Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Матвєєва Євгенія Олександровича*
(ПІБ)

академічної групи *122-20ск-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованої інформаційної системи для організації та надання послуг з оренди житла на базі фреймворку Laravel*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20ск-1* (група) *Матвєєва Євгенія Олександровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованої інформаційної системи для організації та надання послуг з оренди житла на базі фреймворку Laravel*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ (підпис) *доц. Спірінцев В.В* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Матвєєв Є.О* (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 79 с., 39 рис., 3 дод., 24 джерела.

Об'єкт розробки: веб-орієнтована інформаційна система для організації та надання послуг з оренди житла на базі фреймворку Laravel.

Мета кваліфікаційної роботи: розробка програмного забезпечення веб-орієнтованої інформаційної системи для організації та надання послуг з оренди житла на базі фреймворку Laravel.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку веб-орієнтованої інформаційної системи, описана робота системи, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні веб-орієнтованої інформаційної системи, що забезпечує: організації та надання послуг з оренди житла.

Актуальність розробки інформаційної системи для організації та надання послуг з оренди житла не викликає сумніву та визначається потребою у зручному та ефективному способі керування та обліку житлових об'єктів, забезпеченням доступу до актуальної інформації для клієнтів та оптимізацією процесів оренди.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ОНЛАЙН-СЕРВІС, БАЗА ДАНИХ, PHP, HTML, CSS, LARAVEL.

ABSTRACT

Explanatory note: 79 pp., 39 fig., 3 extra, 24 sources.

The object of development: a web-based information system for organizing and providing rental services based on the Laravel framework..

The purpose of the qualification work: software development of a web-based information system for organizing and providing rental services based on the Laravel framework.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, substantiates the relevance of the topic and clarifies the formulation of the problem.

In the first chapter, the subject area is analyzed, the relevance of the task and the purpose of development are determined, the statement of the problem is formulated, the requirements for software implementation, technologies and software are indicated.

In the second section, the available solutions are analyzed, a platform for development is selected, the design and development of a web-oriented information system is carried out, the operation of the system, the algorithm and structure of its functioning, as well as the call and loading of the application are described, the input and output data are determined, the composition of the parameters of the technical means.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating an application is calculated and the time for its creation is calculated.

The practical value lies in the creation of a web-oriented information system that provides: organization and provision of rental services.

The relevance of developing an information system for organizing and providing rental services is undeniable and is determined by the need for a convenient and efficient way to manage and account for residential properties, provide access to up-to-date information for customers, and optimize rental processes.

List of keywords: INFORMATION SYSTEM, ONLINE SERVICE, DATABASE, PHP, HTML, CSS, LARAVEL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

ІС - Інформаційна система

ОС - Операційна система

БД - База даних

СУБД - Система управління базами даних

CSS - Cascading Style Sheets

ВСТУП

Стрімкий розвиток інформаційних технологій та систем сприяє активному розвитку електронної комерції в різних галузях підприємницької діяльності, що веде до значних змін у взаємодії між суб'єктами бізнесу. Цей напрям розвитку стає особливо важливим і актуальним у сучасному світі, де інтернет-технології впроваджуються у всі аспекти життя і сприяють створенню нових можливостей для підприємств та споживачів.

Зокрема, електронна комерція стає все більш популярною у сфері надання послуг з оренди житла. Клієнти шукають зручні та ефективні способи здійснення оренди, а підприємства, в свою чергу, прагнуть оптимізувати процеси управління та забезпечити надійність і зручність для своїх клієнтів. Саме тут виникає необхідність у розробці інформаційних систем, які б допомагали автоматизувати діяльність підприємств у сфері електронної комерції.

Темою даної кваліфікаційної роботи є розробка веб-орієнтованої інформаційної системи для організації та надання послуг з оренди житла на базі фреймворку Laravel. Метою роботи є створення програмного забезпечення, що допоможе підвищити ефективність діяльності компаній з оренди житла, забезпечивши автоматизацію процесів у сфері електронної комерції.

Для досягнення поставленої мети необхідно розглянути особливості електронної комерції в контексті розвитку національної економіки, сформулювати основні вимоги до розроблюваного програмного продукту та здійснити розрахунок трудомісткості та витрат на його розробку.

Практичне значення цієї роботи полягатиме у створенні веб-орієнтованої інформаційної системи, що забезпечить компаніям з оренди житла доступ до нових можливостей у сфері електронної комерції, полегшить процеси управління та покращить взаємодію з клієнтами.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Важливість інтернету в сфері здачі і оренди квартир

У наш час інтернет став невід'ємною складовою частиною нашого життя. Люди використовують його для здійснення покупок, замовлення послуг та здійснення оплат. Квартирна сфера не є виключенням. Здача квартир в інтернеті – важливий крок в напрямку оптимізації та поліпшення житлового ринку.

Перш за все, здача квартир в інтернеті дає можливість власникам нерухомості ефективно рекламувати свої пропозиції. Інтернет дає широкий доступ до інформації для потенційних орендарів, які шукають квартиру. Завдяки тому, що інтернет-платформи здебільшого підтримують деталізовані профілі квартир та зручні фільтри пошуку, орендарі можуть легко знайти квартиру, яка відповідає їх потребам.

По-друге, здача квартир в інтернеті зменшує час, що потрібен для проведення транзакції. Традиційно, процес здачі квартир може займати багато часу, від пошуку клієнтів до укладення угоди. З інтернет-платформами, цей процес стає значно швидшим та простішим. Клієнти можуть зв'язатися з власником квартири через онлайн-чат або електронну пошту та отримати детальну інформацію про квартиру. Якщо клієнт зацікавлений, то оплата та укладання угоди може бути здійснено через онлайн-сервіси. В результаті, цей процес стає швидшим та зручнішим як для власників, так і для орендарів.

Тема цієї кваліфікаційної роботи – розробка веб-додатку для здійснення операцій з оренди та продажу квартир через Інтернет.

Для розробки повного функціоналу веб-сайту з оренди та продажу квартир було ідентифіковано основні бізнес-процеси, які потрібно врахувати при розробці:

– перегляд доступних користувачеві пропозицій;

– отримання інформації про кожну окрему пропозицію;

– можливість пошуку пропозицій;

– можливість сортування та фільтрування пропозицій;

– можливість зв’язатися з менеджером компанії для вирішення питанн;

– можливість додавати власні пропозиції.

1.1.2. Аналіз існуючих рішень

Перед розробкою будь-якої системи важливо провести аналіз існуючих аналогів, з'ясувати їх переваги та недоліки, щоб уникнути можливих помилок та використати переваги вже наявних рішень.

Для аналізу було обрано 2 веб-ресурси:

– веб-сайт «DIM.RIA» (URL: https://dom.ria.com/);

– веб-сайт «RIELTOR» (URL: https://rieltor.ua/).

Почнемо огляд з першого ресурсу («DIM.RIA»). На рис. 1.1 зображено головну сторінку веб-ресурсу.

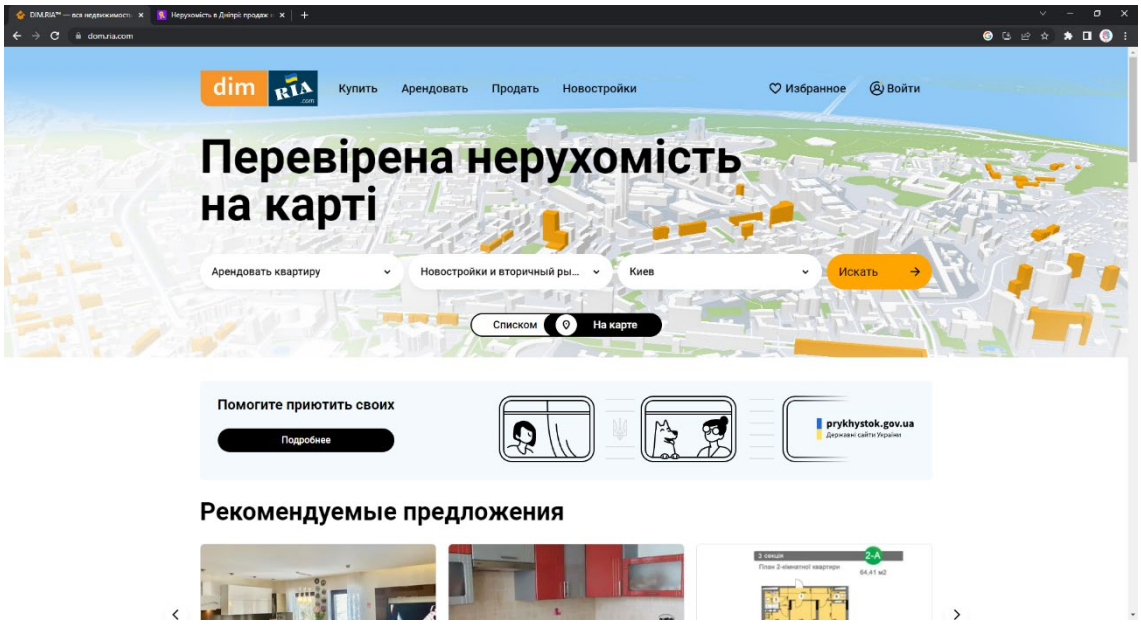

Рис. 1.1. Зовнішній вигляд головної сторінки веб-ресурсу

Головна сторінка містить наступну інформацію:

- верхнє меню з пунктами «Купівля», «Продаж», «Новобудови», «Аренда», «Обране» та особистим кабінетом. Ці пункти дозволяють швидко перейти до необхідної категорії або переглянути спеціальні пропозиції;
- логотип компанії, який розташований на видному місці, а також виконує функцію повертання на головну сторінку;
- пошукова форма, яка дозволяє швидко знайти потрібну нерухомість за різними параметрами, такими як тип, ціна, район, кількість кімнат тощо;
- список рекомендованих пропозицій, який дозволяє переглянути поточні «гарячі» пропозиції;
- список пропозицій нерухомості з можливістю швидкої фільтрації за різними параметрами. Кожна пропозиція містить зображення об'єкту, кількість кімнат, ціну та розташування.
- рейтинг ріелторів, якій зображує список найпопулярніших та найнадійніших ріелторів;
- опис кампанії з посиланням на соціальні мережі.

Загалом, головна сторінка сайту містить достатньо інформації та функціоналу для швидкого та зручного пошуку та розміщення пропозицій нерухомості. Дизайн сайту досить зручний та інтуїтивно зрозумілий,

Тепер зробимо огляд другого ресурсу («RIELTOR»). На рис. 1.2 зображено головну сторінку веб-ресурсу.

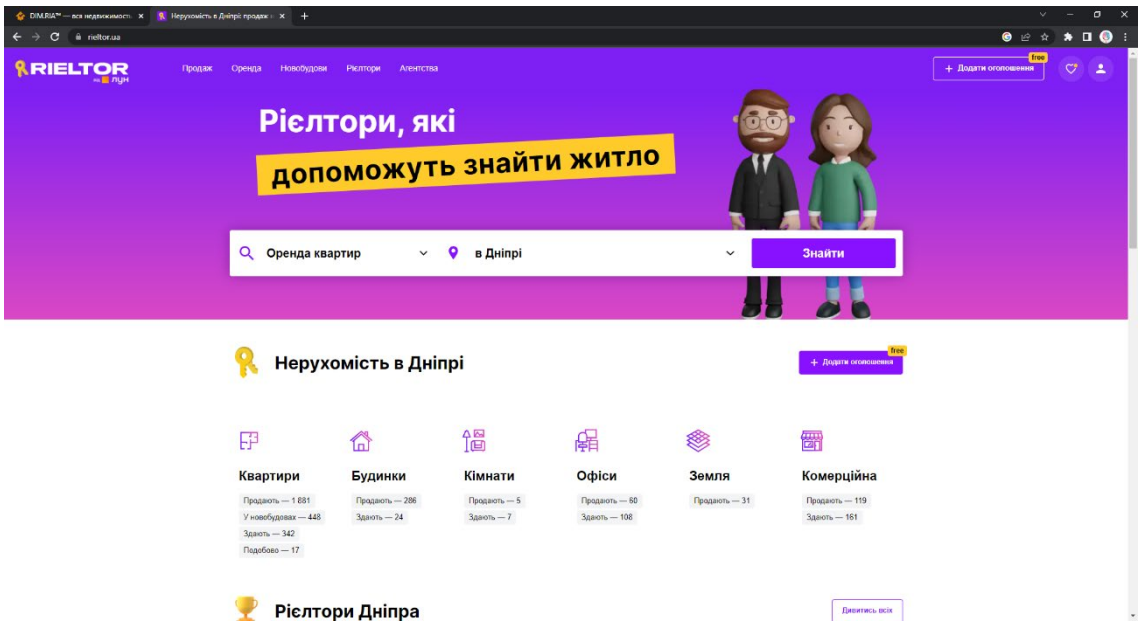

Рис. 1.2. Зовнішній вигляд головної сторінки веб-ресурсу

Головна сторінка містить наступну інформацію:

- верхнє меню з пунктами «Продаж», «Оренда», «Новобудови», «Рієлтори», «Агенства» та особистим кабінетом;
- логотип, який розташований на видному місці, та який також служить для повертання на головну сторінку;
- пошукову форму, яка дозволяє обрати дію з пропозиціями та місто;
- кількість пропозицій в окремому місті;
- список перевірених ріелторів, який допомагає користувачеві обрати надійного ріелтора;
- інформацію про компанію з посиланням на соціальні мережі.

На сайті наведена досить повна та вичерпна інформація про ринок нерухомості України, оголошення про купівлю, продаж і оренду будь-якої нерухомості, а також інформація про будівництво новобудов та інші зв'язані з нерухомістю послуги.

Після аналізу цих ресурсів були визначені наступні ключові аспекти, що потрібно буде реалізувати у проєкті:

– навігаційна панель. Ця панель дозволить користувачеві мати швидкий доступ до будь-якого потрібного йому розділу сайту;
– логотип. Обов'язковий елемент, якій вказує на унікальність продукту;
– інформація про ресурс. Цей елемент буде допомагати користувачеві зрозуміти чим саме займається кампанія. Його було вирішено винести до окремої сторінки;
– декілька випадкових пропозицій на головній сторінці, які можуть зацікавити користувача;
– фільтр пропозицій. Також важливий елемент, який дозволить користувачеві підібрати пропозицій за своїм смаком.

1.2. Призначення розробки та галузь застосування

Розробка інформаційної системи дозволить інтегрувати створений програмний продукт в сферу нерухомості з метою автоматизації робочих процесів. Розробляємий програмний продукт надає широкі можливості для різних суб'єктів, таких як ріелторські агентства, фізичні особи та організації, щоб полегшити процес здачі та оренди квартир.

Основні призначення розробки:

– автоматизація роботи ріелторських агенцій. Веб-сайт надає засоби для керування базою даних квартир, обробки запитів від клієнтів, а також моніторингу стану пропозицій;
– фільтр пропозицій. Також важливий елемент, який дозволить користувачеві підібрати пропозицій за своїм смаком.
– допомога фізичним особам здавати квартиру у оренду. Веб-сайт забезпечує можливість розміщення оголошень про оренду квартир, опису характеристик, визначення ціни та зв'язок з потенційними орендарями.
– детальна інформація про квартири. Кожна пропозиція містить докладний опис квартири, включаючи кількість кімнат, район, краткий

опис, тощо. Це допомагає фізичним особам отримати чітку картину про квартиру перед прийняттям рішення.

1.3. Підстава для розробки

Підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 122 «Комп'ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
- завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованої інформаційної системи для організації та надання послуг з оренди житла на базі фреймворку Laravel».

1.4. Постановка завдання

Кваліфікаційна робота спрямована на розробку інформаційної системи, орієнтованої на веб, для компанії, яка займається здачею та орендою квартир. Метою цієї системи є автоматизація різних процесів в цій компанії з метою полегшення роботи та забезпечення більш ефективного контролю над його діяльністю в цій сфері. Для досягнення цієї мети будуть використані сучасні веб-технології та методи розробки програмного забезпечення.

Основні вимоги до розробленої інформаційної системи передбачають високу швидкість завантаження сторінок, максимальну зручність для користувача, оптимізацію сторінок для пошукових систем, легке сприйняття інформації та простоту управління змістом. Забезпечення цих вимог є важливим для забезпечення ефективності роботи з інформаційною системою та забезпечення задоволення користувачів від її використання. Для досягнення цих вимог будуть використані сучасні технології розробки та методи проектування.

Створення й розробка інформаційної системи повинно включати наступні етапи:

- аналіз вимог та потреб користувачів системи;
- розробка концептуальної моделі системи та визначення її функціональності;
- розробка детальної технічної специфікації системи;
- розробка та верифікація програмного забезпечення;
- тестування та відладка системи;
- впровадження системи та підготовка користувачів до її використання;
- забезпечення підтримки та розвитку системи після її впровадження.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставленої в роботі мети в інформаційній системі, що розробляється, повинні бути реалізовані:

- перегляд списку пропозицій;
- сортування списку пропозицій;
- перегляд інформаційної сторінки про пропозицію;
- написання відгуку про компанію;
- наповнення бази даних новими пропозиціями;
- редагування поточних пропозицій;
- керування запитами користувачів.

1.5.2. Вимоги до інформаційної безпеки

Хешування паролів є процесом перетворення вхідного паролю в неповторний хеш-код, який неможливо зворотно перетворити в початковий

пароль. Це важлива практика безпеки, яка допомагає захистити паролі користувачів від несанкціонованого доступу у випадку, якщо база даних стає компрометованою.

Один з найпоширеніших алгоритмів хешування паролів – bcrypt, який використовує солі і множинне хешування для забезпечення високої захищеності. Крім bcrypt, існують інші алгоритми, такі як Argon2, PBKDF2 і SHA-256, які також використовуються для хешування паролів з метою забезпечення безпеки [1].

На рис. 1.3. наведений приклад зашифрованих паролів у БД проекту кваліфікаційної роботи.

id	login	password	name1	name2	name3	email	phone_number	is_admin
1	Altrimgot	$2y$10$MzRvW6Yh/vu9Nc5d.9mrl.d8LIJZBGS8RlCU3N7JRks...	Євгеній	Матвєєв	Олександрович	altrimgot@gmail.com	0954261213	1
3	RegularUser01	$2y$10$.cAtBkUPG6Zd8lismVBUUu08XVlrTmc1Xbgp5nO6hSX...	Віталій	Свіридов	Альбертович	sv.user01@gmail.com	0678261982	0
4	RegularUser02	$2y$10$vRVNFVniQ7J3Tu31ILVv0OLXRspHi6J6Ta.5.oW2nCR...	Василь	Бервін	Олегович	vavasil14@gmail.com	0985314263	0

Рис. 1.3. Зашифровані паролі користувачів у БД

Також у веб-додатку можуть існувати різні ролі користувачів, такі як адміністратор, користувач та неавторизований користувач. Розподіл ролей визначає доступ та повноваження, які має кожна категорія користувачів.

Адміністратор. Адміністратор має повний доступ та контроль над веб-додатком. Його роль полягає в управлінні системою, керуванні користувачами, налаштуванні прав доступу, додаванні та видаленні ресурсів, а також у вирішенні проблем, пов'язаних з функціональністю додатку.

Користувач. Користувач – це зареєстрований та авторизований користувач, який має обмежений доступ до функціоналу додатку. Він може виконувати дії, які пов'язані з основними функціями додатку, такими як створення облікового запису, редагування персональних даних, створення та управління контентом, взаємодія з іншими користувачами тощо.

Неавторизований користувач. Неавторизований користувач – це відвідувач, який ще не зареєстрований або не авторизований в системі. Він має обмежений доступ і може переглядати загальну інформацію, але не може виконувати дії, що потребують авторизації [2].

1.5.3. Вимоги до складу та параметрів технічних засобів

Для ефективної роботи веб-орієнтованої інформаційної системи необхідно дотримуватися певних вимог щодо технічних засобів, які використовуються для її функціонування.

Для клієнтської частини:

- операційна система: Windows 7 або новіша;
- процесор: Intel Core 2 Duo або AMD Athlon 64 X2 або новіші
- оперативна пам'ять: мінімум 2 ГБ RAM (для 32-бітних ОС) або 4 ГБ RAM (для 64-бітних ОС);
- відеокарта: мінімум 512 МБ VRAM і підтримка WebGL;
- роздільна здатність екрану: мінімум 1024x768 пікселів.

Для серверної частини:

- операційна система: може бути використана будь-яка операційна система, яка підтримує веб-сервер і базу даних, наприклад, Linux або Windows Server;
- процесор: мінімум 2-ядерний процесор з тактовою частотою від 2 ГГц;
- оперативна пам'ять: мінімум 2 ГБ RAM, але може змінюватися в залежності від обсягу трафіку і розміру бази даних;
- диск: достатній обсяг дискового простору для зберігання файлів сайту та бази даних. Рекомендується використання SSD для покращення швидкості роботи;
- веб-сервер: можна використовувати Apache, Nginx, IIS або інші веб-сервери;

– база даних: можна використовувати MySQL, PostgreSQL, Microsoft SQL Server або інші системи управління базами даних;
– безпека: сервер повинен бути захищеним від вторгнень, забезпечення захисту даних та конфіденційності.

Вказані технічні вимоги є рекомендованими та можуть залежати від конкретного проекту та його вимог. Якщо технічні засоби відповідають наведеним характеристикам, то це допоможе забезпечити надійність, швидкість та безпеку розробленого програмного продукту відповідно до вимог замовника. Проте, в окремих випадках, вимоги можуть бути більш жорсткими залежно від складності програмного продукту та навантаження на сервер, що може вимагати більш потужних засобів.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для ефективної роботи веб-орієнтованої програми необхідно, щоб програмне забезпечення технічних засобів, на яких вона буде працювати, відповідало певним технічним характеристикам. Це гарантує правильне та швидке виконання різноманітних завдань, а також забезпечує надійність та безпеку програмного продукту. Вимоги для системи, на якій буде використовуватися продукт:

– операційна система Windows 7+, Unix, Ubuntu;
– браузер Інтернет, такий як Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, Opera, Internet Explorer.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення програми

Метою даної кваліфікаційної роботи є створення веб-орієнтованої системи, яка допомагатиме користувачам знайти пропозиції з оренди житла. Дана система повинна бути доступна на будь-яких пристроях та операційних системах завдяки своїй пристосованості до веб-браузера.

Основні функції програми включають:

- відображення користувачеві сторінок сайту з інформацією;
- виведення даних про оренду житла з бази даних магазину;
- авторизацію адміністратора (ріелтора) для доступу до панелі керування;
- можливість доповнення існуючого контенту сайту за допомогою панелі керування.

2.2. Опис застосованих математичних методів

Для проектування та розробки даної веб-орієнтованої інформаційної системи не були використані математичні методи. Замість цього були використані лише прості арифметичні операції.

2.3. Опис використаних технологій та мов програмування

При створенні цієї інформаційної системи було застосовано різноманітні технології та мови програмування.

HTML. Ця мова є стандартизованою мовою розмітки документів, яка використовується для відображення веб-сторінок у браузерах. Веб-сторінки, написані мовою HTML, передаються браузеру через протоколи HTTP/HTTPS

або можуть бути відкриті з локального диска. Браузери інтерпретують HTML код і відображають його на екрані монітора у вигляді інтерфейсу.

HTML використовується для створення різноманітних елементів на веб-сторінках. Ці елементи служать будівельними блоками, з яких складається структура сторінки. За допомогою HTML можна вставляти зображення та інші об'єкти, такі як інтерактивні форми, у візуальну сторінку. Крім того, HTML надає засоби для позначення структурної семантики тексту, таких як заголовки, абзаци, списки, посилання, цитати та інші елементи. Хоча браузери не показують теги HTML на веб-сторінці, вони використовують їх для правильної інтерпретації вмісту сторінки [3].

HTML має свої переваги і недоліки. Переваги HTML:

- простота використання. HTML є досить простим мовою розмітки, що дозволяє створювати веб-сторінки з легкістю. Він має простий синтаксис, що дозволяє швидше розробляти сторінки;
- підтримка браузерами. HTML є стандартною мовою для розробки веб-сторінок і підтримується всіма основними браузерами. Це означає, що веб-сторінки, написані на HTML, будуть відображатись коректно на різних платформах і пристроях;
- семантична структура. HTML надає можливість створювати семантичну структуру веб-сторінок, що полегшує розуміння їх змісту як людиною, так і пошуковими системами. Це сприяє покращенню доступності і оптимізації для пошукової оптимізації (SEO);
- розширюваність. HTML дозволяє використовувати різноманітні технології і інструменти, такі як CSS для стилізації, JavaScript для додавання інтерактивності, а також підтримує вбудовування зображень, відео, аудіо та інших мультимедійних елементів.

Недоліки HTML:

- обмежена функціональність. HTML в основному використовується для створення статичних веб-сторінок. Він не надає широкого спектру можливостей для розробки складних функцій, таких як обробка форм,

маніпуляція з базами даних або взаємодія з сервером. Для цих завдань зазвичай потрібні додаткові технології, такі як CSS, JavaScript або серверні мови програмування;

- відсутність динамічності. HTML не забезпечує вбудованих механізмів для створення складних динамічних ефектів або інтерактивних інтерфейсів. Для цього потрібно використовувати JavaScript або інші скриптові мови;
- підтримка різних версій: HTML має кілька версій, таких як HTML4, XHTML і HTML5. Це може призвести до проблем з сумісністю і потребує уваги при виборі відповідної версії для веб-проекту [4].

PHP. Ця мова є скриптовою мовою програмування, яка була створена для генерації HTML-сторінок на веб-сервері. Вона є однією з популярних мов, використовуваних у веб-розробці, разом з іншими, такими як Java, .NET, JavaScript, Python і Ruby. PHP підтримується більшістю хостинг-провайдерів і є проєктом відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на бік клієнта. Однак, на відміну від JavaScript, користувач не бачить PHP-коду, оскільки браузер отримує готовий HTML-код. Це сприяє безпеці, але може знизити інтерактивність сторінок. Втім, PHP можна використовувати для генерації JavaScript-коду, який виконується на боці клієнта.

PHP дозволяє вбудовувати HTML-код безпосередньо в код сторінок, що коректно обробляються PHP-інтерпретатором. Виконання PHP-коду розпочинається після відкриваючого тегу (<?php) і продовжується до зустрічі закриваючого тегу [5, 6].

PHP має свої переваги і недоліки. Переваги PHP:

- простота вивчення. PHP є досить простою мовою програмування, що дозволяє швидко освоїти її для розробки веб-додатків. Синтаксис PHP схожий на мову C, що полегшує розуміння інших програмістів і сприяє швидкому впровадженню;

– велика спільнота. PHP має одну з найбільших спільнот програмістів у світі. Це означає, що знайдення документації, підтримки та відповідей на запитання стає легким завдяки активній спільноті, яка допомагає один одному;
– велика кількість фреймворків. PHP має широкий вибір фреймворків, таких як Laravel, Symfony, CodeIgniter та інші, які допомагають в розробці швидких і масштабованих веб-додатків. Вони надають готові компоненти, що спрощують розробку і прискорюють процес;
– широкі можливості: PHP дозволяє використовувати різноманітні функції та бібліотеки для роботи з базами даних, обробки форм, генерації зображень, криптографії, роботи з API та багато іншого. Це дає можливість реалізувати різноманітні функціональність у веб-додатках.

Недоліки PHP:

– вразливість до безпеки. PHP має деякі вразливості безпеки, особливо якщо неправильно використовувати його. Недотримання найкращих практик та застосування ненадійних скриптів можуть створити ризики безпеки для веб-додатків;
– специфічний синтаксис. Синтаксис PHP може бути дещо складним для деяких програмістів, особливо для тих, хто звик до інших мов програмування. Це може вимагати часу на адаптацію та навчання нового синтаксису;
– продуктивність. У порівнянні з деякими іншими мовами програмування, PHP може бути менш продуктивним в деяких сценаріях, особливо при обробці великих обсягів даних. Однак, з використанням оптимізаційних технік та кешування результатів, продуктивність PHP може бути покращена [7].

CSS. Ця мова представляє собою формальну мову для стилізації і опису зовнішнього вигляду документа (наприклад, веб-сторінки), написаного з використанням мови розмітки (зазвичай HTML або XHTML). Він також може застосовуватись до будь-яких XML-документів, таких як SVG або XUL.

Розробники веб-сторінок використовують CSS для визначення кольорів, шрифтів, стилів, розташування окремих блоків та інших аспектів зовнішнього вигляду цих веб-сторінок. Основною метою розробки CSS є відокремлення опису логічної структури веб-сторінки (яке здійснюється за допомогою HTML або інших мов розмітки) від опису її зовнішнього вигляду (яке тепер здійснюється за допомогою формальної мови CSS). Таке розділення може підвищити доступність документа, надати більшу гнучкість та можливість керування його відображенням, а також зменшити складність і повторюваність у структурному вмісті.

CSS-правило (рис 2.1) складається з селектора, CSS-властивостей і їх значень. Селектори – це мітки, які допомагають браузеру зрозуміти, до якої частини HTML-коду потрібно застосувати вказані параметри. CSS-властивості - це конкретні параметри оформлення, наприклад, колір елемента або тексту (color) або колір фону (background) [8, 9].

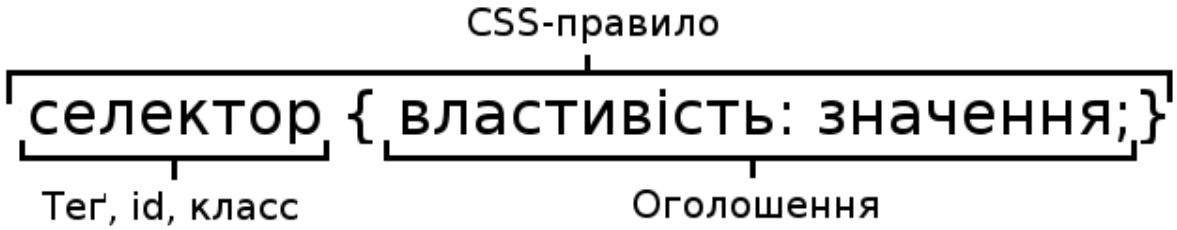

Рис. 2.1. CSS-правило

CSS має свої переваги і недоліки. Переваги CSS:

- розділення змісту і дизайну. CSS дозволяє відокремити стилізацію веб-сторінок від їх змісту. Це спрощує редагування та зміну вигляду сторінок без втручання в HTML-структуру. Завдяки цьому, зміни стилів можна легко застосовувати до всіх сторінок одного веб-сайту;
- повторне використання стилів. CSS дозволяє використовувати стилі на кількох сторінках одночасно. Це дозволяє зберігати і підтримувати

єдиний стиль для всього веб-сайту, що спрощує управління дизайном і забезпечує єдність вигляду;

- гнучкість і контроль. CSS надає широкі можливості для налаштування вигляду елементів на сторінці. З його допомогою можна контролювати розміри, кольори, шрифти, відступи, рамки та інші аспекти дизайну. Це дозволяє створювати унікальні та індивідуальні веб-інтерфейси;
- ефективність завантаження сторінок. CSS-файли можуть бути кешовані браузерами, що дозволяє зберігати їх на стороні клієнта. Після першого завантаження сторінки, стилі будуть застосовуватись до наступних сторінок без повторного завантаження, що покращує швидкість відображення.

Недоліки CSS:

- сумісність з браузерами. Різні браузери можуть розбирати CSS-правила по-різному, що може призводити до некоректного відображення сторінок. Це вимагає тестування і забезпечення сумісності з різними браузерами;
- складність деяких задач. Деякі складні дизайнерські ефекти можуть бути важкими для досягнення за допомогою CSS. Наприклад, створення складних анімацій або тривимірних трансформацій може вимагати використання додаткових технологій, таких як JavaScript або CSS-фреймворки;
- обмеження в оформленні: CSS має певні обмеження в оформленні сторінок, особливо щодо розміщення елементів. Деякі складні макети можуть потребувати використання додаткових технік, таких як флексбокси або сітки, для досягнення потрібного вигляду [10].

Laravel. Це популярний міжплатформовий PHP-фреймворк, який використовується для створення веб-додатків. Цей фреймворк має багато переваг, оскільки надає розробникам доступ до широкої бібліотеки готових функцій, таких як автентифікація, маршрутизація і шаблонування HTML.

Використання цих готових функцій дозволяє швидко створювати надійні веб-додатки, зменшуючи кількість потрібного коду.

Laravel надає зручне середовище розробки, а також інтуїтивно зрозумілі та виразні інтерфейси командного рядка. Фреймворк використовує об'єктно-реляційне відображення (ORM), що спрощує роботу з базами даних.

Програми, розроблені з використанням Laravel, легко масштабовані і мають чистий код, що спрощує обслуговування. Розробники можуть додавати нові функції до своїх додатків завдяки модульній системі упаковки Laravel і надійному управлінню залежностями [11, 12].

Laravel має свої переваги і недоліки. Переваги CSS:

- зручна та елегантна синтаксична структура. Laravel пропонує чистий і зрозумілий синтаксис, який спрощує процес розробки. Його зручність полягає в тому, що він дозволяє швидко створювати потужні функції і функціональність з меншою кількістю коду;
- велика кількість вбудованих функцій. Laravel надає широкий набір вбудованих функцій та інструментів, які полегшують розробку. Наприклад, він має систему маршрутизації, ORM (Object-Relational Mapping), систему шаблонів Blade, аутентифікацію користувачів, кешування, планувальник завдань та багато іншого. Це дозволяє розробникам ефективно виконувати рутинні завдання і швидко створювати функціональні веб-додатки;
- модульність і розширюваність. Laravel побудований на основі принципів модульності, що дозволяє розробникам легко розширювати функціональність за допомогою пакетів та розширень. Велика спільнота Laravel активно створює і підтримує різноманітні пакети, що спрощує розширення функціоналу веб-додатків;
- висока продуктивність. Laravel пропонує ефективну обробку запитів і оптимізований швидкісний показник завдяки використанню кешування, компіляції представлень та інших оптимізаційних підходів. Це дозволяє побудувати швидкі та продуктивні веб-додатки.

Недоліки Laravel:

- навчання. Навчання Laravel може вимагати деякого часу та зусиль, особливо для новачків. Фреймворк має велику кількість функцій і понять, які потребують зрозуміння та освоєння;
- надмірна комплектність. Хоча велика кількість вбудованих функцій є перевагою, вона також може бути недоліком у випадку, коли вам потрібно створити дуже простий веб-додаток, не використовуючи багато функціоналу Laravel. Завдяки своїй комплектності фреймворк може вимагати більше ресурсів, що впливає на продуктивність;
- обмежена контрольованість. Іноді Laravel може мати певні обмеження в контролі над процесами, які відбуваються в основі фреймворку. Це може ускладнити вирішення деяких нетипових або складних сценаріїв [13].

2.4. Опис структури системи та алгоритмів її функціонування

2.4.1. Опис архітектури системи

Для веб-додатку, що розробляється, була обрана клієнт-серверна архітектура.

Клієнт-серверна архітектура є популярним підходом у розробці веб-додатків, де взаємодія між клієнтами (користувачами) та сервером відбувається за певними правилами та взаємодією. У цій архітектурі клієнти і сервери виконують різні функції та взаємодіють між собою для надання та отримання інформації. На рис. 2.2. приведена структура клієнт-серверної архітектури.

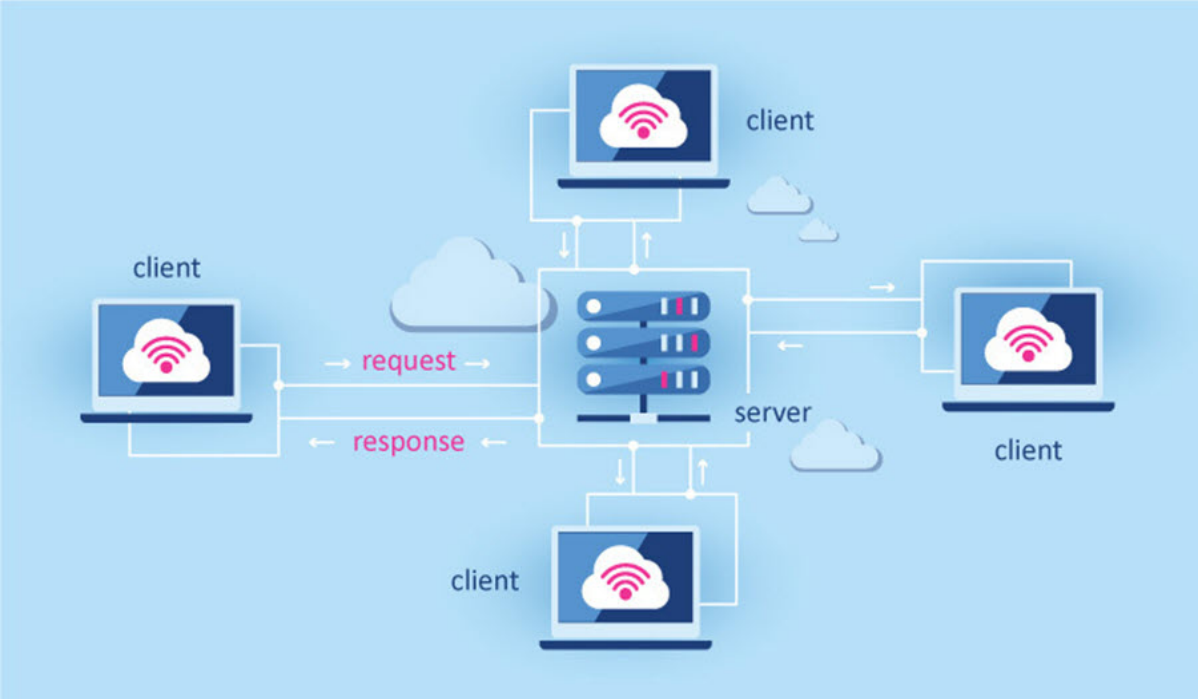

Рис. 2.2. Структура клієнт-серверної архітектури

На стороні клієнта знаходиться пристрій користувача, такий як комп'ютер, смартфон або планшет, який відправляє запити до сервера. Ці запити містять інформацію про те, що клієнт хоче отримати або виконати.

На стороні сервера розташовані потужніші комп'ютери або обладнання, які обробляють запити від клієнтів і надають їм необхідні дані або результати обчислень. Сервери також відповідають за збереження інформації, яка може бути доступна клієнтам.

Клієнт-серверна архітектура передбачає, що сервер може обслуговувати багато клієнтів одночасно, виконуючи їх запити послідовно або за пріоритетами. Цей підхід дозволяє розділити завдання між клієнтом і сервером, що сприяє збільшенню масштабованості та ефективності системи [14, 15].

2.4.2. Опис структури системи

Структура сайту є організованою схемою розташування його сторінок, категорій, підкатегорій і контенту. Це план, що відображає логічний зв'язок між

різними елементами сайту. З технічної точки зору, структура навігації визначається шляхами URL та відображає послідовність сторінок і каталогів. Ця структура є ключовим елементом, що визначає, як користувачі будуть переміщатися та знаходити інформацію на сайті. Схематична структура додатку, що розробляється, наведена на рис 2.3.

Рис. 2.3. Схематична структура додатку

Тепер розглянемо, як це реалізовано у фреймворку Laravel. У Laravel ми можемо використовувати маршрутизацію для обробки посилань на сторінки. У файлі з маршрутами визначаються шляхи (URL) та вказується, яку функцію контролера слід викликати при зверненні до кожного конкретного посилання.

Функція кожного контролера виконує необхідні дії, обробляє запит користувача та повертає йому відповідну сторінку або відображення. Це може бути HTML-шаблон, вміст бази даних, JSON-відповідь або інші типи відображень, залежно від логіки додатку [16]. Цей процес схематично зображений на рис. 2.4.

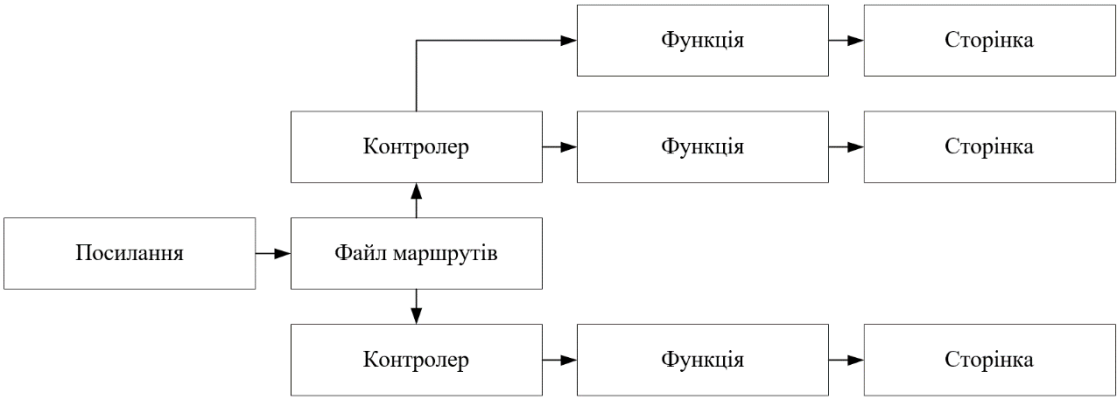

Рис. 2.4. Схематична структура реалізації посилань

Тепер розглянемо це на прикладі проекту кваліфікаційної роботи. В нас є файл «web.php» (рис 2.5), у якому знаходяться всі можливі маршрути.

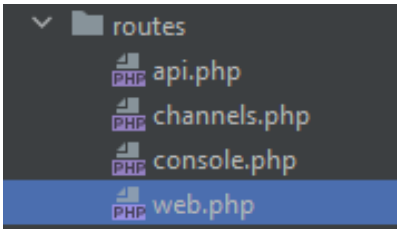

Рис. 2.5. Файл «web.php» у проекті

Цей файл звертається до контролерів (рис 2.6), а саме – до їх функцій, які в свою чергу повертають інформаційні сторінки.

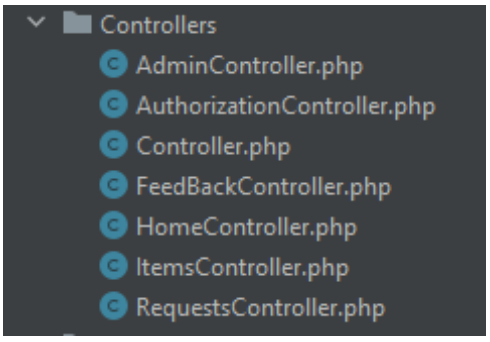

Рис. 2.6. Контролери проекту

Ось невеличкий опис кожного з контролерів:

– adminController. Відповідає за навігацію між сторінок на панелі адміністратора;
– authorisationController. Відповідає за реєстрацію та авторизацію користувача;
– controller. Стандартний контролер, який був створений у проекті за замовчуванням;
– feedBackController. Відповідає за всю взаємодію з відгуками як зі сторони користувача, так і зі сторони адміністратора;
– homeController. Відповідає за навігацію між сторінок звичайного користувача;
– itemsController. Відповідає за всю взаємодію з пропозиціями як зі сторони користувача, так і зі сторони адміністратора;
– requestsController. Відповідає за взаємодію з запитами користувачей.

Тепер розглянемо які саме сторінки (рис 2.7) будуть повертатися користувачам.

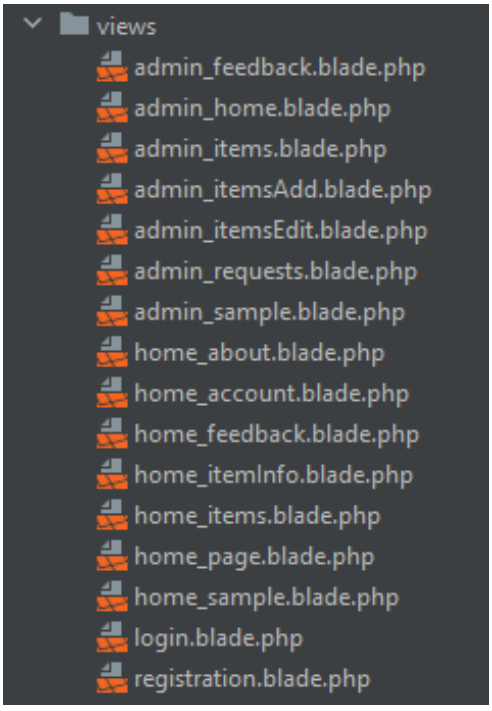

Рис. 2.7. Сторінки проекту

Одразу можна виділити дві основні сторінки, які виступають у якості шаблонів:

- admin_sample. Це шаблон для адміністративною панелі. Використовується разом з усіма сторінками, які бачить адміністратор;
- home_sample. Це шаблон для сторінок користувач. Використовується разом з усіма сторінками, які бачить користувач.

Спочатку розглянемо сторінки адміністратора:

- admin_home. Домашня сторінка на панелі адміністратора;
- admin_feedback. Виводить всі відгуки для подальшої їх публікації або видалення;
- admin_items. Виводить список всіх пропозицій для подальної взаємодії з ними;
- admin_itemsAdd. Сторінка-форма для додавання пропозиції;
- admin_itemsEdit. Сторінка-форма для редагування пропозиції;
- admin_requests. Виводить список всіх запитів користувачів для подальшої взаємодії з ними;
- admin_items. Відповідає за взаємодію з запитами користувачей.

Тепер розглянемо сторінки звичайного користувача:

- home_page. Головна сторінка сайту;
- home_about. Сторінка з інформацією про сервіс;
- home_account. Сторінка з даними про акаунт користувача, а також з даними про пропозиції, якими він цікавиться;
- home_feedback. Сторінка з відгуками користувачів;
- home_items. Сторінка, на якій присутні всі доступні пропозиції;
- home_itemInfo. Сторінка з інформацією про конкретну пропозицію.

Сторінки, які відповідають за реєстрацію та авторизацію:

- registration. Сторінка з формою для реєстрації нового користувача;
- login. Сторінка з формою для авторизації користувача.

Таблиці для бази даних також створюються безпосередньо в самому проекті. Потрібно створити модель (рис 2.8) та файл міграції (рис 2.9) для неї. Файли міграції відповідають за створення або видалення таблиць з бази даних.

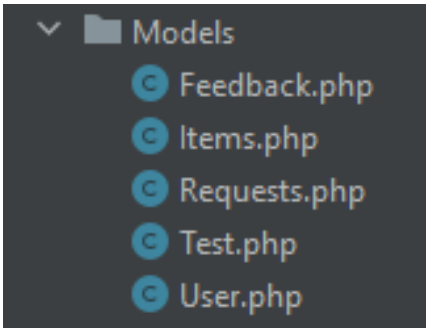

Рис. 2.8. Моделі проекту

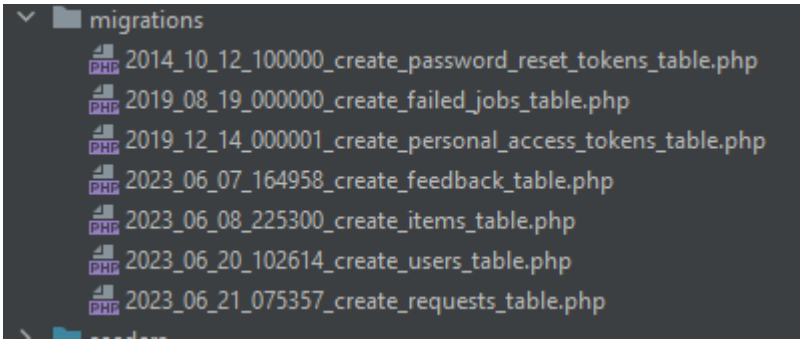

Рис. 2.9. Файли міграції проекту

Ось які дані зберігаються в цих таблицях:

- feedback. Таблиця для зберігання відгуків;
- items. Таблиця для зберігання пропозицій;
- requests. Таблиця для зберігання файлів;
- user. Таблиця для зберігання даних про користувачів.

Повний код вищеописаних файлів наведено в ДОДАТКУ А.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними зі сторони користувача є:

- логін та пароль;
- номер телефона для зв’язку;
- ПІБ клієнту;
- поштова адреса клієнту.

Вхідними даними зі сторони адміністратора є:

- логін та пароль;
- назва пропозиції;
- опис пропозиції;
- ціна пропозиції.

Веб-додаток, який розробляється, має два основних типи вихідних даних: вміст сторінок і пропозиції з оренди.

Вміст сторінок включає інформацію, яка відображається на веб-сторінках, таку як текст, зображення, тощо. Цей вміст є основою, яка допомагає створити користувачам зрозумілі та зручні веб-сторінки для навігації та здійснення різних дій.

Пропозиції з оренди є частиною вмісту сторінок і містять інформацію про доступні квартири чи житлові об'єкти для оренди. Ці пропозиції можуть містити дані, такі як опис об'єкта, його характеристики, фотографії, умови оренди та контактну інформацію для отримання додаткової інформації або здійснення бронювання.

Усі ці дані, які складаються з вмісту сторінок та пропозицій з оренди, є вихідними даними веб-додатку і вони генеруються та передаються клієнтам для відображення та взаємодії.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Для виконання кваліфікаційної роботи було використано персональний комп'ютер (ЕОМ), який має такі характеристики:

- процесор: AMD R3 3200G;
- плата: MSI B450M Pro-VDH Max;
- ОЗП: 2*8 DDR4 3200MHz;
- відеокарта: GTX 970 4Gb;
- накопичувач: 1Tb HDD;
- монітор: Samsung LC27F396FHIXCI;
- інші периферійні пристрої (клавіатура, миша, тощо.).

2.6.2. Використані програмні засоби

Для написання коду програми був використаний PHPStorm.

PhpStorm є продуктивною інтегрованим середовищем розробки (IDE) для мови програмування PHP, розробленим компанією JetBrains.

PhpStorm надає потужний редактор для роботи з кодом PHP, HTML та JavaScript. Він вміє аналізувати код у режимі реального часу, виявляти й запобігати помилкам, а також автоматизувати процес рефакторингу коду PHP та JavaScript. Редактор PhpStorm підтримує різні версії PHP і має функції, які полегшують роботу з новими можливостями мови, такими як генератори, підпрограми, останнє ключове слово, список foreach, простори імен, замикання, ознаки та синтаксис короткого масиву.

Крім того, PhpStorm включає повноцінний редактор SQL з можливістю виконання та редагування запитів. Це IDE, яке дозволяє розробникам ефективно працювати з кодом PHP, HTML та JavaScript, забезпечуючи інструменти для швидкої розробки, аналізу та відладки програмного забезпечення [17, 18].

У програмі небагато зображень, але всі вони створювалися у онлайн-сервісі Figma.

Figma є веб-сервісом, який дозволяє розробникам створювати векторні ілюстрації, інтерактивні дизайни веб-сайтів та мобільних додатків, а також елементи інтерфейсу. Цей сервіс можна використовувати на різних платформах і працювати з ним у режимі онлайн.

Головна перевага Figma полягає у можливості спільної роботи з іншими розробниками. Ви можете створити команду в Figma і одночасно працювати над проектом, спостерігаючи зміни, які роблять усі учасники, в режимі реального часу.

Ще одна особливість Figma – це можливість працювати з сервісом не тільки через десктопну версію, але й у браузері. Обидва варіанти потребують постійного підключення до Інтернету. Таким чином, Figma надає зручний спосіб створювати дизайни і спільно працювати над проектами з командою, забезпечуючи доступ до інструментів у реальному часі [19, 20].

Вся робота з базою даних була реалізована за допомогою OpenServer.

Open Server – це спеціальний сервер, який призначений для зручної розробки та тестування web-проектів. Він містить набір програм, які допомагають у виконанні цих завдань.

У складі Open Server входять різноманітні компоненти, такі як Apache – надійний та гнучкий сервер з відкритим кодом, PHP – популярна мова програмування для web-розробки, PHPMyAdmin – інструмент для керування базами даних, Nginx – ще один гнучкий сервер, MySQL – система управління базами даних та багато інших.

Open Server має деякі особливості, які забезпечують зручну роботу. Наприклад, він підтримує роботу з флеш-накопичувачами, що дозволяє прискорити процес завантаження платформи. Також він дозволяє швидко створювати домени за допомогою простих папок, забезпечує захист сервера від зовнішніх втручань та надає швидкий доступ до доменів та модулів. Крім того, Open Server підтримує роботу з кириличними доменами.

Загалом, Open Server є потужним інструментом, який дозволяє зручно розробляти та тестувати web-проекти, надаючи набір необхідних компонентів та функціональності [21].

PhpMyAdmin – веб-додаток з відкритим вихідним кодом, який написаний мовою PHP і має графічний веб-інтерфейс. Він призначений для адміністрування баз даних MySQL або MariaDB. За допомогою phpMyAdmin можна керувати сервером MySQL, виконувати SQL-запити, переглядати і редагувати дані таблиць баз даних. Цей інструмент є дуже популярним серед веб-розробників, оскільки дозволяє управляти базою даних MySQL через дружній інтерфейс в браузері з будь-якого комп'ютера, підключеного до Інтернету, без необхідності встановлення додаткового програмного забезпечення.

PhpMyAdmin широко використовується в практиці, оскільки розробники постійно вдосконалюють свій продукт, враховуючи нововведення у системі управління базами даних MySQL. Більшість українських хостинг-провайдерів використовують phpMyAdmin як інструмент керування базами даних, щоб надати своїм клієнтам можливість адміністрування їх виділених баз даних [22].

2.6.3. Виклик та завантаження програми

Ініціація програми включає в себе наступні кроки:

- переконатися що запущений Open Server. Він потрібен для встановлення зв’язку з базою даних;
- у терміналі PHP Storm перейти до директорії проекту та використати команду для запуску вбудованого веб-серверу. Процес запуску наведений на рис. 2.10;

Рис. 2.10. Запуск вбудованого веб-серверу

– у браузері до адресної строки потрібно ввести ip-адресу, як на рис. 2.11.

Рис. 2.11. IP-адреса у адресній строці

2.6.4. Опис інтерфейсу програми

Додаток можна умовно поділити на три основні частини: інтерфейс без авторизації, інтерфейс з авторизацією користувача і інтерфейс адміністратора.

Інтерфейс без авторизації дозволяє користувачам переглядати основні сторінки сайту та отримувати необхідну інформацію. Однак, в цьому режимі взаємодія з сайтом обмежена, і користувачі не можуть здійснювати дії, такі як залишення відгуків чи робота з особистими даними.

Інтерфейс з авторизацією користувача включає всі можливості інтерфейсу без авторизації, але додає функціонал, який дозволяє здійснювати взаємодію з відгуками, товарами та особистими даними. Користувачі, які авторизувалися, можуть залишати відгуки про роботу сайту, переглядати та редагувати свої особисті дані та здійснювати дії, пов'язані з пропозиціями.

Інтерфейс адміністратора, відомий також як "обратна сторона" додатку, надає адміністратору повний контроль над пропозиціями, відгуками та замовленнями, які користувачі залишають. Адміністратор може керувати всіма

аспектами сайту, включаючи додавання, редагування та видалення пропозицій, модерацію відгуків та обробку замовлень.

Така структура додатку дозволяє розділити функціональність на рівні доступу користувачів і надає зручні інструменти для взаємодії та керування різними аспектами додатку.

2.6.4.1 Опис інтерфейсу без авторизації

При відвідуванні сайту неавторизованим користувачем відкривається головна сторінка (рис 2.12), на якій розміщений «header» з різними посиланнями. Нижче розміщений логотип сайту зі своїм слоганом, а також кілька випадкових пропозицій, які можуть привернути увагу користувача.

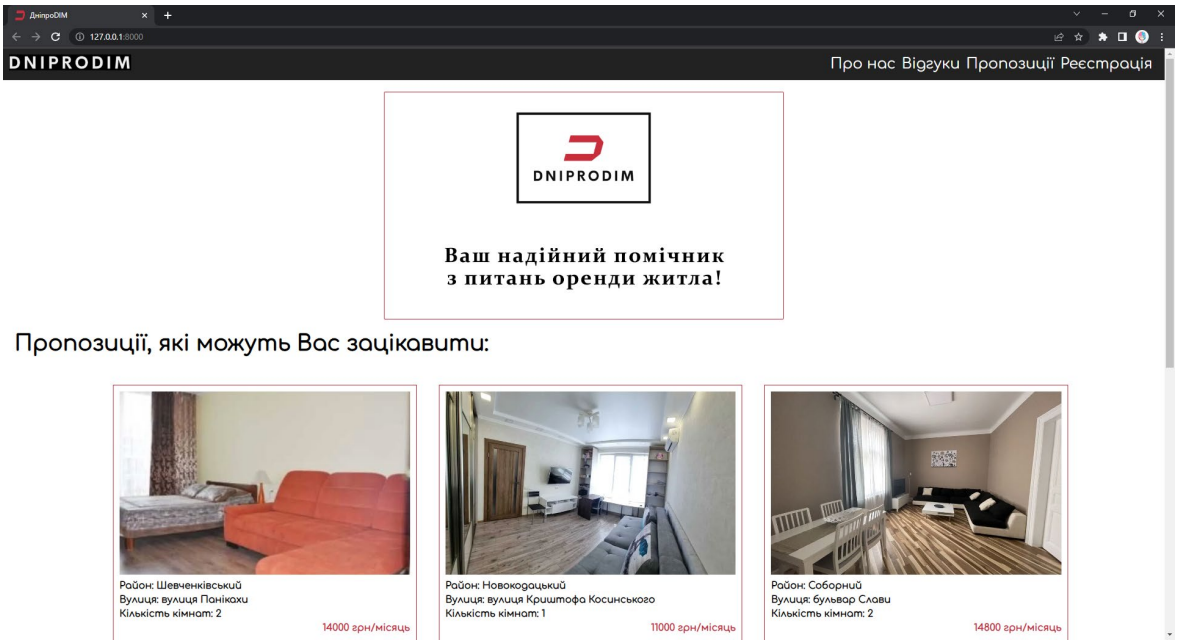

Рис. 2.12. Головна сторінка (неавторизований користувач)

У самому низу цієї та будь-якої сторінки знаходиться «footer» (рис 2.13).

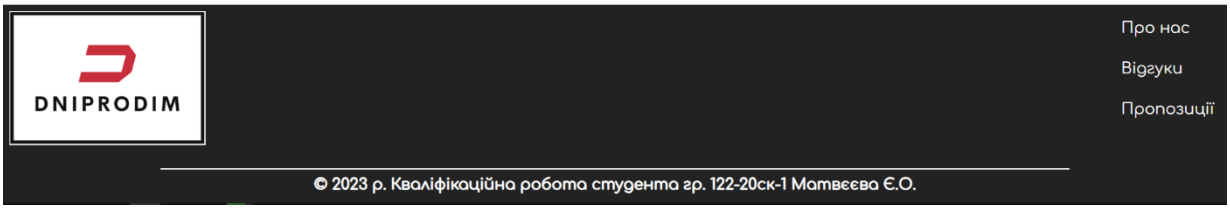

Рис. 2.13. Підвал або «footer»

Розберемо посилання у «header-і». Перше посилання у вигляді логотипу повертає користувача на головну сторінку.

Коли користувач натискає на друге посилання «Про нас», він перенаправляється на окрему сторінку, де можна знайти детальну інформацію про сервіс (рис 2.14).

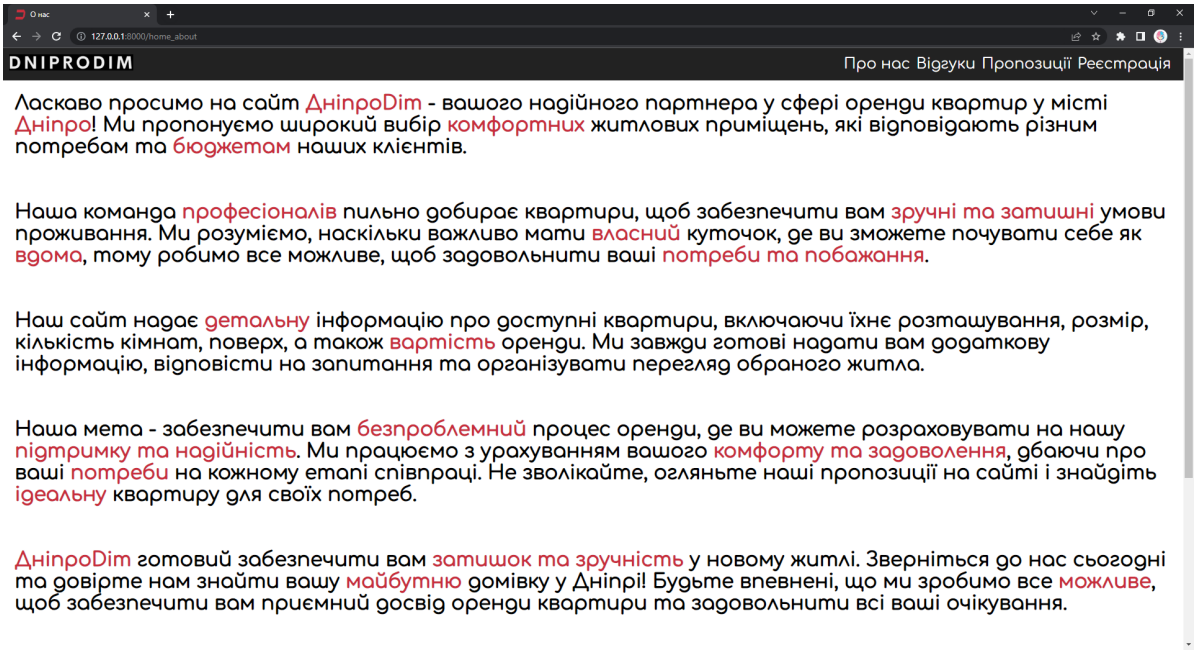

Рис. 2.14. Сторінка з інформацією

При натисканні на третє посилання «Відгуки», користувач буде перенаправлений на сторінку, де можна ознайомитися з відгуками інших клієнтів

(рис 2.15). У зв’язку з тим, що користувач неавторизований, він зможе лише переглядати ці відгуки, а не залишати свої власні.

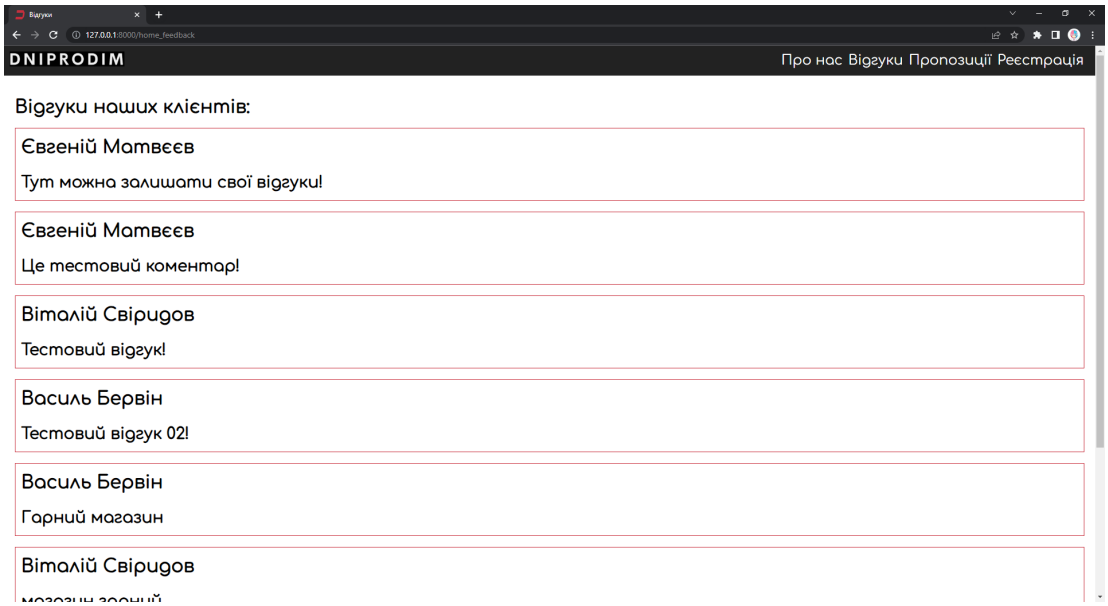

Рис. 2.15. Сторінка з відгуками

Якщо користувач натисне на четверте посилання «Пропозиції», він буде перенаправлений на сторінку, де відображаються всі доступні пропозиції (рис 2.16). На верхній частині цієї сторінки розташована форма, яка дозволяє фільтрувати пропозиції за різними параметрами. Ця форма також доступна для неавторизованого користувача.

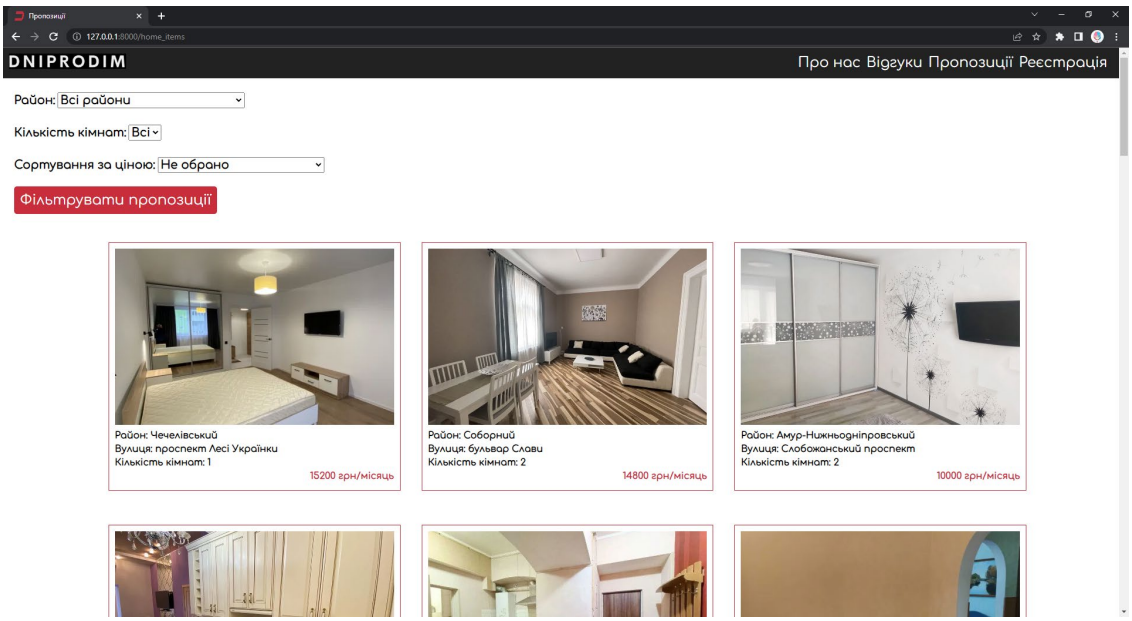

Рис. 2.16. Сторінка з пропозиціями

Наприклад, перевіримо фільтрування по двокімнатним квартирам у будь-якому районі міста та відсортуємо їх від дорогих до дешевих (рис 2.17).

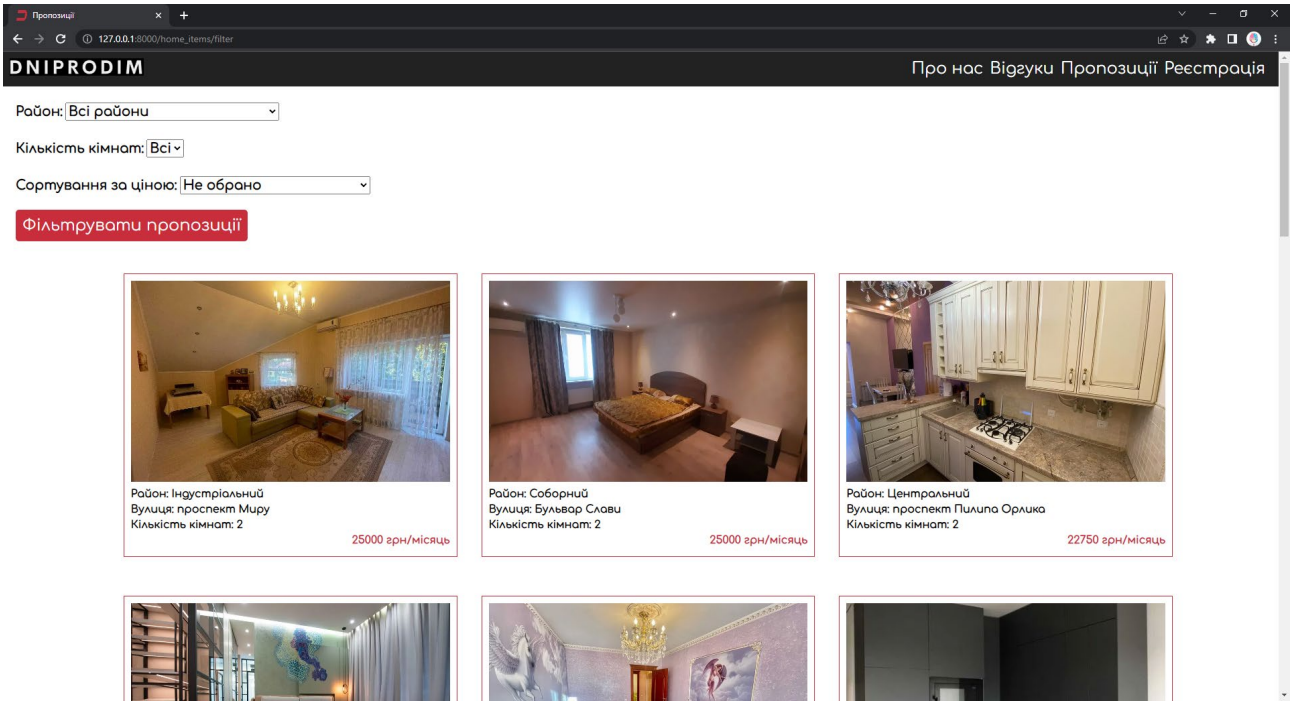

Рис. 2.17. Приклад фільтрації пропозицій

При наведенні на кожну пропозицію висвічується її опис (рис 2.18).

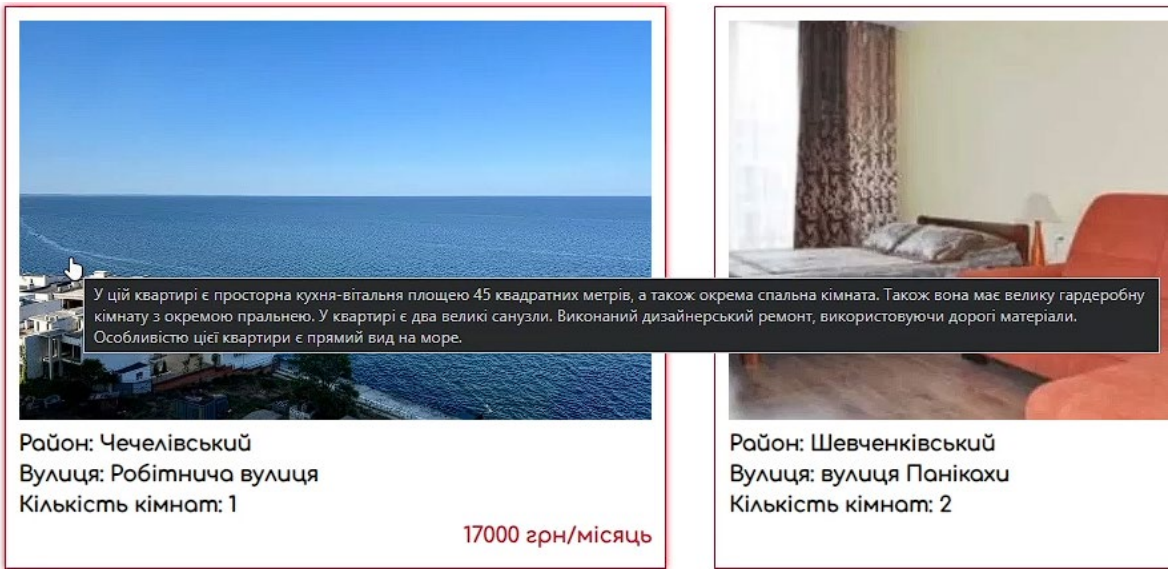

Рис. 2.18. Опис пропозиції при наведенні

При натисканні на кожну пропозицію, користувач буде перенаправлений на сторінку, яка умовно розділена на дві частини. У першій частині (рис 2.19) міститься повна інформація про пропозицію, а також кнопка, яка запрошує користувача авторизуватися на сайті. Друга частина (рис 2.20) містить шість зображень, при чому перше зображення відображається на головній сторінці або сторінці пропозицій як головне.

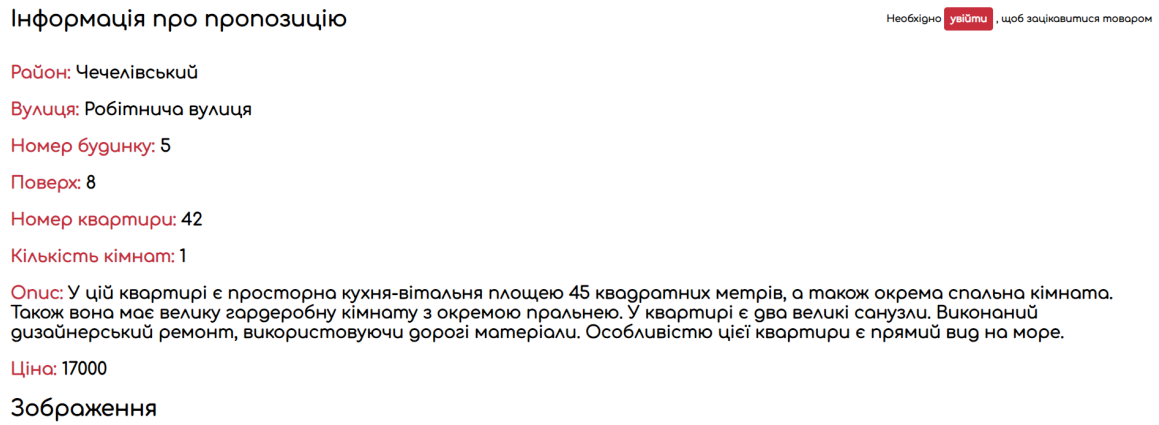

Рис. 2.19. Опис пропозиції

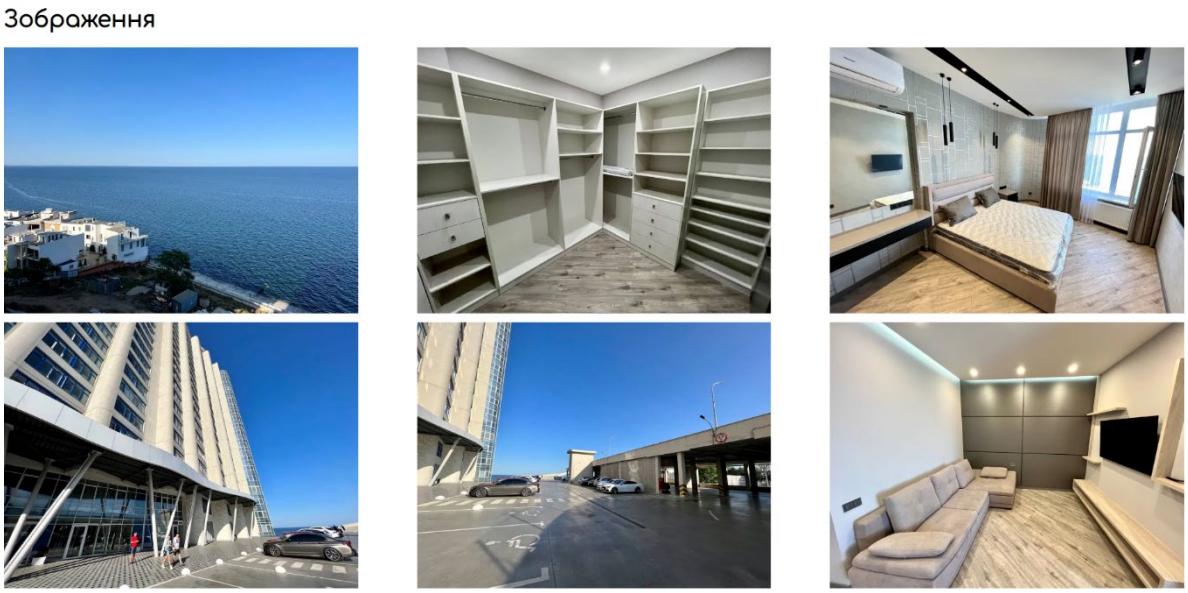

Рис. 2.20. Зображення пропозиції

Розглянемо останнє посилання «Реєстрація». Це посилання перенаправляє неавторизованого користувача на сторінку реєстрації (рис 2.21). На цій сторінці розміщений логотип сайту, а також форма для введення реєстраційних даних.

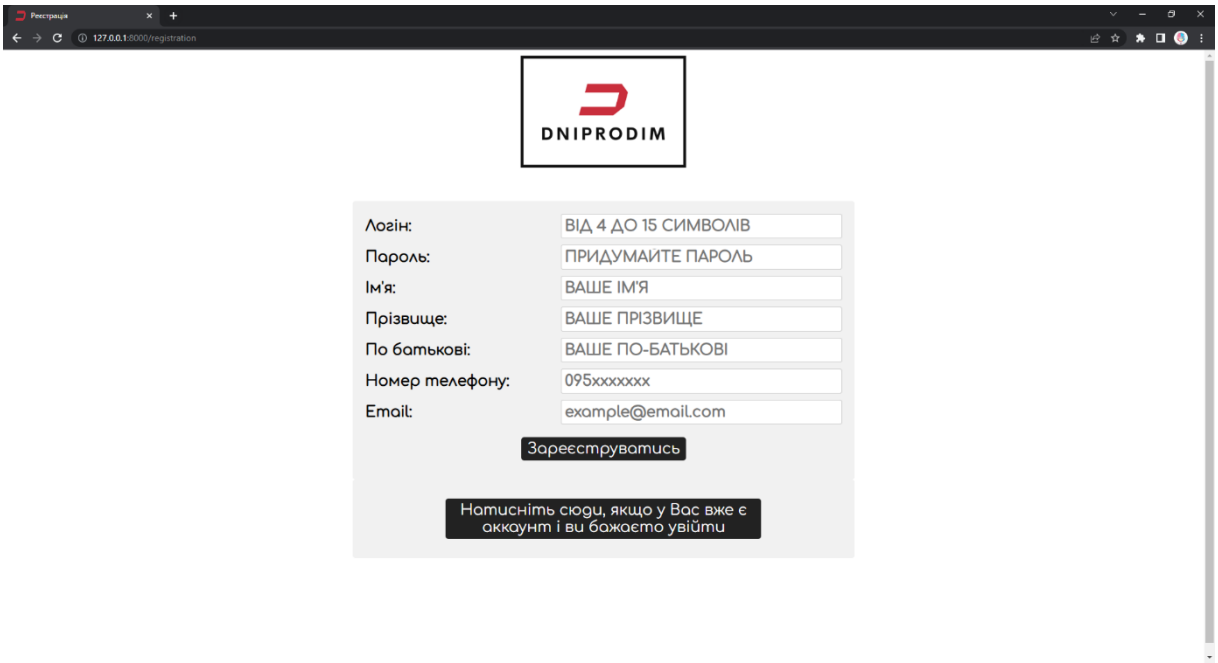

Рис. 2.21. Сторінка реєстрації

Якщо при реєстрації користувач ввів некоректні дані, то під формою реєстрації з’явиться повідомлення з усіма помилковими даними (рис 2.22).

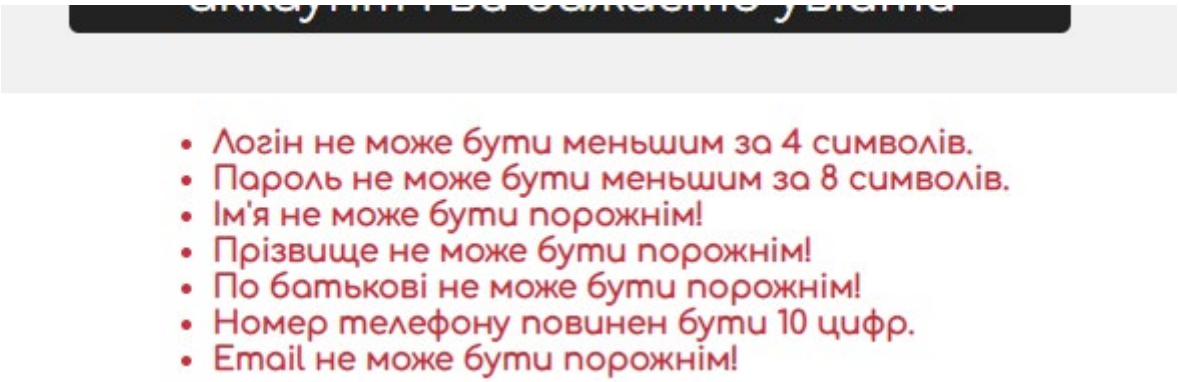

Рис. 2.22. Помилки, які виникли при реєстрації

При натисненні на нижню кнопку користувач перенаправляється до сторінки авторизації (рис 2.23), на якій також присутній логотип сайту, та сама форма авторизації.

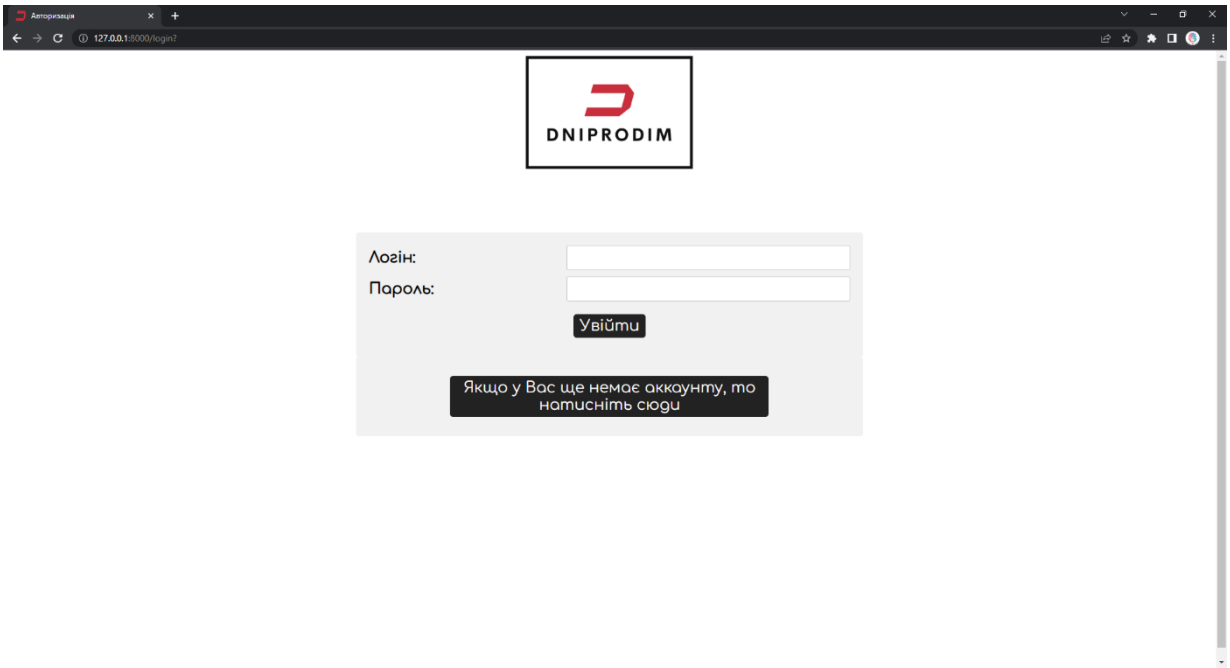

Рис. 2.23. Сторінка авторизації

Саме на цій сторінці відбувається перевірка даних користувача та пересилання на частину сайту з авторизацією або на адміністративну частину.

2.6.4.2 Опис інтерфейсу з авторизацією

Коли користувач авторизується, він також переходить на головну сторінку сайту. На цьому етапі він відзначає зміни. Замість кнопки реєстрації тепер відображаються дві інші кнопки (рис 2.24): кнопка з його ім'ям акаунту та кнопка для виходу.

Рис. 2.24. Дві кнопки, замість кнопки «Реєстрація»

Кнопка з ім’ям пересилає до особистого кабінету користувача, а кнопка «Вийти» завершує сесію та повертає користувача до неавторизованої частини сайту.

На сторінці з відгуками з’явилася можливість залишати відгук (рис 2.25).

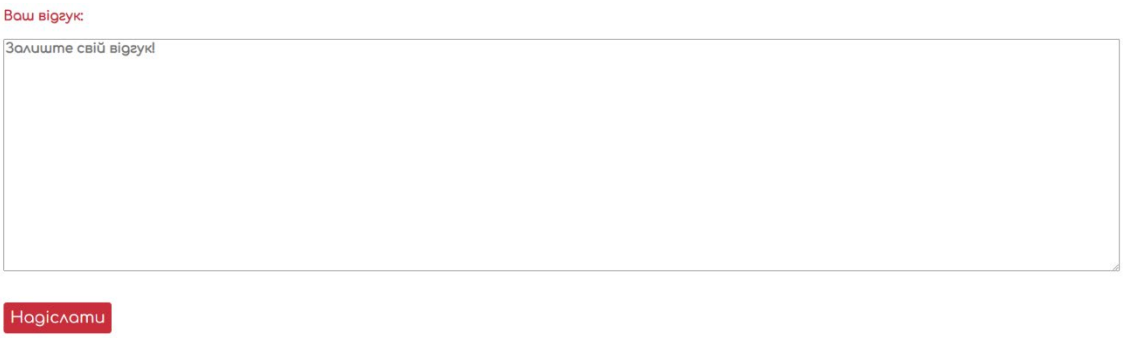

Рис. 2.25. Форма для залишення відгуку

Залишений відгук не публікується одразу, спочатку він повинен пройти перевірку зі сторони адміністартора.

Також зміни стосуються сторінки опису товару. Тепер, у правій верхній частині з’явилася кнопка «Цікавить» (рис 2.26).

Рис. 2.26. Опис пропозиції з кнопкою «Цікавить»

Вона додає обрану пропозицію до списку пропозицій, які зацікавили даного користувача.

Після того, як користувач виявив зацікавленість у пропозиції, він має можливість змінити свою думку. Замість кнопки «Цікавить» з'являється кнопка «Не цікавить» (рис 2.27), яка видаляє дану пропозицію зі списку пропозицій, які цікавлять даного користувача.

Рис. 2.27. Опис пропозиції з кнопкою «Не цікавить»

Переходимо на сторінку, де можна переглянути інформацію про акаунт користувача. Цю сторінку, так само як і сторінку з описом пропозиції, можна

умовно розділити на дві частини. Перша частина (рис 2.28) містить дані про користувача, які він може змінити або оновити. Друга частина (рис 2.29) – це список пропозицій, які зацікавили користувача. В цьому списку містяться всі пропозиції, до яких поточний користувач виявив зацікавленість.

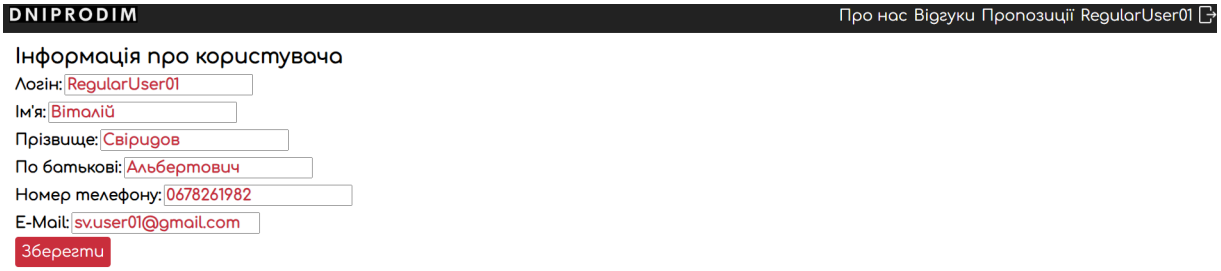

Рис. 2.28. Інформація про користувача

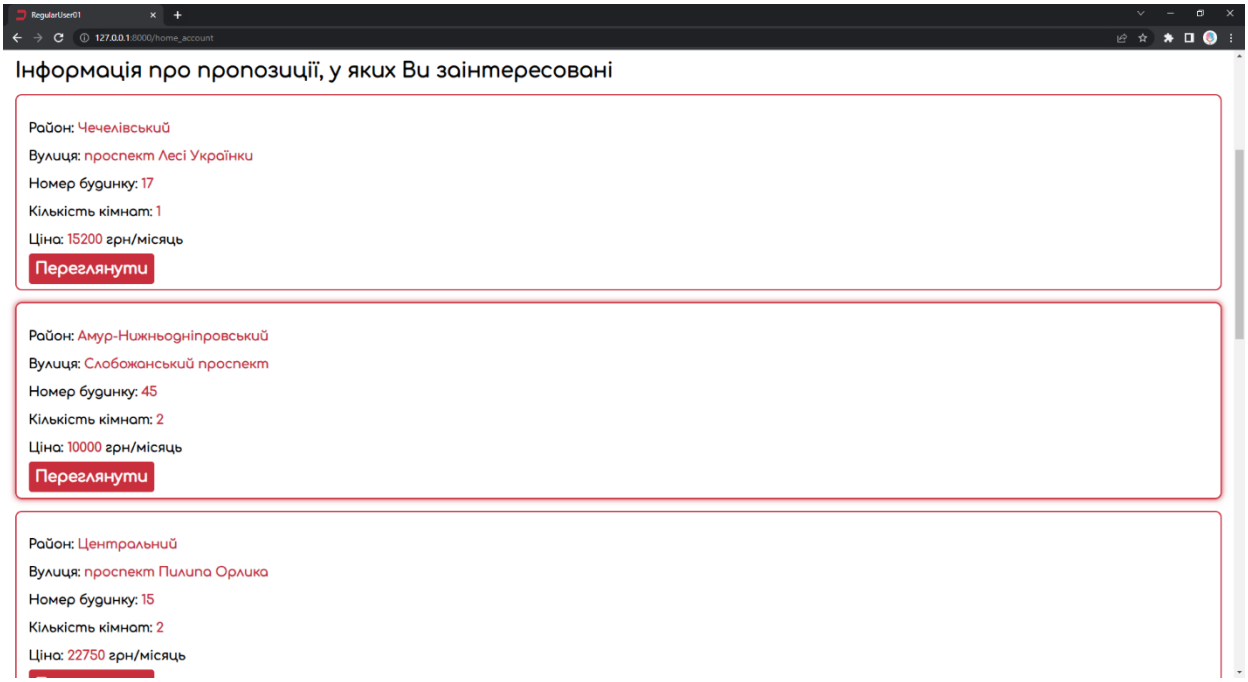

Рис. 2.29. Пропозиції, які цікавлять користувача

Кожна кнопка «Переглянути» пересилає на сторінку з описом відповідної пропозиції.

2.6.4.3 Опис інтерфейсу адміністратора

Розглянемо сторінку адміністратора (рис 2.30). Цю сторінку можуть побачити лише користувачі, у яких є доступ до неї.

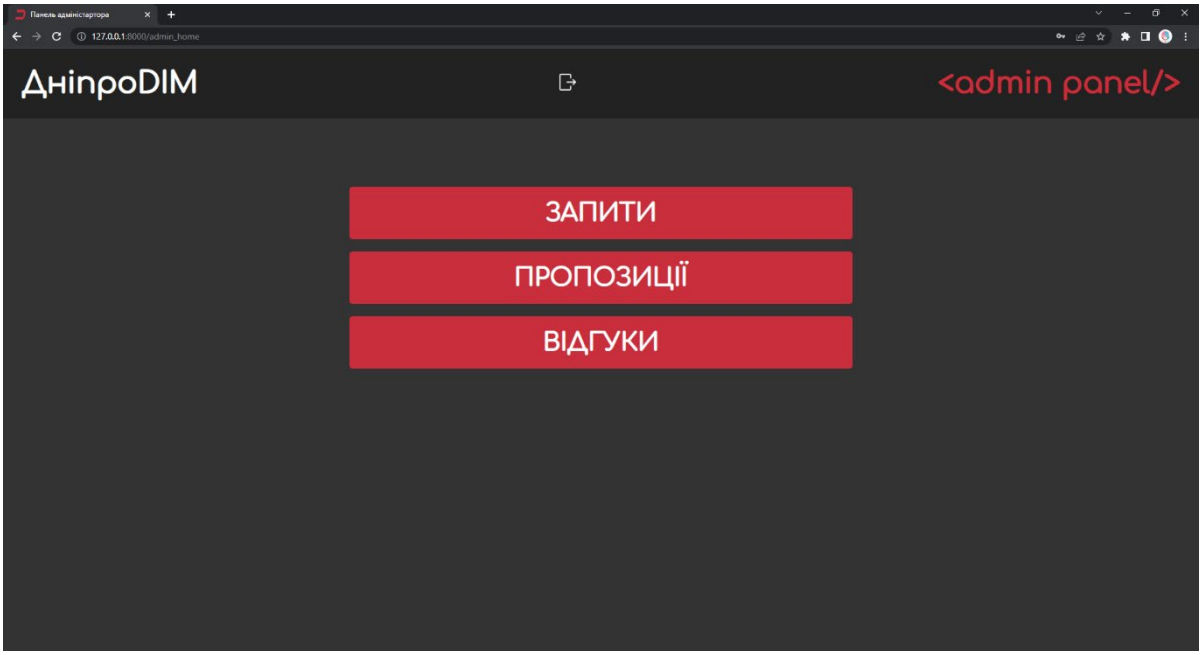

Рис. 2.30. Головна сторінка панелі адміністратора

На цій сторінці можна побачити три кнопки: «ЗАПИТИ», «ПРОПОЗИЦІЇ» та «ВІДГУКИ». Також у «header-і» активною є надпис «ДніпроDIM», яка пересилає адміністратора на головну сторінку та кнопка посередині, яка завершує сесію.

При переході на сторінку запитів (рис 2.31) відображаються всі поточні запити користувачів. Кожен запит містить інформацію про пропозицію, а також контактні дані користувача для зв'язку з ним. Якщо розмова з користувачем пройшла успішно, є дві праві кнопки, які дозволяють видалити всі запити поточного користувача. Це стосується випадків, коли було досягнуто домовленості про певну пропозицію і інші запити стають непотрібними. Друга кнопка робить пропозицію неактивною, що означає, що вона більше не буде відображатися для інших користувачів.

У випадку невдалої розмови є кнопка «Видалити запит», яка дозволяє видалити конкретний запит.

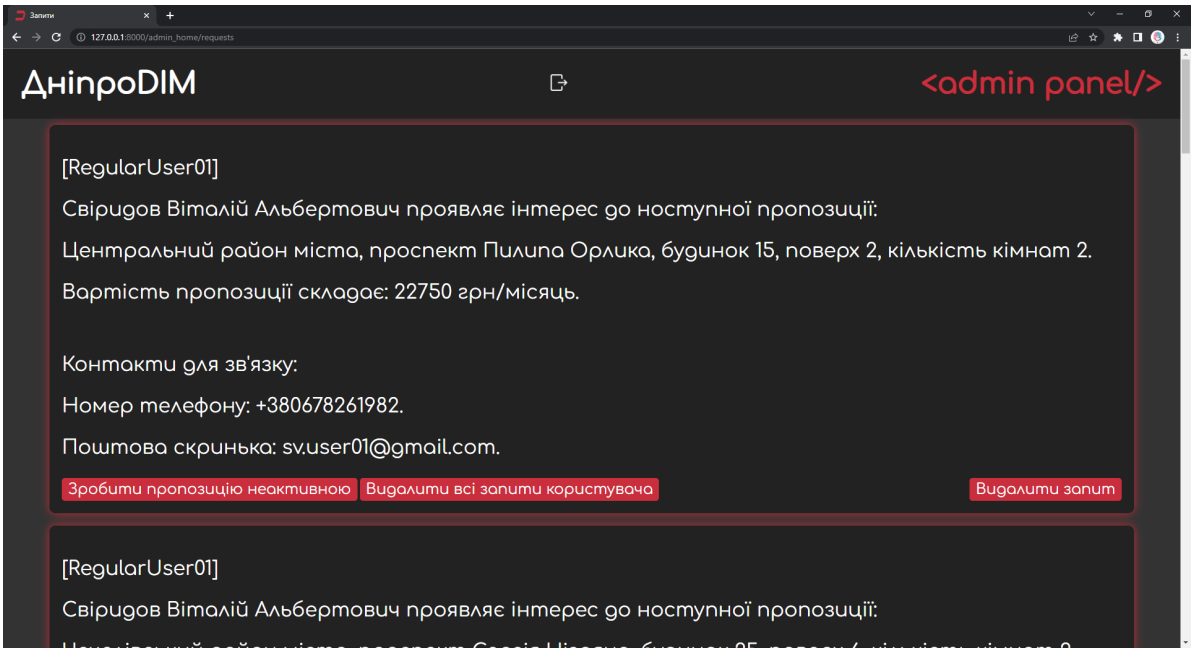

Рис. 2.31. Сторінка запитів

При натисканні на кнопку «Пропозиції» адміністратор переходить до сторінки з усіма пропозиціями (рис 2.32).

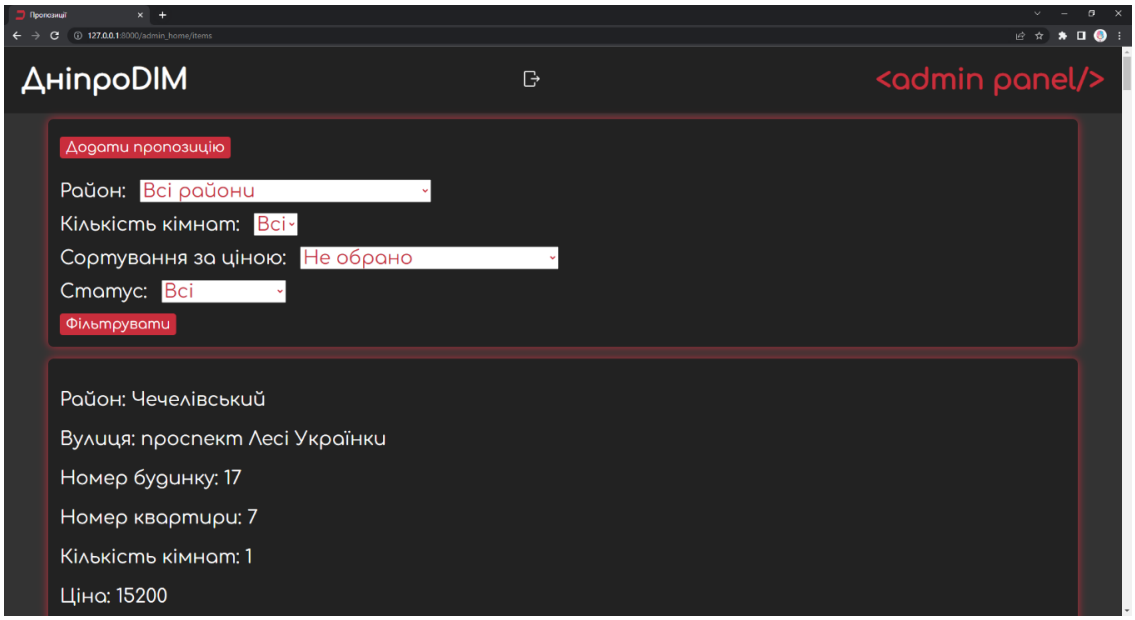

Рис. 2.32. Сторінка пропозицій

Вони представлені у вигляді списку. Також у верхній частині є кнопка «Додати пропозицію» та можливість відфільтрувати їх за різними параметрами.

При натисканні на кнопку «Додати пропозицію» відкривається вікно з додаванням (рис 2.33).

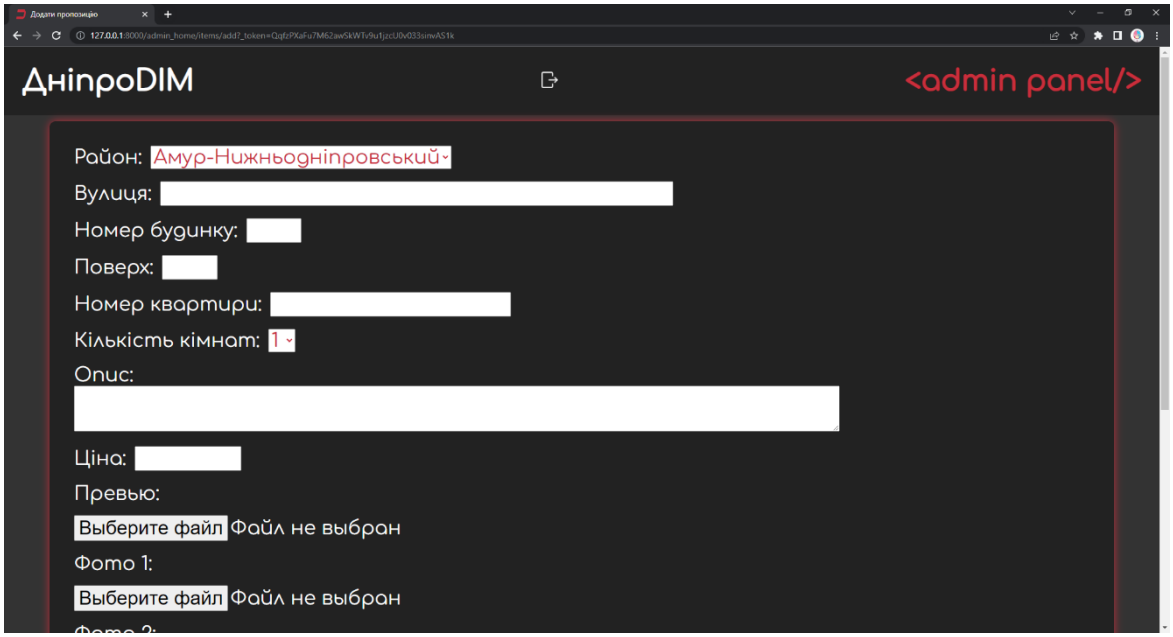

Рис. 2.33. Сторінка додавання пропозиції

Після заповнення всіх даних та додавання картинок, адміністратор натискає кнопку «Зберегти» (рис 2.34) щоб додати нову пропозицію. Важливо виставити статус «Активне», щоб нова пропозиція одразу висвічувалась користувачам.

Рис. 2.34. Кнопка збереження пропозиції

Також, на попередній сторінці біля кожної пропозиції є дві кнопки «Редагувати» та «Видалити» (рис 2.35).

Рис. 2.35. Кнопки біля пропозиції

При натисканні кнопки «Редагувати» відкриється вікно, аналогічне вікну додавання (рис 2.36), але в ньому вже будуть внесені дані, а кнопка «Зберегти» буде оновлювати пропозицію.

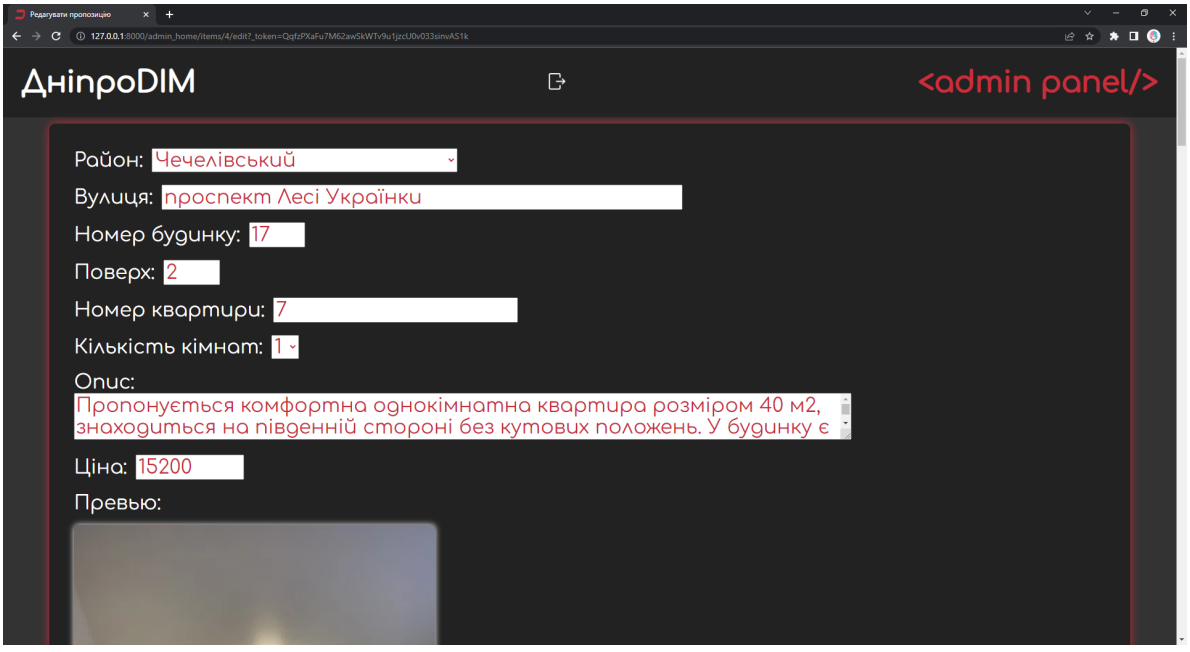

Рис. 2.36. Сторінка редагування пропозиції

При натисненні кнопки «Видалити» з бази буде видалятися відповідна пропозиція.

Нарешті розглянемо сторінку, яка відкривається по натисканню кнопки «Відгуки» (рис 2.37).

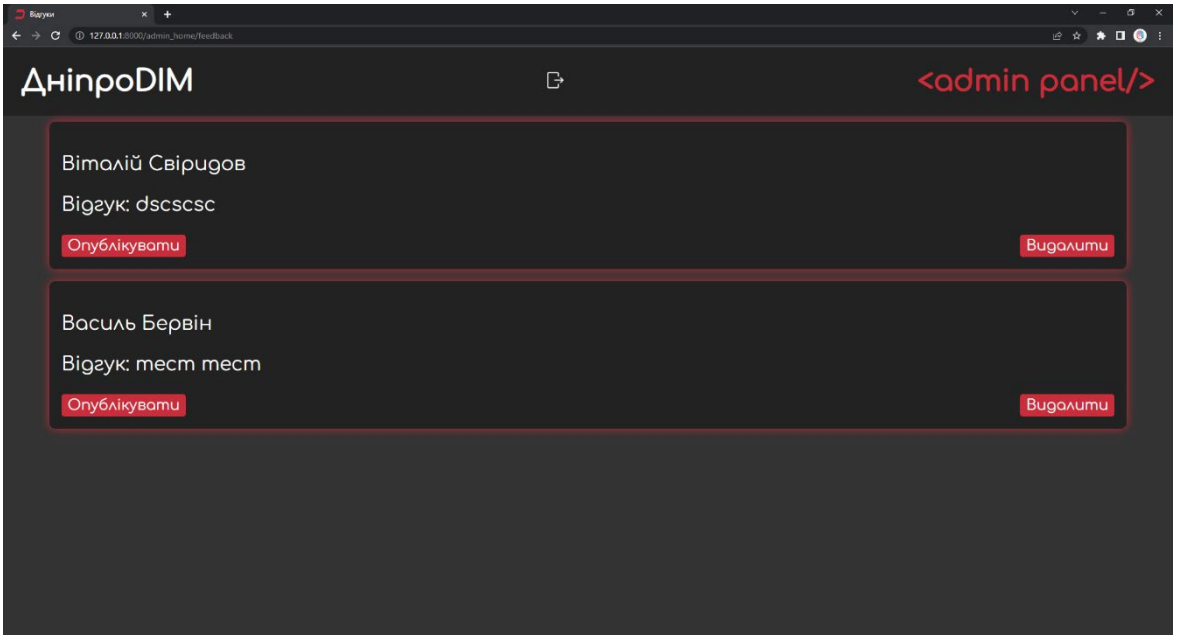

Рис. 2.37. Сторінка відгуків

Як можна побачити, тут знаходиться список усіх відгуків, які залишають користувачі. Біля кожного з них є дві кнопки: кнопка «Опублікувати» робить цей відгук видимий для всіх інших користувачів, а кнопка «Видалити», відповідно, видаляє цей відгук.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

- передбачуване число операторів програми – 1959;
- коефіцієнт складності програми – 1,25;
- коефіцієнт корекції програми в ході її розробки – 0,1;
- годинна заробітна плата програміста – 124 грн/год;

Відповідно до даних, наданих "Українською спільнотою програмістів (DOU)", початківець веб-розробник отримує середньомісячну заробітну плату у розмірі приблизно 600 доларів США. Зважаючи на курс валют Національного банку України на кінець червня 2023 року, що становить 36,57 гривень за один долар США, можна припустити, що середня заробітна плата в гривнях складає 21 942 гривень. З урахуванням восьмигодинного робочого дня (загалом 176 робочих годин на місяць), можна припустити, що середня заробітна плата за годину становить 124 гривні [23].

- коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,25;
- коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,3;
- вартість машино-години ЕОМ – 12 грн/год.

Оскільки за годину комп’ютер витрачає 0,6 кВт, а вартість електроенергії за кВт/год складає 2,64 грн [24], то за годину комп’ютер споживає електроенергії на 0,6 * 2,64 = 1.58 грн. Якщо додати інші витрати на забезпечення роботи комп’ютера, то вартість машино-години ЕОМ становить приблизно 15 грн/год.

Окрім витрат на комп'ютер, приміщення, комунікації та інші витрати, ціна була збільшена до рівня зразків, а саме до 12 гривень за годину.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{\text{отл}} + t_{\text{д}}, \quad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{\text{отл}}$ - витрати праці на налагодження програми на ЕОМ;

$t_{\text{д}}$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q * C * (1 + p) = q * C * (1 + p), \quad (3.2)$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

За формулою (3.2) умовне число операторів складає:

$$Q = 1959 * 1{,}25 * (1 + 0{,}1) = 2693{,}63$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q*B}{(75..85)*k}, \text{ людино-годин,} \quad (3.3)$$

де В – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на вивчення опису задачі за формулою (3.3) складають:

$$t_u = \frac{2696{,}63*1{,}25}{85*1{,}3} \approx 30{,}5, \text{ людино-годин,}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25)*k}, \text{ людино-годин,} \quad (3.4)$$

Виходячи з формули (3.4), витрати праці на розробку алгоритму рішення задачі складають:

$$t_a = \frac{2696{,}63}{25*1{,}3} \approx 82{,}97, \text{ людино-годин,}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)*k}, \text{ людино-годин,} \quad (3.5)$$

За формулою (3.5) витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{2696{,}63}{25*1{,}3} \approx 82{,}97, \text{ людино-годин},$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5)*k}, \text{ людино-годин}, \quad (3.6)$$

Використовуючи формулу (3.6) витрати праці на налагодження програми на ЕОМ за умови автономного налагодження одного завдання дорівнюють:

$$t_{\text{отл}} = \frac{2696{,}63}{5*1{,}3} \approx 414{,}87, \text{ людино-годин},$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1{,}5 * t_{\text{отл}}, \text{ людино-годин}, \quad (3.7)$$

Витрати праці на налагодження програми на ЕОМ за умови комплексного налагодження завдання за формулою (3.7) складають:

$$t_{\text{отл}}^{k} = 1{,}5 * 414{,}87 \approx 622{,}31, \text{ людино-годин},$$

Витрати праці на підготовку документації:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \text{ людино-годин}, \quad (3.8)$$

де $t_{\text{др}}$ - трудомісткість підготовки матеріалів і рукопису.

$$t_{\text{др}} = \frac{Q}{(15..20)*k}, \text{ людино-годин,} \quad (3.9)$$

$t_{\text{до}}$ - трудомісткість редагування, печатки й оформлення документації

$$t_{\text{до}} = 0{,}75 * t_{\text{др}}, \text{ людино-годин,} \quad (3.10)$$

За формулами (3.8) – (3.10) витрати праці на підготовку документації складають:

$$t_{\text{др}} = \frac{2696{,}63}{20*1{,}3} \approx 103{,}72, \text{ людино-годин,}$$

$$t_{\text{до}} = 0{,}75 * 103{,}72 \approx 77{,}79, \text{ людино-годин,}$$

$$t_{\text{д}} = 103{,}72 + 77{,}79 = 181{,}51, \text{ людино-годин,}$$

Розрахувавши всі показники, за формулою (3.1) отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 30{,}5 + 82{,}97 + 82{,}97 + 414{,}87 + 181{,}51 = 842{,}82,$$

людино-годин.

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми *Зз/п* і витрат машинного часу, необхідного на налагодження програми на ЕОМ

$$\text{К}_{\text{ПО}} = \text{З}_{\text{ЗП}} + \text{З}_{\text{МВ}}, \text{ грн.} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * С_{ПР}, \text{ грн}, \quad (3.12)$$

де: t – загальна трудомісткість, людино-годин;

$С_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

Враховуючи те, що середня годинна зарплата Junior Full-Stack розробника становить 124 грн/год, за формулою (3.12) заробітна плата виконавців:

$$З_{ЗП} = 842{,}82 * 124 \approx 104\,432{,}8, \text{ грн},$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} * С_{Мч}, \text{ грн}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.;

$С_{Мч}$ – вартість машино-години ЕОМ, грн/год.

За формулою (3.13) вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = 414{,}87 * 12 \approx 4\,978{,}44, \text{ грн},$$

Виходячи з цього, за формулою (3.11) витрати на створення програмного забезпечення:

$$К_{ПО} = 104\,432{,}8 + 4\,978{,}44 = 109\,411{,}24, \text{ грн},$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{ міс}, \tag{3.14}$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p =176 годин).

За формулою (3.14) очікуваний період створення програмного забезпечення:

$$T = \frac{842{,}82}{1 * 176} \approx 4{,}79 \text{ міс.}$$

Висновок: для розробки програмного продукту знадобиться витратити 842,82 людино-годин. З урахуванням вхідних даних можна припустити, що розробка продукту займе приблизно 4,79 місяців за стандартного робочого тижня та робочого місяця. Вартість цього програмного продукту складатиме 109411,24 гривень, що включає заробітну плату фахівців, витрати на придбання та обслуговування ЕОМ.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи була розроблена веб-орієнтована інформаційна система для організації та надання послуг з оренди житла на базі фреймворку Laravel. Результатом роботи є повнофункціональний web-сайт, який готовий до використання та орієнтований на клієнтів, які шукають можливостей для оренди житла.

У процесі розробки системи були використані сучасні технології для створення веб-сайтів та вивчене програмне забезпечення, що застосовується для розробки таких систем. Було проведено аналіз розміщення різноманітної інформації на веб-сторінках та дотримані основні правила і рекомендації при розробці веб-сайтів.

Структура та вміст веб-сторінок сайту були визначені з метою забезпечення зручності користувачів та логічного розташування інформації. Також був проведений аналіз працездатності розробленого веб-сайту, щоб переконатися у його ефективності та надійності.

Розроблений сайт відповідає всім вимогам, які були поставлені на етапі постановки завдання. Він реалізований на базі фреймворку Laravel та використовує мови програмування, такі як HTML, CSS і PHP.

Завдяки розробленій веб-орієнтованій інформаційній системі, підприємства з оренди житла зможуть забезпечити більш ефективну організацію та надання послуг, а також полегшити процеси управління та покращити взаємодію з клієнтами.

Крім того, під час виконання кваліфікаційної роботи було встановлено трудомісткість розробленого програмного продукту, яка складає 842,82 людино-години. Також було проведено розрахунок вартості розробки програмного продукту, яка оцінена в суму 109411,24 грн. Визначений час, необхідний для створення програмного продукту, складає 4,79 місяці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Key Derivation Function [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Key_derivation_function#Password_hashing (дата звернення 13.05.2023)
2. Користувацькі ролі на сайті. Хто є хто [Електронний ресурс] // URL: https://te-st.org/2020/11/10/wordpress-user-roles/ (дата звернення 20.05.2023)
3. HTML5 – що це? [Електронний ресурс] // URL: https://armedsoft.com/ua/blog/html5-shcho-ce (дата звернення 13.06.2023)
4. Що таке HTML та як за допомогою нього увійти в IT [Електронний ресурс] // URL: https://mc.today/uk/shho-take-html-ta-yak-za-dopomogoyu-nogo-uvijti-do-it/ (дата звернення 13.06.2023)
5. Що таке PHP? [Електронний ресурс] // URL: https://hyperhost.ua/uk/wiki/chto-takoe-php (дата звернення 13.06.2023)
6. PHP [Електронний ресурс] // URL: http://programming.in.ua/web-design/allphp/30-about-php.html (дата звернення 14.06.2023)
7. Переваги та недоліки використання мови PHP для Backend розробки [Електронний ресурс] // URL: https://smart-solutions.com.ua/archives/637 (дата звернення 17.06.2023)
8. Що таке CSS [Електронний ресурс] // URL: https://css.in.ua/article/shcho-take-html_10 (дата звернення 11.06.2023)
9. CSS [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://css.in.ua/article/shcho-take-css_3 (дата звернення 14.06.2023)
10. Каскадні таблиці стилів [Електронний ресурс] // URL: https://studfile.net/preview/1624162/ (дата звернення 18.06.2023)
11. The PHP Framework for Web Artisans [Електронний ресурс] // URL: https://laravel.com/ (дата звернення 19.06.2023)

12. Installation [Електронний ресурс] // URL: https://laravel.com/docs/10.x (дата звернення 19.06.2023)
13. Laravel vs Yii Framework: що краще для вашого бізнесу? [Електронний ресурс] // URL: https://wezom.com.ua/ua/blog/laravel-vs-yii-framework-chto-luchshe-dlja-vashego-biznesa (дата звернення 19.06.2023)
14. Клієнт-серверна архітектура [Електронний ресурс] // URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення 21.06.2023)
15. Клієнт-серверна архітектура [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура (дата звернення 22.06.2023)
16. Routing [Електронний ресурс] // URL: https://laravel.com/docs/10.x/routing (дата звернення 23.06.2023)
17. PHPStorm [Електронний ресурс] // URL: https://www.jetbrains.com/phpstorm/ (дата звернення 23.06.2023)
18. JetBrains [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JetBrains#IDEs (дата звернення 23.06.2023)
19. Що таке Figma [Електронний ресурс] // URL: https://avada-media.ua/ua/figma/ (дата звернення 23.06.2023)
20. Figma [Електронний ресурс] // Вікіпедія. Вільна енциклопедія – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Figma (дата звернення 23.06.2023)
21. Налаштування OpenServer [Електронний ресурс] // URL: https://armedsoft.com/ua/blog/nalashtuvannya-open-server (дата звернення 24.06.2023)
22. Bringing MySQL to the web [Електронний ресурс] // URL: https://www.phpmyadmin.net/ (дата звернення 24.06.2023)

23. Статистика зарплат програмістів [Електронний ресурс] // URL: https://jobs.dou.ua/salaries/?period=2022-12&position=Middle%20SE (дата звернення 29.06.2023)
24. Тарифи для населення [Електронний ресурс] // URL: https://yasno.com.ua/b2c-tariffs (дата звернення 29.06.2023)

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг файлу «web.php»

```
<?php

use App\Http\Controllers\AdminController;
use App\Http\Controllers\AuthorizationController;
use App\Http\Controllers\FeedBackController;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\ItemsController;
use App\Http\Controllers\RequestsController;
use Illuminate\Support\Facades\Route;

/** АВТОРИЗАЦІЯ **/
Route::get('/registration', [AuthorizationController::class, 'registration'])->name('registration');
Route::post('/registration/check', [AuthorizationController::class, 'registration_check'])->name('registration_check');
Route::get('/login', [AuthorizationController::class, 'login'])->name('login');
Route::post('/login/check', [AuthorizationController::class, 'login_check'])->name('login_check');
Route::post('/logout', [AuthorizationController::class, 'logout'])->name('logout');
Route::put('/home_account/{id}', [AuthorizationController::class, "update"])->name('update_account'); /** ЗМІНА ДАНИХ КОРИСТУВАЧА **/

/** ПЕРЕХІД МІЖ СТОРІНОК НА ДОМАШНІЙ СТОРІНЦІ САЙТУ **/
Route::get('/', [HomeController::class, 'home'])->name('home');
Route::get('/home_about', [HomeController::class, 'about_us'])->name('home_about_us');
Route::get('/home_feedback', [HomeController::class, 'feedback'])->name('home_feedback');
Route::post('/home_feedback/save', [FeedBackController::class, "save_feedback"])->name('save_feedback'); /** ЦЕЙ ПРО ВІДГУК **/
Route::get('/home_items', [HomeController::class, 'items'])->name('home_items');
Route::match(['get', 'post'], '/home_items/filter', [ItemsController::class, 'filter_home'])->name('filter_itemsHome'); /** ЦЕЙ ПРО ФІЛЬТРАЦІЮ **/
Route::get('/home_account', [HomeController::class, 'account'])->name('home_account');
Route::get('/home_itemInfo/{id}', [HomeController::class, 'itemInfo'])->name('home_itemInfo');

/** ЗАПИТИ **/
Route::post('/create_request', [RequestsController::class, 'create'])->name('create_request');
Route::delete('/delete_request/{id}', [RequestsController::class, 'delete'])->name('delete_request');
Route::delete('/admin_home/requests/{userId}/delete', [RequestsController::class, 'deleteRequests'])->name('admin_deleteRequests');
Route::delete('/admin/requests/{requestId}/delete', [RequestsController::class, 'deleteRequest'])->name('admin_deleteRequest');

/** ПЕРЕХІД МІЖ СТОРІНОК НА ПАНЕЛІ АДМІНА **/
Route::get('/admin_home', [AdminController::class, 'admin_home'])->name('admin_home');
Route::get('/admin_home/requests', [AdminController::class, "admin_requests"])->name('admin_requests');
Route::get('/admin_home/items', [AdminController::class, "admin_items"])->name('admin_items');
Route::get('/admin_home/feedback', [AdminController::class, "admin_feedback"])->name('admin_feedback');

/** ВСЕ ПРО ВІДГУКИ (АДМІН) **/
Route::get('/admin_home/feedback', [FeedBackController::class, "feedback"])->name('feedback');
Route::post('/admin_home/feedback/{id}/publish', [FeedBackController::class, "publish"])->name('publish_feedback');
Route::delete('/admin_home/feedback/{id}/delete', [FeedBackController::class, "delete"])->name('delete_feedback');
```

```
/** ВСЕ ПРО ПРОПОЗИЦІЇ (АДМІН) **/
Route::get('/admin_home/items', [ItemsController::class, "items"])->name('items');
Route::get('/admin_home/items/{item}/edit', [ItemsController::class, "edit"])->name('edit_item');
Route::put('/admin_home/items/{id}', [ItemsController::class, "update"])->name('update_item');
Route::get('/admin_home/items/add', [ItemsController::class, "add"])->name('add_item');
Route::post('/admin_home/items/add/save', [ItemsController::class, "save"])->name('save_item');
Route::delete('/admin_home/items/{id}', [ItemsController::class, "delete"])->name('delete_item');
Route::match(['get', 'post'], '/admin_home/filter_items', [ItemsController::class, 'filter'])->name('filter_items');
Route::put('/admin_home/items/{id}/deactivate', [ItemsController::class, 'deactivateItem'])-
>name('admin_deactivateItem');
```

Лістинг файлу «AdminController.php»

```
<?php

namespace App\Http\Controllers;

use App\Models\Requests;
use Illuminate\Http\Request;

class AdminController extends Controller
{
   public function admin_home() { /** ПЕРЕХІД НА ДОМАШНЮ СТОРІНКУ АДМІНПАНЕЛІ **/
      return view('admin_home');
   }
   public function admin_requests() { /** ПЕРЕХІД НА СТОРІНКУ ЗАПИТІВ АДМІНПАНЕЛІ **/
      $requests = Requests::with(['user', 'item'])->get();
      return view('admin_requests', compact('requests'));
   }
   public function admin_items() { /** ПЕРЕХІД НА СТОРІНКУ ПРОПОЗИЦІЙ АДМІНПАНЕЛІ **/
      return view('admin_items');
   }
   public function admin_feedback() { /** ПЕРЕХІД НА СТОРІНКУ ВІДГУКІВ АДМІНПАНЕЛІ **/
      return view('admin_feedback');
   }
}
```

Лістинг файлу «AuthorisationController.php»

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

class AuthorizationController extends Controller
{
   public function registration() { /** ПЕРЕХІД НА СТОРІНКУ РЕЄСТРАЦІЇ **/
      return view('registration');
   }

   public function registration_check(Request $request) { /** ВАЛІДАЦІЯ ДАНИХ НОВОГО
КОРИСТУВАЧА ТА ЗБЕРІГАННЯ ДО БД **/
      $customAttributes = [
         'login' => 'Логін',
         'password' => 'Пароль',
         'name1' => 'Ім\'я',
         'name2' => 'Прізвище',
         'name3' => 'По батькові',
```

```
            'phone_number' => 'Номер телефону',
            'email' => 'Email',
        ];

        $validatedData = $request->validate([
            'login' => 'required|min:4|max:15|unique:users',
            'password' => 'required|min:8',
            'name1' => 'required',
            'name2' => 'required',
            'name3' => 'required',
            'phone_number' => 'required|digits:10',
            'email' => 'required|email|unique:users',
        ], [], $customAttributes);


        $hashedPassword = Hash::make($validatedData['password']);

        $user = new User();
        $user->login = $validatedData['login'];
        $user->password = $hashedPassword;
        $user->name1 = $validatedData['name1'];
        $user->name2 = $validatedData['name2'];
        $user->name3 = $validatedData['name3'];
        $user->phone_number = $validatedData['phone_number'];
        $user->email = $validatedData['email'];

        $user->save();

        return redirect()->route('registration')->withErrors(['success' => 'Реєстрація пройшла успішно!']);
    }

    public function login() { /** ПЕРЕХІД НА СТОРІНКУ АВТОРИЗАЦІЇ **/
        return view('login');
    }

    public function login_check(Request $request) { /** ПЕРЕВІРКА АВТОРИЗАЦІЇ **/
        $credentials = $request->only('login', 'password');

        if (Auth::attempt($credentials)) {
            $user = Auth::user();
            $request->session()->put('user_id', $user->id);

            if ($user->is_admin == 1) {
                return redirect()->route('admin_home');
            } elseif ($user->is_admin == 0) {
                return redirect()->route('home');
            }
        } else {
            return redirect()->route('login')->withErrors(['message' => 'Неправильні дані авторизації']);
        }

    }

    public function logout(Request $request) { /** ВИХІД З АКАУНТУ **/
        $request->session()->forget('user_id');

        Auth::logout();

        return redirect()->route('home');
    }

    public function update(Request $request, $id) { /** ЗБЕРЕЖЕННЯ ЗМІНЕНИХ ДАННИХ КОРИСТУВАЧА **/
```

```
        $user = User::find($id);
        $user->login = $request->input('login');
        $user->name1 = $request->input('name1');
        $user->name2 = $request->input('name2');
        $user->name3 = $request->input('name3');
        $user->phone_number = $request->input('phone_number');
        $user->email = $request->input('email');

        $user->save();

        return redirect()->route('home_account')->with('success', 'Інформацію оновлено!');
    }
}
```

Лістинг файлу «FeedBackController.php»

```
<?php

namespace App\Http\Controllers;

use App\Models\Feedback;
use App\Models\User;
use Illuminate\Http\Request;

class FeedBackController extends Controller
{
    public function feedback() { /** ВИВВІД ВСІХ ВІДГУКІВ НА СТОРІНЦІ ВІДГУКІВ АДМІНПАНЕЛІ
**/
        $feedback = new Feedback();
        return view('admin_feedback', ['feedback' => $feedback->all()]);
    }

    public function publish($id) { /** ПУБЛІКАЦІЯ ВІДГУКУ **/
        $feedback = Feedback::findOrFail($id);
        $feedback->is_verified = 1;
        $feedback->save();

        return redirect()->back();
    }

    public function delete($id) { /** ВИДАЛЕННЯ ВІДГУКУ **/
        $feedback = Feedback::findOrFail($id);
        $feedback->delete();

        return redirect()->back();
    }

    public function save_feedback(Request $request) { /** ПЗБЕРЕЖЕННЯ ВІДГУКУ **/
        $userId = $request->session()->get('user_id');
        $user = User::findOrFail($userId);

        $feedback = new Feedback();
        $feedback->first_name = $user->name1;
        $feedback->last_name = $user->name2;
        $feedback->comment = $request->input('comment');
        $feedback->save();

        return redirect()->route('home_feedback')->with('success', 'Відгук успішно додано!');
    }
}
```

Лістинг файлу «HomeController.php»

```
<?php

namespace App\Http\Controllers;

use App\Models\Feedback;
use App\Models\Items;
use App\Models\Requests;
use App\Models\User;
use Illuminate\Http\Request;

class HomeController extends Controller
{
    public function home() { /** ВИВЕДЕННЯ : ВИПАДКОВИХ ПРОПОЗИЦІЙ **/
        $random = Items::where('is_active', 1)
            ->inRandomOrder()
            ->limit(6)
            ->get();

        return view('home_page', ['random' => $random]);
    }

    public function about_us() { /** ПЕРЕХІД НА СТОРІНКУ З ІНФОРМАЦІЄЮ О НАС **/
        return view('home_about');
    }

    public function feedback() {/** ВИВЕДЕННЯ ЛИШЕ ПІДТВЕРДЖЕНИХ ВІДГУКІВ **/
            $feedback = Feedback::where('is_verified', 1)->get();
            return view('home_feedback', ['feedback' => $feedback]);
    }

    public function itemInfo($id) {/** ВИВЕДЕННЯ ІНФОРМЦІЇ ПРО ПРОПОЗИЦІЮ **/
        $item = Items::find($id);

        return view('home_itemInfo', ['item' => $item]);
    }

    public function items() { /** ВИВЕДЕННЯ СПИСКУ ПРОПОЗИЦІЙ **/
        $items = new Items();
        return view('home_items', ['items' => $items->all()]);
    }

    public function account() { /** ВИВЕДЕННЯ ІНФОРМАЦІЇ ПРО АКАУНТ **/
        $userId = session('user_id');
        $user = User::findOrFail($userId);

        $requests = Requests::where('user_id', $userId)->get();

        $items = Items::whereIn('id', function ($query) use ($requests) {
            $query->select('item_id')
                ->from('requests')
                ->whereIn('id', $requests->pluck('id'));
        })->get();

        return view('home_account', [
            'user' => $user,
            'requests' => $requests,
            'items' => $items,
        ]);
    }
}
```

Лістинг файлу «ItemsController.php»

```
<?php

namespace App\Http\Controllers;

use App\Models\Items;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\File;

class ItemsController extends Controller
{
   public function items() { /** ВИВЕДЕННЯ ІНФОРМАЦІЇ ПРО ВІДГУКИ **/
      $items = Items::all();
      return view('admin_items', ['items' => $items]);
   }

   public function add() { /** ПЕРЕХІД НА СТОРІНКУ ДОДАВАННЯ ПРОПОЗИЦІЇ **/
      return view('admin_itemsAdd');
   }

   public function save(Request $request) { /** ЗБЕРЕЖЕННЯ НОВОЇ ПРОПОЗИЦІЇ **/
      $item = new Items();
      $item->district = $request->input('district');
      $item->street = $request->input('street');
      $item->house_number = $request->input('house_number');
      $item->floor = $request->input('floor');
      $item->apartment_number = $request->input('apartment_number');
      $item->room_count = $request->input('room_count');
      $item->description = $request->input('description');
      $item->price = $request->input('price');
      $item->is_active = $request->input('is_active');

      if ($request->hasFile('image_preview')) {
         $image = $request->file('image_preview');
         $imageData = File::get($image);
         $item->image_preview = $imageData;
      }

      if ($request->hasFile('image_1')) {
         $image = $request->file('image_1');
         $image_1Data = File::get($image);
         $item->image_1 = $image_1Data;
      }

      if ($request->hasFile('image_2')) {
         $image = $request->file('image_2');
         $image_2Data = File::get($image);
         $item->image_2 = $image_2Data;
      }

      if ($request->hasFile('image_3')) {
         $image = $request->file('image_3');
         $image_3Data = File::get($image);
         $item->image_3 = $image_3Data;
      }

      if ($request->hasFile('image_4')) {
         $image = $request->file('image_4');
         $image_4Data = File::get($image);
         $item->image_4 = $image_4Data;
      }
```

```
        if ($request->hasFile('image_5')) {
            $image = $request->file('image_5');
            $image_5Data = File::get($image);
            $item->image_5 = $image_5Data;
        }

        $item->save();

        return redirect()->route('items')->with('success', 'Пропозиція була успішно додана!');
    }

    public function delete($id) { /** ВИДАЛЕННЯ ПРОПОЗИЦІЇ **/
        $item = Items::find($id);
        $item->delete();

        return redirect()->route('items');
    }

    public function edit(Items $item) { /** ПЕРЕХІД НА СТОРІНКУ РЕДАГУВАННЯ ПРОПОЗИЦІЇ **/
        return view('admin_itemsEdit', ['item' => $item]);
    }

    public function update(Request $request, $id) { /** ЗБЕРЕЖЕННЯ ОНОВЛЕНИХ ДАНИХ ПРО ПРОПОЗИЦІЮ **/
        $item = Items::find($id);
        $item->district = $request->input('district');
        $item->street = $request->input('street');
        $item->house_number = $request->input('house_number');
        $item->floor = $request->input('floor');
        $item->apartment_number = $request->input('apartment_number');
        $item->room_count = $request->input('room_count');
        $item->description = $request->input('description');
        $item->price = $request->input('price');
        $item->is_active = $request->input('is_active');

        if ($request->hasFile('image_preview')) {
            $image = $request->file('image_preview');
            $imageData = File::get($image);
            $item->image_preview = $imageData;
        }

        if ($request->hasFile('image_1')) {
            $image = $request->file('image_1');
            $image_1Data = File::get($image);
            $item->image_1 = $image_1Data;
        }

        if ($request->hasFile('image_2')) {
            $image = $request->file('image_2');
            $image_2Data = File::get($image);
            $item->image_2 = $image_2Data;
        }

        if ($request->hasFile('image_3')) {
            $image = $request->file('image_3');
            $image_3Data = File::get($image);
            $item->image_3 = $image_3Data;
        }

        if ($request->hasFile('image_4')) {
            $image = $request->file('image_4');
            $image_4Data = File::get($image);
            $item->image_4 = $image_4Data;
```

```
        }

        if ($request->hasFile('image_5')) {
            $image = $request->file('image_5');
            $image_5Data = File::get($image);
            $item->image_5 = $image_5Data;
        }

        $item->save();

        return redirect()->route('items')->with('success', 'Пропозицію було успішно оновлено!');
    }

    public function filter(Request $request) {/** ФІЛЬТРУВАННЯ ПРОПОЗИЦІЙ **/
        $district = $request->input('district');
        $roomCount = $request->input('room_count');
        $sortByPrice = $request->input('sort_by_price');
        $status = $request->input('status');

        $query = Items::query();

        if ($district) {
            $query->where('district', $district);
        }

        if ($roomCount) {
            $query->where('room_count', $roomCount);
        }

        if ($status) {
            if ($status === 'active') {
                $query->where('is_active', 1);
            } elseif ($status === 'inactive') {
                $query->where('is_active', 0);
            }
        }

        if ($sortByPrice) {
            $query->orderBy('price', $sortByPrice);
        }

        $filteredItems = $query->get();

        return view('admin_items', ['items' => $filteredItems]);
    }

    public function deactivateItem($id) { /** ПЕРЕХІД ПРОПОЗИЦІЇ ДО НЕАКТИВНОГО СТАТУСУ **/
        {
            $item = Items::findOrFail($id);
            $item->is_active = 0;
            $item->save();

            return redirect()->route('admin_requests');
        }
    }

    public function filter_home(Request $request) {/** ФІЛЬТРУВАННЯ ПРОПОЗИЦІЙ НА ДОМАШНІЙ
СТОРІНЦІ **/
        $district = $request->input('district');
        $roomCount = $request->input('room_count');
        $sortByPrice = $request->input('sort_by_price');

        $query = Items::query();
```

```
        if ($district) {
            $query->where('district', $district);
        }

        if ($roomCount) {
            $query->where('room_count', $roomCount);
        }


        if ($sortByPrice) {
            $query->orderBy('price', $sortByPrice);
        }

        $filteredItems = $query->get();

        return view('home_items', ['items' => $filteredItems]);
    }
}
```

Лістинг файлу «RequestsController.php»

```
<?php

namespace App\Http\Controllers;

use App\Models\Requests;
use Illuminate\Http\Request;

class RequestsController extends Controller
{
    public function create(Request $request) {/** ЗБЕРЕЖЕННЯ НОВОГО ЗАПИТУ **/
        $requestData = $request->only(['user_id', 'item_id']);

        Requests::create($requestData);

        return redirect()->back()->with('success', 'Запит успішно створено');
    }

    public function delete($id) { /** ВИДАЛЕНЯ ЗАПИТУ **/
        $request = Requests::find($id);

        if ($request) {
            $request->delete();
            return redirect()->back()->with('success', 'Запит успішно видалено');
        } else {
            return redirect()->back()->with('error', 'Запит не знайдено');
        }
    }

    public function deleteRequests($id) { /** ВИДАЛЕННЯ ВСІХ ЗАПИТІВ ВІД ПЕВНОГО КОРИСТУВАЧА **/
        Requests::where('user_id', $id)->delete();

        return redirect()->route('admin_requests');
    }

    public function deleteRequest($id) {/** ВИДАЛЕННЯ ПЕВНОГО ЗАПИТУ **/
        Requests::findOrFail($id)->delete();

        return redirect()->route('admin_requests');
    }
```

Лістинг файлу «admin_requests.blade.php»

```
@extends('admin_sample')

@section('title', 'Запити')

@section('main')
    @foreach ($requests as $req)
        <div class="feedback-block">

            <!-- ОПИС -->
            <div class="">
                <p>[{{ $req->user->login }}]</p>
                <p>{{ $req->user->name2 }} {{ $req->user->name1 }} {{ $req->user->name3 }} проявляє інтерес до ноступної пропозиції:</p>
                <p>{{ $req->item->district }} район міста, {{ $req->item->street }}, будинок {{ $req->item->house_number }}, поверх {{ $req->item->floor }}, кількість кімнат {{ $req->item->room_count }}.</p>
                <p>Вартість пропозиції складає: {{ $req->item->price }} грн/місяць.</p>
                <br><p>Контакти для зв'язку:</p>
                <p>Номер телефону: +38{{ $req->user->phone_number }}.</p>
                <p>Поштова скринька: {{ $req->user->email }}.</p>
            </div>

            <div class="feedback-buttons">
                <div>
                <form action="{{ route('admin_deactivateItem', $req->item->id) }}" method="POST" style="display: inline-block">
                    @csrf
                    @method('PUT')

                    <!-- КНОПКА РОБИТЬ ПРОПОЗИЦІЮ НЕАКТИВНОЮ -->
                    <button type="submit" class="btn btn-danger">Зробити пропозицію неактивною</button>
                </form>
                <form action="{{ route('admin_deleteRequests', $req->user->id) }}" method="POST" style="display: inline-block">
                    @csrf
                    @method('DELETE')

                    <!-- КНОПКА ВИДАЛЯЄ ВСІ ЗАПИТИ КОРИСТУВАЧА -->
                    <button type="submit" class="btn btn-primary">Видалити всі запити користувача</button>
                </form>
                </div>
                <form action="{{ route('admin_deleteRequest', $req->id) }}" method="POST" style="display: inline-block">
                    @csrf
                    @method('DELETE')

                    <!-- КНОПКА ВИДАЛЯЄ ЗАПИТ -->
                    <button type="submit" class="btn btn-warning">Видалити запит</button>
                </form>
            </div>
        </div>
    @endforeach
@endsection
```

Лістинг файлу «home_account.blade.php»

```
@extends('home_sample')

@section('title', $user->login)

@section('main')
  <div>
    <form action="{{ route('update_account', $user->id)}}" method="POST" enctype="multipart/form-data">
      @csrf
      @method('PUT')

      <!-- ЗМІНА ІНФОРМАЦІЇ ПРО КОРИСТУВАЧА -->
      <div style="margin: 1%">
        <h1 class="black-text" style="font-size: 2.0em">Інформація про користувача</h1>
        <div>
          <label for="login" class="black-text" style="font-size: 1.5em">Логін:</label>
          <input name="login" id="login" class="red-text" style="font-size: 1.5em; text-align: left" value="{{
$user->login }}">
        </div>

        <div>
          <label for="name1" class="black-text" style="font-size: 1.5em">Ім'я:</label>
          <input name="name1" id="name1" class="red-text" style="font-size: 1.5em; text-align: left"
value="{{ $user->name1 }}">
        </div>

        <div>
          <label for="name2" class="black-text" style="font-size: 1.5em">Прізвище:</label>
          <input name="name2" id="name2" class="red-text" style="font-size: 1.5em; text-align: left"
value="{{ $user->name2 }}">
        </div>

        <div>
          <label for="name3" class="black-text" style="font-size: 1.5em">По батькові:</label>
          <input name="name3" id="name3" class="red-text" style="font-size: 1.5em; text-align: left"
value="{{ $user->name3 }}">
        </div>

        <div>
          <label for="phone_number" class="black-text" style="font-size: 1.5em"> Номер
телефону:</label>
          <input name="phone_number" id="phone_number" class="red-text" style="font-size: 1.5em; text-
align: left" value="{{ $user->phone_number }}">
        </div>

        <div>
          <label for="email" class="black-text" style="font-size: 1.5em">E-Mail:</label>
          <input name="email" id="email" class="red-text" style="font-size: 1.5em; text-align: left"
value="{{ $user->email }}">
        </div>

        <!-- КНОПКА ЗБЕРЕЖЕННЯ -->
        <button type="submit" class="red-button">Зберегти</button>
      </div>

    </form>
  </div>
  <br>

  <h1 class="black-text" style="font-size: 2.0em; margin: 1%">Інформація про пропозиції, у яких Ви
заінтересовані</h1>
  <div style="margin: 1%" class="black-text">
```

```
        @foreach ($items as $item)

            <!-- ВСІ ПРОПОЗИЦІЇ, ЯКИМИ ЦІКАВИТЬСЯ КОРИСТУВАЧ -->
            <div class="aaaaa" style="border: 2px solid #C92E3C; padding: 1%; border-radius: 10px; transition: box-shadow 0.2s;">
                <p style="font-size: 1.25em">Район: <span class="red-text">{{ $item->district }}</span></p>
                <p style="font-size: 1.25em">Вулиця: <span class="red-text">{{ $item->street }}</span></p>
                <p style="font-size: 1.25em">Номер будинку: <span class="red-text">{{ $item->house_number }}</span></p>
                <p style="font-size: 1.25em">Кількість кімнат: <span class="red-text">{{ $item->room_count }}</span></p>
                <p style="font-size: 1.25em">Ціна: <span class="red-text">{{ $item->price }}</span> грн/місяць</p>
                <a href="/home_itemInfo/{{ $item->id }}" class="red-button" style="text-decoration: none">Переглянути</a><br>
            </div><br>
        @endforeach
    </div>
@endsection
```

Лістинг файлу «home_itemInfo.blade.php»

```
@extends('home_sample')

@section('title', 'Інформація про пропозицію')

@section('main')
    @php
        $user_id = session('user_id');
        $request = \App\Models\Requests::where('user_id', $user_id)
                    ->where('item_id', $item->id)
                    ->first();
    @endphp

    <div class="black-text" style="margin: 1%">
        <div style="display: flex; align-items: center; justify-content: space-between;">
            <h1 style="font-size: 2.25em; margin-right: auto;">Інформація про пропозицію</h1>
            <div>
                @if ($user_id)
                    @if ($request === null)

                        <!-- КНОПКА ЗАЦІКАВЛЕНОСТІ, ЯКЩО КОРИСТУВАЧ НЕ ЦІКАВИВСЯ ЦІЄЮ ПРОПОЗИЦІЄЮ -->
                        <form action="{{ route('create_request') }}" method="POST">
                            @csrf
                            <input type="hidden" name="user_id" value="{{ $user_id }}">
                            <input type="hidden" name="item_id" value="{{ $item->id }}">
                            <button type="submit" class="red-button">Цікавить</button>
                        </form>
                    @else
                        @if ($request->user_id == $user_id)

                            <!-- КНОПКА НЕЗАЦІКАВЛЕНОСТІ, ЯКЩО КОРИСТУВАЧ ЦІКАВИВСЯ ЦІЄЮ ПРОПОЗИЦІЄЮ -->
                            <form action="{{ route('delete_request', ['id' => $request->id]) }}" method="POST">
                                @csrf
                                @method('DELETE')
                                <button type="submit" class="red-button">Не цікавить</button>
                            </form>
                        @else
                            <p>Ви вже подали запит на цю пропозицію</p>
                        @endif
                    @endif
```

```
                @else
                  <p class="black-text" style="font-size: 1em">Необхідно <a href="/login" class="red-button" style="font-size: 1em; text-decoration: none"> увійти</a> , щоб зацікавитися товаром</p>
                @endif
              </div>
            </div>

            <!-- ОПИС ПРОПОЗИЦІЇ -->
            <p style="font-size: 1.75em"><span class="red-text">Район:</span> {{ $item->district }}</p>
            <p style="font-size: 1.75em"><span class="red-text">Вулиця:</span> {{ $item->street }}</p>
            <p style="font-size: 1.75em"><span class="red-text">Номер будинку:</span> {{ $item->house_number }}</p>
            <p style="font-size: 1.75em"><span class="red-text">Поверх:</span> {{ $item->floor }}</p>
            <p style="font-size: 1.75em"><span class="red-text">Номер квартири:</span> {{ $item->apartment_number }}</p>
            <p style="font-size: 1.75em"><span class="red-text">Кількість кімнат:</span> {{ $item->room_count }}</p>
            <p style="font-size: 1.75em"><span class="red-text">Опис:</span> {{ $item->description }}</p>
            <p style="font-size: 1.75em"><span class="red-text">Ціна:</span> {{ $item->price }}</p>

            <!-- ВИВІД ЗОБРАЖЕНЬ -->
            <h1 style="font-size: 2.25em">Зображення</h1>
            <div class="images">
              <div class="image">
                <img src="{{ base64_encode($item->image_preview) }}" alt="NO IMAGE">
              </div>
              <div class="image">
                <img src="{{ base64_encode($item->image_1) }}" alt="NO IMAGE">
              </div>
              <div class="image">
                <img src="{{ base64_encode($item->image_2) }}" alt="NO IMAGE">
              </div>
              <div class="image">
                <img src="{{ base64_encode($item->image_3) }}" alt="NO IMAGE">
              </div>
              <div class="image">
                <img src="{{ base64_encode($item->image_4) }}" alt="NO IMAGE">
              </div>
              <div class="image">
                <img src="{{ base64_encode($item->image_5) }}" alt="NO IMAGE">
              </div>
            </div>
          </div>

        @endsection
```

Лістинг файлу «home_page.blade.php»

```
        @extends('home_sample')

        @section('title', 'ДніпроDIM')

        @section('main')
          <div style="display: flex; justify-content: center; margin: 1%">
            <div style="max-width: 35%; border: 1px solid #C92E3C; box-shadow: 0 0 10px #C92E3C; border-radius: 2px">
              <img src="{{ asset('storage/images/img.png') }}" alt="NO IMAGE" style="width: 100%;">
            </div>
          </div>

          <h1 class="black-text" style="font-size: 2.25em; margin: 1%">Пропозиції, які можуть Вас зацікавити:</h1>
```

```
        <div style="display: flex; flex-wrap: wrap; justify-content: center;">
            @foreach($random as $rand)
                @if($rand->is_active)
                    <a href="{{ route('home_itemInfo', ['id' => $rand->id]) }}" style="text-decoration: none; color: inherit; width: 25%; padding: 10px; margin: 1%;">
                        <div class="item-container">
                            <img src="{{ base64_encode($rand->image_preview) }}" alt="NO IMAGE" width="100%" height="300px" title="{{ $rand->description }}" style="object-fit: cover;">
                            <p class="black-text">Район: {{ $rand->district }}</p>
                            <p class="black-text">Вулиця: {{ $rand->street }}</p>
                            <p class="black-text">Кількість кімнат: {{ $rand->room_count }}</p>
                            <p class="red-text">{{ $rand->price }} грн/місяць</p>
                        </div>
                    </a>
                @endif
            @endforeach
        </div>
    @endsection
```

Решту файлів можна переглянути в прикріпленому до роботи носії на якому буде архів з повною роботою.

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра

на тему:

«Розробка веб-орієнтованої інформаційної системи для організації та надання послуг з оренди житла на базі фреймворку Laravel»

студента групи 122-20ск-1 Матвєєва Євгеній Олександровича

Керівник економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_Матвєєв.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_Матвєєв.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
PZ_Laravel.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Матвєєв.ppt	Презентація кваліфікаційної роботи