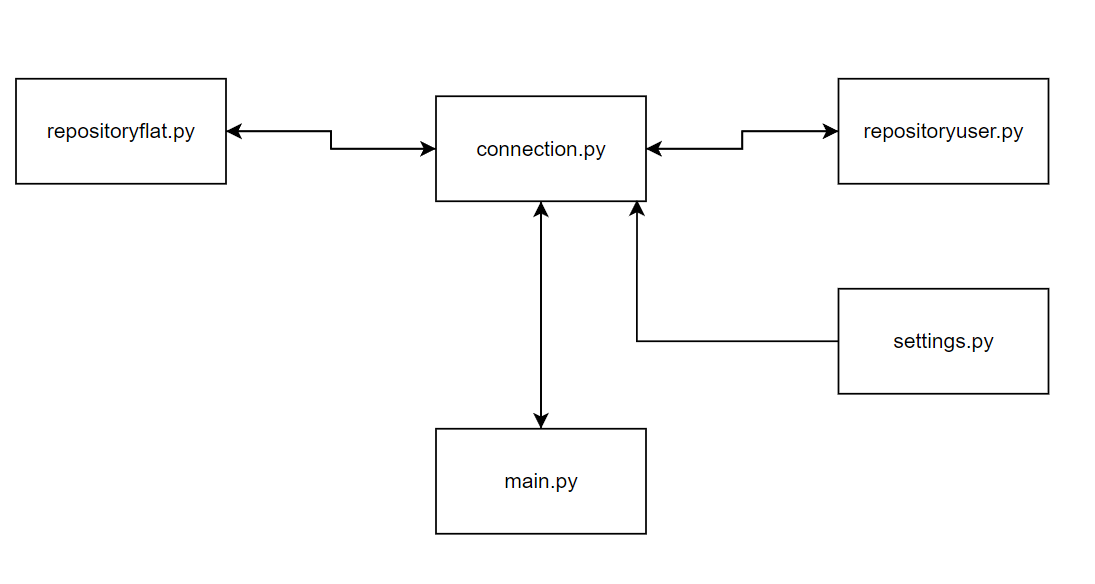


Рисунок 4.2 – Основні сутності та інтерфейси рівня доступу до даних

## 4.3 Компоненти рівня бізнес-логіки

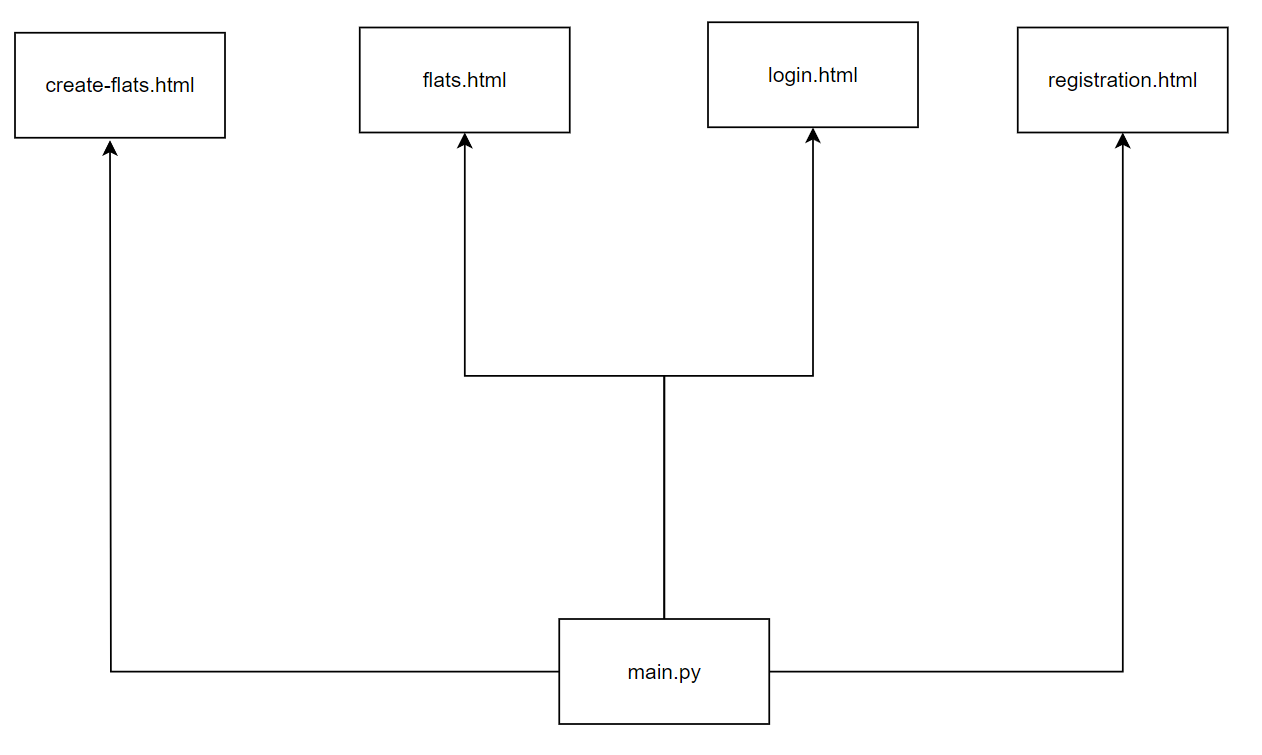


Рисунок 4.3 – Основні сутності рівня бізнес-логіки

## 4.4 Компоненти рівня інтерфейсу користувача

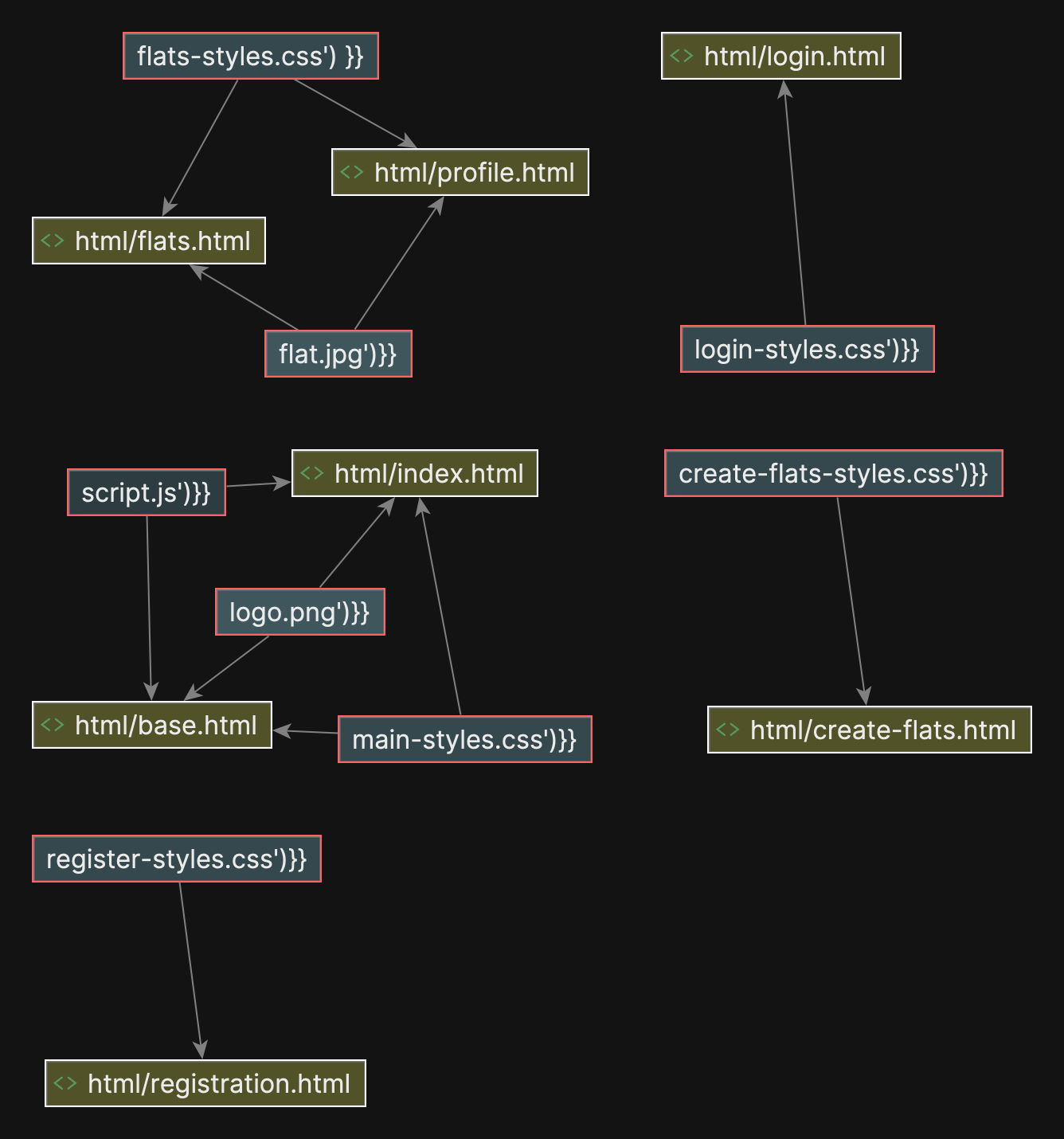


Рисунок 4.4 – Основні сутності рівня інтерфейсу користувача

# ВИСНОВОК

Під час виконання даної роботи ми переглянули велику кількість ресурсів для того, аби зрозуміти що і навіщо є в інтернет-проектах для оренди квартир. Ми проаналізували багато сучасних підходів до оренди квартир.

Перш за все, ми переглянули технічне завдання та сформулювали функціональні та нефункціональні вимоги до системи. Далі, ми обрали технології та мову програмування. Нами була обрана мова Python для роботи сервера, а точніше фреймворк FastAPI. Ми обрали цей фреймворк через те, що він пропонує нам швидко створювати функціонал, необхідний нам, тестувати, вносити зміни майже миттєво. Для самої структури графічного інтерфейсу та його зв’язок з сервером ми використали HTML, CSS та JavaScript. В якості бази даних ми використовували PostgreSQL, через свою зручність та можливість інтуїтивного використання користувацького інтерфейсу. Спочатку планувалось використання бази даних Redis, але ця база даних має багато обмежень, наприклад відсутність структури, обмежена масштабованість та необхідність коштів для великої кількості даних. Саме через те, що PostgreSQL таких мінусів не має, ми обрали її.

Після всього вище переліченого ми описали сценарії використання нашої системи. Через цей сценарій нам вдалось успішно та безпомилково реалізувати необхідний нам продукт. В результаті ми отримали застосунок, що підтримує функціонал перегляду квартир користувачем, оренди квартир та здачі квартир в оренду. Також, ми внесли в даний проект авторизацію, що хоч і не було вимогою в технічному завданні, але додало проекту зручності використання.

Підсумовуючи, можемо сказати, що наш проект був успішно спланований та створений для комфортного використання користувачами. Цей проект має багато переваг, таких як зручність читання, зрозумілість синтаксису та загального принципу роботи, але все ще має куди розвиватись. В майбутньому планується додати покращену архітектуру та додавання нових можливостей.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація FastAPI: <https://fastapi.tiangolo.com/>
2. Кольори: <https://visme.co/blog/website-color-schemes/>
3. Приклад для оформлення форм: <https://www.olx.ua/uk/>
4. Документація Python: <https://docs.python.org/3/>
5. Документація Python українською: <https://github.com/python/python-docs-uk>
6. Документація JavaScript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
7. Документація Jinja2Templates: <https://fastapi.tiangolo.com/advanced/templates/>

# ДОДАТОК А Лістінг програми

<https://github.com/iamthegoose/first-course>

base.html

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <link  
 rel="stylesheet"  
 href="{{ url\_for('static', path='styles/main-styles.css')}}"  
 />  
 {% block head %} {% endblock %}  
 <title>Ecstatic Estate</title>  
 </head>  
 <body>  
 <header id="header" class="top">  
 <div class="container head">  
 <a href="/">  
 <img  
 class="logo-black"  
 src="{{ url\_for('static', path='images/logo.png')}}"  
 alt="Logo"  
 height="50px"  
 /></a>  
 <nav class="menu">  
 <ul class="menu-list">  
 <li class="menu-item">  
 <a class="menu-item-link menu-item-rent" href="/rent"  
 >Орендувати</a  
 >  
 </li>  
 <li class="menu-item">  
 <a  
 class="menu-item-link menu-item-create-rent"  
 href="/create-flats"  
 >Здати в оренду</a  
 >  
 </li>  
   
 </ul>  
 </nav>  
 </div>  
 </header>  
  
 {% block body %} {% endblock %}  
  
 <footer class="footer">  
 <div class="container flex-footer">  
 <div class="footer-information">  
 <a class="information-logo-link" href=""> </a>  
 <p class="information-text">  
 Знайдіть житло, що буде підходити саме вам, тут і зараз!  
 </p>  
 </div>  
 <div class="footer-social">  
 <p class="footer-social-text">Соціальні мережі</p>  
 <ul class="footer-social-list">  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#instagram')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#twitter')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#facebook')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#linkedin')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 </ul>  
 </div>  
 <div class="footer-user-contact">  
 <p class="footer-social-text">Підпишіться!</p>  
 <form class="footer-form">  
 <label class="footer-form-label">  
 <input  
 class="footer-form-input"  
 type="text"  
 name="footer\_email"  
 placeholder="E-mail"  
 />  
 </label>  
 <button class="footer-form-btn" type="submit">  
 Підписатись  
 <svg class="footer-form-btn-icon" width="24" height="20">  
 <use  
 href="{{ url\_for('static', path='images/icons/send.svg')}}"  
 ></use>  
 </svg>  
 </button>  
 </form>  
 </div>  
 </div>  
 </footer>  
  
 <script src="{{ url\_for('static', path='scripts/script.js')}}"></script>  
 </body>  
</html>

create-flats.html

{% extends "base.html" %} {% block head %}  
<link  
 rel="stylesheet"  
 href="{{ url\_for('static', path='styles/create-flats-styles.css')}}"  
/>  
{% endblock head%} {% block body %}  
 <section class="create-flats-section section">  
 <div class="container">  
 <div class="box-create-flats">  
 <form id="createFlatForm" class="create-flats-form">  
 <input  
 type="text"  
 id="name"  
 name="name"  
 placeholder="Назва"  
 class="create-flats-input"  
 required  
 />  
 <input  
 type="text"  
 id="location"  
 name="location"  
 placeholder="Розташування"  
 class="create-flats-input"  
 required  
 />  
 <input  
 type="text"  
 id="description"  
 name="description"  
 placeholder="Опис"  
 class="create-flats-input"  
 required  
 />  
 <input  
 type="text"  
 id="area"  
 name="area"  
 placeholder="Площа"  
 class="create-flats-input"  
 required  
 />  
 <input  
 type="text"  
 id="price"  
 name="price"  
 class="create-flats-input"  
 placeholder="Ціна за місяць"  
 required  
 />  
 <input  
 type="text"  
 id="rooms"  
 name="rooms"  
 class="create-flats-input"  
 placeholder="Кількість кімнат"  
 required  
 />  
 <button id="submitBtn" class="create-flats-btn">  
 Здати в оренду  
 </button>  
 </form>  
 </div>  
 </div>  
 </section>  
  
<script>  
 const ***createFlatForm*** = ***document***.getElementById("createFlatForm")**;  
 *createFlatForm***.addEventListener("submit"**,** handleSubmit)**;** function handleSubmit(e) {  
 e.preventDefault()**;** const token = ***localStorage***.getItem("token")**;** const formData = {  
 name: ***document***.getElementById("name").value**,** location: ***document***.getElementById("location").value**,** description: ***document***.getElementById("description").value**,** area: ***document***.getElementById("area").value**,** price: ***document***.getElementById("price").value**,** rooms: ***document***.getElementById("rooms").value**,** }**;** fetch("/user/create-flat"**,** {  
 method: "POST"**,** headers: {  
 "Content-Type": "application/json"**,** Authorization: "Bearer " + ***localStorage***.getItem("token")**,** }**,** body: ***JSON***.stringify(formData)**,** })  
 .then((response) => {  
 if (response.ok) {  
 return response.json()**;** } else {  
 return response.text().then((text) => {  
 throw new Error(text)**;** })**;** }  
 })  
 .then((data) => {  
 ***window***.location.href = "/rent"**;** })  
 .catch((error) => {  
 alert(error.message)**;  
 *console***.error("Error:"**,** error)**;** })**;** }  
</script>  
  
{% endblock body %}

flats.html

{% extends "base.html" %} {% block head %}  
<link  
 rel="stylesheet"  
 href="{{ url\_for('static', path='styles/flats-styles.css') }}"  
/>  
{% endblock head %} {% block body %}  
  
<section class="filter-section">  
 <div class="container">  
 <div class="filter">  
 <form class="filter-form" method="post">  
 <label>  
 <p class="label-txt">Загальна площа</p>  
 <div class="input-group">  
 <input  
 type="text"  
 placeholder="від"  
 class="input-field"  
 name="area\_from"  
 id="area\_from"  
 inputmode="numeric"  
 pattern="[0-9]\*"  
 />  
 <input  
 type="text"  
 placeholder="до"  
 class="input-field"  
 name="area\_to"  
 id="area\_to"  
 inputmode="numeric"  
 pattern="[0-9]\*"  
 />  
 </div>  
 </label>  
  
 <label>  
 <p class="label-txt">Ціна</p>  
 <div class="input-group">  
 <input  
 type="text"  
 placeholder="від"  
 class="input-field"  
 name="price\_from"  
 id="price\_from"  
 inputmode="numeric"  
 pattern="[0-9]\*"  
 />  
 <input  
 type="text"  
 placeholder="до"  
 class="input-field"  
 name="price\_to"  
 id="price\_to"  
 inputmode="numeric"  
 pattern="[0-9]\*"  
 />  
 </div>  
 </label>  
  
 <label for="">  
 <p class="label-txt">Кількість кімнат</p>  
 <select class="select-form" name="rooms" id="rooms">  
 <option value="1">1</option>  
 <option value="2">2</option>  
 <option value="3">3</option>  
 <option value="4">4</option>  
 <option value="5">5</option>  
 </select>  
 </label>  
  
 <button class="submit-filter-btn">Застосувати</button>  
 </form>  
 </div>  
 <hr />  
 </div>  
 <section class="section">  
 <div class="container">  
 <div id="flats-container">  
 <ul class="list-flats">  
 {% for flat in flats %}  
 <li class="flats-item">  
 <h4 class="flats-item-title">{{ flat.name }}</h4>  
 <img  
 src="{{ url\_for('static', path='images/flat.jpg')}}"  
 alt="Flat Image"  
 class="flats-item-image"  
 />  
 <div class="flats-item-content">  
 <p class="flats-item-text">{{ flat.description }}</p>  
 <p class="flats-item-text"><b>{{ flat.location }}</b></p>  
 <p class="flats-item-text">Площа: {{ flat.area }} кв. м.</p>  
 <p class="flats-item-text">Ціна: {{ flat.price }} ₴/місяць</p>  
 <p class="flats-item-text">Кількість кімнат: {{ flat.rooms }}</p>  
 </div>  
 </li>  
  
 {% endfor %}  
 </ul>  
   
 </section>  
 </div>  
 </div>  
</section>  
<script>  
 document  
 .querySelector(".filter-form")  
 .addEventListener("submit"**,** async function (event) {  
 event.preventDefault()**;** const formData = {  
 area\_from: document.getElementById("area\_from").value**,** area\_to: document.getElementById("area\_to").value**,** price\_from: document.getElementById("price\_from").value**,** price\_to: document.getElementById("price\_to").value**,** rooms: document.getElementById("rooms").value**,** }**;** console.log(JSON.stringify(formData))**;** try {  
 const response = await fetch("/user/flats"**,** {  
 method: "POST"**,** headers: {  
 "Content-Type": "application/json"**,** }**,** body: JSON.stringify(formData)**,** })**;** const html = await response.text()**;** const flatsContainer = document.getElementById("flats-container")**;** const tempDiv = document.createElement("div")**;** tempDiv.innerHTML = html**;** const newFlatsList = tempDiv.querySelector(".list-flats")**;** if (newFlatsList && flatsContainer) {  
 flatsContainer.innerHTML = newFlatsList.outerHTML**;** } else {  
 console.error(  
 "Помилка: не вдалося знайти контейнер або список квартир."  
 )**;** }  
 } catch (error) {  
 console.error("Помилка:"**,** error)**;** }  
 })**;**const listFlats = document.querySelector('.list-flats')**;**const listFlatsChildren = listFlats.children**;**function itemClick() {  
 const res = confirm("Орендувати цю квартиру?")**;** if (res) {  
 alert("Заявка подана успішно!")**;** } else {  
 return**;** }  
}  
  
  
  
for(const el of listFlatsChildren) {  
 el.addEventListener('click'**,** itemClick)  
}  
  
  
</script>  
  
{% endblock body %}

index.html

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>Ecstatic Estate</title>  
 <link  
 rel="stylesheet"  
 href="{{ url\_for('static', path='styles/main-styles.css')}}"  
 />  
 </head>  
 <body>  
 <header id="header" class="top">  
 <div class="container head">  
 <a href="/">  
 <img  
 class="logo-black"  
 src="{{ url\_for('static', path='images/logo.png')}}"  
 alt="Logo"  
 height="50px"  
 /></a>  
 <nav class="menu">  
 <ul class="menu-list">  
 <li class="menu-item">  
 <a class="menu-item-link menu-item-rent" href="/rent"  
 >Орендувати</a  
 >  
 </li>  
 <li class="menu-item">  
 <a  
 class="menu-item-link menu-item-create-rent"  
 href="/create-flats"  
 >Здати в оренду</a  
 >  
 </li>  
   
 </ul>  
 </nav>  
 </div>  
 </header>  
 <section class="hero">  
 <div class="">  
 <div class="main-title">  
 <p class="rent-now-text">  
 Живіть в <span class="unical-text">унікальних</span> місцях  
 </p>  
 <form action="/login" id="catalogueForm">  
 <input  
 type="submit"  
 value="КАТАЛОГ"  
 class="rent-now-btn"  
 type="button"  
 />  
 </form>  
 </div>  
 </div>  
 </section>  
 <section class="img-content">  
 <div class="container"></div>  
 </section>  
  
 <footer class="footer">  
 <div class="container flex-footer">  
 <div class="footer-information">  
 <a class="information-logo-link" href=""> </a>  
 <p class="information-text">  
 Знайдіть житло, що буде підходити саме вам, тут і зараз!  
 </p>  
 </div>  
 <div class="footer-social">  
 <p class="footer-social-text">Соціальні мережі</p>  
 <ul class="footer-social-list">  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#instagram')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#twitter')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#facebook')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 <li>  
 <a class="footer-social-link" href="">  
 <svg class="form-icon-check" width="24" height="24">  
 <use  
 href="{{ url\_for('static', path='images/icons/socials-icons.svg#linkedin')}}"  
 ></use>  
 </svg>  
 </a>  
 </li>  
 </ul>  
 </div>  
 <div class="footer-user-contact">  
 <p class="footer-social-text">Підпишіться!</p>  
 <form class="footer-form">  
 <label class="footer-form-label">  
 <input  
 class="footer-form-input"  
 type="text"  
 name="footer\_email"  
 placeholder="E-mail"  
 />  
 </label>  
 <button class="footer-form-btn" type="submit">  
 Підписатись  
 <svg class="footer-form-btn-icon" width="24" height="20">  
 <use  
 href="{{ url\_for('static', path='images/icons/send.svg')}}"  
 ></use>  
 </svg>  
 </button>  
 </form>  
 </div>  
 </div>  
 </footer>  
  
 <script src="{{ url\_for('static', path='scripts/script.js')}}"></script>  
 </body>  
</html>

login.html

{% extends "base.html" %} {% block head %}  
<link  
 rel="stylesheet"  
 href="{{ url\_for('static', path='styles/login-styles.css')}}"  
/>  
{% endblock head%} {% if message %}  
<script>  
 alert(message)**;**</script>  
{%endif%} {% block body %}  
<section class="login-section section">  
 <div class="container">  
 <div class="login-box">  
 <form id="login-form" class="login-form" method="post">  
 <h2>Увійдіть</h2>  
 <input  
 type="email"  
 class="login-input"  
 placeholder="Електронна пошта"  
 name="email"  
 id="email"  
 required  
 />  
 <input  
 type="password"  
 class="login-input"  
 placeholder="Пароль"  
 name="password"  
 id="password"  
 required  
 />  
 <button class="login-btn">Увійти</button>  
 <p>Ще не зареєструвались? Зробіть це <a href="/signup">тут</a>!</p>  
 </form>  
 </div>  
 </div>  
</section>  
<script>  
 ***document*** .querySelector(".login-form")  
 .addEventListener("submit"**,** async function (event) {  
 event.preventDefault()**;** const formData = {  
 email: document.getElementById("email").value**,** password: document.getElementById("password").value**,** }**;** try {  
 const response = await fetch("/user/signin"**,** {  
 method: "POST"**,** headers: {  
 "Content-Type": "application/json"**,** }**,** body: JSON.stringify(formData)**,** })**;** if (response.ok) {  
 const data = await response.json()**;** const token = data.token**;** localStorage.setItem("token"**,** token)**;** window.location.href = `/success?token=${data.token}`**;** } else {  
 const errorMessage = await response.text()**;** alert("Дані введено некоректно!")**;** }  
 } catch (error) {  
 alert("Сталася помилка під час входу.")**;** console.error("Error:"**,** error)**;** }  
 })**;**</script>  
{% endblock body %}

profile.html

{% extends "base.html" %} {% block head %}  
<link  
 rel="stylesheet"  
 href="{{ url\_for('static', path='styles/flats-styles.css') }}"  
/>  
{% endblock head %} {% block body %}  
<section class="section">  
 <div class="container">  
 <h3>Мої квартири</h3>  
 <div id="flats-container">  
 <ul class="list-flats">  
 {% if flats %} {% for flat in flats %}  
 <li class="flats-item">  
 <h4 class="flats-item-title">{{ flat.name }}</h4>  
 <img  
 src="{{ url\_for('static', path='images/flat.jpg')}}"  
 alt=""  
 width="300px"  
 height="150px"  
 />  
 <p class="flats-item-text">{{ flat.description }}</p>  
 <p class="flats-item-text">Площа: {{ flat.area }} кв. м.</p>  
 <p class="flats-item-text">Ціна: {{ flat.price }} ₴/місяць</p>  
 <p class="flats-item-text">Кількість кімнат: {{ flat.rooms }}</p>  
 </li>  
 {% endfor %} {% else %}  
 <p>Ви ще не маєте жодної квартири</p>  
 {% endif %}  
 </ul>  
 </div>  
 </div>  
</section>  
{% endblock body %}

profile-red.html

<script>  
 document.addEventListener("DOMContentLoaded"**,** function () {  
 const token = localStorage.getItem("token")**;** if (token) {  
 window.location.href = "/profile"**;** } else {  
 window.location.href = "/login"**;** }  
 })**;**</script>

registration.html

{% extends "base.html" %} {% block head %}  
<link  
 rel="stylesheet"  
 href="{{ url\_for('static', path='styles/register-styles.css')}}"  
/>  
{% endblock head%} {% block body %}  
<section class="register-section section">  
 <div class="container">  
 <div class="box-register">  
 <form action="" class="register-form">  
 <h4 class="register-title">Зареєструйтесь</h4>  
 <input  
 type="text"  
 placeholder="Імʼя"  
 class="register-input"  
 name="name"  
 id="name"  
 required  
 />  
 <input  
 type="text"  
 placeholder="Прізвище"  
 class="register-input"  
 name="surname"  
 id="surname"  
 required  
 />  
 <input  
 type="email"  
 placeholder="Пошта"  
 class="register-input"  
 name="email"  
 id="email"  
 required  
 />  
 <input  
 type="password"  
 placeholder="Пароль"  
 class="register-input"  
 name="password"  
 id="password"  
 required  
 />  
  
 <button class="register-btn">Зареєструйтесь</button>  
 <p class="register-txt">  
 Вже зареєстровані?  
 <a href="/login">Увійти</a>  
 </p>  
 </form>  
 </div>  
 </div>  
</section>  
  
<script>  
 ***document*** .querySelector(".register-form")  
 .addEventListener("submit"**,** async function (event) {  
 event.preventDefault()**;** const formData = {  
 name: ***document***.getElementById("name").value**,** surname: ***document***.getElementById("surname").value**,** email: ***document***.getElementById("email").value**,** password: ***document***.getElementById("password").value**,** }**;** try {  
 const response = await fetch("/user/signup"**,** {  
 method: "POST"**,** headers: {  
 "Content-Type": "application/json"**,** }**,** body: ***JSON***.stringify(formData)**,** })**;** if (response.ok) {  
 const data = await response.json()**;** const token = data.token**;  
 *localStorage***.setItem("token"**,** token)**;  
 *window***.location.href = `/success?token=${token}`**;** } else {  
 const errorMessage = await response.text()**;** alert(errorMessage)**;** }  
 } catch (error) {  
 alert("Сталася помилка під час реєстрації.")**;  
 *console***.error("Error:"**,** error)**;** }  
 })**;**</script>  
  
{% endblock body %}

script.js

***window***.addEventListener("scroll"**,** function () {  
 const header = ***document***.getElementById("header")**;** if (***window***.scrollY > **0**) {  
 header.classList.add("scrolled")**;** header.classList.remove("top")**;** } else {  
 header.classList.add("top")**;** header.classList.remove("scrolled")**;** }  
})**;**const ***token*** = ***localStorage***.getItem("token")**;**function handleClick(e) {  
 e.preventDefault()**;** if (***token***) {  
 ***window***.location.href = "/rent"**;** } else {  
 ***window***.location.href = "/login"**;** }  
}  
  
const ***catalogue*** = ***document***.getElementById("catalogueForm")**;**const ***itemRent*** = ***document***.querySelector(".menu-item-rent")**;**const ***itemCreateRent*** = ***document***.querySelector(".menu-item-create-rent")**;  
  
*catalogue***.addEventListener("click"**,** handleClick)**;  
*itemRent***.addEventListener("click"**,** handleClick)**;  
*itemCreateRent***.addEventListener("click"**,** itemClick)**;**

create-flats-styles.css

.create-flats-section {  
 margin-top: **200**px**;**}  
  
.box-create-flats {  
 display: flex**;** justify-content: center**;** align-items: center**;**}  
  
.create-flats-form {  
 display: flex**;** justify-content: center**;** align-items: center**;** flex-direction: column**;** gap: **30**px**;** width: **400**px**;** height: **450**px**;** border-radius: **10**px**;** box-shadow: **0**px **0**px **10**px rgba(**0, 0, 0, 0.1**)**;** background-color: **#ffffff;**}  
  
.create-flats-input {  
 width: **300**px**;** height: **30**px**;** border: none**;** padding-left: **10**px**;** border-bottom: solid black **1**px**;**}  
  
.create-flats-btn {  
 width: **200**px**;** height: **30**px**;** border-radius: **5**px**;** border: none**;** transition: **0.3**s**;**}  
  
.create-flats-btn:hover {  
 background-color: rgba(**76, 75, 75, 0.2**)**;**}

flats-styles.css

.filter-section {  
 margin-top: **100**px**;**}  
  
.filter {  
 display: flex**;** align-items: center**;** justify-content: center**;**}  
  
.filter-form {  
 display: flex**;** gap: **20**px**;** padding: **20**px**;** border-radius: **10**px**;**}  
  
.filter-form label {  
 display: flex**;** flex-direction: column**;** font-size: **14**px**;** color: **#333;**}  
  
.input-group {  
 display: flex**;** gap: **10**px**;**}  
  
.input-field {  
 padding: **10**px**;** border: **1**px solid **#ccc;** border-radius: **5**px**;** width: **100**px**;**}  
  
.select-form {  
 padding: **10**px**;** border: **1**px solid **#ccc;** border-radius: **5**px**;** height: **37**px**;** width: **150**px**;**}  
  
.label-txt {  
 margin-bottom: **5**px**;**}  
  
.submit-filter-btn {  
 padding: **8**px **24**px**;** background-color: **#ffffff;** border-radius: **4**px**;** border: none**;** font-family: inherit**;** transition: **0.3**s ease-in-out**;** height: **35**px**;** margin-top: **25**px**;**}  
  
.submit-filter-btn:hover {  
 background-color: rgba(**76, 75, 75, 0.2**)**;**}  
  
.input-field:focus {  
 border-color: **#007bff;** outline: none**;**}  
  
.select-field {  
 padding: **10**px**;** border: **1**px solid **#ccc;** border-radius: **5**px**;**}  
  
.list-flats {  
 display: flex**;** justify-content: center**;** align-items: center**;** gap: **50**px**;** margin-top: **80**px**;** flex-wrap: wrap**;**}  
  
.flats-item {  
 width: **450**px**;** height: **450**px**;** background-color: **#fff;** border: solid black **1**px**;** border-radius: **10**px**;** display: flex**;** flex-direction: column**;** justify-content: space-between**;** overflow: hidden**;** transition: **0.3**s ease-in-out**;**}  
  
.flats-item:hover {  
 cursor: pointer**;** box-shadow: **12**px **12**px **2**px **1**px rgba(**76, 75, 75, 0.2**)**;**}  
  
.flats-item-title {  
 font-size: **1.5**rem**;** margin: **10**px**;** display: flex**;** justify-content: center**;** align-items: center**;** text-align: center**;**}  
  
.flats-item-image {  
 width: **100**%**;** height: auto**;** max-width: **300**px**;** max-height: **150**px**;** margin: **0** auto **10**px**;**}  
  
.flats-item-content {  
 flex-grow: **1;** overflow-y: auto**;** padding: **0 10**px **10**px**;**}  
  
.flats-item-text {  
 font-size: **1**rem**;** margin-bottom: **5**px**;**}

login-styles.css

.login-section {  
 margin-top: **150**px**;** display: flex**;** justify-content: center**;** align-items: center**;**}  
  
.login-title {  
 margin-bottom: **40**px**;** display: flex**;** justify-content: center**;** align-items: center**;**}  
.login-form {  
 width: **400**px**;** height: **300**px**;** display: flex**;** justify-content: center**;** align-items: center**;** flex-direction: column**;** gap: **20**px**;** border-radius: **10**px**;** box-shadow: **0**px **0**px **10**px rgba(**0, 0, 0, 0.1**)**;** background-color: **#ffffff;**}  
  
.login-box {  
 display: flex**;** justify-content: center**;** align-items: center**;**}  
  
.login-input {  
 width: **300**px**;** height: **30**px**;** border: none**;** padding-left: **10**px**;** border-bottom: solid black **1**px**;**}  
  
.login-btn {  
 width: **100**px**;** height: **30**px**;** border-radius: **5**px**;** border: none**;** transition: **0.3**s**;**}  
  
.login-btn:hover {  
 background-color: rgba(**76, 75, 75, 0.2**)**;**}

main-styles.css

@import url("https://fonts.googleapis.com/css2?family=Merriweather:wght@300;400;700;900&family=Roboto+Mono:wght@100..700&display=swap")**;**h1**,**h2**,**h3**,**h4**,**h5**,**h6**,**p {  
 margin: **0;**}  
  
body {  
 background-color: **#ddd0c8;** font-family: "Roboto Mono"**,** monospace**;** font-optical-sizing: auto**;** font-weight: **350;** font-style: normal**;** margin: **0;**}  
  
img {  
 display: block**;** bottom: none**;**}  
  
button {  
 display: block**;** cursor: pointer**;** bottom: none**;** border-radius: **4**px**;**}  
.scrolled {  
 background-color: **#fff;**}  
  
ul**,**ol {  
 list-style: none**;** padding-left: **0;** margin: **0;**}  
  
a {  
 text-decoration: none**;**}  
  
.container {  
 width: **1800**px**;** padding-left: **15**px**;** padding-right: **15**px**;** margin-left: auto**;** margin-right: auto**;**}  
  
header {  
 position: fixed**;** top: **0;** left: **0;** width: **100**%**;** height: **90**px**;** background-color: **#ddd0c8;** transition: background-color **0.5**s ease**;** display: flex**;**}  
.head {  
 display: flex**;** justify-content: space-between**;** align-items: center**;**}  
  
.menu-list {  
 display: flex**;** gap: **80**px**;**}  
  
.menu-item-link {  
 color: **#000;** position: relative**;** font-size: **20**px**;**}  
  
.menu-item-link::after {  
 content: ""**;** position: absolute**;** left: **0;** bottom: -**7**px**;** width: **100**%**;** height: **2**px**;** background-color: **#000000;** transform: scaleX(**0**)**;** transition: transform **0.3**s ease**;**}  
  
.menu-item-link:hover::after {  
 transform: scaleX(**1**)**;**}  
  
.language-switcher {  
 position: relative**;**}  
  
.language-options {  
 position: absolute**;** right: -**20**px**;** display: none**;** background-color: **#ffffff;** padding: **5**px **0;**}  
  
.language-options.show {  
 display: block**;**}  
  
.language-option {  
 display: block**;** padding: **5**px **20**px**;** color: **#fff;** text-decoration: none**;**}  
  
.hero {  
 height: **170**px**;**}  
  
.main-title {  
 display: flex**;** justify-content: center**;** align-items: center**;** margin-top: **150**px**;** flex-direction: column**;**}  
  
.rent-now-text {  
 margin-bottom: **20**px**;** font-size: **40**px**;**}  
  
.unical-text {  
 letter-spacing: **7**px**;** font-weight: **400;**}  
  
.rent-now-btn {  
 padding: **15**px **35**px **15**px **35**px**;** border-radius: **24**px**;** font-size: **18**px**;** font-weight: **380;** cursor: pointer**;** background-color: **#323232;** transition: **0.3**s ease-in-out**;** color: **#fff;** letter-spacing: **5**px**;**}  
  
.rent-now-btn:hover {  
 background-color: **#524d4b;**}  
  
.img-content {  
 max-width: **1440**px**;** height: **600**px**;** margin: **0** auto**;** background-image: url("../images/main-img.png")**;** background-repeat: no-repeat**;** background-position: **50**%**;** background-size: cover**;**}  
  
.section {  
 min-height: **600**px**;**}  
  
.footer {  
 margin-top: **20**px**;** padding-top: **100**px**;** padding-bottom: **100**px**;** background-color: **#323232;**}  
.flex-footer {  
 display: flex**;** flex-wrap: wrap**;**}  
/\* phone screen \*/  
@media screen and (min-width: **320**px) and (max-width: **767**px) {  
 .flex-footer {  
 justify-content: center**;** }  
 .footer-user-contact {  
 text-align: center**;** }  
}  
/\* tablet screen \*/  
@media screen and (min-width: **768**px) and (max-width: **1157**px) {  
 .flex-footer {  
 max-width: **590**px**;** }  
 .footer-user-contact {  
 margin-left: **58**px**;** }  
}  
.information-logo-link {  
 font-size: **18**px**;** color: var(--blue-txt-cl)**;** letter-spacing: **0.03**em**;** line-height: **1.3;** font-weight: **800;** text-transform: uppercase**;** font-family: var(--secondery-font)**;**}  
.information-logo-link-light {  
 color: **#ffffff;** font-size: **18**px**;** letter-spacing: **0.03**em**;** line-height: **1.3;** font-weight: **800;** font-family: var(--secondery-font)**;**}  
/\* phone screen \*/  
@media screen and (min-width: **320**px) and (max-width: **767**px) {  
 .information-logo-link {  
 display: flex**;** justify-content: center**;** }  
}  
.information-text {  
 margin-top: **16**px**;** color: **#ffffff;** line-height: **1.5;** width: **264**px**;**}  
/\* phone screen \*/  
@media screen and (min-width: **320**px) and (max-width: **767**px) {  
 .footer-information {  
 margin: **0** auto**;** margin-bottom: **72**px**;** }  
}  
/\* tablet screen \*/  
@media screen and (min-width: **768**px) and (max-width: **1158**px) {  
 .footer-information {  
 margin-left: **58**px**;** margin-right: **24**px**;** margin-bottom: **72**px**;** }  
}  
.footer-social {  
 margin: **0** auto**;**}  
/\* phone screen \*/  
@media screen and (min-width: **320**px) and (max-width: **767**px) {  
 .footer-social {  
 text-align: center**;** margin-bottom: **72**px**;** }  
}  
.footer-social-list {  
 display: flex**;** gap: **16**px**;**}  
.footer-social-text {  
 font-weight: **500;** line-height: **1.5;** color: **#ffffff;** margin-bottom: **16**px**;**}  
.footer-social-link {  
 width: **40**px**;** height: **40**px**;** background: rgba(**255, 255, 255, 0.1**)**;** display: flex**;** align-items: center**;** justify-content: center**;** border-radius: **50**%**;** fill: var(--light-txt-cl)**;** transition-property: background-color**;** transition: **0.3**s**;**}  
  
.footer-social-link:hover**,**.footer-social-link:focus {  
 background-color: **#524d4b;**}  
.footer-form {  
 display: flex**;** gap: **24**px**;**}  
/\* phone screen \*/  
@media screen and (min-width: **320**px) and (max-width: **767**px) {  
 .footer-form {  
 display: flex**;** flex-wrap: wrap**;** gap: **40**px**;** justify-content: center**;** }  
 .footer-form-label {  
 width: **100**%**;** }  
}  
.footer-form-input {  
 width: **264**px**;** height: **40**px**;** background-color: var(--dark-bg-cl)**;** border: **1**px solid rgba(**255, 255, 255, 0.3**)**;** filter: drop-shadow(**0**px **4**px **4**px rgba(**0, 0, 0, 0.15**))**;** border-radius: **4**px**;** padding: **0 16**px**;** color: **#ffffff;**}  
/\* phone screen \*/  
@media screen and (min-width: **320**px) and (max-width: **767**px) {  
 .footer-form-input {  
 width: **100**%**;** }  
}  
  
.form-icon-check {  
 fill: **#fcfcfc;**}  
  
.footer-form-btn {  
 padding: **8**px **24**px**;** background-color: **#ffffff;** border-radius: **4**px**;** border: none**;** color: var(--main-bg-cl)**;** display: flex**;** align-items: center**;** font-family: inherit**;** transition: **0.2**s ease-in-out**;**}  
.footer-form-btn-icon {  
 fill: **#323232;** margin-left: **16**px**;**}  
.footer-form-btn:hover**,**.footer-form-btn:focus {  
 background-color: **#ddd0c8;** box-shadow: **0**px **4**px **4**px rgba(**0, 0, 0, 0.15**)**;**}

register-styles.css

.register-section {  
 margin-top: **100**px**;**}  
  
.box-register {  
 display: flex**;** justify-content: center**;** align-items: center**;**}  
  
.register-form {  
 display: flex**;** justify-content: center**;** align-items: center**;** flex-direction: column**;** gap: **20**px**;** width: **400**px**;** height: **400**px**;** border-radius: **10**px**;** box-shadow: **0**px **0**px **10**px rgba(**0, 0, 0, 0.1**)**;** background-color: **#ffffff;**}  
  
.register-title {  
 display: flex**;** justify-content: center**;** align-items: center**;** margin-bottom: **20**px**;**}  
  
.register-input {  
 width: **300**px**;** height: **30**px**;** border: none**;** padding-left: **10**px**;** border-bottom: solid black **1**px**;**}  
  
.register-btn {  
 width: **120**px**;** height: **30**px**;** border-radius: **5**px**;** border: none**;** margin-bottom: **30**px**;** transition: **0.3**s**;**}  
  
.register-btn:hover {  
 background-color: rgba(**76, 75, 75, 0.2**)**;**}  
  
.register-txt {  
 display: flex**;** justify-content: center**;** align-items: center**;**}

base.py

from sqlalchemy.orm import DeclarativeBase  
  
  
class Base(DeclarativeBase):  
 pass

flat.py

from app.services.models.base import Base  
from sqlalchemy.orm import Mapped**,** mapped\_column**,** relationship  
from sqlalchemy import ForeignKey  
  
  
class Flat(Base):  
 \_\_tablename\_\_ = "flats"  
 id: Mapped[int] = mapped\_column(  
 primary\_key=True**,** index=True**,** autoincrement=True)  
 name: Mapped[str] = mapped\_column(index=True**,** nullable=False)  
 location: Mapped[str] = mapped\_column(index=True**,** nullable=False)  
 description: Mapped[str] = mapped\_column(nullable=False)  
 area: Mapped[int] = mapped\_column(nullable=False)  
 price: Mapped[int] = mapped\_column(nullable=False)  
 rooms: Mapped[int] = mapped\_column(nullable=False)  
 user\_id: Mapped[int] = mapped\_column(  
 ForeignKey("users.id")**,** nullable=False)  
  
 owner = relationship("User"**,** back\_populates="flats")

requests.py

from pydantic import BaseModel**,** EmailStr  
  
  
class UserSignupRequest(BaseModel):  
 email: EmailStr  
 name: str  
 surname: str  
 password: str  
  
  
class UserSigninRequest(BaseModel):  
 email: EmailStr  
 password: str  
  
  
class FlatCreateRequest(BaseModel):  
 name: str  
 location: str  
 description: str  
 area: int  
 price: int  
 rooms: int  
  
 class Config:  
 orm\_mode = True

user.py

from app.services.models.base import Base  
from sqlalchemy.orm import Mapped**,** mapped\_column  
from app.services.types.schema import Roles  
from sqlalchemy.orm import relationship  
  
  
class User(Base):  
 \_\_tablename\_\_ = "users"  
 id: Mapped[int] = mapped\_column(  
 primary\_key=True**,** index=True**,** autoincrement=True)  
 email: Mapped[str] = mapped\_column(unique=True**,** index=True**,** nullable=False)  
 name: Mapped[str] = mapped\_column(index=True**,** nullable=False)  
 surname: Mapped[str] = mapped\_column(index=True**,** nullable=False)  
 password: Mapped[str] = mapped\_column(nullable=False)  
 role: Mapped[Roles] = mapped\_column(default=Roles.user)  
  
 flats = relationship("Flat"**,** back\_populates="owner")

base.py

from pydantic import BaseModel**,** ConfigDict  
  
class Base(BaseModel):  
 model\_config = ConfigDict(populate\_by\_name=True)

flatfilters.py

from pydantic import BaseModel  
from typing import Optional  
  
  
class FlatFilters(BaseModel):  
 area\_from: Optional[int] = None  
 area\_to: Optional[int] = None  
 price\_from: Optional[int] = None  
 price\_to: Optional[int] = None  
 rooms: Optional[int] = None

schema.py

from pydantic import BaseModel**,** EmailStr  
from enum import Enum  
  
class Roles(Enum):  
 user = "user"  
 admin = "admin"  
  
class UserSchema(BaseModel):  
 email: EmailStr  
 name: str  
 surname: str  
 password: str  
  
 class Config:  
 orm\_mode = True

connection.py

from sqlalchemy import create\_engine  
from sqlalchemy.orm import sessionmaker  
from app.settings import settings  
from sqlalchemy import MetaData  
  
dbcon = f"postgresql://{settings.DB\_USER}:{settings.DB\_PASSWORD.get\_secret\_value()}@{settings.DB\_HOST}/{settings.DB\_NAME}"  
  
engine = create\_engine(dbcon**,** echo=True)  
metadata\_object = MetaData()  
SessionFactory = sessionmaker(autocommit=False**,** autoflush=False**,** bind=engine)

repositoryflat.py

# repositoryflat.py  
from sqlalchemy.orm import Session  
from sqlalchemy.exc import SQLAlchemyError  
from app.services.models.flat import Flat  
import logging  
  
logging.basicConfig(level=logging.DEBUG)  
  
  
class FlatRepository:  
 def \_\_init\_\_(self**,** sess: Session):  
 self.sess: Session = sess  
  
 def create\_flat(self**,** flat: Flat) -> bool:  
 try:  
 self.sess.add(flat)  
 self.sess.commit()  
 except SQLAlchemyError as e:  
 logging.error(f"Error creating flat: {e}")  
 self.sess.rollback()  
 return False  
 finally:  
 self.sess.close()  
 return True  
  
 def get\_all\_flats(self**,** filters=None**,** order\_by=None):  
 query = self.sess.query(Flat)  
 if filters:  
 for key**,** value in filters.items():  
 if key == "name":  
 query = query.filter(Flat.name.ilike(f"%{value}%"))  
 elif key == "location":  
 query = query.filter(Flat.location.ilike(f"%{value}%"))  
 elif key == "area\_min":  
 query = query.filter(Flat.area >= value)  
 elif key == "area\_max":  
 query = query.filter(Flat.area <= value)  
 elif key == "price\_min":  
 query = query.filter(Flat.price >= value)  
 elif key == "price\_max":  
 query = query.filter(Flat.price <= value)  
 elif key == "rooms":  
 query = query.filter(Flat.rooms == value)  
 if order\_by:  
 query = query.order\_by(order\_by)  
 return query.all()  
  
 def get\_flats\_by\_user\_id(self**,** user\_id: int):  
 try:  
 return self.sess.query(Flat).filter(Flat.user\_id == user\_id).all()  
 except SQLAlchemyError as e:  
 logging.error(f"Error fetching flats: {e}")  
 return []

repositoryuser.py

from typing import Dict**,** Any  
from app.services.models.user import User  
from sqlalchemy.orm import Session  
from sqlalchemy.exc import SQLAlchemyError  
import logging  
  
logging.basicConfig(level=logging.DEBUG)  
  
  
class UserRepository:  
 def \_\_init\_\_(self**,** sess: Session):  
 self.sess: Session = sess  
  
 def create\_user(self**,** signup: User) -> bool:  
 try:  
 self.sess.add(signup)  
 self.sess.commit()  
 except SQLAlchemyError as e:  
 logging.error(f"Error creating user: {e}")  
 self.sess.rollback()  
 return False  
 finally:  
 self.sess.close()  
 return True  
  
 def get\_user\_by\_id(self**,** user\_id):  
 return self.sess.query(User).filter(User.id == user\_id).first()  
  
 def get\_user(self):  
 return self.sess.query(User).all()  
  
 def get\_user\_by\_email(self**,** email: str):  
 return self.sess.query(User).filter(User.email == email).first()  
  
 def update\_user(self**,** id: int**,** details: Dict[str**,** Any]) -> bool:  
 try:  
 self.sess.query(User).filter(  
 User.id == id).update(details)  
 self.sess.commit()  
 except:  
 return False  
 return True  
  
 def delete\_user(self**,** id: int) -> bool:  
 try:  
 self.sess.query(User).filter(User.id == id).delete()  
 self.sess.commit()  
 except:  
 return False  
 return True

security.py

from passlib.context import CryptContext  
from fastapi.security import OAuth2PasswordBearer  
from jose import jwt  
from fastapi import Depends**,** Request  
from fastapi.exceptions import HTTPException  
from app.settings import settings  
from app.services.repositoryuser import UserRepository  
from app.services.connection import SessionFactory  
from datetime import datetime**,** timedelta  
from app.services.models.user import User  
  
ACCESS\_TOKEN\_EXPIRE\_MINUTES = **3000**pwd\_context = CryptContext(  
 schemes=["bcrypt"]**,** deprecated="auto")  
oauth2\_scheme = OAuth2PasswordBearer(tokenUrl="token")  
COOKIE\_NAME = "Authorization"  
  
# create Token  
  
  
def create\_access\_token(session**,** user\_email: str):  
 try:  
 with SessionFactory() as sess:  
 userRepository = UserRepository(sess)  
 user = userRepository.get\_user\_by\_email(user\_email)  
 expiration\_time = datetime.utcnow() + timedelta(weeks=**2**)  
 payload = {  
 "user\_id": user.id**,** "email": user.email**,** "name": user.name**,** "surname": user.surname**,** "password": user.password  
 }  
 token\_payload = {\*\*payload**,** "exp": expiration\_time}  
 return jwt.encode(token\_payload**,** key=settings.JWT\_SECRET.get\_secret\_value()**,** algorithm=settings.ALGORITHM.get\_secret\_value())  
 except Exception as ex:  
 print(str(ex))  
 raise ex  
  
# create verify Token  
  
  
def verify\_token(token):  
 try:  
 payload = jwt.decode(token**,** key=settings.JWT\_SECRET.get\_secret\_value())  
 user\_id = payload.get("user\_id")  
 return user\_id  
 except Exception as ex:  
 raise ex  
  
  
# password hash  
  
  
def get\_password\_hash(password):  
 return pwd\_context.hash(password)  
  
# password verify  
  
  
def verify\_password(plain\_password**,** hashed\_password):  
 return pwd\_context.verify(plain\_password**,** hashed\_password)  
  
  
def get\_current\_user\_from\_token(token: str = Depends(oauth2\_scheme)):  
 user\_id = verify\_token(token)  
 if not user\_id:  
 raise HTTPException(status\_code=**401,** detail="Invalid token")  
  
 with SessionFactory() as sess:  
 userRepository = UserRepository(sess)  
 user = userRepository.get\_user\_by\_id(user\_id)  
 if not user:  
 raise HTTPException(status\_code=**404,** detail="User not found")  
  
 return user  
  
  
def get\_current\_user\_from\_cookie(request: Request) -> User:  
 token = request.cookies.get(COOKIE\_NAME)  
 if token:  
 user = verify\_token(token)  
 return user  
  
  
async def get\_current\_user(token: str = Depends(oauth2\_scheme)):  
 return token

main.py

from fastapi import FastAPI**,** Depends**,** Request  
from fastapi.templating import Jinja2Templates  
import logging  
from fastapi.middleware.cors import CORSMiddleware  
from fastapi.staticfiles import StaticFiles  
from fastapi.responses import HTMLResponse**,** JSONResponse  
from fastapi.exceptions import HTTPException  
from app.services.connection import SessionFactory  
from app.services.security import get\_password\_hash**,** verify\_password**,** create\_access\_token**,** get\_current\_user\_from\_token**,** get\_current\_user  
  
# repository  
from app.services.repositoryuser import UserRepository  
from app.services.repositoryflat import FlatRepository  
  
# types & models  
from app.services.models.user import User  
from app.services.models.flat import Flat  
from app.services.types.flatfilters import FlatFilters  
from app.services.models.requests import UserSigninRequest**,** UserSignupRequest**,** FlatCreateRequest  
  
app = FastAPI()  
app.mount("/static"**,** StaticFiles(directory="app/front/static"**,** html=True)**,** name="static")  
  
templates = Jinja2Templates(directory="app/front/html")  
  
origins = [  
 "http://localhost:5500"**,** "http://127.0.0.1:8000/"  
]  
  
app.add\_middleware(  
 CORSMiddleware**,** allow\_origins=origins**,** allow\_credentials=True**,** allow\_methods=["\*"]**,** allow\_headers=["\*"]**,**)  
  
  
@app.get("/")  
def index\_file(request: Request):  
 return templates.TemplateResponse("index.html"**,** {"request": request})  
  
  
@app.get("/login")  
def login(request: Request):  
 return templates.TemplateResponse("login.html"**,** {"request": request})  
  
  
@app.get("/signup")  
def signup(request: Request):  
 return templates.TemplateResponse("registration.html"**,** {"request": request})  
  
  
@app.get("/profile-red")  
def profile\_red(request: Request):  
 return templates.TemplateResponse("profile-red.html"**,** {"request": request})  
  
  
@app.get("/profile")  
def get\_profile(request: Request**,** token: str = Depends(get\_current\_user)):  
 print(token)  
 if token:  
 current\_user = get\_current\_user\_from\_token(token)  
 with SessionFactory() as sess:  
 flatRepository = FlatRepository(sess)  
 user\_flats = flatRepository.get\_flats\_by\_user\_id(current\_user.id)  
 return templates.TemplateResponse("profile.html"**,** {"request": request**,** "flats": user\_flats})  
 else:  
 return templates.TemplateResponse("login.html"**,** {"request": request})  
  
  
@app.get("/rent")  
def rent(request: Request):  
 with SessionFactory() as sess:  
 flatRepository = FlatRepository(sess)  
 flats = flatRepository.get\_all\_flats()  
 return templates.TemplateResponse("flats.html"**,** {"request": request**,** "flats": flats})  
  
  
@app.get("/create-flats")  
def create\_flats(request: Request):  
 return templates.TemplateResponse("create-flats.html"**,** {"request": request})  
  
  
@app.post("/user/signup")  
def signup\_user(user\_request: UserSignupRequest**,** request: Request):  
 with SessionFactory() as sess:  
 userRepository = UserRepository(sess)  
 db\_user = userRepository.get\_user\_by\_email(user\_request.email)  
  
 if db\_user:  
 return templates.TemplateResponse("registration.html"**,** {"request": request**,** "message": "Користувач з такою поштою вже існує!"})  
  
 signup = User(  
 email=user\_request.email**,** name=user\_request.name**,** surname=user\_request.surname**,** password=get\_password\_hash(user\_request.password)  
 )  
 success = userRepository.create\_user(signup)  
  
 token = create\_access\_token(sess**,** user\_request.email)  
  
 return JSONResponse(status\_code=**201,** content={"message": "Користувач створений успішно!"**,** "token": token})  
  
  
@app.post("/user/signin")  
def signin\_user(user\_request: UserSigninRequest**,** request: Request):  
 with SessionFactory() as sess:  
 userRepository = UserRepository(sess)  
 db\_user = userRepository.get\_user\_by\_email(user\_request.email)  
 if not db\_user:  
 return JSONResponse(status\_code=**400,** content={"message": "Користувача з такою поштою не існує!"})  
  
 if verify\_password(user\_request.password**,** db\_user.password):  
 token = create\_access\_token(sess**,** user\_request.email)  
 return JSONResponse(status\_code=**200,** content={"message": "Успішний вхід"**,** "token": token})  
  
 return JSONResponse(status\_code=**400,** content={"message": "Неправильний пароль!"})  
  
  
@app.get("/success"**,** response\_class=HTMLResponse)  
def success\_page(request: Request**,** token: str):  
 return templates.TemplateResponse("index.html"**,** {"request": request**,** "token": token})  
  
  
@app.post("/user/create-flat")  
def create\_flat(flat\_request: FlatCreateRequest**,** current\_user: User = Depends(get\_current\_user\_from\_token)):  
 with SessionFactory() as sess:  
 flatRepository = FlatRepository(sess)  
 new\_flat = Flat(  
 name=flat\_request.name**,** location=flat\_request.location**,** description=flat\_request.description**,** area=int(flat\_request.area)**,** price=int(flat\_request.price)**,** rooms=int(flat\_request.rooms)**,** user\_id=current\_user.id  
 )  
 success = flatRepository.create\_flat(new\_flat)  
 if success:  
 return JSONResponse(status\_code=**201,** content={"message": "Квартира створена успішно"})  
 else:  
 raise HTTPException(  
 status\_code=**500,** detail="Помилка при створенні квартири")  
  
  
@app.post("/user/flats"**,** response\_class=HTMLResponse)  
def flats(request: Request**,** filters: FlatFilters):  
 filter\_dict = {}  
  
 if filters.area\_from is not None:  
 filter\_dict["area\_min"] = filters.area\_from  
 if filters.area\_to is not None:  
 filter\_dict["area\_max"] = filters.area\_to  
  
 if filters.price\_from is not None:  
 filter\_dict["price\_min"] = filters.price\_from  
 if filters.price\_to is not None:  
 filter\_dict["price\_max"] = filters.price\_to  
  
 if filters.rooms is not None:  
 filter\_dict["rooms"] = filters.rooms  
  
 with SessionFactory() as sess:  
 flatRepository = FlatRepository(sess)  
 flats = flatRepository.get\_all\_flats(filters=filter\_dict)  
  
 return templates.TemplateResponse("flats.html"**,** {"request": request**,** "flats": flats})  
  
  
logging.basicConfig(level="DEBUG")

settings.py

from pydantic import SecretStr  
from pydantic\_settings import BaseSettings**,** SettingsConfigDict  
  
  
class Settings(BaseSettings):  
 DB\_NAME: str  
 DB\_USER: str  
 DB\_PASSWORD: SecretStr  
 DB\_HOST: str  
 DB\_PORT: str  
 # SERVER\_NAME: str  
 SECRET\_KEY: SecretStr  
 JWT\_SECRET: SecretStr  
 ALGORITHM: SecretStr  
 model\_config = SettingsConfigDict(  
 env\_file=('.env'**,** 'stack.env')**,** env\_file\_encoding='utf-8'**,** extra='ignore'  
 )  
  
  
settings = Settings()