

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ВИЗНАЧЕННЯ ВАРТОСТІ ДОВГОСТРОКОВОГО
ВИНАЙМАННЯ БУДИНКІВ. RESEARCH AND SOFTWARE
IMPLEMENTATION OF METHODS TO SET THE PRICE
FOR A LONG-TERM HOUSE RENTAL

Виконав(ла): студент(ка) 2 курсу, групи КНТ-122м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

СОКОЛ Р.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ ФКНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« _____ » _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СОКОЛА Романа Володимировича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів визначення вартості довгострокового винаймання будинків. Research and Software Implementation of Methods to Set the Price for a Long-term House Rental

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

[illegible]

Студент(ка)

Підпис
 (підпис)

Роман СОКОЛ
 (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи) _____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
113 с., 31 рис., 4 табл., 3 дод., 13 джерел.

БУДИНОК, ДОВГОСТРОКОВЕ ВИНАЙМАННЯ, ВІДВІДУВАННЯ
БУДИНКУ, ДЕРЕВА РІШЕНЬ, RANDOM FOREST.

Об'єкт роботи – процес довгострокового винаймання будинків.
Предмет роботи – методи визначення вартості довгострокового винаймання будинків. Мета роботи – розробити програмне забезпечення довгострокового винаймання будинків для надання інформації для обґрунтованого прийняття рішень та підвищення взаємодії між власниками будинків та винаймачами.

Матеріали, методи та технічні засоби: мова Python, база даних на основі системи PostgreSQL, фреймворк Django, метод Random Forest.

Результати. Розроблена програма призначена для підтримки довгострокового винаймання будинків, оголошення про яке створюється власником будинку, надаючи для цього інструменти пошуку оголошень, створення і визначення параметрів оголошень, створення запитів на відвідування будинку, консультування без відвідування, створення відгуків за результатами відвідування або винаймання, створення скарг на відвідувачів або власників будинків за необхідності та підтримки процесів вирішення конфліктів менеджерами.

Висновки. Наукова новизна роботи полягає в методі визначення вартості довгострокового винаймання будинків, який ґрунтується на створенні моделі прогнозування на основі методу Random Forest, даних відібраних інших будинків у заданому місті за визначеними ознаками, що дозволяє визначити очікувану місячну вартість винаймання будинку.

Галузь використання. Ринок винаймання нерухомості, довгострокове винаймання будинків.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 113 pages, 31 figures, 4 tables, 3 appendixes, 13 sources.

HOUSE, LONG-TERM RENTAL, HOUSE VISIT, DECISION TREE, RANDOM FOREST.

The object of the work is the process of long-term renting of houses. The subject of the work is methods of determining the cost of long-term renting of buildings. The goal of the work is to develop software for long-term housing rentals to provide information for informed decision-making and improve interaction between homeowners and renters.

Materials, methods and technical means: Python language, database based on the PostgreSQL system, Django framework, Random Forest method.

The results. The developed program is designed to support the long-term rental of houses, the announcement of which is created by the owner of the house, providing tools for searching for ads, creating and defining the parameters of ads, creating requests for home visits, consulting without visiting, creating reviews based on the results of visits or renting, creating complaints about visitors or homeowners as necessary and support of conflict resolution processes by managers.

Conclusions. The scientific novelty of the work is in the method of determining the cost of long-term rental of houses, which is based on the creation of a prediction model based on the Random Forest method, data sample of other houses selected in a given city according to certain characteristics, which allows determining the expected monthly cost of renting a house.

Field of use. Real estate rental market, long-term rental of houses.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Огляд програмного забезпечення підтримки довгострокового винаймання будинків	11
1.2 Порівняння аналогів	15
1.3 Висновки за розділом 1	16
2 Матеріали і методи	18
2.1 Задача визначення вартості довгострокового винаймання будинків	18
2.2 Метод визначення вартості довгострокового винаймання будинків	21
2.3 Висновки за розділом 2	23
3 Проєктування програмного забезпечення	24
3.1 Вибір рішень для веброзробки	24
3.2 Моделювання програмного забезпечення на основі функціональних вимог	25
3.3 Проєктування структури бази даних	30
3.4 Висновки за розділом 3	35
4 Основні рішення щодо реалізації програмного забезпечення	36
4.1 Порядок роботи програмного забезпечення	36
4.2 Рішення з реалізації вебзастосунку	38
4.3 Висновки за розділом 4	56
5 Експлуатація, тестування та експериментальне дослідження програми	57
5.1 Опис програми	57
5.2 Експлуатація програми	57
5.3 Експериментальне дослідження	66
5.4 Висновки за розділом 5	71
Висновки	72
Перелік джерел посилення	73

Додаток А Технічне завдання.....	75
Додаток Б Текст програми.....	81
Додаток В Слайди презентації	106

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ДР – дерево рішень.

ВСТУП

Винаймання будинків є важливим і актуальним процесом, який має певні особливості в залежності від того, на який період відбувається винаймання. Короткострокове винаймання має достатньо багато відомих у світі рішень для підтримки такої роботи, а довгострокове є достатньо обмеженим, адже зазвичай представлено вебсайтами, де виставляються оголошення, які не мають відгуків, оцінок, що характерно для короткострокового винаймання, весь процес взаємодії з користувачем обмежений отриманням даних оголошення, що сприяє тому, що інформація на вебсайті нерідко є застарілою, не відповідає дійсності. Невідповідність дійсності як раз обумовлена тим, що фактично користувачі не можуть перевірити дані, а коли перевірили, то не можуть їх представити іншим користувачам. Відповідно комунікація відбувається через зовнішні засоби, а тому момент, коли винаймання оформлено, залишається поза програмою. Тож оголошення надається всім користувачам у той момент, коли воно фактично вже є неактивним. Це все є значними проблемами, які впливають на результати.

Тому потреба в програмному забезпеченні, яке могло б вплинути на цей процес взаємодії, перенести його на довгострокове винаймання та врахувати при цьому потреби винаймання саме будинків, що загалом є більш широким процесом ніж винаймання квартир через те, що будинки можуть містити багато додаткової інформації, додаткові споруди, які впливають на вибір клієнта в результаті, є достатньо великою.

При цьому процес взаємодії з користувачами є частиною характерних проблем, окрім того власники будинків зіштовхуються з тим, що потрібно визначити вартість винаймання, а клієнти вирішують, скільки таке винаймання за певних їх вимог може коштувати. Для отримання підґрунтя для такого рішення фактично користувачі зазвичай можуть тільки виконати пошук за оголошеннями і на основі власного прийняття альтернативних

пропозицій визначити вартість у оголошенні або визначити, які оголошення їм підходять, з іншого боку. Наявність програмного інструменту для підтримки таких рішень могло б спростити та допомогти у вирішенні задач, які виникають перед власниками будинків і перед клієнтами. Тому все це говорить про актуальність розроблення програмного забезпечення, що виконується у роботі, яке б дозволило підтримувати довгострокове винаймання будинків.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд програмного забезпечення підтримки довгострокового винаймання будинків

Короткострокове винаймання житла має ряд сервісів, які можна вважати лідерами у цій сфері, однак їх не доцільно використовувати для довгострокового винаймання житла, адже тоді зазвичай вартість є надто високою. Тому в огляді виділено ті програми, які надають можливість пошуку саме будинків і саме на довгий строк. В Україні одним з лідерів є Olx [1] (рис. 1.1).

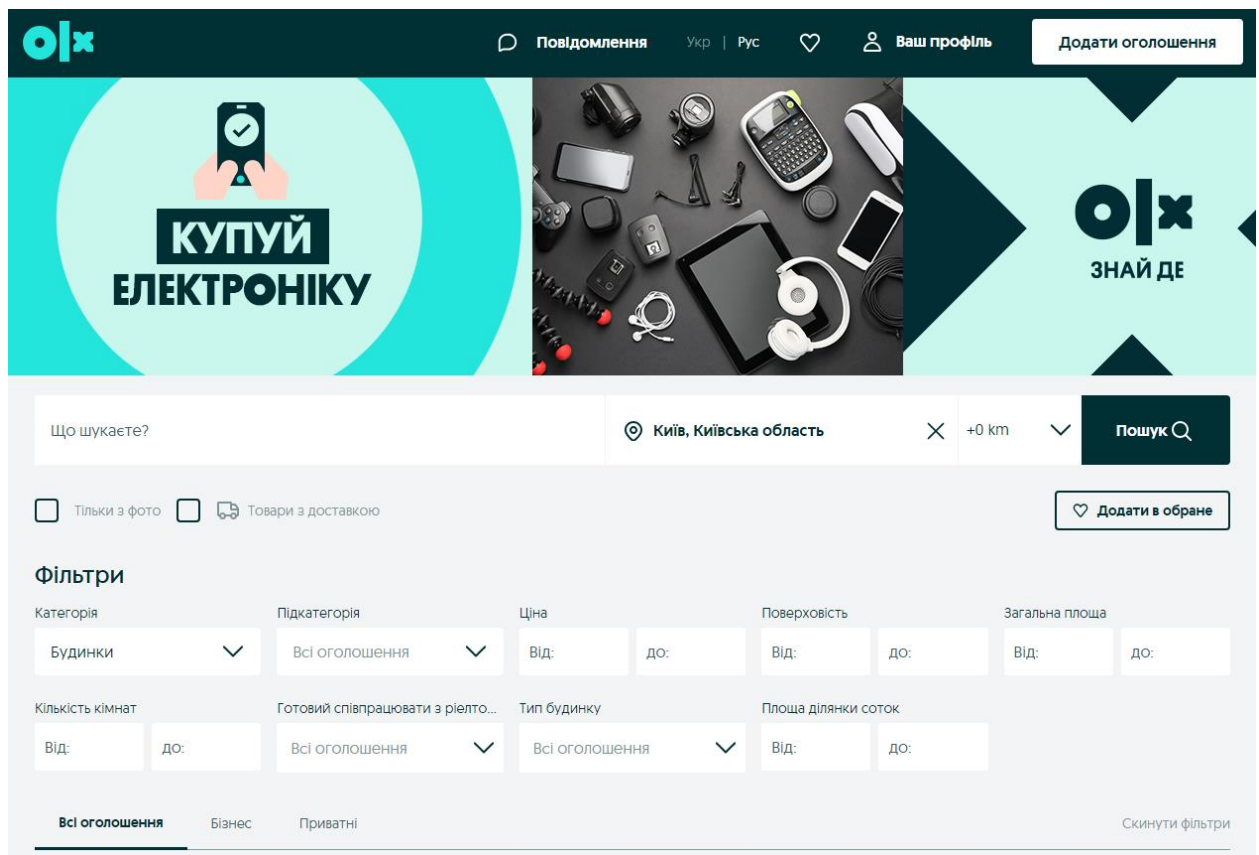


Рисунок 1.1 – Пошук будинків для довгострокового винаймання

Можливості пошуку в Olx є достатньо обмеженими, включають вартість, поверховість, загальну площу, кількість кімнат, тип будинку, площу ділянок. Будинки є однією з категорій пошуку. Оголошення, що

представляють самі будинки, є дуже обмеженими, а побудовані за принципом загального оголошення, яке стосується різних областей: включено загальний опис та ряд міток, якими представляється будинок.

HousingAnywhere [2] (рис. 1.2) дозволяє виконувати пошук будинків у багатьох країнах світу для довгострокового винаймання. Хоча програма дозволяє під час пошуку встановити будь-які дати, в тому числі один день, але розрахунок вартості винаймання виконується мінімально на один місяць. Фільтри пошуку включають дати винаймання, інтервал вартості, тип житла (для будинку визначається як апартаменти, встановити інші типи будинків не можна), рейтинг представника (оцінювання відбувається тільки для представника, не для конкретного житла), наявність меблів, загальну площу (будинку, не ділянки), тип винаймача, зручності.

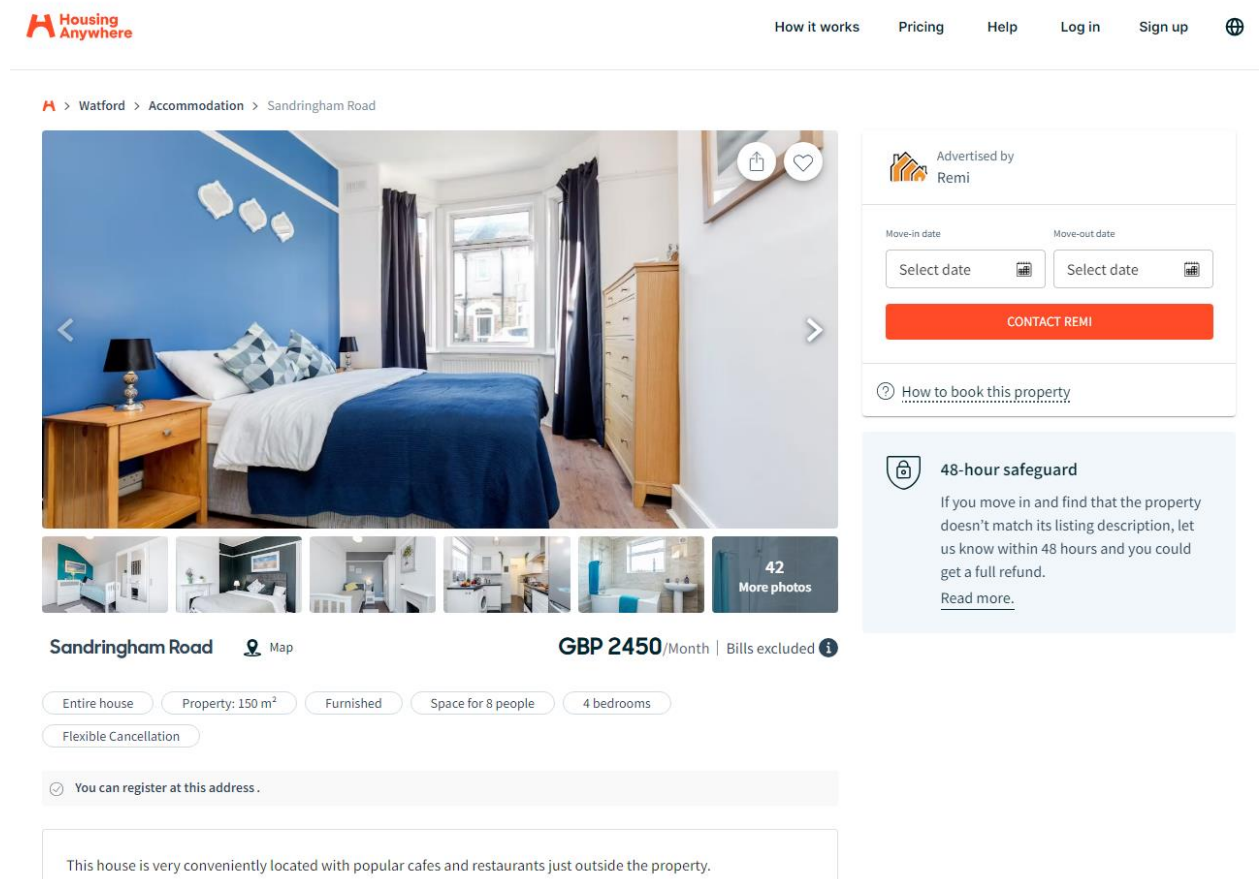


Рисунок 1.2 – Вебсторінка будинку в програмі HousingAnywhere

На відміну від Olx HousingAnywhere передбачає виділення окремих характеристик будинку, зручностей, правила винаймання (рис. 1.3). Програма дозволяє надати власні документи та сплатити за винаймання після підтвердження власником. Однак напряду забороняється відвідування перед цим будинку, при цьому допускається контактування з представником, який виставив будинок, щоб поставити йому необхідні запитання.

The screenshot displays the 'Facilities' section of a HousingAnywhere listing. It is divided into two columns: 'Facilities' and 'Amenities'. The 'Facilities' column lists 'Private parking', 'Private living room', 'Private garden', and 'Private Toilet', with a 'Show more' link. The 'Amenities' column lists 'Stone flooring', 'Central heating', 'Private kitchenware', and 'WiFi'. Below this is the 'Rules and preferences' section, which lists various rules such as 'Age: No Preference', 'Gender: No Preference', 'Tenant type: Students, Working Professionals', 'Suitable for couples: Yes', 'Playing Musical Instruments allowed: Negotiable', 'Pets allowed: No', and 'Smoking allowed: Outside Only'. At the bottom is the 'Documents required to book' section, which states that the landlord will ask for documents to accept the booking request. It lists two required documents: 'Proof of identity' (Government issued ID, passport, driver's license) and 'Proof of enrolment or occupation' (University enrolment certificate, internship or employment contract). On the right side of the screenshot, there is a section for 'Advertised by Amarpreet' with 'Move-in date' and 'Move-out date' fields, a 'CONTACT AMARPREET' button, and a '48-hour safeguard' section explaining the refund policy.

Facilities

Facilities

- Private parking
- Private living room
- Private garden
- Private Toilet
- [Show more](#)

Amenities

- Stone flooring
- Central heating
- Private kitchenware
- WiFi

Rules and preferences

- Age: **No Preference**
- Gender: **No Preference**
- Tenant type: **Students, Working Professionals**
- Suitable for couples: **Yes**
- Playing Musical Instruments allowed: **Negotiable**
- Pets allowed: **No**
- Smoking allowed: **Outside Only**

Documents required to book

The landlord will ask for them to accept your booking request.

- Proof of identity**
Government issued ID, passport, driver's license.
- Proof of enrolment or occupation**
University enrolment certificate, internship or employment contract.

Advertised by Amarpreet

Move-in date: Select date

Move-out date: Select date

[CONTACT AMARPREET](#)

48-hour safeguard

If you move in and find that the property doesn't match its listing description, let us know within 48 hours and you could get a full refund.

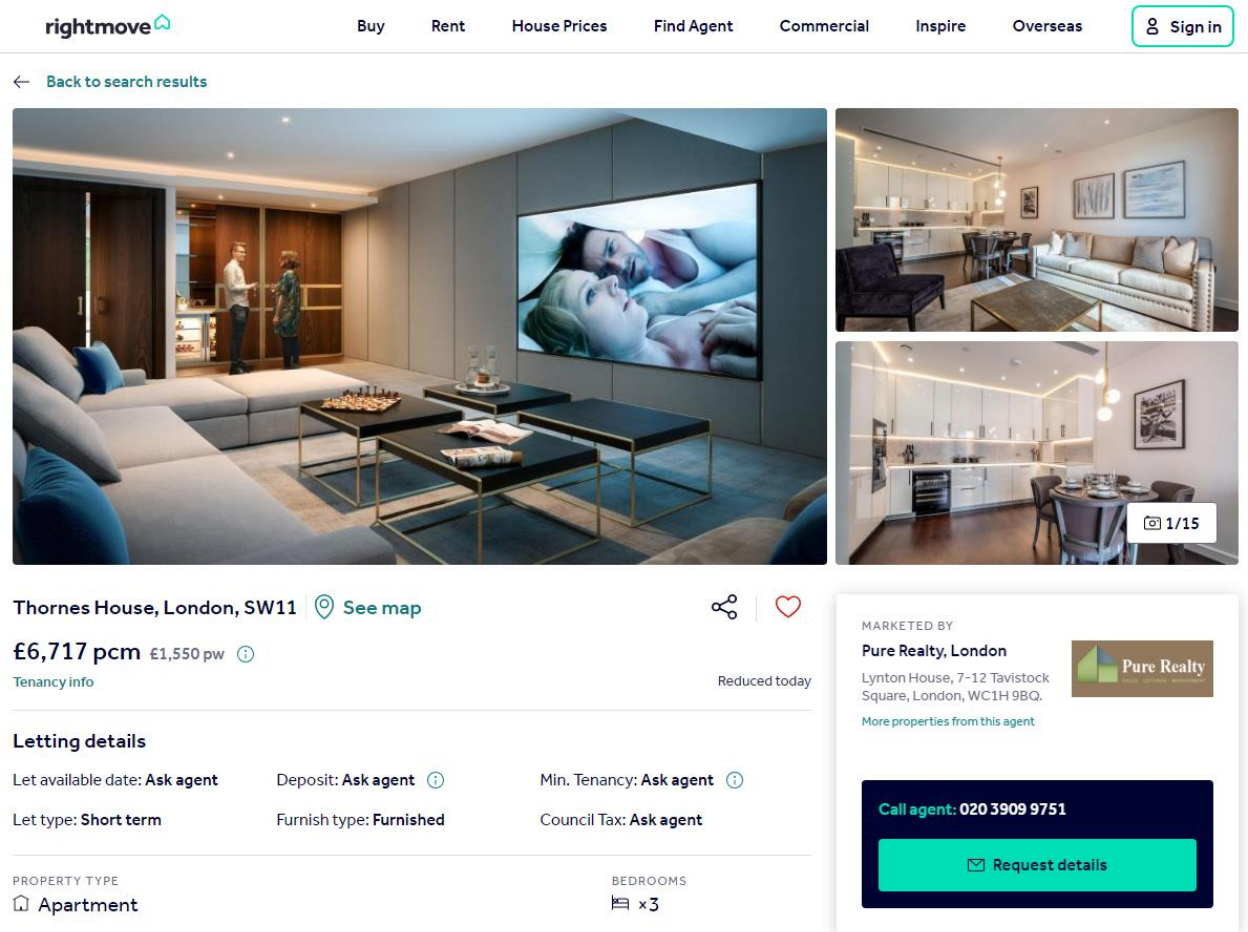
[Read more.](#)

Рисунок 1.3 – Додаткові характеристики будівлі на вебсторінці програми HousingAnywhere

Rightmove [3] (рис. 1.4) дозволяє шукати житло в різних містах Великобританії, проте в самому пошуку не можна вказати тип будинку, хоча тип визначений і на сторінці будинку. Тобто виконати пошук за типом будинку не можна, але отримати таку інформацію можна. У програмі можна подати заявку на консультування або відвідування будинку, при цьому заявка

полягає як стандартна форма, тобто фактично це форма для того, щоб створити контакт.

За будинком подаються значення за великою кількістю характеристик, які описують будинок, якщо звичайно такі характеристики присутні в конкретному будинку.



rightmove Buy Rent House Prices Find Agent Commercial Inspire Overseas Sign in

← Back to search results

Thornes House, London, SW11 See map

£6,717 pcm £1,550 pw

Tenancy info Reduced today

Letting details

Let available date: Ask agent	Deposit: Ask agent	Min. Tenancy: Ask agent
Let type: Short term	Furnish type: Furnished	Council Tax: Ask agent

PROPERTY TYPE: Apartment BEDROOMS: x3

MARKETED BY: Pure Realty, London
Lynton House, 7-12 Tavistock Square, London, WC1H 9BQ.
More properties from this agent

Call agent: 020 3909 9751

Request details

Рисунок 1.4 – Вебсторінка будинку в програмі Rightmove

Trulia [4] (рис. 1.5) надає можливості довгострокового винаймання будинків у США. Через програму можна подати заявку з відповідним текстом на відвідування будинку, проте час через програму не вказується. Надається можливість створити типову заявку, яку потім швидко можна повторити для інших оголошень. Також можна подати заявку на винаймання житла, завантажити ряд документів.

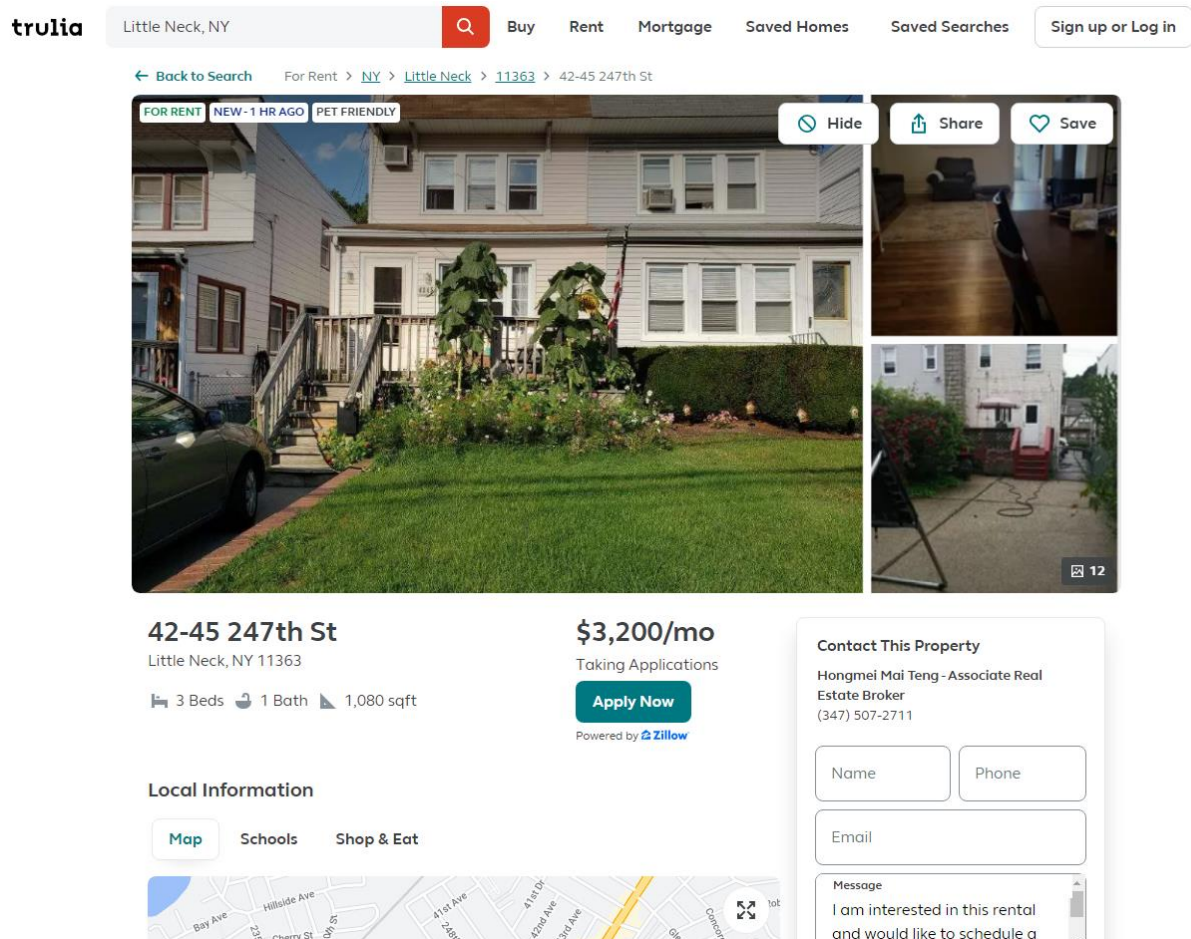


Рисунок 1.5 – Вебсторінка будинку в програмі Trulia

1.2 Порівняння аналогів

До порівняння аналогів було внесено HousingAnywhere, Rightmove та Olx (табл. 1.1). Olx має розширені можливості фільтрування як і розробка з цієї роботи, що відрізняє їх від інших аналогів, проте якщо Olx не має спеціалізованих засобів для роботи над створенням запитів на відвідування будинку, консультування, створення відгуків, то виконувана розробка має. У деяких аналогах частина цих засобів присутня, однак не всі одразу. До того ж розробка повинна містити можливості визначення тих споруд, які є на території будинку, щоб деталізувати, що належить до самого будинку, а що є додатковим, та засоби визначення вартості довгострокового винаймання будинків.

Таблиця 1.1 – Порівняння програмного забезпечення підтримки довгострокового винаймання будинків

Позиція порівняння	Housing-Anywhere	Rightmove	Olx	Розробка
Пошук будинків на довгий строк	+	+	+	+
Фільтрування за типом будинку	+	—	+	+
Фільтрування за площею ділянки	—	—	+	+
Запит на відвідування будинку	—	+	—	+
Опис споруд на території будинку	—	—	—	+
Відгуки та оцінювання будинків	Відгуки про користувачів	—	—	Відгуки про будинки та користувачів
Консультавання за будинком	+	+	—	+
Визначення вартості довгострокового винаймання будинків	—	—	—	+

1.3 Висновки за розділом 1

Аналоги програми підтримки довгострокового винаймання будинків, що створюється в роботі, не мають можливостей опису споруд на території

будинку та визначення вартості довгострокового винаймання будинків, що і було віднесено до функціональних особливостей програми.

Завданнями кваліфікаційної роботи є:

- проаналізувати аналоги програм підтримки довгострокового винаймання будинків;
- розробити метод визначення вартості довгострокового винаймання будинків;
- виконати проєктування програмного забезпечення підтримки довгострокового винаймання будинків;
- реалізувати програмне забезпечення підтримки довгострокового винаймання будинків.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача визначення вартості довгострокового винаймання будинків

Визначення вартості довгострокового винаймання будинків є достатньо важливим, але і одночасно складним практичним завданням. Воно виникає перед потенційними винаймачами та перед власниками будинків:

– винаймачі щонайменше на першому етапі, коли фактично вивчають ринок, потребують розуміння, скільки загалом може коштувати необхідне їм житло: необхідне житло визначається рядом параметрів, які вони хочуть мати під час винаймання (наприклад, площа приміщення, кількість ванних кімнат, спальних кімнат тощо), а винаймач повинен вивчити ринок, знайти подібні оголошення, узагальнити вартість з цих оголошень, зрозуміти, які додаткові (наприклад, тимчасові) фактори впливають на цю вартість, зрозуміти, що можливо на ринку тимчасово відсутні певні пропозиції, а на поточні результати впливає така відсутність, відповідно сформована вартість викривлена, а якщо людина має можливість очікування, то наприклад, через тиждень ситуація може змінитись;

– власники будинків мають встановити певну вартість винаймання, відповідно потребують певний орієнтир також: власники будинків найчастіше проводять додаткове вивчення ринку, однак встановлення певного орієнтира також є важливим.

З іншого боку варто зазначити, що для довгострокового винаймання будинків вплив різних тимчасових факторів значно зменшується. Якщо при короткостроковому винайманні свята, будь-які невеликі події можуть значно впливати на вартість, то за довгострокового винаймання такий вплив щонайменше значно зменшується, а можливо і нівелюється. Тож отримання певного орієнтира щодо вартості винаймання є дуже важливим. Таким орієнтиром є вартість винаймання будинку на місяць.

Розв'язання задачі визначення вартості довгострокового винаймання будинків пов'язано з отриманням значення середньомісячної вартості винаймання будинку протягом найближчого періоду часу за значеннями параметрів винаймання або параметрів будинку, який буде винайматися. Така задача в машинному навчанні розглядається як задача регресії або задача прогнозування. Для розв'язання цієї задачі необхідні дані про вартість будинків з певними встановленими параметрами, тобто накопичений історичний досвід, який не враховує або з меншою мірою враховує коливання, тимчасові фактори, які впливали на вартість протягом певного періоду.

Оскільки в даній задачі заплановано створювати рекомендацію щодо визначення вартості довгострокового винаймання будинків, не спираючись на конкретну дату прогнозування, адже як було описано вище, цей вплив значно менший за довгострокового винаймання ніж за короткострокового, то рекомендація має виконуватись на основі певного зрізу ринку. Зріз ринку має виконуватись на деяку дату, коли відомо, що або вплив тимчасових факторів є незначним, а очікується, що дана тенденція буде зберігатися і в той період, на який відбувається прогнозування, або вплив буде, але і в період прогнозування передбачається також збереження такого впливу. Відповідно у випадках, коли вплив додаткових факторів значно збільшується, варто використовувати дані щодо стану ринку у відповідний момент.

Окрім того слід вказати також на те, що при створенні зрізу ринку деякі спостереження можуть не відповідати загальним тенденціям. Фактично заради уникнення впливу таких спостережень на прийняття рішень і виникає описана задача. Тобто якщо певний власник вирішив значно підняти вартість винаймання будинку порівняно з наявними цінами на ринку, то потенційний винаймач повинен зрозуміти, що така вартість є неприйнятною, і якщо у нього є вибір у поточний момент або можливість почекати зміни стану, то скористатись альтернативними варіантами. З іншого боку вартість може бути і нижчою за очікувану. Це може надавати винаймачу розуміння того, що

такою пропозицією варто скористатися. Проте в залежності від умов винаймання це може також вказувати на те, що вартість винаймання в наступні місяці може бути зміненою. Загалом це говорить про те, що вартість винаймання, що значно вибивається з очікуваних тенденцій на ринку, є нехарактерною, і такі спостереження не мають використовуватися при створенні моделі, яка таку задачу зможе розв'язати.

Оскільки задача визначення вартості довгострокового винаймання будинків є задачею регресії, як було описано вище, то існує ряд методів і моделей, які можуть використовуватися для розв'язання такої задачі, включаючи зокрема наступні.

Базовою моделлю, яка використовується для розв'язання задачі регресії, є лінійна регресія.

Метод опорних векторів – це контрольований (з учителем) алгоритм машинного навчання, який може використовуватися як для розв'язання завдань класифікації (класичне застосування даного методу) та регресії, а основна логіка його полягає в тому, щоб знайти гіперплощину у високовимірному просторі, що якнайкращим чином розділяє дані на потрібні класи, спрямований на максимізацію відстані між гіперплощиною та найближчими точками даних кожного класу, мінімізуючи при цьому помилки класифікації, а для задачі регресії відбувається пошук гіперплощини, яка найкраще відповідає екземплярам даних під час навчання, що досягається шляхом відображення вхідних змінних у просторі ознак великої розмірності та пошуку гіперплощини, яка максимізує відстань між гіперплощиною та найближчими точками даних, а також мінімізує помилку прогнозування [5].

Дерева рішень (ДР) – це модель машинного навчання у вигляді блок-схеми, яка починається з однієї головної ідеї, а потім розгалужується відповідно до наслідків рішень (названа ДР, оскільки модель зазвичай виглядає як дерево з гілками) [6].

Метод Random Forest поєднує набір ДР для створення ансамблю, коли кожне дерево в такому ансамблі будується з випадковими підмножинами даних, процес випадкового вибору допомагає зменшити дисперсію та перенавчання всередині моделі, роблячи її більш надійною та стійкою до шуму в даних, відповідно під час прогнозування усереднюються прогнози кожного ДР, щоб обчислити результат для даного екземпляра [7].

2.2 Метод визначення вартості довгострокового винаймання будинків

Для реалізації методу визначення вартості довгострокового винаймання будинків потрібно створити вибірку даних, яка характеризує вартість винаймання будинків на певний момент з відповідним рівнем впливу тимчасових факторів, як це було описано в попередньому підрозділі, для певного міста.

Для створення моделі мають використовуватися наступні характеристики, які таким чином характеризуються відповідні екземпляри вибірки:

- вартість винаймання будинку на місяць;
- загальні додаткові витрати – це витрати, які стандартно несе винаймач та на які він фактично не впливає: наприклад, це може бути вартість прибирання території будинку, сплата за послуги з охорони (наприклад, територія, де розташований будинок, фактично район з декількох будинків, може бути закритим, а доступ на територію відбувається або з використанням певної системи доступу, або людиною, яка представляє охоронну компанію, відповідно вартість їх послуг сплачується окремо);
- кількість ванних кімнат;
- кількість спальних кімнат;
- кількість гаражів;
- наявність басейну;

- загальна площа власності – площа, яка враховує повну площу будинку (внутрішній простір) та площу земельної ділянки: у якості даного фактору може бути встановлена сама площа приміщення окремо;
- площа земельної ділянки – площа землі, на якій розташовано будинок та яка доступна винаймачу;
- тип системи опалення: якщо дані про систему опалення недоступні, то може використовуватися категоріальне значення, яке визначає, чи наявне опалення у будинку, чи ні.

Щодо останнього фактору варто зазначити, що його вплив залежить від місцевості, для якої відбувається прогнозування. У деяких країнах наявність системи опалення може бути некритичним фактором, адже через теплу погоду будинку опалення не потребують. В інших це є обов'язковою вимогою, а відповідно вже те, яка це система опалення, впливає на фактичну вартість винаймання. Тому даний фактор має бути врахований у відповідності з об'єктом прогнозування та його характеристиками.

У факторах не враховується окремо кількість кімнат, це пов'язано з тим, що фактично це значення корелює з площею приміщення. При цьому вже кількість ванних кімнат, кількість спальних кімнат визначають додаткові характеристики, які безпосередньо з самою площею не корелюють значно.

За кожним будинком має бути визначено значення кожної з описаних вище характеристик. Тоді потрібно визначити процентилі, які відповідають 25 % та 75 % значень вартості винаймання. Далі потрібно обчислити різницю між цими процентилями, а тоді відняти від процентилі в 25 % цю обчислену різницю, помножену на 1,5, а відповідно до процентилі в 75 % додати цю різницю, помножену на 1,5. Це дозволяє виключити екземпляри, які значно відхиляються від загальних тенденцій. Такі екземпляри є нехарактерними і мають бути виключені з навчання, адже на основі них складно і можливо некоректно робити узагальнення. Тобто значення, які випадають за межі створеного інтервалу, мають бути виключені з вибірки. В залежності від значення лівої межі (зазвичай вона є від'ємною) має бути додатково

перевірено наявність будинків з нульовою вартістю. Такі екземпляри також мають бути виключені. Окрім того мають бути виключені всі дублюючі екземпляри, до яких відносяться екземпляри зі всіма однаковими значеннями характеристик та екземпляри, які представляють однакові будинки (однакові адреси – такі екземпляри не мають початково включатися у вибірку).

Після нормування даних вибірки вона використовується для навчання моделі, яка має організовуватись на основі виділення вартості винаймання будинку на місяць як вихідної ознаки, а всіх інших – як вхідних. Модель створюється на основі ДР за допомогою методу Random Forest з визначенням кількості дерев у ансамблі. Після створення моделі та її навчання кожен новий екземпляр має формуватися на основі нормованих значень вхідних ознак.

2.3 Висновки за розділом 2

Описано задачу визначення вартості довгострокового винаймання будинків, розглянуто потенційні методи і моделі її розв'язання та визначено метод визначення вартості довгострокового винаймання будинків.

Наукова новизна роботи полягає в методі визначення вартості довгострокового винаймання будинків, який ґрунтується на створенні моделі прогнозування на основі методу Random Forest, даних відібраних інших будинків у заданому місті за визначеними ознаками, що дозволяє визначити очікувану місячну вартість винаймання будинку.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір рішень для веброзробки

Зваживши популярність мови PHP у веброзробці, врахувавши наростаючу популярність Python у цій сфері, а також стабільну популярність Python для роботи з даними, машинного навчання, одні з основних рішень для веброзробки на основі цих мов [8]-[10] були визначені для виконання вибору (табл. 3.1). Django [8] виявився кращим за здатністю до налаштувань, адже окрім налаштування інтерфейсів та можливості підключення різних готових рішень, має велику кількість налаштувань сервера та дозволяє визначати моделі предметної області, функціональність, також надає кращий рівень безпеки та характеризується через Python кращими бібліотеками роботи з даними та машинного навчання.

Таблиця 3.1 – Вибір рішення для веброзробки

Позиція порівняння	Drupal	Django
Призначення	Типові рішення	Рішення з особливою функціональністю
Здатність до налаштувань	Середній рівень	Високий рівень
Система керування вмістом	Так	Ні, але має такі системи за необхідності
Фреймворк для розробки повного стеку	Ні	Так
Безпека	Середній рівень	Високий рівень
Відкритий доступ	Так	Так
Мова програмування	PHP	Python
Бібліотеки роботи з даними та машинного	Середній рівень	Високий рівень

навчання		
----------	--	--

3.2 Моделювання програмного забезпечення на основі функціональних вимог

При моделюванні вимог спочатку було вирішено питання участі користувачів у роботі з програмою. Було проаналізовано дві концепції:

- перша концепція передбачала участь користувача в роботі програми як одночасно і можливого власника будинку, і винаймача;
- друга концепція передбачала розділення цих ролей та необхідність створення окремих облікових записів для цього.

Проте як відомо зараз, велика кількість облікових записів є проблемою для користувачів, яка частково вирішується за рахунок використання універсальних способів активації за рахунок зокрема облікових записів Google. Однак у такому випадку користувач має для того, щоб виставити власний будинок для можливості винаймання на вебсайті, організувати додатковий обліковий запис, а відповідно і додатковий запис Google. Тобто це потребує додаткових кроків, які можуть бути додатковою проблемою, а користувач замість того, щоб вже спробувати таку функціональність у програмі, маючи базовий запис, вирішить відмовитися від такої ідеї. Тому в результаті ці ролі об'єднано.

Додатково в програмі було визначено окрему роль, яку названо менеджером, який фактично відповідальний за слідкування за роботою всього сервісу загалом та за вирішення конфліктів, які можуть виникати між користувачами, які є винаймачами або відвідувачами (під час розгляду варіантів винаймання будинків), та користувачами, які є власниками будинків.

Користувач є основним джерелом активностей у програмі, оскільки він виставляє для винаймання власні будинки, шукає, винаймає будинки інших користувачів (рис. 3.1).

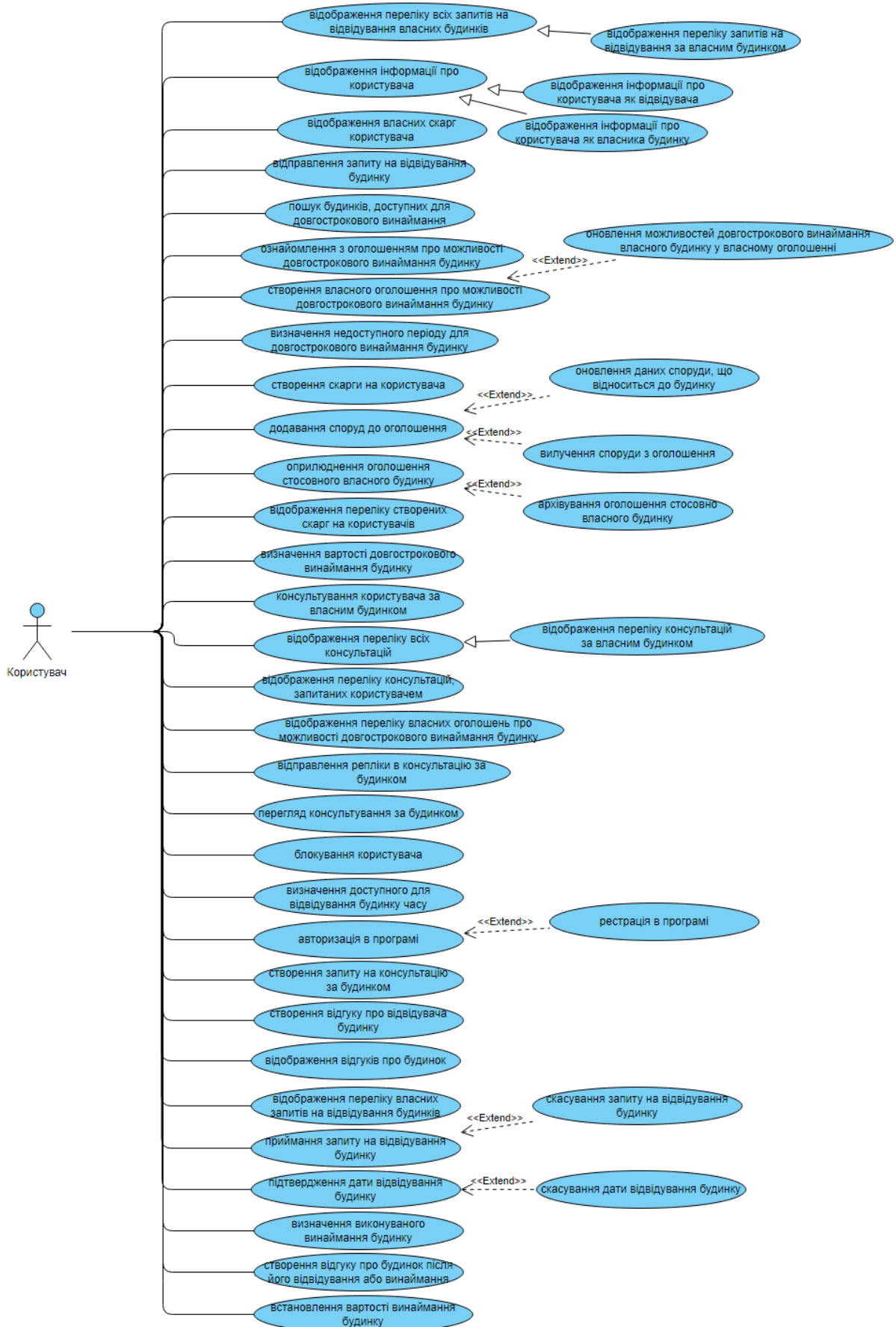


Рисунок 3.1 – Діаграма варіантів використання користувачем програми підтримки довгострокового винаймання будинків

Винаймач або відвідувач та власник будинку об'єднані в роль користувача, для якої надаються сценарії:

- пошук будинків, доступних для довгострокового винаймання;
- ознайомлення з оголошенням про можливості довгострокового винаймання будинку;
- створення власного оголошення про можливості довгострокового винаймання будинку, що має альтернативним сценарієм оновлення можливостей довгострокового винаймання власного будинку у власному оголошенні;
- додавання споруд до оголошення про можливості довгострокового винаймання будинку, що має два альтернативні сценарії, якими є оновлення даних споруди, що відноситься до будинку та вилучення споруди з оголошення;
- оприлюднення оголошення стосовного власного будинку, що має альтернативним сценарієм архівування оголошення стосовно власного будинку;
- визначення недоступного періоду для довгострокового винаймання будинку;
- визначення вартості довгострокового винаймання будинку;
- консультування користувача за власним будинком;
- відображення переліку консультацій за власним будинком;
- відображення переліку всіх консультацій;
- відображення переліку консультацій, запитаних користувачем;
- відображення переліку власних оголошень про можливості довгострокового винаймання будинку;
- створення запиту на консультацію за будинком;
- відправлення репліки в консультацію за будинком;
- перегляд консультування за будинком;
- блокування користувача;
- визначення доступного для відвідування будинку часу;

- авторизація в програмі, що має альтернативним сценарієм реєстрацію в програмі;
 - відправлення запиту на відвідування будинку;
 - відображення переліку запитів на відвідування за власним будинком;
 - відображення переліку всіх запитів на відвідування власних будинків;
 - відображення переліку власних запитів на відвідування будинків;
 - створення відгуку про будинок після його відвідування або винаймання;
 - відображення відгуків про будинок, створених після його відвідування або винаймання;
 - створення відгуку про відвідувача будинку;
 - відображення інформації про користувача як відвідувача;
 - відображення інформації про користувача як власника будинку;
 - приймання запиту на відвідування будинку, що має альтернативним сценарієм скасування запиту на відвідування будинку;
 - підтвердження дати відвідування будинку, що має альтернативним сценарієм скасування дати відвідування будинку;
 - визначення виконуваного винаймання будинку;
 - створення скарги на користувача, зокрема на власників будинків або на відвідувачів;
 - перегляд створеної скарги;
 - відображення скарг на користувача;
 - відображення власних скарг користувача;
 - встановлення вартості винаймання будинку.
- Менеджеру мають бути доступні сценарії (рис. 3.2):
- відображення переліку створених скарг на власників будинків;
 - відображення переліку створених скарг на відвідувачів;
 - перегляд створеної скарги;

- надання відповіді на скаргу;
- скасування скарги, що є фактично результатом надання відповіді на скаргу і фіксує завершення вирішення процесу, який відбувається поза межами програми і повинен призвести до залагодження конфлікту, однак якщо конфлікт не вдалося вирішити, користувач порушує правила роботи, то альтернативним сценарієм є підтвердження скарги, за якого повністю блокується доступ такого користувача до всіх засобів;
- пошук користувачів;
- перегляд запитів на відвідування будинку за будинком або за відвідувачем менеджером;
- перегляд консультацій менеджером;
- авторизація в програмі.

Реєстрація менеджера не доступна, вона виконується централізовано через адміністративну панель адміністратором.

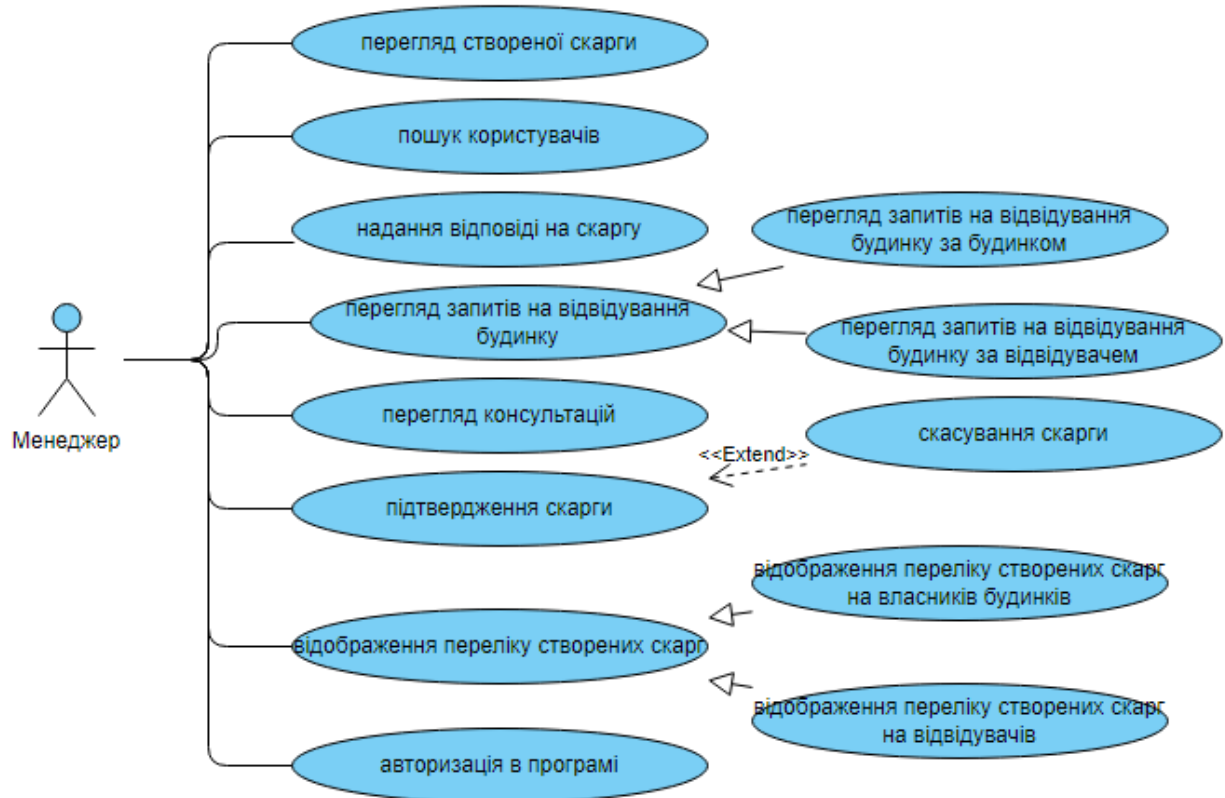


Рисунок 3.2 – Діаграма варіантів використання менеджером програми підтримки довгострокового винаймання будинків

3.3 Проектування структури бази даних

Робота з даними в програмі підтримки довгострокового винаймання будинків охоплює велику кількість сутностей, а з часом при активному використанні може з'явитися і велика кількість даних, тому потрібно було створити базу даних (БД). Для роботи з БД застосовано систему керування БД PostgreSQL.

Фактично головною сутністю цієї предметної області є безпосередньо сам будинок, який виставляється для довгострокового винаймання в програмі через створення відповідного оголошення. Ці дані відповідають сутності Cottage та відповідній таблиці БД. Будинком є узагальнена сутність, а фактично винаймаються будинки конкретних типів (котедж, особняк, дачний будинок тощо). Тому типи будинків, які виставляються для винаймання, представлені окремою сутністю CottageType. Будинок може бути лише самим одним будинком, а може включати різні споруди (гараж, сарай, окремий спортивний зал тощо), які розташовані на території, відведених під цей будинок, та винаймаються разом з цим будинком одночасно. Усі такі споруди відповідають сутності CottageStructure. При цьому як видно з приведених прикладів споруд, вони також можуть мати свої власні окремі типи, тому їх також виділено у окрему сутність StructureType.

Будинки можуть знаходитись як і квартири у містах, а можуть розташовуватись і поруч з містом, в селах, у віддалених районах. Це все варіанти розміщення, тому для пошуку потрібно було визначити те, яка фактично адміністративна система використовується. На верхньому рівні у підсумку визначена країна, якій відповідає сутність Country. Далі знаходиться область (або інший тип регіону, який присутній в цій країні), яку внесено в сутність Region. Наступний рівень може бути різним. Це місто, село або інший тип населеного пункту, який є в цій країні. Ці дані відповідають сутності Geopoint. Найнижчим рівнем є район, який може бути

районом міста, складовою населеного пункту, в якому як раз і знаходиться будівля. Райони відповідають сутності District.

Оголошення про можливості довгострокового винаймання будинку представляється не тільки самим будинком, але і додатковими даними, які визначають, як саме це винаймання має відбуватися. Такими складовими є вартість винаймання та час, коли винаймання будинку недоступне. Вартість винаймання будинку представлена сутністю Monthprice. У цій таблиці з одного боку зберігаються дані про те, якою була вартість в минулому, а з іншого боку – варіанти вартості винаймання з деякої дати (вартість може бути різною в залежності від періоду, тобто якщо винаймання відбувається на рік, а не на місяць, то вартість винаймання будинку на місяць може бути нижчою).

Час, коли винаймання будинку недоступне, визначається сутністю Cottageinaccessible. Фактично якщо оголошення неоприлюднене, то будинок для винаймання недоступний, ці дані не використовуються. Якщо оголошення оприлюднене і будинок потенційно може бути винайнятий, то дані з цієї таблиці використовуються для визначення того, чи можна в конкретні дати винайняти цей будинок. Якщо дати не зазначені в таблиці, то вважається, що будинок в такі дати можна винаймати.

Титульна фотографія будинку знаходиться в таблиці Cottage, проте таких фотографій може бути декілька. Усі інші фотографії будинку знаходяться в таблиці Cottagephoto.

Будинки можуть мати різний стиль оформлення внутрішнього приміщення. Самі стилі відповідають сутності Design.

Відвідування будинків є однією з основних точок взаємодії між власниками будинків та користувачами, які шукають собі будинок для довгострокового винаймання. У програмі це визначається запитом на відвідування будинку, що знаходиться в таблиці БД Visit. Відвідування необхідне для того, щоб зрозуміти, чи точно людина хоче жити в цьому

будинку, і відповідно буде його винаймати. Відвідування може бути безкоштовним, а може бути за визначену плату.

Відвідування відбувається не в довільний час, а у час, встановлений через взаємодію між власником будинку та користувачем програми, який цікавиться цим будинком. Однак користувач обирає не довільний час як пропозицію, а з варіантів, встановлених власником будинку. Ці варіанти доступного часу для відвідування відповідають сутності `Visitperiod`.

Запити на відвідування є одним з основних джерел для подальшого встановлення зворотного зв'язку відвідувачем відносно будинку, а власником будинку – відносно користувача як відвідувача або винаймача. Однак для підготовки відвідування, продовження спілкування в подальшому можуть також використовуватися консультавання за будинком, коли власник може дати відповіді на додаткові запитання за будинком, якщо готовий надати таку інформацію, яку запитує інший користувач. Таке спілкування реалізується через створення окремих консультавань, що відповідають сутності `Conference`. Саме спілкування, тобто створення запитань, відповідей реалізується через внесення реплік до консультавання, які відповідають сутності `Conferencemessage`.

Усі користувачі програми, включаючи, як було описано під час моделювання, користувачів і менеджерів, відповідають сутності `User`, яка побудована над сутністю `User` з підпаketу аутентифікації `Django django.contrib.auth`. Користувачі можуть блокувати один одного з тих чи інших причин. Тоді користувач, який входить до такого списку іншого користувача, не зможе відправити йому запит на консультавання або відвідування. Такі дані знаходяться в таблиці `Blockeduser`. Блокування також може виконуватися і менеджером як заборона загального доступу. Тоді користувач не зможе створювати нові оголошення, а існуючі стають одразу заархівованими. Таке блокування є результатом оброблення менеджером скарги.

Безпосередньо скарги можуть створюватися на власників будинків або на відвідувачів протилежною стороною. Вони відповідають сутності Complaint. Скарги є основною сутністю, з якою працюють менеджери.

Оформлення винаймання будинків реалізується не через програму, адже довгострокове винаймання зазвичай потребує укладання угоди, окремих юридичних моментів, а реалізується нечасто. Проте дані про те, хто який будинок винаймав, можуть бути внесені в програму. Винаймання будинку є сутністю Cottagerental. Ці дані є фактично результатом відвідування, тому для архівації оголошення логічним виглядає в такому випадку підтвердження успішного завершення відвідування і швидке внесення дати початку винаймання. Після повторного оприлюднення оголошення власник будинку буде запитаний про надання даних про попереднє винаймання, тож це сприяє внесенню даних про самі винаймання в програму, однак безпосередньо винаймання через програму не відбувається. Окрім того надати інформацію про успішне завершення відвідування будинку та виконання винаймання може і сам винаймач. Тоді власнику будинку пропонується підтвердити ці дані. Внесення даних про винаймання будинку важливе для власника також тим, що це дозволяє винаймачу створити відгук про житло. Оскільки відгук можна створити тільки протягом місяця від події (відвідування або завершення винаймання), то власник додатково зацікавлений в тому, щоб все-таки надати дані про винаймання будинків.

Тож кінцевим результатом взаємодії власників з користувачами щодо винаймання будинків є відгуки в програмі. Відгуки додатково стимулюють власників надавати дані про винаймання. Адже відгуки дозволяють змінити оцінку будинку, привернути до нього увагу додатково, підвищують довіру до інформації, приведеної в оголошенні про можливість довгострокового винаймання будинку.

Відгуки про будинок після його відвідування або винаймання створюються звичайними користувачами (відвідувачами або винаймачами)

та відповідають сутності БД Cottagereview, а відгуки про відвідувача будинку відповідають сутності Visitorreview і створюються власниками будинків стосовно тих користувачів, які винаймали у них будинок або відвідували його (рис. 3.3).

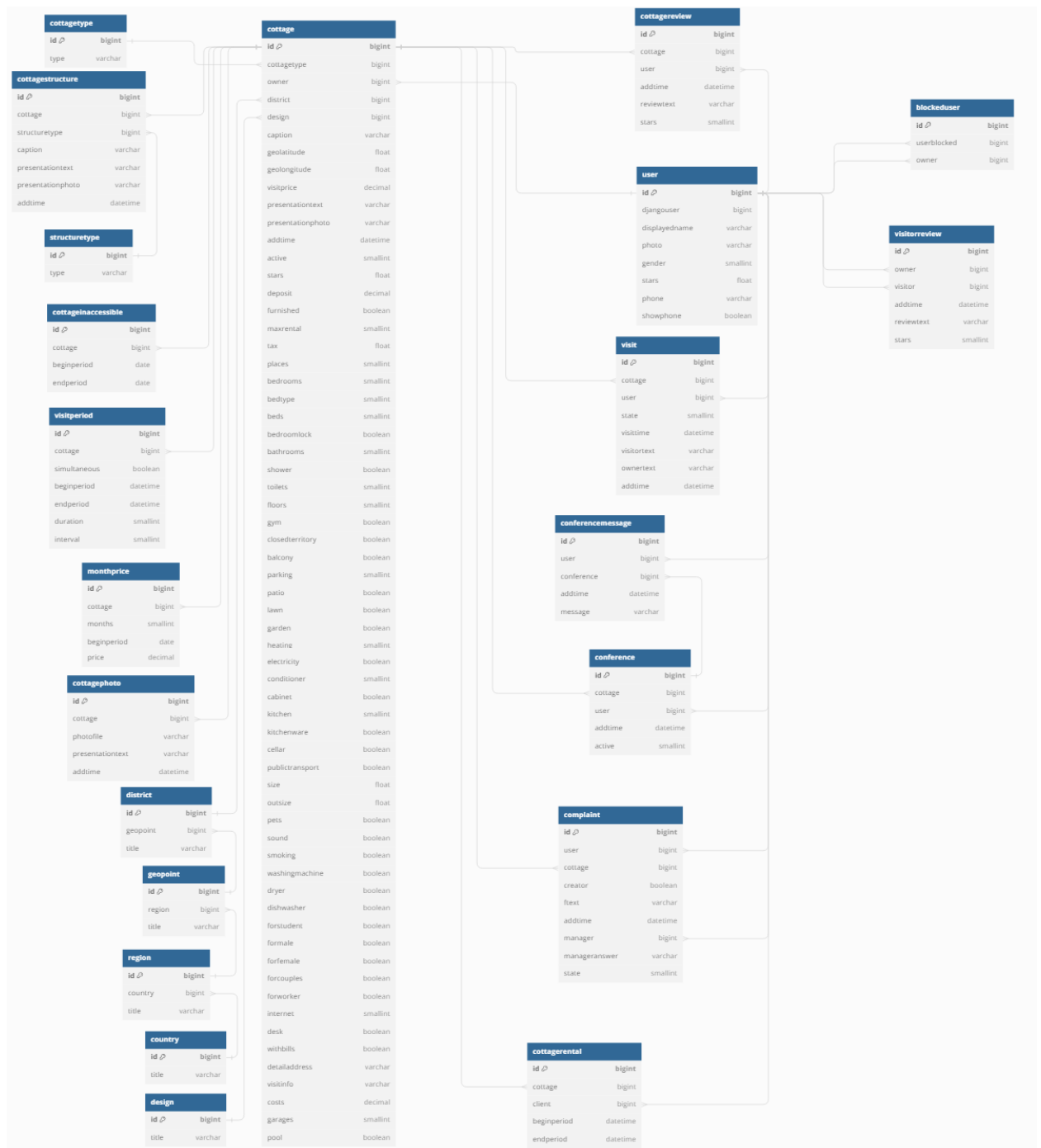


Рисунок 3.3 – Схема реляційної БД

3.4 Висновки за розділом 3

Аналіз наявних засобів веброзробки, який в підсумку було зведено до Django і Drupal, було завершено вибором для використання фреймворку Django, а відповідно і мови програмування Python, що в подальшому було розширено системою PostgreSQL. Сценарії використання програми було об'єднано окремо для звичайного користувача, і окремо для менеджера, який займається керуванням проблемами, які виникають під час роботи звичайних користувачів. Виділені сценарії стали основою визначення структури створеної реляційної БД.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Порядок роботи програмного забезпечення

Оголошення про можливість довгострокового винаймання будинку (рис. 4.1) після збереження користувачем переходить до стану створеного. Далі воно може бути оприлюднено тоді, коли будинок стає доступним для винаймання. Коли хтось винайняв будинок, то оголошення переходить до стану заархівованого. Також це може зробити сам власник будинку. Однак потім власник може заново його оприлюднити, коли будинок стане повторно доступним.

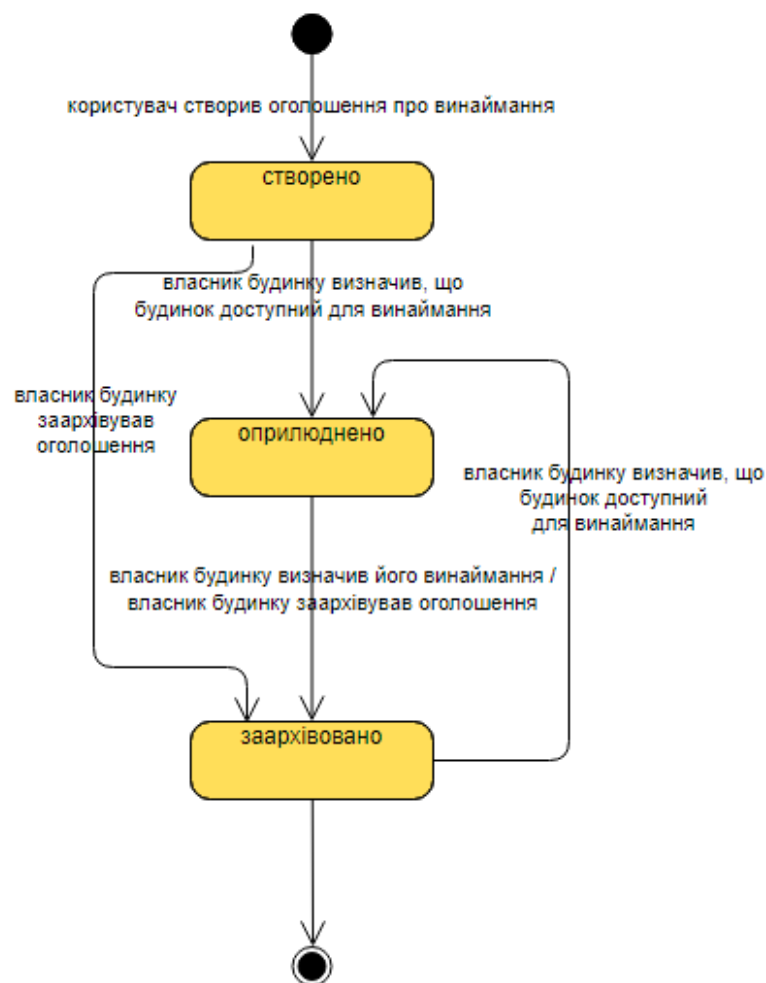


Рисунок 4.1 – Діаграма станів оголошення про можливість довгострокового винаймання будинку

Відвідування будинку (рис. 4.2) є окремим об'єктом, який може знаходитися в різних станах.

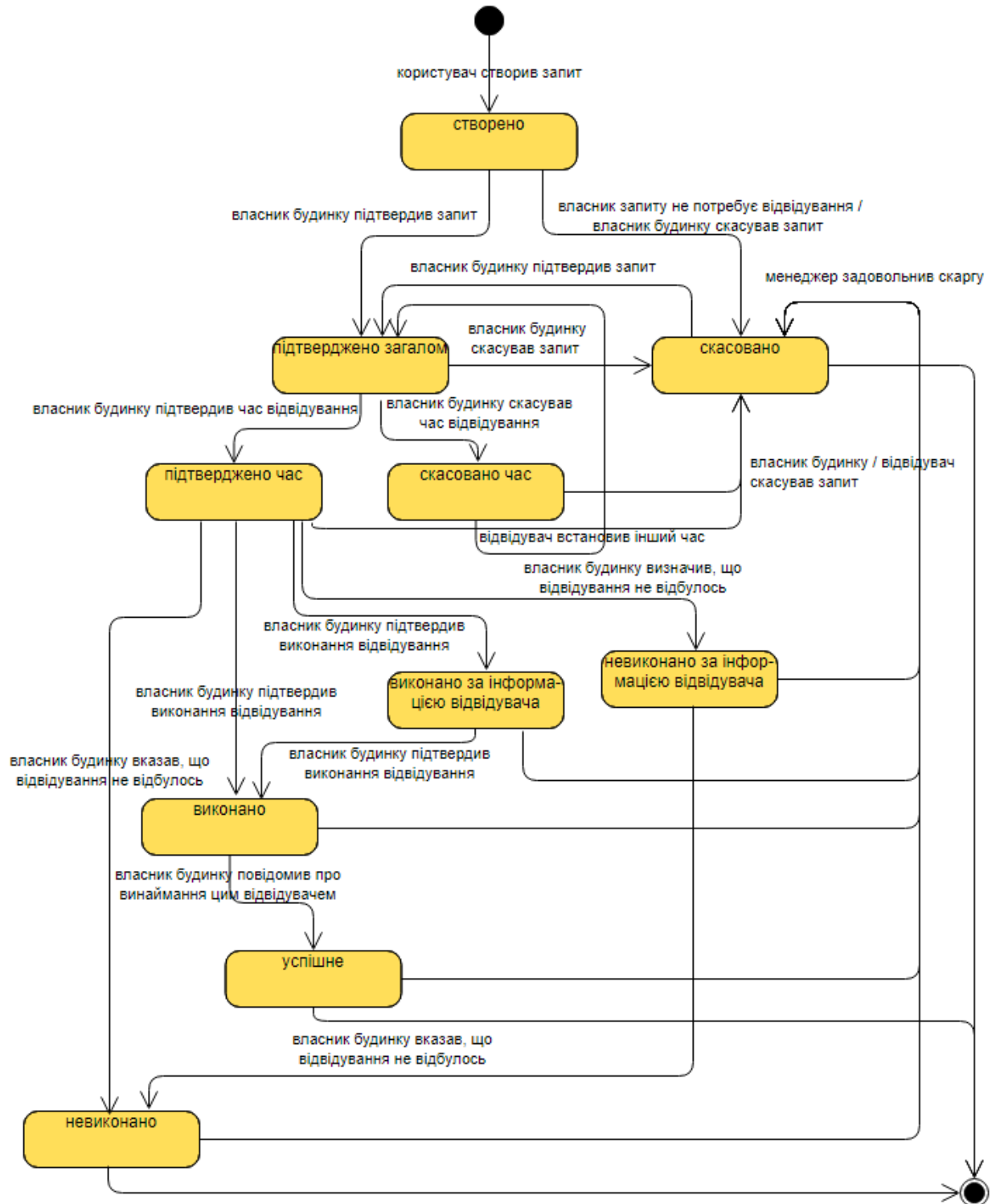


Рисунок 4.2 – Діаграма станів відвідування будинку

У момент, коли користувач створює запит на відвідування, відвідування стає створеним. Після цього власник будинку може або підтвердити цей запит, або скасувати його. Якщо власник будинку підтвердив запит, то далі користувач, який створив запит, може внести конкретні дати і час відвідування, тоді власник далі може або підтвердити час відвідування, або скасувати його. Окрім того власник будинку може змінити думку та скасувати підтверджений запит. Після того, як відвідування перейшло до стану підтвердженого за часом, то воно може стати або виконаним, якщо про це повідомив власник будинку, або невиконаним, або у випадку, якщо це повідомив відвідувач, то перейти до окремих станів виконаного за інформацією відвідувача або навпаки такого ж невиконаного. З цих станів у випадку, якщо інша сторона подала скаргу, відвідування може перейти до стану скасованого, якщо цю скаргу підтвердить менеджер.

Якщо відвідування виконано за інформацією відвідувача або не виконано за інформацією відвідувача, то далі власник будинку може цей факт підтвердити, перевіривши відвідування до станів відповідно виконано або не виконано.

Якщо відвідування виконано, а після цього реалізовано винаймання, то власник будинку може перевести таким чином відвідування до стану успішного.

4.2 Рішення з реалізації вебзастосунку

Створення застосунку в Django одразу передбачає певну його структуру, тому далі потрібно було визначити необхідні моделі, що описані одразу нижче, представлення, а до них відповідні шаблони вебсторінок, для розробки макетів використано шаблон [11], та менеджер для визначення перенаправлення адрес запитів на представлення.

Класи моделей створені у чіткій відповідності зі структурою БД, яку було представлено загалом і описано як сутності в розділі 3. Класи моделей Cottage, Structure, Country, Region, Geopoint, District включають текстове поле з назвою та наявності посилання на іншу модель.

Модель Cottage реалізована за рахунок класу оголошення про можливості довгострокового винаймання будинку з полями:

- id – поле для забезпечення зовнішнього ключа моделі будинку, тип BigAutoField;
- cottage – тип будинку, який визначений моделлю Cottage, тип ForeignKey;
- owner – користувач, який є власником оголошення про можливості довгострокового винаймання будинку, реалізовано поле через зв'язування з моделлю User, тип ForeignKey;
- caption – назва оголошення (фактично назва будинку, який пропонується для довгострокового винаймання), тип CharField;
- geolatitude – широта, на якій знаходиться цей будинок, тип FloatField;
- geolongitude – довгота, на якій знаходиться цей будинок, тип FloatField;
- district – район, до якого відноситься цей будинок, реалізовано поле через зв'язування з моделлю District, тип ForeignKey;
- visitprice – вартість відвідування будинку для подальшого вибору (може бути і безкоштовним), тип DecimalField;
- presentationtext – текст, який використовується в оголошенні для опису будинку, тип CharField;
- presentationphoto – файл фотографії будинку, яка застосовується як титульна фотографія оголошення, тобто відображається в результатах пошуку і в будь-яких позиціях, де відображається тільки одна фотографія, тип ImageField;

- `addtime` – час, коли було створено (оприлюднено оголошення могло бути пізніше) оголошення про можливості довгострокового винаймання будинку, тип `DateTimeField`;
- `active` – стан, в якому перебуває оголошення про можливості довгострокового винаймання будинку в поточний момент (не оприлюднено, оприлюднено, архівне), тип `SmallIntegerField`;
- `deposit` – сума депозиту, який має бути одноразово внесено для виконання довгострокового винаймання будинку з оголошення, тип `DecimalField`;
- `furnished` – визначення того, чи будинок є мебльованим, чи потребує використання в'їзду з власними меблями, тип `BooleanField`;
- `maxrental` – максимальна кількість місяців поспіль, протягом якої власник допускає винаймання будинку, тип `SmallIntegerField`;
- `tax` – відсоток податків, які необхідно сплатити безпосередньо при винайманні цього будинку з суми винаймання, тип `FloatField`;
- `places` – максимальна кількість людей, які можуть жити в будинку, тип `SmallIntegerField`;
- `bedrooms` – кількість спальних кімнат у будинку, тип `SmallIntegerField`;
- `bedtype` – найкращий тип спального місця, наявного в будинку (ліжка, диван тощо), тип `SmallIntegerField`;
- `beds` – кількість ліжок або диванів, які знаходяться в цьому будинку, тип `SmallIntegerField`;
- `bedroomlock` – визначення того, чи закриваються спальні кімнати в будинку на замок, чи ні, тип `BooleanField`;
- `bathrooms` – кількість санвузлів, які знаходяться в цьому будинку, тип `SmallIntegerField`;
- `shower` – визначення того, чи є в цьому будинку душ, тип `BooleanField`;
- `toilets` – кількість туалетних кімнат у будинку, тип `SmallIntegerField`;

- floors – кількість поверхів у будинку, тип `SmallIntegerField`;
- gym – визначення того, чи є в цьому будинку спортивний зал або певні можливості для занять спортом, тип `BooleanField`;
- closedterritory – визначення того, чи розташований цей будинок на закритій території (доступ через перевірку з шлагбаумом або певний варіант обмеженого доступу), чи ні, тип `BooleanField`;
- balcony – визначення того, чи є в цьому будинку балкон, тип `BooleanField`;
- parking – тип парковки доступний в цьому будинку (приватний гараж, паркування на території власного двору, паркування на території спільного двору, паркування на вулиці поруч платно, паркування на приватній парковці з дахом або під землею поруч платно, паркування на приватній парковці поруч платно, паркування на вулиці поруч безкоштовно), тип `SmallIntegerField`;
- ratio – визначення того, чи є на території (на вулиці) цього будинку власний майданчик зі стільцями та столиком, тип `BooleanField`;
- lawn – визначення того, чи є на території цього будинку власний газон, тип `BooleanField`;
- garden – визначення того, чи є на території цього будинку власний сад, тип `BooleanField`;
- heating – тип обігріву приміщення, що використовується, тип `SmallIntegerField` (газ, електричне опалення, дрова або інший природний спосіб з власним котлом, тільки обігрівач);
- electricity – тип доступу до електричної мережі (через централізовану мережу, можливість використання електричних батарей), тип `BooleanField`;
- conditioner – тип кондиціонування повітря в будинку (загальна система, окремі блоки, тільки вентилятори, відсутнє), тип `SmallIntegerField`;
- design – стиль дизайну, використаний всередині будинку, який визначений моделлю `Design`, тип `ForeignKey`;

- cabinet – визначення того, чи є в будинку власний кабінет, тип BooleanField;
- kitchen – тип кухні (окреме приміщення, невелика кімната, спільна кімната з їдальнею, відсутня окрема облаштована кухня), тип SmallIntegerField;
- kitchenware – визначення того, чи надається за винаймання будинку кухонне обладнання, чи потрібно використовувати тільки власне, тип BooleanField;
- cellar – визначення того, чи є в будинку підвал, тип BooleanField;
- publictransport – визначення того, чи доступний поруч громадський транспорт для проїзду, чи він відсутній, тип BooleanField;
- size – загальна площа внутрішнього приміщення будинку, тип FloatField;
- outsize – загальна площа землі, доступна під будинком, тип FloatField;
- pets – визначення того, чи можна винаймати будинок з власними домашніми тваринами, тип BooleanField;
- sound – визначення того, чи можна гучно слухати музику, грати на музичних інструментах у законно відведений для цього час, тип BooleanField;
- smoking – визначення того, чи можна палити в самому будинку, чи тільки на зовнішній території, тип BooleanField;
- washingmachine – визначення того, чи доступна пральна машина під час винаймання, чи необхідно мати власну, тип BooleanField;
- dryer – визначення того, чи доступна сушарка під час винаймання, чи необхідно мати власну, тип BooleanField;
- dishwasher – визначення того, чи доступна машина для миття посуду під час винаймання, чи необхідно мати власну, тип BooleanField;
- forstudent – визначення того, чи допускає (чи заохочує) власник будинку його винаймання студентами, тип BooleanField;

- `formale` – визначення того, чи допускає (чи заохочує) власник будинку його винаймання жінками, тип `BooleanField`;
- `forfemale` – визначення того, чи допускає (чи заохочує) власник будинку його винаймання чоловіками, тип `BooleanField`;
- `forcouples` – визначення того, чи допускає (чи заохочує) власник будинку його винаймання парами, тип `BooleanField`;
- `forworker` – визначення того, чи допускає (чи заохочує) власник будинку його винаймання працюючими особами (відповідно не пенсіонерами та не студентами), тип `BooleanField`;
- `internet` – тип доступу до інтернету, який вже наявний у цьому будинку, тобто не потребує додаткового підключення, налаштування, тип `SmallIntegerField`;
- `desk` – визначення того, чи доступне окреме робоче місце в цьому будинку, тип `BooleanField`;
- `withbills` – визначення того, чи входить в оплату за місяць оплата комунальних послуг, тип `BooleanField`;
- `detailaddress` – повна адреса будинку, з деталями, тип `CharField`;
- `visitinfo` – деталі щодо відвідування будинку, які мають бути надані користувачу, якому було підтверджено відвідування та визначено час відвідування, тип `CharField`;
- `stars` – середня оцінка, отримана цим будинком за відгуками після його відвідування або винаймання користувачами, тип `FloatField`;
- `costs` – загальні витрати на обслуговування будинку, тип `DecimalField`;
- `garages` – кількість гаражів на території будинку, тип `SmallIntegerField`;
- `pool` – наявність або відсутність басейну на території будинку, тип `BooleanField`.

Модель `Cottagephoto` реалізована за рахунок класу додаткової фотографії будинку з полями:

- id – поле для забезпечення зовнішнього ключа моделі фотографії будинку, тип BigAutoField;
- cottage – будинок, який показано на фотографії (додаткова фотографія з оголошення), визначений моделлю Cottage, тип ForeignKey;
- photofile – сама фотографія будинку, яка додатково представляється в оголошенні, тип ImageField;
- presentationtext – текст опису фотографії будинку, тип CharField;
- addtime – час збереження цієї фотографії будинку в програмі, тип DateTimeField.

Модель Cottagestructure реалізована за рахунок класу споруди, що відноситься до будинку, з полями:

- id – поле для забезпечення зовнішнього ключа моделі споруди з оголошення про можливості довгострокового винаймання будинку, тип BigAutoField;
- cottage – будинок, до якого відноситься ця споруда, визначений моделлю Cottage, тип ForeignKey;
- structuretype – тип споруди, визначений моделлю Structuretype, тип ForeignKey;
- caption – назва споруди, що відноситься до будинку, тип CharField;
- addtime – час збереження даних цієї споруди до оголошення, тип DateTimeField;
- presentationtext – текст, який використовується в оголошенні для опису саме цієї споруди будинку, тип CharField;
- presentationphoto – файл фотографії споруди будинку, яка застосовується для відображення цієї споруди, тип ImageField;

Модель Monthprice реалізована за рахунок класу вартості винаймання будинку на місяць з полями:

- id – поле для забезпечення зовнішнього ключа моделі вартості винаймання будинку на місяць, тип BigAutoField;

- cottage – будинок, до якого відноситься цей запис про його вартість, визначений моделлю Cottage, тип ForeignKey;

- months – кількість місяців винаймання, з якої використовується цей варіант, тип SmallIntegerField;

- beginperiod – дата, з якої встановлена така вартість (одночасно діє тільки один варіант для будинку за однакової кількості місяців), тип DateField;

- price – вартість місяця винаймання цього будинку, тип DecimalField.

Модель Cottageinaccessible реалізована за рахунок класу, в якому визначаються періоди, коли будинок недоступний для винаймання, з полями:

- id – поле для забезпечення зовнішнього ключа моделі недоступності будинку для винаймання, тип BigAutoField;

- cottage – будинок, до якого відноситься цей запис про недоступність будинку для винаймання, визначений моделлю Cottage, тип ForeignKey;

- beginperiod – дата, починаючи з якої будинок не може бути винайнятий, тип DateField;

- endperiod – остання дата, до якої будинок не може бути винайнятий, тип DateField.

Модель Conference реалізована за рахунок класу консультування за будинком з полями:

- id – поле для забезпечення зовнішнього ключа моделі консультування за будинком, тип BigAutoField;

- user – користувач, який запросив це консультування щодо будинку, визначений моделлю User, тип ForeignKey;

- cottage – будинок, стосовно якого було запитано це консультування, визначений моделлю Cottage, тип ForeignKey;

- addtime – час, коли було створено початковий запит на консультування, тип DateTimeField;

- active – стан, в якому це консультування перебуває (створене, підтверджене, заборонене), тип SmallIntegerField.

Модель Conferencemessage реалізована за рахунок класу репліки в консультуванні за будинком з полями:

- id – поле для забезпечення зовнішнього ключа моделі репліки в консультуванні за будинком, тип BigAutoField;
- conference – консультування, якого стосується ця репліка, визначене моделлю Conference, тип ForeignKey;
- user – користувач, який створив цю репліку, визначений моделлю User, тип ForeignKey;
- message – репліка в консультуванні за будинком, тип CharField;
- addtime – час відправлення репліки, тип DateTimeField.

Модель User реалізована за рахунок класу користувача програми з полями:

- id – поле для забезпечення зовнішнього ключа моделі користувача в програмі, тип BigAutoField;
- djangouser – користувацький обліковий запис з системи Django, яким ідентифікується цей користувач, визначений моделлю за замовчуванням в Django з назвою User, тип ForeignKey;
- displayedname – ім'я користувача, яким він бажає відображатися в програмі, тип CharField;
- gender – стать користувача, тип SmallIntegerField (користувач може в тому числі відмовитись надавати цю інформацію, однак і пошук тоді буде виконаний без врахування цієї інформації за замовчуванням);
- photo – фотографія, яка використовується в профілі користувача, тип ImageField;
- phone – номер телефону користувача, тип CharField;
- showphone – визначення того, чи можна показувати номер телефону власнику будинку одразу, а відвідувачу до фіксування дати відвідування будинку, тип BooleanField;
- stars – середня оцінка, отримана цим користувачем від власників будинків, які він оглядав або винаймав, тип FloatField;

– blocks – користувачі, які були заблоковані цим користувачем для взаємодії (надсилання запитів цьому користувачу щодо його будинків, надсилання реплік у консультування), представляє собою перелік користувачів, пов’язаний з моделлю User, тип ManyToManyField.

Модель Visit реалізована за рахунок класу відвідування будинку з полями:

– id – поле для забезпечення зовнішнього ключа моделі відвідування будинку, тип BigAutoField;

– user – користувач, який збирається відвідати цей будинок, визначений моделлю User, тип ForeignKey;

– cottage – будинок, відвідування якого відбувається, визначений моделлю Cottage, тип ForeignKey;

– addtime – час створення запиту на відвідування будинку, тип DateTimeField;

– visittime – час відвідування будинку, який було вже підтверджено або який запитано, тип DateTimeField;

– state – стан запиту щодо відвідування будинку, тип SmallIntegerField (створено, підтверджено, скасовано, виконано за інформацією відвідувача, виконано, успішне);

– visitortext – текст, який адресував користувач, який бажає відвідати будинок, власнику будинку в запиті, тип CharField;

– ownertext – текст, який адресував користувачу, який бажає відвідати будинок, власник будинку в запиті (пряма відповідь власника), тип CharField.

Модель Visitperiod реалізована за рахунок класу допустимого часу відвідування будинку з полями:

– id – поле для забезпечення зовнішнього ключа моделі допустимого часу відвідування будинку, тип BigAutoField;

– cottage – будинок, відвідування якого відбувається за цим допустимим часом, визначений моделлю Cottage, тип ForeignKey;

- simultaneous – можливість одночасного відвідування декількох будинків цього власника (тобто чи можна за наявності підтвердженого на той же час запиту на відвідування за іншим будинком власника допускати запит за ще одним будинком власника на той же самий час), тип BooleanField;

- beginperiod – час, з якого розпочинається відвідування або з якого можливі відвідування (тобто декілька за відведений час), тип DateTimeField;

- endperiod – час, до якого допустиме відвідування будинку, тип DateTimeField;

- duration – максимальна тривалість одного відвідування (кількість хвилин), тип SmallIntegerField;

- interval – інтервал часу між декількома відвідуваннями, якщо за цей період може відбутися декілька відвідувань, тип SmallIntegerField.

Модель Cottage review реалізована за рахунок класу відгуку про будинок після його відвідування або винаймання з полями:

- id – поле для забезпечення зовнішнього ключа моделі відгуку про будинок після його відвідування або винаймання, тип BigAutoField;

- cottage – будинок, стосовно відвідування або винаймання якого створено цей відгук, визначений моделлю Cottage, тип ForeignKey;

- user – користувач, який залишив цей відгук про будинок після його відвідування або винаймання, визначений моделлю User, тип ForeignKey;

- addtime – час, коли було збережено цей відгук про будинок після його відвідування або винаймання, тип DateTimeField;

- reviewtext – текст відгуку про будинок після його відвідування або винаймання, тип CharField;

- stars – цілочисельна оцінка з відгуку про будинок після його відвідування або винаймання, тип SmallIntegerField.

Модель Visitor review реалізована за рахунок класу відгуку про відвідувача будинку з полями:

- id – поле для забезпечення зовнішнього ключа моделі відгуку про відвідувача будинку, тип BigAutoField;

- owner – користувач, який залишив цей відгук про відвідувача будинку, визначений моделлю User, тип ForeignKey;
- visitor – користувач, про якого як відвідувача будинку створений власником будинку відгук, визначений моделлю User, тип ForeignKey;
- addtime – час, коли було збережено цей відгук про відвідувача будинку, тип DateTimeField;
- reviewtext – текст відгуку про відвідувача будинку, тип CharField;
- stars – цілочисельна оцінка з відгуку про відвідувача будинку, тип SmallIntegerField.

Модель Cottagerental реалізована за рахунок класу винаймання будинку, що може бути зафіксовано в програмі, з полями:

- id – поле для забезпечення зовнішнього ключа моделі винаймання будинку, тип BigAutoField;
- cottage – будинок, який було винайнято користувачем, визначений моделлю Cottage, тип ForeignKey;
- client – користувач, який винайняв цей будинок, визначений моделлю User, тип ForeignKey;
- beginperiod – час, з якого розпочинається це винаймання будинку, тип DateTimeField;
- endperiod – час, коли було завершено це винаймання, якщо воно завершене (інакше порожнє значення), тип DateTimeField.

Модель Complaint реалізована за рахунок класу скарги, що може подаватися на власників будинків або на відвідувачів, з полями:

- id – поле для забезпечення зовнішнього ключа скарги, тип BigAutoField;
- cottage – будинок, стосовно оголошення за яким або стосовно взаємодії з власником якого створено скаргу (або протилежно будинок, власником якого було створено скаргу), визначений моделлю Cottage, тип ForeignKey;

- user – користувач, який створив скаргу на оголошення про можливість довгострокового винаймання будинку або на його власника, визначений моделлю User, тип ForeignKey;

- creator – визначення того, що скарга була створена власником будинку (True) або користувачем, який був відвідувачем або винаймачем (False), тип BooleanField;

- ftext – текст, у якому викладено суть скарги, тип CharField;

- addtime – час, коли було збережено цю скаргу, тип DateTimeField;

- manager – користувач, який знаходиться в ролі менеджера та обробив цю скаргу, визначений моделлю User, тип ForeignKey;

- manageranswer – відповідь від менеджера стосовно цієї скарги, тип CharField;

- state – стан роботи над скаргою (створена, оброблена, підтверджена, скасована), тип SmallIntegerField.

Пошук будинків, доступних для довгострокового винаймання, потребував створення представлення `seek_cottage`, функція якого не має параметрів, але отримує необхідні значення через GET-запит, включаючи ідентифікатори типів будинків (обирається один з), ідентифікатори потрібних типів споруд (обираються всі одночасно), країну, область, населений пункт, район розташування, максимальну вартість відвідування, на яку готовий погодитись користувач, максимальний депозит, потребу в меблях, мінімальний рейтинг, мінімальну потрібну кількість місць, мінімальну потрібну кількість спальних кімнат, тип ліжка не менше ніж заданий, мінімальну потрібну кількість ліжок, потребу в замках на дверях спальні, потребу в наявності душа, мінімальної кількості туалетних кімнат, мінімальну і максимальну кількість поверхів, потребу в наявності спортивного залу, закритої території, паркінгу з типом не гірше заданого, наявність майданчика зі столиком і стільцями, наявність газону, наявність саду, потрібний тип обігріву приміщення, наявність електричних батарей, потрібний тип кондиціонування повітря, потрібний стиль оформлення

приміщення, наявність кабінету, наявність кухні потрібного типу, наявність кухонного приладдя, наявність підвалу, доступ до громадського транспорту, мінімальну і максимальну потрібну площу приміщення і окремо території будинку, можливість винаймання з домашніми тваринами, можливість прослуховувати гучно музику і грати на музичних інструментах, можливість палити в приміщенні, наявність пральної, сушильної машини, машини для миття посуду, можливість винаймання студентами, чоловіками, жінками, парами, працівниками, потрібний варіант доступу до мережі інтернет, наявність робочого столу, дати, в які планується винаймати будинок, мінімальна і максимальна вартість винаймання будинку за цей період та порядок сортування (за вартістю винаймання, за рейтингом, за площею приміщення). Якщо ці параметри не задані, то повертаються їх можливі значення для формування повної форми пошуку. Якщо задані, то серед всіх оголошень вибираються тільки оприлюднені, а від них відфільтровуються ті, що не відповідають заданим параметрам пошуку.

Представлення `show_cottage` реалізує ознайомлення з оголошенням про можливості довгострокового винаймання будинку з номером `cottage`, що передається через GET-запит. Окрім витягання всіх даних про цей будинок, перед цим перевіривши, що користувач є або власником оголошення, або оголошення є оприлюдненим, у випадку передачі дат винаймання через запит також виконується визначення можливості такого винаймання і вартості.

Оприлюднення оголошення стосовного власного будинку потребувало створення представлення `publish_cottage` з параметром `cottageid`, що визначає номер раніше створеного оголошення щодо цього будинку. Відповідно архівування оголошення стосовно власного будинку реалізовано представленням `archive_cottage`. Дії виконуються, якщо авторизований користувач є власником будинку.

Приймання запиту на відвідування будинку реалізовано представленням `visit_confirm`. Дії виконуються тоді, коли авторизований користувач є власником будинку, а стан є створеним або скасованим. У всіх

інших випадках зміна стану на підтверджений загалом не відбувається. Скасування запиту на відвідування будинку реалізовано подібним представленням `visit_cancel`. Підтвердження дати відвідування будинку реалізовано представленням `visit_date_confirm` стосовно відвідування з номером `visitid`, переданого через запит. Виконується перевірка того, що авторизований користувач є власником цього будинку, а стан є підтвердженим або таким, коли скасовано час відвідування. Також перевіряється, що такий час відвідування дозволений власником. Скасування дати відвідування будинку реалізовано представленням `visit_date_cancel` і діє аналогічно попередньому. Усі ці представлення мають параметр `visitid` з номером відвідування будинку.

Відображення переліку запитів на відвідування за власним будинком з номером `cottageid` реалізовано представленням `cottage_visits`. Параметр `visitstate` визначає те, в яких станах потрібно повернути запити, а нульове значення відповідає всім запитам. Відображення переліку всіх запитів на відвідування власних будинків реалізовано представленням `owner_visits` з аналогічним параметром, при цьому повертаються запити з заданим станом, які стосуються будинків, якими володіє авторизований користувач. Відображення переліку власних запитів на відвідування будинків реалізовано представленням `user_visits`, параметр `visitstate` працює аналогічно.

Відображення переліку власних оголошень про можливості довгострокового винаймання будинку реалізовано представленням `owner_cottages`.

Створення відгуку про будинок після його відвідування або винаймання реалізовано представленням `cottage_review_create`, а номер будинку задається параметром `cottageid`. У представленні виконується перевірка, що користувач відвідував цей будинок протягом місяця з поточної дати або він виконував його винаймання. Відображення відгуків про будинок, створених після його відвідування або винаймання, виконується представленням `cottage_reviews` за будинком з номером `cottageid`. Створення

відгуку про відвідувача будинку реалізовано представленням `user_review_create` з перевіркою того, що цей користувач був задіяний у винайманні або відвідуванні будинку з номером `cottageid`, а логін користувача задається параметром `visitorusername`.

Відображення інформації про користувача як відвідувача реалізовано представленням `visitor_profile` з параметром `visitorusername`, що визначає логін цього користувача. Одночасно з даними користувача повертаються і відгуки про нього. Відображення інформації про користувача як власника будинку реалізовано представленням `owner_profile` з параметром `ownerusername`, що визначає логін цього користувача, повертаючи окрім даних користувача ще 3 найкращі за середнім рейтингом будинки цього користувача, 10 відгуків про будинки цього користувача.

Відображення переліку консультацій за власним будинком реалізовано представленням `cottage_conferences` з перевіркою власника та за будинком з номером `cottageid`. Відображення переліку всіх консультацій для користувача стосовно його будинків реалізовано представленням `owner_conferences`. Відображення переліку консультацій, запитаних користувачем, що авторизувався, створено представленням `user_conferences`. Перегляд консультацій менеджером реалізовано представленням `manager_conferences` з параметрами `username` логіну користувача, `conferenceid` з номером конференції, `ownerusername` з логіном користувача, який є власником будинку, стосовно якого відбувалось консультування. Відправлення репліки в консультацію за будинком з номером `conferenceid` відбувається за представленням `conference_message_create`. Створення самого запиту на консультацію за будинком реалізовано представленням `conference_create` стосовно будинку з номером `cottageid`. Перегляд консультування за будинком реалізовано представленням `conference_show`.

Створення власного оголошення про можливості довгострокового винаймання будинку реалізовано представленням `cottage_create`, що працює на основі форми. Оновлення можливостей довгострокового винаймання

власного будинку у власному оголошенні утворено аналогічно на основі представлення `cottage_edit`. Додавання споруд до оголошення з номером `cottageid` реалізовано окремо через представлення `cottage_structure_create`, а оновлення даних споруди з номером `cottagestructureid`, що відноситься до будинку, реалізовано представленням `cottage_structure_edit`, відповідно вилучення споруди з оголошення – представленням `cottage_structure_delete`.

Визначення недоступного періоду для довгострокового винаймання будинку реалізовано 3 представленнями: `cottageinaccessible_create` для внесення нового періоду для будинку з номером `cottageid`, `cottageinaccessible_delete` для вилучення такої внесеної дати з номером `cottageinaccessibleid` та `cottage_inaccessibles` для перелічення всіх визначених дат за будинком з номером `cottageid`.

Визначення вартості довгострокового винаймання будинку реалізовано 4 представленнями: `cottage_price_create` для внесення нового запису про вартість для будинку з номером `cottageid`, `cottage_price_edit` для оновлення даних, `cottage_price_delete` для вилучення такого запису про вартість з номером `cottagepriceid` та `cottage_prices` для перелічення всіх записів про вартість за будинком з номером `cottageid`. Окремо створено представлення `predict_cottage_price` для отримання доступу до прогнозування вартості на основі запропонованого методу.

Визначення доступного для відвідування будинку часу реалізовано аналогічно через 4 представлення, при цьому при редагуванні виконується перевірка дат порівняно з існуючими, а якщо вони сумісні з якимись, то тоді зберігається тільки одна позиція, а дати розширюються (тобто встановлюється один єдиний період замість того, щоб зберігати декілька).

Визначення виконуваного винаймання будинку реалізовано представленням `cottage_rental_create`, параметр якого `cottageid` визначає номер будинку, авторизований користувач повинен бути його власником, а параметр `visitorusername` визначає логін клієнта, який винаймає будинок, який може бути незаданий, якщо наприклад, клієнт не з цієї програми. При

внесенні цих даних оголошення одразу переводиться до заархівованого, а всі відвідування будинку, які були визначені, скасовуються.

Блокування користувача стосовно власної сторінки відбувається представленням `block_user` стосовно користувача з логіном `username`.

Перегляд запитів на відвідування будинку за будинком з номером `cottageid` менеджером реалізовано через представлення `manager_cottage_visits`, а відповідно за відвідувачем з логіном `visitorusername` – `manager_user_visits`.

Відправлення запиту на відвідування будинку з номером `cottageid` реалізовано представленням `cottage_visit_create`. У випадку, якщо вказані дати не входять в доступні, то повертаються дані про помилку. Встановлення після погодження власником загальному запиту, але не погодження дати, нової дати відбувається через представлення `visit_date_set` для відвідування з номером `visit_date_set`.

Створення скарги на користувача реалізовано представленням `complaint_create`, використовуючи дані авторизації, відображення переліку створених скарг на власників будинків реалізовано представленням `owner_complaints` і підтримується тільки, якщо користувач є менеджером або надаються тільки по авторизованому користувачу, створених скарг на відвідувачів реалізовано представленням `visitor_complaints`, перегляд створеної скарги реалізовано представленням `complaint_show` за номером `complaintid` з перевіркою прав доступу (автори скарги, менеджер або адресат), підтвердження скарги реалізовано представленням `complaint_confirm`, надання відповіді на скаргу – `complaint_answer`, скасування скарги – `complaint_cancel`. У останніх представленнях перевіряється, що це менеджер авторизувався, а в `complaint_confirm` блокується доступ користувача до програми, а всі його будинку переводяться до стану заархівованих оголошень зі скасуванням всіх відвідувань його будинків. Пошук користувачів реалізовано представленням `user_search`, а підтримується воно тільки для авторизованого менеджера.

Для експериментального дослідження, для реалізації моделі використано пакет `scikit-learn`, в якому клас `RandomForestRegressor` реалізує метод Random Forest з параметром `n_estimators` для встановлення кількості дерев, клас `LinearRegression` реалізує лінійну регресію, а метод опорних векторів реалізований класом `SVR`.

4.3 Висновки за розділом 4

Порядок роботи з програмою визначено за діаграмами станів оголошення про можливість довгострокового винаймання будинку та винаймання будинку. Розроблені функції Django-вебзастосунку створені за сценаріями роботи користувача і менеджера та описані.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис програми

Програма призначена для підтримки довгострокового винаймання будинків, оголошення про яке створюється власником будинку, надаючи для цього інструменти пошуку оголошень, створення і визначення параметрів оголошень, створення запитів на відвідування будинку, консультування без відвідування, створення відгуків за результатами відвідування або винаймання, створення скарг на відвідувачів або власників будинків за необхідності та підтримки процесів вирішення конфліктів менеджерами.

5.2 Експлуатація програми

Програма надає користувачам інтерфейси у чіткій відповідності з запланованими сценаріями для роботи. Окремо надається інтерфейс для роботи менеджера, який має бути доступний за окремим посиланням. За базовим посиланням вебзастосунку доступний інтерфейс клієнта, він не передбачає переходу з нього на інтерфейс менеджера. Тобто це фактично два окремі варіанти використання програми, адже створення облікових записів менеджерів відбувається адміністратором через адміністративну панель, вони мають свої окремі сценарії використання програми, тому логічним є те, що доступ до цих інтерфейсів повністю розділений. Для отримання доступу потрібно ввести у браузер адресу вебсайту (інтерфейс користувача), тоді з'явиться сторінка пошуку. Ця сторінка містить ряд полів форми пошуку: тип будинків (обирається один з), типи споруд у будинку (додаються до переліку з вибором з випадającego переліку), країна, область, населений пункт, район розташування (може бути заданий набір тільки до певного рівня, тобто

наприклад, країна і область, не деталізуючи нижчі), максимальна вартість відвідування, на яку готовий погодитись користувач, максимальний депозит, потреба в меблях, мінімальний рейтинг, мінімальна потрібна кількість місць, мінімальна потрібна кількість спальних кімнат, тип ліжка не менше ніж заданий, мінімальна потрібна кількість ліжок, потреба в замках на дверях спальні, потреба в наявності душа, мінімальна кількість туалетних кімнат, мінімальна і максимальна кількість поверхів, потреба в наявності спортивного залу, закритої території, паркінгу з типом не гірше заданого, наявність майданчика зі столиком і стільцями, наявність газону, наявність саду, потрібний тип обігріву приміщення, наявність електричних батарей, потрібний тип кондиціонування повітря, потрібний стиль оформлення приміщення, наявність кабінету, наявність кухні потрібного типу, наявність кухонного приладдя, наявність підвалу, доступ до громадського транспорту, мінімальна і максимальна потрібна площа приміщення і окремо території будинку, можливість винаймання з домашніми тваринами, можливість прослуховувати гучно музику і грати на музичних інструментах, можливість палити в приміщенні, наявність пральної, сушильної машини, машини для миття посуду, можливість винаймання студентами, чоловіками, жінками, парами, працівниками, потрібний варіант доступу до мережі інтернет, наявність робочого столу, дати, в які планується винаймати будинок, мінімальна і максимальна вартість винаймання будинку за цей період та порядок сортування (за вартістю винаймання, за рейтингом, за площею приміщення), наявність басейну, мінімальна кількість гаражів. Проте не всі ці параметри потрібно задавати. По-перше, вони можуть бути пропущені і не встановлені, по-друге, пошук спочатку відображається в скороченому вигляді, включаючи тільки інтервал вартості винаймання, місто пошуку та плановані дати винаймання. Для того щоб розширити кількість фільтрів потрібно обрати розширений пошук, що відобразить всю форму. Після встановлення параметрів і натискання на кнопку Шукати відображаються

результати (рис. 5.1), а в формі можна встановити нові значення та відобразити нові результати кнопкою Змінити результати.

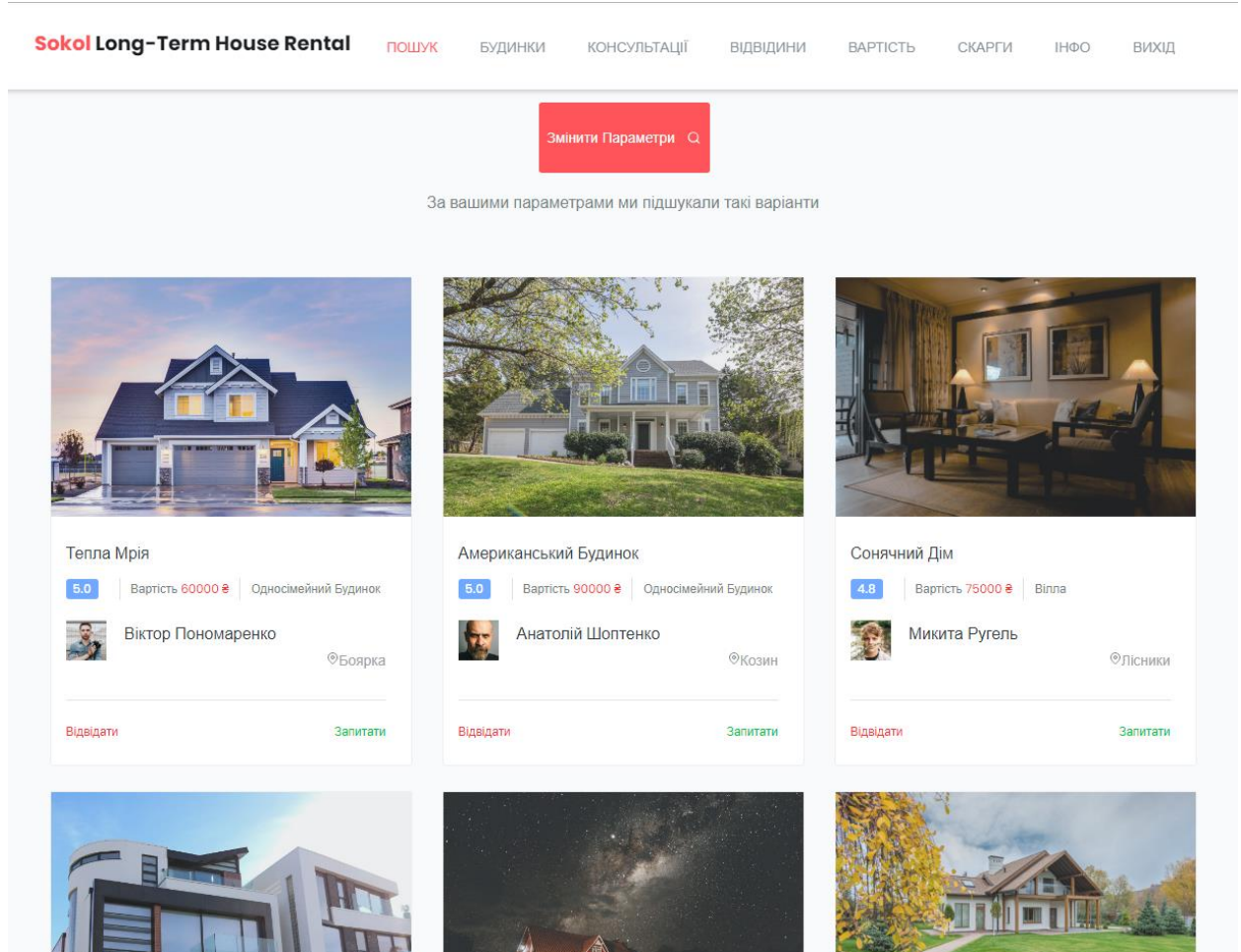


Рисунок 5.1 – Пошук будинків для довгострокового винаймання

У результатах демонструються всі знайдені будинки, за кожною позицією вказуючи основну фотографію, назву оголошення, середню оцінку, вартість винаймання, тип будинку, господаря (за посиланням можна перейти на сторінку з інформацією про господаря), місце, де розташовано будинок та посилання для користувача, що включають створення запиту для отримання консультації від власника будинку (посилання запитати) та оформлення запиту на відвідування будинку (посилання відвідати). Якщо обрати назву оголошення, то відкривається окрема сторінка з повною інформацією про будинок (рис. 5.2). Зверху знаходиться основна фотографія, а нижче на сторінці є галерея з іншими фотографіями будинку. На фотографії ж поля для визначення дат винаймання будинку та кнопка вартість для обчислення

вартості такого винаймання. Внизу фотографії знаходяться значення основних параметрів, які використані в створеній моделі, також тип будинку.

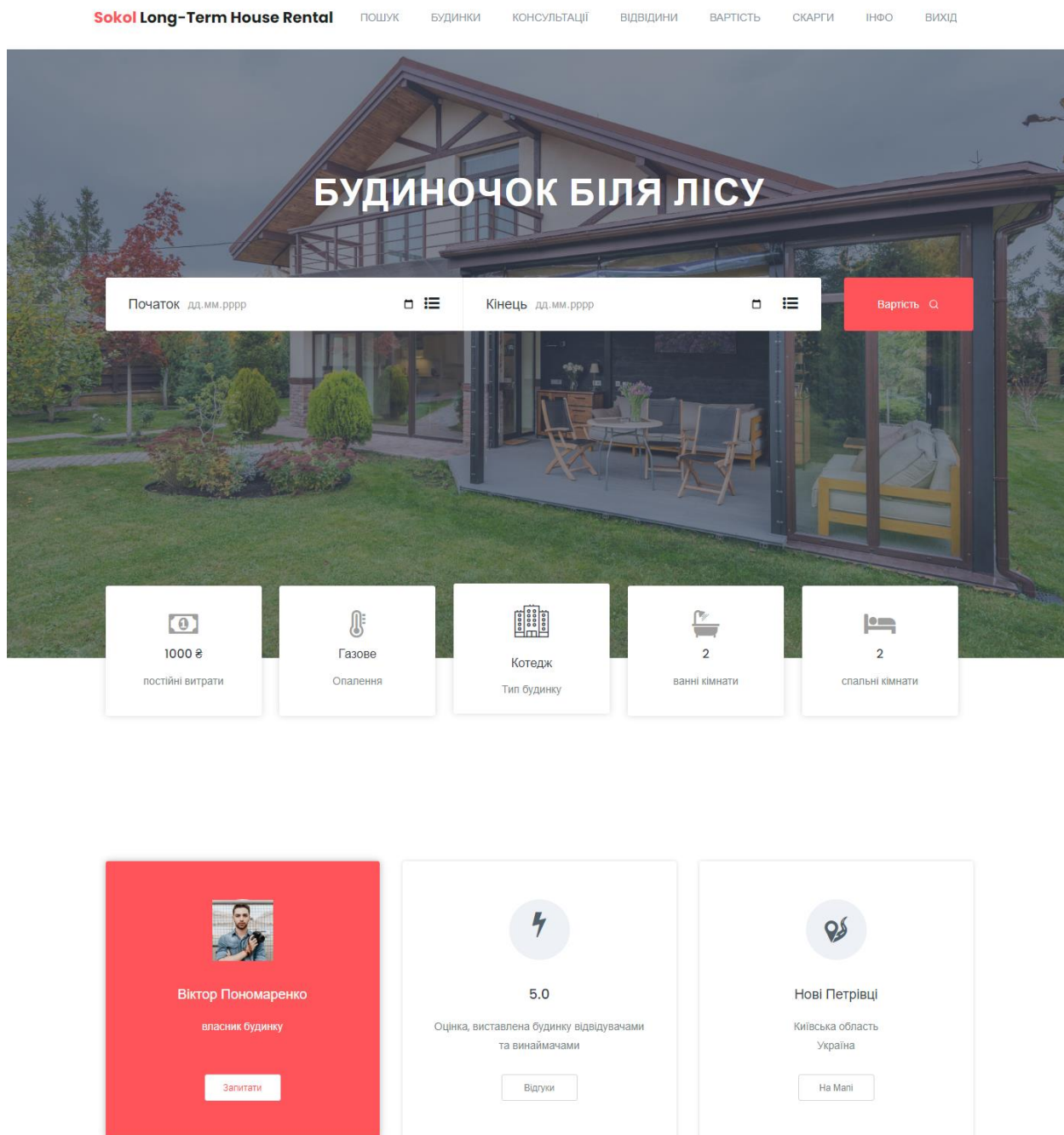


Рисунок 5.2 – Ознайомлення з оголошенням про можливості довгострокового винаймання будинку

На позицію нижче знаходяться дані про власника (натискаючи на ім'я, можна відкрити повну його сторінку, натискаючи на запитати можна перейти до отримання консультації у цього власника), середня оцінка будинку з посиланням для переходу до переліку самих відгуків, звідки можна перейти до повного переліку відгуків та перейти до сторінки користувача, який створив відгук, а також повна інформація про місце розташування будинку з можливістю відобразити в Google Maps за посиланням адресу на мапі. Ще нижче знаходиться перелік значень всіх інших параметрів будинку з оголошення, після цього перелік споруд на території будинку. Відповідно нижче, як було зазначено, є результати для обчислення вартості винаймання, кнопка для створення відвідування.

Верхній рядок сторінок містить основні інструменти, що дозволяють переходити між основними сценаріями. Пошук дозволяє повернутись в будь-який момент до сторінки пошуку будинків. Розділ Відвідини відображає свої заплановані відвідини будинку або відвідини за власними будинками (рис. 5.3). Вибір виконується натисканням однієї з кнопок (стає червоною).

Sokol Long-Term House Rental

ПОШУК

БУДИНКИ

КОНСУЛЬТАЦІЇ

ВІДВІДИНИ

ВАРТІСТЬ

СКАРГИ

ІНФО

ВИХІД

Клієнт

Власник

ВІДВІДУВАННЯ ВАШИХ БУДИНКІВ

Будинок

Будиночок біля лісу

Початок відвідувань

дд.мм.рррр

Кінець відвідувань

дд.мм.рррр

Стан

будь-які

Порядок

час створення

Запити

Будиночок Біля Лісу

4.5

Час Не Встановлено

Створено

Сгор Писаренко
консультація

Скасувати

Підтвердити

Будиночок Біля Лісу

4.5

30.11.2023 19:00-19:15

Підтверджено

Павло Чорний
консультація

Скасувати Час

Скасувати

Підтвердити Час

Будиночок Біля Лісу

5.0

28.11.2023 19:00-19:15

Виконано

Ігор Панасов
консультація

Створити Відгук

Винайнято

Рисунок 5.3 – Запити на відвідування будинку

Щоб не працювати з великим переліком запитів на відвідування одночасно, а до того ж тут і фактичні внесені винаймання можуть відображатися, то створено засоби фільтрування, які включають, по-перше, вибір самого будинку (зі сторінки з переліком власних будинків можна перейти до відвідувань всіх будинків відповідним посиланням або до переліку відвідувань вже за конкретним будинком, біля якого таке посилання знаходиться), також дати, відвідування за які мають бути показані, стан запитів та порядок відображення (час створення або час відвідування). За кожним запитом відображається назва будинку, оцінка користувача, який цей запит зробив, дата і час відвідування, стан запиту, ім'я користувача та фотографія, посилання для переходу до консультувань цього користувача за цим будинком, посилання для підтвердження або скасування запиту (для створеного), посилання для скасування час, підтвердження часу відвідування або скасування всього запиту (для підтвердженого запиту), тобто у відповідності зі станами, описаними для відвідувань. Якщо час відвідування

вийшов, а власник підтвердив виконання відвідування, то відображається посилання для визначення того, що відвідування завершилось винайманням. Раніше відповідно можна позначити відвідування як виконане або невиконане. Те саме може зробити і користувач, який відвідував цей будинок. Якщо ці позиції розходяться, то інший користувач може подати скаргу. Скарга доступна у випадку початкового підтвердження стану виконання однією зі сторін для іншої сторони. Окрім того скарга доступна зі сторінки оголошення (в самому низу відповідне посилання). Скарги розглядаються менеджером, він виносить рішення.

Розділ Будинки містить перелік всіх будинків, які належить користувачу, та можливість переходу до створення нового оголошення. Звідси можна змінювати стан оголошення (архівувати його або навпаки оприлюднювати), визначати за відповідним посиланням недоступний для винаймання період, вартість, додавати, оновлювати та редагувати споруди в оголошенні, переходити до запитів на відвідування за конкретним будинком або за всіма за відповідним посиланням, переходити до консультацій за певним будинком.

Розділ Консультації надає можливість відобразити всі консультації, які також обираються або для клієнта (той, хто запитує консультацію), або для власника будинку. Консультації відображаються або за будинком (фотографія будинку вказує на це), або за користувачем для власників (фотографія власника вказує на це). Консультації з більш новими повідомленнями демонструються вище. Розділ Скарги дозволяє переглянути власні скарги, щоб побачити результат їх опрацювання менеджером, та скарги, подані на користувача, який авторизований. Розділ Інформація надає доступ до зміни даних профілю.

Розділ Вартість дозволяє задати параметри будинку та отримати інформацію про орієнтовну вартість такого будинку (рис. 5.4). Моделі створюються під кожний доступний у програмі населений пункт за наявними даними. Тобто це вже моделі, створені з даних, зібраних у програмі.

Sokol Long-Term House Rental
ПОШУК
БУДИНКИ
КОНСУЛЬТАЦІЇ
ВІДВІДИНИ
ВАРТІСТЬ
СКАРГИ
ІНФО
ВИХІД

ВАРТІСТЬ МІСЯЦЯ ВІНАЙМАННЯ БУДИНКУ

Місто

Львів

Кількість ванних кімнат

1

Кількість спальних кімнат

2

Кількість гаражів

1

Наявність басейну

ні

Загальна площа

80

Площа земельної ділянки

120

Допустимі стандартні витрати

0

Система опалення

газ

Визначити

ОЧІКУВАНА ВАРТІСТЬ: 34529 €

Рисунок 5.4 – Визначення вартості довгострокового винаймання будинку

Менеджер з власного інтерфейсу після авторизації (рис. 5.5) має зверху розділи, що включають користувачів, звідки можна виконати пошук потрібно користувача, а за ним переглянути доступні консультації, запити, скарги, інформацію для зміни профілю.

Sokol Long-Term House Rental

КОРИСТУВАЧІСКАРГИІНФОВИХІД

СКАРГИ

Початок створення

30.11.2023

Кінець створення

02.12.2023

Стан

будь-які

Скарги

Будиночок Біля Лісу

КонсультаціяЗапитиВласник Будинку

Дата Створення: 01.12.2023 20:19



відадвувач

Павло Чорний

Власник не з'явився у зазначений час та не відповідає на мої запити, при цьому встановив, що відвідування виконано.

Скасувати

Відповідь

Підтвердити

Рисунок 5.5 – Оброблення скарг менеджером

Для того щоб працювати не зі всіма скаргами одночасно, менеджеру потрібно задати період створення скарги, тоді скарги тільки за вказаний період показуються. Також можна визначити стан скарг. Це може бути підтверджена, скасована, розглянута (тоді написана відповідь, але не визначений остаточний стан), нерозглянута (нові скарги). За кожною скаргою фактично менеджер має можливість або її підтвердити, тоді користувачу може бути заблокований доступ до керування власними оголошеннями, створення запитів, скасувати або написати відповідь. Посилання можуть змінюватися в залежності від попередніх дій. Окрім блокування користувача менеджером самі користувачі також можуть обмежити доступ до власних сторінок, відповідне блокування доступне з консультувань.

5.3 Експериментальне дослідження

Дані про вартість винаймання будинків у Монтевідео [12], столиці Уругваю, використані при проведенні експериментального дослідження. Вибірка доступна за ліцензією [13], що дозволяє зокрема змінювати дані, але не дозволяє комерційне використання. Тому за комерційного застосування програма має створити і використовувати модель на основі даних, які зібрані з самої програми.

Для експериментального дослідження використані не всі дані повністю, а спочатку виділено тільки ті екземпляри, які стосуються будинків. Оскільки у вибірці даних присутні дані стосовно офісних приміщень, складів, гаражів, інших типів приміщень, то вони були виключені на перших кроках. Окрім того з даних було усунуто всі дублікати, а також екземпляри, які мають нульову вартість винаймання будинку та вартість, що значно виділяється з множини, як це визначено у методі в розділі 2. На рис. 5.6 показано розподіл вартості винаймання будинків, які було залишено для подальшого дослідження.

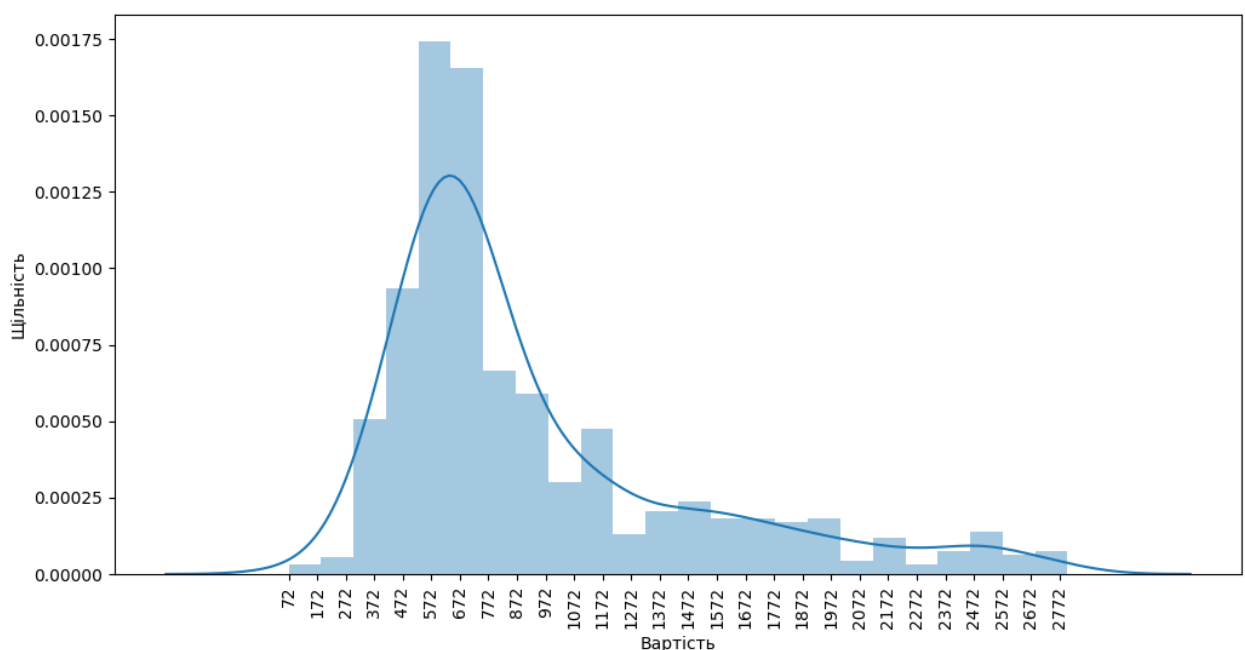


Рисунок 5.6 – Розділ вартості винаймання будинків

Оскільки вибірка має велику кількість ознак, було визначено коефіцієнт кореляції для всіх числових ознак стосовно вартості винаймання будинку (табл. 5.1) окрім категоріальних ознак.

Таблиця 5.1 – Кореляція між вартістю винаймання будинку та ознаками

Ознака вибірки даних	Коефіцієнт кореляції
Загальні додаткові витрати	0,53493
Кількість ванних кімнат	0,73813
Кількість спальних кімнат	0,52488
Загальна площа власності	0,54128
Площа тераси	0,20026
Площа земельної ділянки	0,56529
Рік будівництва	0,10797
Кількість гаражів	0,64654
Вид на море	0,20397
Наявність вітальні	0,13150
Наявність гардеробної	0,16888
Наявність басейну	0,32753
Наявність гаража або паркувального місця	0,10203
Можливість винаймання з домашніми тваринами	-0,10588
Під'єднання до газопостачання	-0,11771
Наявність ігрової кімнати або зони відпочинку	-0,02117
Наявність меблів	0,05743
Дуплекс	-0,00642
Тип системи опалення	0,38662
Наявність Wi-Fi	0,01170

Наявність зони барбекю	0,12392
------------------------	---------

Продовження таблиці 5.1

Ознака вибірки даних	Коефіцієнт кореляції
Наявність спортивного залу	0,05527
Наявність балкону	0,03177
Наявність складського приміщення	0,03194
Наявність кондиціонування повітря	0,04479
Наявність саду	0,28131
Наявність підігріву підлоги	0,26869

Використовуючи отримані дані, було створено ряд моделей на основі використання лінійної регресії, регресії на основі методу опорних векторів та дерев рішень на основі Random Forest (табл. 5.2). Також для Random Forest перевірено ряд варіантів, включаючи різні ознаки до створення моделі за методом з розділу 2.

Таблиця 5.2 – Експериментальне дослідження

Модель	Характеристики моделі	Середня абсолютна похибка
Лінійна регресія	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки	0,0826669272
Лінійна регресія	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат,	0,0861494262

	гаражів, загальна площа власності, площа земельної ділянки	
--	---	--

Продовження таблиці 5.2

Модель	Характеристики моделі	Середня абсолютна похибка
Метод опорних векторів	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки	0,0775624772
Метод опорних векторів	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, загальна площа власності, площа земельної ділянки	0,0793501855
Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, 100 дерев	0,0699638615
Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, загальна площа власності, площа земельної ділянки, 100 дерев	0,0708163836
Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, загальна площа власності, 100 дерев	0,0732172246
Random	Ознаки: загальні додаткові витрати, кіль-	0,0711231827

Forest	кість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, категорія стану будинку, 100 дерев	
--------	---	--

Продовження таблиці 5.2

Модель	Характеристики моделі	Середня абсолютна похибка
Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, тип системи опалення, 100 дерев	0,0705509
Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, 2000 дерев	0,0691457803
Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, тип системи опалення, 200 дерев	0,0687375081

Найкращі результати отримані за застосування методу з розділу 2 та створення моделі з ансамблем з 200 дерев, що мають наступні ознаки:

- загальні додаткові витрати;
- кількість ванних кімнат;
- кількість спальних кімнат;
- кількість гаражів;

- наявність басейну;
- загальна площа власності;
- площа земельної ділянки;
- тип системи опалення.

Ця модель має середню абсолютну похибку на 20,26 % меншу ніж у лінійної регресії та на 12,84 % меншу ніж у методу опорних векторів.

5.4 Висновки за розділом 5

Описано порядок роботи з програмою, реалізовано експериментальне дослідження за допомогою даних про довгострокове винаймання будинків, включаючи до нього лінійну регресію, метод опорних векторів, Random Forest. Запропонований метод мав у результаті абсолютну похибку на 20,26 % меншу ніж у лінійної регресії та на 12,84 % меншу ніж у методу опорних векторів.

ВИСНОВКИ

Виконана дипломна кваліфікаційна робота магістра дозволила розробити програмне забезпечення довгострокового винаймання будинків для надання інформації для обґрунтованого прийняття рішень та підвищення взаємодії між власниками будинків та винаймачами.

Розроблена програма призначена для підтримки довгострокового винаймання будинків, оголошення про яке створюється власником будинку, надаючи для цього інструменти пошуку оголошень, створення і визначення параметрів оголошень, створення запитів на відвідування будинку, керування цими запитами, консультування без відвідування або додатково до відвідування, створення відгуків за результатами відвідування або винаймання, створення скарг на відвідувачів або власників будинків за необхідності та підтримки процесів вирішення конфліктів менеджерами. Користувачам надається можливість визначити очікувану вартість винаймання будинку за один місяць на основі параметрів такого винаймання, включаючи загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, визначаючи загальну площу власності та земельної ділянки, тип системи опалення. Розробка застосовує мову Python, базу даних на основі системи PostgreSQL, фреймворк Django та базові засоби створення вебсторінок.

Наукова новизна роботи полягає в методі визначення вартості довгострокового винаймання будинків, який ґрунтується на створенні моделі прогнозування на основі методу Random Forest, даних відібраних інших будинків у заданому місті за визначеними ознаками, що дозволяє визначити очікувану місячну вартість винаймання будинку. Запропонований метод мав у результаті середню абсолютну похибку на 20,26 % меншу ніж у лінійної регресії та на 12,84 % меншу ніж у методу опорних векторів за результатами експериментального дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Приватні будинки [Електрон. ресурс] : знайти дешевий будинок – кращі ціни на будинки в Україні на OLX.ua. – Режим доступу : <https://www.olx.ua/uk/nedvizhimost/doma/?currency=UAH>.
2. HousingAnywhere [Electronic resource] : Mid to Long-Term Rentals in 400+ cities. – Access mode : <https://housinganywhere.com/>.
3. Rightmove – UK's number one property website for properties for sale and to rent [Electronic resource]. – Access mode : <https://www.rightmove.co.uk/>.
4. Rental Listings in Your Neighborhood. Trulia.com [Electronic resource]. – Access mode : <https://www.trulia.com/rent/>.
5. Sethi, A. Support Vector Regression Tutorial for Machine Learning [Electronic resource] / A. Sethi. – Access mode : <https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/>.
6. What is decision tree analysis? 5 steps to make better decisions [Electronic resource]. – Access mode : <https://asana.com/resources/decision-tree-analysis>.
7. Samarth, V. What is Random Forest In Data Science and How Does it Work? [Electronic resource] / V. Samarth. – Access mode : <https://emeritus.org/in/learn/data-science-random-forest/>.
8. Django Tutorial [Electronic resource]. – Access mode : <https://www.w3schools.com/django/>.
9. Django Web Framework (Python) [Electronic resource]. – Access mode : <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>.
10. Drupal – Open Source CMS. Drupal.org [Electronic resource]. – Access mode : <https://www.drupal.org/>.
11. Listrace – Directory Template. ThemeSINE [Electronic resource]. – Access mode : <https://themesine.com/downloads/listtrace-directory-template/>.
12. House rental Montevideo [Electronic resource]. – Access mode :

[https://www.kaggle.com/datasets/lasm1984/house-rental-price-in-montevideo?
select=houses.csv](https://www.kaggle.com/datasets/lasm1984/house-rental-price-in-montevideo?select=houses.csv).

13. CC BY-NC-SA 4.0 DEED Attribution-NonCommercial-ShareAlike
4.0 International [Electronic resource]. – Access mode :
<https://creativecommons.org/licenses/by-nc-sa/4.0/>.

ДОДАТОК А
Технічне завдання

Вступ

Винаймання будинків є важливим і актуальним процесом, який має певні особливості в залежності від того, на який період відбувається винаймання. Короткострокове винаймання має достатньо багато відомих у світі рішень для підтримки такої роботи, а довгострокове є достатньо обмеженим, адже зазвичай представлено вебсайтами, де виставляються оголошення, які не мають відгуків, оцінок, що характерно для короткострокового винаймання, весь процес взаємодії з користувачем обмежений отриманням даних оголошення, що сприяє тому, що інформація на вебсайті нерідко є застарілою, не відповідає дійсності. Тому потреба в програмному забезпеченні, яке могло б вплинути на цей процес взаємодії, перенести його на довгострокове винаймання та врахувати при цьому потреби винаймання саме будинків, що загалом є більш широким процесом ніж винаймання квартир через те, що будинки можуть містити багато додаткової інформації, додаткові споруди, які впливають на вибір клієнта в результаті, є достатньо великою.

A.1 Підстави для розробки

Наказ № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка» з зазначенням теми роботи «Дослідження та програмна реалізація методів визначення вартості довгострокового винаймання будинків» та завдання на роботу є підставами для розробки.

A.2 Призначення розробки

Програма повинна мати призначення у вигляді підтримки довгострокового винаймання будинків, оголошення про яке створюється

власником будинку, надаючи для цього інструменти пошуку оголошень, створення і визначення параметрів оголошень, створення запитів на відвідування будинку, консультування без відвідування, створення відгуків за результатами відвідування або винаймання, створення скарг на відвідувачів або власників будинків за необхідності та підтримки процесів вирішення конфліктів менеджерами.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Забезпечення функціональних характеристик повинно вкладатися в наступний набір функцій:

- пошук будинків, доступних для довгострокового винаймання;
- ознайомлення з оголошенням про можливості довгострокового винаймання будинку;
- створення власного оголошення про можливості довгострокового винаймання будинку;
- додавання споруд до оголошення;
- оновлення даних споруди, що відноситься до будинку;
- вилучення споруди з оголошення;
- оприлюднення оголошення стосовного власного будинку;
- архівування оголошення стосовно власного будинку;
- оновлення можливостей довгострокового винаймання власного будинку у власному оголошенні;
- визначення недоступного періоду для довгострокового винаймання будинку;
- визначення вартості довгострокового винаймання будинку;
- консультування користувача за власним будинком;
- відображення переліку консультацій за власним будинком;
- відображення переліку всіх консультацій;

- відображення переліку консультацій, запитаних користувачем;
- відображення переліку власних оголошень про можливості довгострокового винаймання будинку;
- створення запиту на консультацію за будинком;
- відправлення репліки в консультацію за будинком;
- перегляд консультування за будинком;
- блокування користувача;
- визначення доступного для відвідування будинку часу;
- реєстрація та авторизація в програмі;
- відправлення запиту на відвідування будинку;
- відображення переліку запитів на відвідування за власним будинком;
- відображення переліку всіх запитів на відвідування власних будинків;
- відображення переліку власних запитів на відвідування будинків;
- створення відгуку про будинок після його відвідування або винаймання;
- відображення відгуків про будинок, створених після його відвідування або винаймання;
- створення відгуку про відвідувача будинку;
- відображення інформації про користувача як відвідувача;
- відображення інформації про користувача як власника будинку;
- приймання запиту на відвідування будинку;
- скасування запиту на відвідування будинку;
- підтвердження дати відвідування будинку;
- скасування дати відвідування будинку;
- визначення виконуваного винаймання будинку;
- створення скарги на користувача, зокрема на власників будинків або на відвідувачів;

- відображення переліку створених скарг на користувачів, зокрема на власників будинків або на відвідувачів;
- перегляд створеної скарги;
- підтвердження скарги;
- надання відповіді на скаргу;
- скасування скарги;
- пошук користувачів;
- перегляд запитів на відвідування будинку за будинком або за відвідувачем менеджером;
- перегляд консультацій менеджером;
- відображення скарг на користувача;
- відображення власних скарг користувача;
- встановлення вартості винаймання будинку.

A.3.2 Умови експлуатації

Програмне забезпечення в підсумку має підтримуватись під час експлуатації через:

- організацію хостингу для роботи з сервером;
- розміщення на сервері вебзастосунку з базовим налаштуванням Django;
- наявність інтерпретатора python 3.

Або у складі цього серверу, або як окремий сервер БД має бути використано і налаштовано систему PostgreSQL.

A.3.3 Вимоги до складу та параметрів технічних засобів

Вимоги до організації технічних засобів сервера включають мінімально процесор Dual Intel Xeon, який має частоту 3,2 ГГц, 4 Гб доступної оперативної пам'яті, 10 Гб доступної пам'яті на жорсткому диску.

A.4 Порядок контролю та приймання

Строки виконання дипломної кваліфікаційної роботи магістра визначені в завданні на роботу безпосередньо, включаючи строк подання роботи, календарний план виконання роботи з зазначеними етапами, результатами та строками. захист роботи відбувається за оголошеним завчасно графіком захисту дипломних кваліфікаційних робіт і відповідно графіком роботи екзаменаційної комісії.

ДОДАТОК Б
Текст програми

```

    from .models import Cottage, CottageType, StructureType,
Country, Region, Geopoint, District, Design, CottageType,
CottageInaccessible, MonthPrice, User, Visit, VisitorReview,
CottageReview, CottageStructure, CottageInaccessible, VisitPeriod,
CottageRental
    from .forms import CottageForm, CottageStructureForm,
CottageInaccessibleForm, MonthPriceForm, VisitPeriodForm, VisitForm
    import operator
    from django.db.models import Q
    from django.shortcuts import render, get_object_or_404
    from django.urls import reverse
    from datetime import date, datetime
    from dateutil.relativedelta import relativedelta
    from django.http import HttpResponseRedirect
    from django.http.response import HttpResponseRedirect

    def seek_cottage(request):
        if request.method == 'GET' and 'cottageTypes' not in
request.GET and 'structureTypes' not in request.GET and 'country' not
in request.GET and 'region' not in request.GET and 'geopoint' not in
request.GET and 'district' not in request.GET and 'visit' not in
request.GET and 'pricemi' not in request.GET and 'pricema' not in
request.GET and 'stars' not in request.GET and 'deposit' not in
request.GET and 'furnished' not in request.GET and 'datemi' not in
request.GET and 'datema' not in request.GET and 'places' not in
request.GET and 'bedrooms' not in request.GET and 'bedtype' not in
request.GET and 'beds' not in request.GET and 'bedroomlock' not in
request.GET and 'bathrooms' not in request.GET and 'shower' not in
request.GET and 'toilets' not in request.GET and 'floors' not in
request.GET and 'gym' not in request.GET and 'closedterritory' not in
request.GET and 'parking' not in request.GET and 'patio' not in
request.GET and 'lawn' not in request.GET and 'garden' not in
request.GET and 'heating' not in request.GET and 'electricity' not in
request.GET and 'conditioner' not in request.GET and 'design' not in
request.GET and 'cabinet' not in request.GET and 'kitchen' not in
request.GET and 'kitchenware' not in request.GET and 'cellar' not in
request.GET and 'publictransport' not in request.GET and 'sizemi' not
in request.GET and 'sizema' not in request.GET and 'outsizemi' not in
request.GET and 'outsizema' not in request.GET and 'pets' not in
request.GET and 'sound' not in request.GET and 'smoking' not in
request.GET and 'washmachine' not in request.GET and 'dryer' not in
request.GET and 'dishwasher' not in request.GET and 'student' not in
request.GET and 'male' not in request.GET and 'female' not in
request.GET and 'couples' not in request.GET and 'worker' not in
request.GET and 'internet' not in request.GET and 'desk' not in
request.GET:
            cottage_types = CottageType.objects.all()
            structure_types = StructureType.objects.all()

```

```

        countries = Country.objects.all()
        regions = Region.objects.select_related('country').all()
        geopoints =
Geopoint.objects.select_related('region__country').select_related('region').all()
        districts =
District.objects.select_related('geopoint__region__country').select_related('geopoint__region').select_related('geopoint').all()
        designs = Design.objects.all()
        data_seek = {
            'cottageypes': cottage_types,
            'structuretypes': structure_types,
            'countries': countries,
            'regions': regions,
            'geopoints': geopoints,
            'districts': districts,
            'designs': designs
        }
        return render(request, "longtermcottage/search.html",
data_seek)
        cottage_types_seek = self.request.GET.get('cottageypes')
        cottages_results = Cottage.objects.filter(active = 1)
        cottages_results =
cottages_results.select_related('cottageype')
        cottages_results =
cottages_results.prefetch_related('cottagestructures')
        cottages_results =
cottages_results.select_related('district')
        cottages_results =
cottages_results.select_related('district__geopoint')
        cottages_results =
cottages_results.select_related('district__geopoint__region')
        cottages_results =
cottages_results.select_related('district__geopoint__region__country')
        cottages_results = cottages_results.select_related('design')
        cottages_results = cottages_results.select_related('owner')
        if cottage_types_seek:
            cottage_types = cottage_types_seek.split(',')
            cottage_types_list = []
            for cottage_type in cottage_types:
                cottage_types_list.append(int(cottage_type))
            cottage_types = Cottageype.objects.filter(id__in =
cottage_types_list)
            cottages_results =
cottages_results.filter(cottageype__in = cottage_types)
            structure_types_seek =
self.request.GET.get('structuretypes')
            if structure_types_seek:
                structure_types = structure_types_seek.split(',')

```



```

        pool_seek = cottages_results.filter(pool = True)
places_seek = self.request.GET.get('places')
if places_seek:
    places_min = int(places_seek)
    cottages_results = cottages_results.filter(places__gte =
places_min)
    garages_seek = self.request.GET.get('garages')
    if garages_seek:
        garages_min = int(garages_seek)
        cottages_results = cottages_results.filter(garages__gte
= garages_min)
    bedrooms_seek = self.request.GET.get('bedrooms')
    if bedrooms_seek:
        bedrooms_min = int(bedrooms_seek)
        cottages_results = cottages_results.filter(bedrooms__gte
= bedrooms_min)
    bedtype_seek = self.request.GET.get('bedtype')
    if bedtype_seek:
        bedtype_min = int(bedtype_seek)
        cottages_results = cottages_results.filter(bedtype__gte
= bedtype_min)
    beds_seek = self.request.GET.get('beds')
    if beds_seek:
        beds_min = int(beds_seek)
        cottages_results = cottages_results.filter(beds__gte =
beds_min)
    bedroomlock_seek = self.request.GET.get('bedroomlock')
    if bedroomlock_seek:
        cottages_results = cottages_results.filter(bedroomlock =
True)
    bathrooms_seek = self.request.GET.get('bathrooms')
    if bathrooms_seek:
        bathrooms_min = int(bathrooms_seek)
        cottages_results =
cottages_results.filter(bathrooms__gte = beds_min)
    shower_seek = self.request.GET.get('shower')
    if shower_seek:
        cottages_results = cottages_results.filter(shower =
True)
    toilets_seek = self.request.GET.get('toilets')
    if toilets_seek:
        toilets_min = int(toilets_seek)
        cottages_results = cottages_results.filter(toilets__gte
= toilets_min)
    floors_min_seek = self.request.GET.get('floorsmi')
    if floors_min_seek:
        floors_min = int(floors_min_seek)
        cottages_results = cottages_results.filter(floors__gte =
floors_min)

```

```

        floors_max_seek = self.request.GET.get('floorsma')
        if floors_max_seek:
            floors_max = int(floors_max_seek)
            cottages_results = cottages_results.filter(floors__lte =
floors_max)
        gym_seek = self.request.GET.get('gym')
        if gym_seek:
            cottages_results = cottages_results.filter(gym = True)
        closedterritory_seek =
self.request.GET.get('closedterritory')
        if closedterritory_seek:
            cottages_results =
cottages_results.filter(closedterritory = True)
        parking_seek = self.request.GET.get('parking')
        if parking_seek:
            cottage_parking = int(parking_seek)
            cottages_results = cottages_results.filter(parking__lte
= cottage_parking)
        patio_seek = self.request.GET.get('patio')
        if patio_seek:
            cottages_results = cottages_results.filter(patio = True)
        lawn_seek = self.request.GET.get('lawn')
        if lawn_seek:
            cottages_results = cottages_results.filter(lawn = True)
        garden_seek = self.request.GET.get('garden')
        if garden_seek:
            cottages_results = cottages_results.filter(garden =
True)
        heating_seek = self.request.GET.get('heating')
        if heating_seek:
            cottage_heating = int(heating_seek)
            cottages_results = cottages_results.filter(heating =
cottage_heating)
        electricity_seek = self.request.GET.get('electricity')
        if electricity_seek:
            cottages_results = cottages_results.filter(electricity =
True)
        conditioner_seek = self.request.GET.get('conditioner')
        if conditioner_seek:
            cottage_conditioner = int(conditioner_seek)
            cottages_results = cottages_results.filter(conditioner =
cottage_conditioner)
        design_seek = self.request.GET.get('design')
        if design_seek:
            cottage_design = Design.objects.get(id =
int(design_seek))
            cottages_results = cottages_results.filter(design =
cottage_design)
        cabinet_seek = self.request.GET.get('cabinet')

```

```

        if cabinet_seek:
            cottages_results = cottages_results.filter(cabinet =
True)
        kitchen_seek = self.request.GET.get('kitchen')
        if kitchen_seek:
            cottage_kitchen = int(kitchen_seek)
            cottages_results = cottages_results.filter(kitchen =
cottage_kitchen)
        kitchenware_seek = self.request.GET.get('kitchenware')
        if kitchenware_seek:
            cottages_results = cottages_results.filter(kitchenware =
True)
        cellar_seek = self.request.GET.get('cellar')
        if cellar_seek:
            cottages_results = cottages_results.filter(cellar =
True)
        publictransport_seek =
self.request.GET.get('publictransport')
        if publictransport_seek:
            cottages_results =
cottages_results.filter(publictransport = True)
        size_min_seek = self.request.GET.get('sizemi')
        if size_min_seek:
            size_min = float(size_min_seek)
            cottages_results = cottages_results.filter(size__gte =
size_min)
        size_max_seek = self.request.GET.get('sizema')
        if size_max_seek:
            size_max = float(size_max_seek)
            cottages_results = cottages_results.filter(size__lte =
size_max)
        outsize_min_seek = self.request.GET.get('outsizemi')
        if outsize_min_seek:
            outsize_min = float(outsize_min_seek)
            cottages_results = cottages_results.filter(size__gte =
outsize_min)
        outsize_max_seek = self.request.GET.get('outsizema')
        if outsize_max_seek:
            outsize_max = float(outsize_max_seek)
            cottages_results = cottages_results.filter(outsize__lte =
outsize_max)
        pets_seek = self.request.GET.get('pets')
        if pets_seek:
            cottages_results = cottages_results.filter(pets = True)
        sound_seek = self.request.GET.get('sound')
        if sound_seek:
            cottages_results = cottages_results.filter(sound = True)
        smoking_seek = self.request.GET.get('smoking')
        if smoking_seek:

```

```

        cottages_results = cottages_results.filter(smoking =
True)
        washingmachine_seek = self.request.GET.get('washmachine')
        if washingmachine_seek:
            cottages_results =
cottages_results.filter(washingmachine = True)
        dryer_seek = self.request.GET.get('dryer')
        if dryer_seek:
            cottages_results = cottages_results.filter(dryer = True)
        dishwasher_seek = self.request.GET.get('dishwasher')
        if dishwasher_seek:
            cottages_results = cottages_results.filter(dishwasher =
True)
        forstudent_seek = self.request.GET.get('student')
        if forstudent_seek:
            cottages_results = cottages_results.filter(forstudent =
True)
        formale_seek = self.request.GET.get('male')
        if formale_seek:
            cottages_results = cottages_results.filter(formale =
True)
        forfemale_seek = self.request.GET.get('female')
        if forfemale_seek:
            cottages_results = cottages_results.filter(forfemale =
True)
        couples_seek = self.request.GET.get('couples')
        if couples_seek:
            cottages_results = cottages_results.filter forcouples =
True)
        forworker_seek = self.request.GET.get('worker')
        if forworker_seek:
            cottages_results = cottages_results.filter(forworker =
True)
        internet_seek = self.request.GET.get('internet')
        if internet_seek:
            cottage_internet = int(internet_seek)
            cottages_results = cottages_results.filter(internet =
cottage_internet)
        desk_seek = self.request.GET.get('desk')
        if desk_seek:
            cottages_results = cottages_results.filter(desk = True)
        withbills_seek = self.request.GET.get('bills')
        if withbills_seek:
            cottages_results = cottages_results.filter(withbills =
True)
        cottage_rental_start_seek = self.request.GET.get('datemi')
        cottage_rental_end_seek = self.request.GET.get('datema')
        if not cottage_rental_start_seek:
            cottage_rental_start_seek = datetime.now().date()

```

```

        else:
            cottage_rental_start =
cottage_rental_start_seek.split('-')
            cottage_rental_start_seek =
date(int(cottage_rental_start[0]), int(cottage_rental_start[1]),
int(cottage_rental_start[2]))
            if not cottage_rental_end_seek:
                cottage_rental_end_seek = cottage_rental_start_seek +
relativedelta(months = 1)
            else:
                cottage_rental_end = cottage_rental_end_seek.split('-')
                cottage_rental_end_seek =
date(int(cottage_rental_end[0]), int(cottage_rental_end[1]),
int(cottage_rental_end[2]))
                cottages = []
                for cottages_result in cottages_results:
                    cottage_dates_begin =
Cottageinaccessible.objects.filter(cottage = cottages_result,
beginperiod__gte = cottage_rental_start_seek, beginperiod__lte =
cottage_rental_end_seek)
                    cottage_dates_end =
Cottageinaccessible.objects.filter(cottage = cottages_result,
endperiod__gte = cottage_rental_start_seek, endperiod__lte =
cottage_rental_end_seek)
                    if not cottage_dates_begin and not cottage_dates_end:
                        cottages.append(cottages_result.id)
                datedelta =
relativedelta.relativedelta(cottage_rental_end_seek,
cottage_rental_start_seek)
                months = datedelta.months
                if cottage_rental_end_seek.day !=
cottage_rental_start_seek.day:
                    months = months + 1
                cottages_results = cottages_results.filter(id__in =
cottages)
                price_min_seek = self.request.GET.get('pricemi')
                price_max_seek = self.request.GET.get('pricema')
                final_cottages = []
                for cottages_result in cottages_results:
                    nearest_price = Monthprice.objects.filter(cottage =
cottages_result, beginperiod__lte = cottage_dates_begin).order_by('-
beginperiod').values()
                    nearest_date = nearest_price[0].beginperiod
                    nearest_price = Monthprice.objects.filter(cottage =
cottages_result, beginperiod = nearest_date, months__lte =
months).order_by('-months').values()
                    cottages_result.price = nearest_price[0].price * months
                    if cottages_result.price >= price_min_seek and
cottages_result.price <= price_max_seek:

```

```

        final_cottages.append(cottages_result)
        cottage_order_seek = self.request.GET.get('order')
        cottage_order_seek = int(cottage_order_seek)
        if cottage_order_seek == 1:
            final_cottages.sort(reverse = True, key = lambda x:
x.price)
        elif cottage_order_seek == 2:
            final_cottages.sort(reverse = True, key = lambda x:
x.stars)
        elif cottage_order_seek == 3:
            final_cottages.sort(reverse = True, key = lambda x:
x.size)
        return render(request, "longtermcottage/search.html",
final_cottages)

def get_redirect(redirect_view, *args, param = None, **kwargs):
    name = reverse(redirect_view, *args, **kwargs)
    if param:
        full_name = '{}?{}'.format(name, param.urlencode())
    return HttpResponseRedirect(full_name)

def show_cottage(request):
    cottageid = self.request.GET.get('cottage')
    cottageid = int(cottageid)
    cottage = Cottage.objects.select_related('cottage_type')
    cottage = cottage.prefetch_related('cottage_structures')
    cottage = cottage.prefetch_related('cottage_reviews')
    cottage = cottage.select_related('district')
    cottage = cottage.select_related('owner')
    cottage = cottage.select_related('design')
    cottage = cottage.select_related('district_geopoint')
    cottage =
cottage.select_related('district_geopoint_region')
    cottage =
cottage.select_related('district_geopoint_region_country')
    cottage = cottage.prefetch_related('cottage_photos')
    cottage = cottage.get(pk = cottageid)
    if cottage.active != 1 and request.user:
        user_in = User.objects.get(djangouser = request.user)
        if cottage.owner != user_in:
            return render(request,
"longtermcottage/cottage.html", {'cottage': None})
    if request.method == 'GET' and 'datemi' not in request.GET
and 'datema' not in request.GET:
        return render(request, "longtermcottage/cottage.html",
{'cottage': cottage})
    cottage_rental_start_seek = self.request.GET.get('datemi')
    cottage_rental_end_seek = self.request.GET.get('datema')
    if not cottage_rental_start_seek:

```

```

        cottage_rental_start_seek = datetime.now().date()
    else:
        cottage_rental_start =
cottage_rental_start_seek.split('-')
        cottage_rental_start_seek =
date(int(cottage_rental_start[0]), int(cottage_rental_start[1]),
int(cottage_rental_start[2]))
        if not cottage_rental_end_seek:
            cottage_rental_end_seek = cottage_rental_start_seek +
relativedelta(months = 1)
        else:
            cottage_rental_end = cottage_rental_end_seek.split('-')
            cottage_rental_end_seek =
date(int(cottage_rental_end[0]), int(cottage_rental_end[1]),
int(cottage_rental_end[2]))
            accessible = True
            cottage_dates_begin =
Cottageinaccessible.objects.filter(cottage = cottage, beginperiod__gte
= cottage_rental_start_seek, beginperiod__lte =
cottage_rental_end_seek)
            cottage_dates_end =
Cottageinaccessible.objects.filter(cottage = cottage, endperiod__gte =
cottage_rental_start_seek, endperiod__lte = cottage_rental_end_seek)
            if cottage_dates_begin or cottage_dates_end:
                accessible = False
            datedelta =
relativedelta.relativedelta(cottage_rental_end_seek,
cottage_rental_start_seek)
            months = datedelta.months
            if cottage_rental_end_seek.day !=
cottage_rental_start_seek.day:
                months = months + 1
            nearest_price = Monthprice.objects.filter(cottage = cottage,
beginperiod__lte = cottage_dates_begin).order_by('-
beginperiod').values()
            nearest_date = nearest_price[0].beginperiod
            nearest_price = Monthprice.objects.filter(cottage = cottage,
beginperiod = nearest_date, months__lte = months).order_by('-
months').values()
            cottage.price = nearest_price[0].price * months
            return render(request, "longtermcottage/cottage.html",
{'cottage': cottage})

def publish_cottage(request, cottageid):
    cottage = Cottage.objects.get(pk = cottageid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if cottage.owner == user_in:
            cottage.active = 1

```

```

        cottage.save()
        return get_redirect ("cottage-show", param =
cottage.id)

def archive_cottage(request, cottageid):
    cottage = Cottage.objects.get(pk = cottageid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if cottage.owner == user_in:
            cottage.active = 2
            cottage.save()
            return get_redirect ("cottage-show", param =
cottage.id)

def visit_date_confirm(request, visitid):
    visit = Visit.objects.select_related('cottage')
    visit = visit.select_related('cottage__owner')
    visit = visit.get(pk = visitid)
    if request.user:
        inside = Visitperiod.objects.filter(cottage =
visit.cottage, beginperiod__lte = visit.beginperiod, endperiod__gte =
visit.endperiod).first()
        user_in = User.objects.get(djangouser = request.user)
        if inside and visit.cottage.owner == user_in and
(visit.state == 1 or visit.state == 4):
            visit.state = 3
            visit.save()
            return get_redirect ("visit-show", param = visit.id)

def visit_date_cancel(request, visitid):
    visit = Visit.objects.select_related('cottage')
    visit = visit.select_related('cottage__owner')
    visit = visit.get(pk = visitid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if visit.cottage.owner == user_in and (visit.state == 1
or visit.state == 3):
            visit.state = 4
            visit.save()
            return get_redirect ("visit-show", param = visit.id)

def visit_confirm(request, visitid):
    visit = Visit.objects.select_related('cottage')
    visit = visit.select_related('cottage__owner')
    visit = visit.get(pk = visitid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if visit.cottage.owner == user_in and (visit.state == 0
or visit.state == 2):

```

```

        visit.state = 1
        visit.save()
        return get_redirect ("visit-show", param = visit.id)

def visit_cancel(request, visitid):
    visit = Visit.objects.select_related('cottage')
    visit = visit.select_related('cottage__owner')
    visit = visit.get(pk = visitid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if visit.cottage.owner == user_in and (visit.state == 0
or visit.state == 1) or ((visit.user == user_in or visit.cottage.owner
== user_in) and (visit.state == 3 or visit.state == 4)):
            visit.state = 2
            visit.save()
            return get_redirect ("visit-show", param = visit.id)

def visit_done(request, visitid):
    visit = Visit.objects.select_related('cottage')
    visit = visit.select_related('cottage__owner')
    visit = visit.get(pk = visitid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if visit.cottage.owner == user_in and (visit.state == 3
or visit.state == 7):
            visit.state = 5
            visit.save()
            return get_redirect ("visit-show", param = visit.id)
        if visit.user == user_in and visit.state == 3:
            visit.state = 7
            visit.save()
            return get_redirect ("visit-show", param = visit.id)

def visit_notdone(request, visitid):
    visit = Visit.objects.select_related('cottage')
    visit = visit.select_related('cottage__owner')
    visit = visit.get(pk = visitid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if visit.cottage.owner == user_in and (visit.state == 3
or visit.state == 8):
            visit.state = 6
            visit.save()
            return get_redirect ("visit-show", param = visit.id)
        if visit.user == user_in and visit.state == 3:
            visit.state = 8
            visit.save()
            return get_redirect ("visit-show", param = visit.id)

```

```

def visit_success(request, visitid):
    visit = Visit.objects.select_related('cottage')
    visit = visit.select_related('cottage__owner')
    visit = visit.get(pk = visitid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        if visit.cottage.owner == user_in and visit.state == 5:
            visit.state = 9
            visit.save()
            return get_redirect ("visit-show", param = visit.id)

def user_visits(request, visitstate, order = 1):
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        visits = Visit.objects.select_related('cottage')
        visits = visits.select_related('user')
        visits = visits.select_related('cottage__owner')
        visits = visits.select_related('cottage__cottage_type')
        visits = visits.select_related('cottage__district')
        visits =
visits.select_related('cottage__district__geopoint')
        visits =
visits.select_related('cottage__district__geopoint__region')
        visits =
visits.select_related('cottage__district__geopoint__region__country')
        if visitstate != 0:
            visits = visits.filter(state = visitstate)
        visits = visits.filter(user = user_in)
        if order:
            visits = visits.order_by('-addtime')
        else:
            visits = visits.order_by('-visittime')
        return render(request,
"longtermcottage/user_visits.html", {'visits': visits })

def owner_visits(request, visitstate, order = 1):
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        visits = Visit.objects.select_related('cottage')
        visits = visits.select_related('user')
        visits = visits.select_related('cottage__owner')
        visits = visits.select_related('cottage__cottage_type')
        visits = visits.select_related('cottage__district')
        visits =
visits.select_related('cottage__district__geopoint')
        visits =
visits.select_related('cottage__district__geopoint__region')
        visits =
visits.select_related('cottage__district__geopoint__region__country')

```

```

        cottage_rentals =
Cottagerental.objects.select_related('cottage')
        cottage_rentals =
cottage_rentals.select_related('client')
        cottage_rentals =
cottage_rentals.select_related('cottage__owner')
        cottage_rentals =
cottage_rentals.select_related('cottage__cottage_type')
        cottage_rentals =
cottage_rentals.select_related('cottage__district')
        cottage_rentals =
cottage_rentals.select_related('cottage__district__geopoint')
        cottage_rentals =
cottage_rentals.select_related('cottage__district__geopoint__region')
        cottage_rentals =
cottage_rentals.select_related('cottage__district__geopoint__region__c
ountry')
        cottage_rentals = cottage_rentals.filter(cottage__owner
= user_in)
        cottage_rentals = cottage_rentals.order_by('-
beginperiod')
        if visitstate != 0:
            visits = visits.filter(state = visitstate)
            visits = visits.filter(cottage__owner = user_in)
            if order:
                visits = visits.order_by('-addtime')
            else:
                visits = visits.order_by('-visittime')
        return render(request,
"longtermcottage/owner_visits.html", {'visits': visits, 'rentals':
cottage_rentals })

def cottage_visits(request, cottageid, visitstate, order = 1):
    cottage = Cottage.objects.select_related('owner').get(pk =
cottageid)
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        visits = Visit.objects.select_related('cottage')
        visits = visits.select_related('user')
        visits = visits.select_related('cottage__owner')
        visits = visits.select_related('cottage__cottage_type')
        visits = visits.select_related('cottage__district')
        visits =
visits.select_related('cottage__district__geopoint')
        visits =
visits.select_related('cottage__district__geopoint__region')
        visits =
visits.select_related('cottage__district__geopoint__region__country')
        if visitstate != 0:

```

```

        visits = visits.filter(state = visitstate)
        visits = visits.filter(cottage__owner = user_in, cottage
= cottage)
        if order:
            visits = visits.order_by('-addtime')
        else:
            visits = visits.order_by('-visittime')
        cottage_rentals =
Cottagerental.objects.select_related('cottage')
        cottage_rentals =
cottage_rentals.select_related('client')
        cottage_rentals =
cottage_rentals.select_related('cottage__owner')
        cottage_rentals =
cottage_rentals.select_related('cottage__cottage_type')
        cottage_rentals =
cottage_rentals.select_related('cottage__district')
        cottage_rentals =
cottage_rentals.select_related('cottage__district__geopoint')
        cottage_rentals =
cottage_rentals.select_related('cottage__district__geopoint__region')
        cottage_rentals =
cottage_rentals.select_related('cottage__district__geopoint__region__c
ountry')
        cottage_rentals = cottage_rentals.filter(cottage =
cottage
        cottage_rentals = cottage_rentals.order_by('-
beginperiod')
        return render(request,
"longtermcottage/cottage_visits.html", {'visits': visits, 'rentals':
cottage_rentals })

def owner_cottages(request):
    if request.user:
        user_in = User.objects.get(djangouser = request.user)
        cottages = Cottage.objects.select_related('owner')
        cottages = cottages.select_related('cottage_type')
        cottages = cottages.select_related('district')
        cottages = cottages.select_related('district__geopoint')
        cottages =
cottage_rentals.select_related('district__geopoint__region')
        cottages =
cottage_rentals.select_related('district__geopoint__region__country')
        cottages = cottages.filter(owner = user_in)
        cottages = cottages.order_by('-addtime')
        return render(request,
"longtermcottage/owner_cottages.html", {'cottages': cottages })

```

```

def visitor_profile(request, visitorusername):
    if request.user:
        visitor = User.objects.get(djangouser__username =
visitorusername)
        visitor_reviews =
Visitorreview.objects.select_related('owner')
        visitor_reviews = visitor_reviews.filter(visitor =
visitor)
        visitor_reviews = visitor_reviews.order_by('-addtime')
        return render(request, "longtermcottage/visitor.html",
{'visitor': visitor, 'reviews': visitor_reviews })

def owner_profile(request, ownerusername):
    if request.user:
        owner = User.objects.get(djangouser__username =
visitorusername)
        cottages = Cottage.objects.filter(owner =
owner).order_by('-stars').values()[:3]
        owner_cottages_reviews =
Cottagereview.objects.select_related('cottage')
        owner_cottages_reviews =
owner_cottages_reviews.select_related('user')
        owner_cottages_reviews =
owner_cottages_reviews.filter(cottage__owner = owner)
        owner_cottages_reviews =
owner_cottages_reviews.order_by('-addtime')
        owner_cottages_reviews =
owner_cottages_reviews.values()[:10]
        return render(request, "longtermcottage/owner.html",
{'owner': owner, 'cottages': cottages, 'reviews':
owner_cottages_reviews })

def cottage_create(request):
    if request.user:
        owner = User.objects.get(djangouser = request.user)
        if request.method == "POST":
            cottageform = CottageForm(request.POST)
            if cottageform.is_valid():
                cottage = cottageform.save(commit = False)
                cottage.owner = owner
                cottage.save()
                return get_redirect ("cottage-show", param =
cottage.id)
            else:
                cottageform = None
        return render(request,
"longtermcottage/cottage_create.html", {"form": cottageform })
    else:

```

```

        return render(request,
"longtermcottage/cottage_create.html", {"form": None })

    def cottage_edit(request, cottageid):
        if request.user:
            cottage = Cottage.objects.select_related('cottage_type')
            cottage = cottage.prefetch_related('cottage_structures')
            cottage = cottage.select_related('district')
            cottage = cottage.select_related('owner')
            cottage = cottage.select_related('design')
            cottage = cottage.select_related('district__geopoint')
            cottage =
cottage.select_related('district__geopoint__region')
            cottage =
cottage.select_related('district__geopoint__region__country')
            cottage = cottage.prefetch_related('cottage_photos')
            cottage = cottage.get(pk = cottageid)
            owner = User.objects.get(djangouser = request.user)
            if request.method == "POST" and cottage.owner == owner:
                cottageform = CottageForm(request.POST or None,
instance = cottage)
                if cottageform.is_valid():
                    cottageform.save()
                    return get_redirect("cottage-show", param =
cottage.id)
                else:
                    cottageform = None
                    return render(request,
"longtermcottage/cottage_edit.html", {"form": cottageform })

    def cottage_structure_create(request, cottageid):
        if request.user:
            cottage = Cottage.objects.get(pk = cottageid)
            owner = User.objects.get(djangouser = request.user)
            if request.method == "POST" and cottage.owner == owner:
                cottagestructureform =
CottageStructureForm(request.POST)
                if cottagestructureform.is_valid():
                    cottagestructure =
cottagestructureform.save(commit = False)
                    cottagestructure.cottage = cottage
                    cottagestructure.save()
                    return get_redirect ("cottage-show", param =
cottage.id)
                else:
                    cottagestructureform = None
                    return render(request,
"longtermcottage/cottage_struct_create.html", {"form":
cottagestructureform })

```

```

        else:
            return render(request,
                "longtermcottage/cottage_struct_create.html", {"form": None })

    def cottage_structure_edit(request, cottagestructureid):
        if request.user:
            cottagestructure =
Cottagestructure.objects.select_related('cottage')
            cottagestructure =
cottagestructure.select_related('cottage__owner')
            cottagestructure = cottagestructure.get(pk =
cottagestructureid)
            owner = User.objects.get(djangouser = request.user)
            if request.method == "POST" and
cottagestructure.cottage.owner == owner:
                cottagestructureform =
CottageStructureForm(request.POST or None, instance =
cottagestructure)
                if cottagestructureform.is_valid():
                    cottagestructureform.save()
                    return get_redirect("cottage-show", param =
cottagestructure.cottage.id)
                else:
                    cottagestructureform = None
            return render(request,
                "longtermcottage/cottage_struct_edit.html", {"form":
cottagestructureform })

    def cottage_structure_delete(request, cottagestructureid):
        if request.user:
            cottagestructure =
Cottagestructure.objects.select_related('cottage')
            cottagestructure =
cottagestructure.select_related('cottage__owner')
            cottagestructure = cottagestructure.get(pk =
cottagestructureid)
            owner = User.objects.get(djangouser = request.user)
            if cottagestructure.cottage.owner == owner:
                cottage = cottagestructure.cottage
                cottagestructure.delete()
                return get_redirect ("cottage-show", param =
cottage.id)

    def cottageinaccessible_create(request, cottageid):
        if request.user:
            cottage = Cottage.objects.get(pk = cottageid)
            owner = User.objects.get(djangouser = request.user)
            if request.method == "POST" and cottage.owner == owner:

```

```

        cottageinaccessibleform =
CottageinaccessibleForm(request.POST)
        if cottageinaccessibleform.is_valid():
            cottageinaccessible=
cottageinaccessibleform.save(commit = False)
            cottageinaccessible.cottage = cottage
            cottageinaccessible.save()
            return get_redirect ("inaccessibles-show", param
= cottage.id)
        else:
            cottageinaccessibleform = None
            return render(request,
"longtermcottage/cottage_inaccessible_create.html", {"form":
cottageinaccessibleform })

def cottageinaccessible_delete(request, cottageinaccessibleid):
    if request.user:
        cottageinaccessible =
Cottageinaccessible.objects.select_related('cottage')
        cottageinaccessible =
cottageinaccessible.select_related('cottage__owner')
        cottageinaccessible = cottageinaccessible.get(pk =
cottageinaccessibleid)
        owner = User.objects.get(djangouser = request.user)
        if cottageinaccessible.cottage.owner == owner:
            cottage = cottageinaccessible.cottage
            cottageinaccessible.delete()
            return get_redirect ("inaccessibles-show", param =
cottage.id)

def cottage_inaccessibles(request, cottageid):
    if request.user:
        cottage = Cottage.objects.get(pk = cottageid)
        owner = User.objects.get(djangouser = request.user)
        if cottage.owner == owner:
            cottageinaccessibles =
Cottageinaccessible.objects.filter(cottage = cottage, beginperiod__gte
= datetime.now().date())
            cottageinaccessibles =
cottageinaccessibles.order_by('beginperiod')
            return render(request,
"longtermcottage/cottage_inaccessibles.html", {"dates":
cottageinaccessibles })

def cottage_price_create(request, cottageid):
    if request.user:
        cottage = Cottage.objects.get(pk = cottageid)
        owner = User.objects.get(djangouser = request.user)
        if request.method == "POST" and cottage.owner == owner:

```

```

        monthprice_form = MonthpriceForm(request.POST)
        if monthprice_form.is_valid():
            monthprice = monthprice_form.save(commit =
False)

            monthprice.cottage = cottage
            monthprice.save()
            return get_redirect ("cottage-prices-show",
param = cottage.id)
        else:
            monthprice_form = None
            return render(request,
"longtermcottage/cottage_price_create.html", {"form": monthprice_form
})

def cottage_price_edit(request, cottagepriceid):
    if request.user:
        cottage_price =
Monthprice.objects.select_related('cottage')
        cottage_price =
cottage_price.select_related('cottage__owner')
        cottage_price = cottage_price.get(pk =
cottagestructureid)
        owner = User.objects.get(djangouser = request.user)
        if request.method == "POST" and
cottage_price.cottage.owner == owner:
            monthprice_form = MonthpriceForm (request.POST or
None, instance = cottagestructure)
            if monthprice_form.is_valid():
                cottage_price.price = monthprice_form.price
                cottage_price.save()
                return get_redirect("cottage-prices-show", param
= cottage_price.cottage.id)
            else:
                monthprice_form = None
                return render(request,
"longtermcottage/cottage_price_edit.html", {"form": monthprice_form })

def cottage_price_delete(request, cottagepriceid):
    if request.user:
        cottage_price =
Monthprice.objects.select_related('cottage')
        cottage_price =
cottage_price.select_related('cottage__owner')
        cottage_price = cottage_price.get(pk = cottagepriceid)
        owner = User.objects.get(djangouser = request.user)
        if cottage_price.cottage.owner == owner:
            cottage = cottage_price.cottage
            cottage_price.delete()

```

```

        return get_redirect ("cottage-prices-show", param =
cottage.id)

def cottage_prices(request, cottageid):
    if request.user:
        cottage = Cottage.objects.get(pk = cottageid)
        owner = User.objects.get(djangouser = request.user)
        if cottage.owner == owner:
            month_prices = Monthprice.objects.filter(cottage =
cottage, beginperiod__gte = datetime.now().date())
            month_prices = month_prices.order_by('beginperiod')
            return render(request,
"longtermcottage/cottage_prices.html", {"prices": month_prices })

def visit_period_create(request, cottageid):
    if request.user:
        cottage = Cottage.objects.get(pk = cottageid)
        owner = User.objects.get(djangouser = request.user)
        if request.method == "POST" and cottage.owner == owner:
            visitperiod_form = VisitperiodForm(request.POST)
            if visitperiod_form.is_valid():
                visitperiod = visitperiod_form.save(commit =
False)

                visitperiod.cottage = cottage
                visitperiod.save()
                return get_redirect ("visit-periods-show", param
= cottage.id)
            else:
                visitperiod_form = None
                return render(request,
"longtermcottage/visit_period_create.html", {"form": visitperiod_form
})

def visit_period_edit(request, visitperiodid):
    if request.user:
        visit_period =
Visitperiod.objects.select_related('cottage')
        visit_period =
visit_period.select_related('cottage__owner')
        visit_period = visit_period.get(pk = visitperiodid)
        owner = User.objects.get(djangouser = request.user)
        if request.method == "POST" and
visit_period.cottage.owner == owner:
            visitperiod_form = VisitperiodForm (request.POST or
None, instance = visit_period)
            if visitperiod_form.is_valid():
                if visitperiod_form.beginperiod ==
visit_period.beginperiod and visitperiod_form.endperiod ==
visit_period.endperiod:

```

```

        visitperiod_form.save()
    else:
        visit_periods =
Visitperiod.objects.filter(cottage = visit_period.cottage,
beginperiod__gte = visit_period.beginperiod, endperiod__lte =
visit_period.endperiod).exclude(id = visit_period.id)
        if visit_periods:
            visitperiod_form.save()
            for visitperiod in visit_periods:
                visitperiod.delete()
        else:
            visit_periods =
Visitperiod.objects.filter(cottage = visit_period.cottage,
beginperiod__lte = visit_period.beginperiod, endperiod__gte =
visit_period.endperiod).exclude(id = visit_period.id)
            if visit_periods:
                visit_period.delete()
            else:
                visit_periods =
Visitperiod.objects.filter(cottage = visit_period.cottage,
beginperiod__lte = visit_period.beginperiod, endperiod__lte =
visit_period.endperiod, endperiod__gte = visit_period.beginperiod)
                if visit_periods:
                    period = visit_periods.first()
                    period.endperiod =
visitperiod_form.endperiod
                    visit_period.delete()
                else:
                    visit_periods =
Visitperiod.objects.filter(cottage = visit_period.cottage,
beginperiod__gte = visit_period.beginperiod, endperiod__gte =
visit_period.endperiod, beginperiod__lte = visit_period.beginperiod)
                    if visit_periods:
                        period =
visit_periods.first()
                        period.beginperiod =
visitperiod_form.beginperiod
                        visit_period.delete()
                    else:
                        visit_period.save()
            return get_redirect ("visit-periods-show", param
= visit_period.cottage.id)
        else:
            visitperiod_form = None
            return render(request,
"longtermcottage/visit_period_edit.html", {"form": visitperiod_form })

def visit_period_delete(request, visitperiodid):
    if request.user:

```

```

        visit_period =
Visitperiod.objects.select_related('cottage')
        visit_period =
visit_period.select_related('cottage__owner')
        visit_period = visit_period.get(pk = visitperiodid)
        owner = User.objects.get(djangouser = request.user)
        if visit_period.cottage.owner == owner:
            cottage = cottage_price.cottage
            visit_period.delete()
            return get_redirect ("visit-periods-show", param =
visit_period.cottage.id)

def cottage_visitperiods(request, cottageid):
    if request.user:
        cottage = Cottage.objects.get(pk = cottageid)
        owner = User.objects.get(djangouser = request.user)
        if cottage.owner == owner:
            visit_periods = Visitperiod.objects.filter(cottage =
cottage, beginperiod__gte = datetime.now().date())
            visit_periods =
visit_periods.order_by('beginperiod')
            return render(request,
"longtermcottage/visit_periods.html", {"periods": visit_periods })

def cottage_rental_create(request, cottageid, visitorusername =
None):
    if request.user:
        cottage = Cottage.objects.get(pk = cottageid)
        visitor = User.objects.get(djangouser = request.user)
        if request.method == "POST" and cottage.owner == owner:
            cottagerental_form = CottagerentalForm(request.POST)
            if cottagerental_form.is_valid():
                cottagerental = cottagerental_form.save(commit =
False)

                if cottagerental.beginperiod > datetime.now():
                    cottage.active = 2
                    cottage.save()
                visits = Visit.objects.filter(cottage = cottage,
state__in = [0, 1, 3, 4])
                for visit in visits:
                    visit.state = 2
                    visit.save()
                cottagerental.cottage = cottage
                if visitorusername:
                    cottagerental.client = visitor
                cottagerental.save()
                return get_redirect ("visit-periods-show", param
= cottage.id)
            else:

```

```

        cottagerental_form = None
        return render(request,
"longtermcottage/rental_create.html", {"form": cottagerental_form })

    def cottage_visit_create(request, cottageid):
        cottage = Cottage.objects.get(pk =
cottageid).select_related("owner").prefetch_related("owner__blocks")
        visit_periods = Visitperiod.objects.filter(cottage =
cottage, beginperiod__gte = datetime.now().date())
        visit_periods = visit_periods.order_by('beginperiod')
        error = False
        if request.user:
            visitor = User.objects.get(djangouser = request.user)
            if request.method == "POST" and cottage.active == 1 and
(visitor not in cottage.owner.blocks):
                visit_form = VisitForm(request.POST)
                if visit_form.is_valid():
                    inside = Visitperiod.objects.filter(cottage =
cottage, beginperiod__lte = visit_form.beginperiod, endperiod__gte =
visit_form.endperiod).first()
                    if inside and visit_form.duration <=
inside.duration:
                        visit = visit_form.save(commit = False)
                        visit.cottage = cottage
                        visit.user = visitor
                        visit.save()
                        return get_redirect ("visit-periods-show",
param = cottage.id)
                    else:
                        error = True
                else:
                    visit_form = None
            return render(request,
"longtermcottage/visit_create.html", {"form": visit_form, "periods":
visit_periods, "error": error })

    def visit_date_set(request, visitid):
        visit = Visit.objects.select_related('cottage')
        visit = visit.select_related('cottage__owner')
        visit = visit.get(pk = visitid)
        if request.user:
            inside = Visitperiod.objects.filter(cottage =
visit.cottage, beginperiod__lte = visit.beginperiod, endperiod__gte =
visit.endperiod).first()
            user_in = User.objects.get(djangouser = request.user)
            if inside and visit.user == user_in and (visit.state ==
1 or visit.state == 4):
                visit.save()
                return get_redirect ("visit-show", param = visit.id)

```

ДОДАТОК В
Слайди презентації

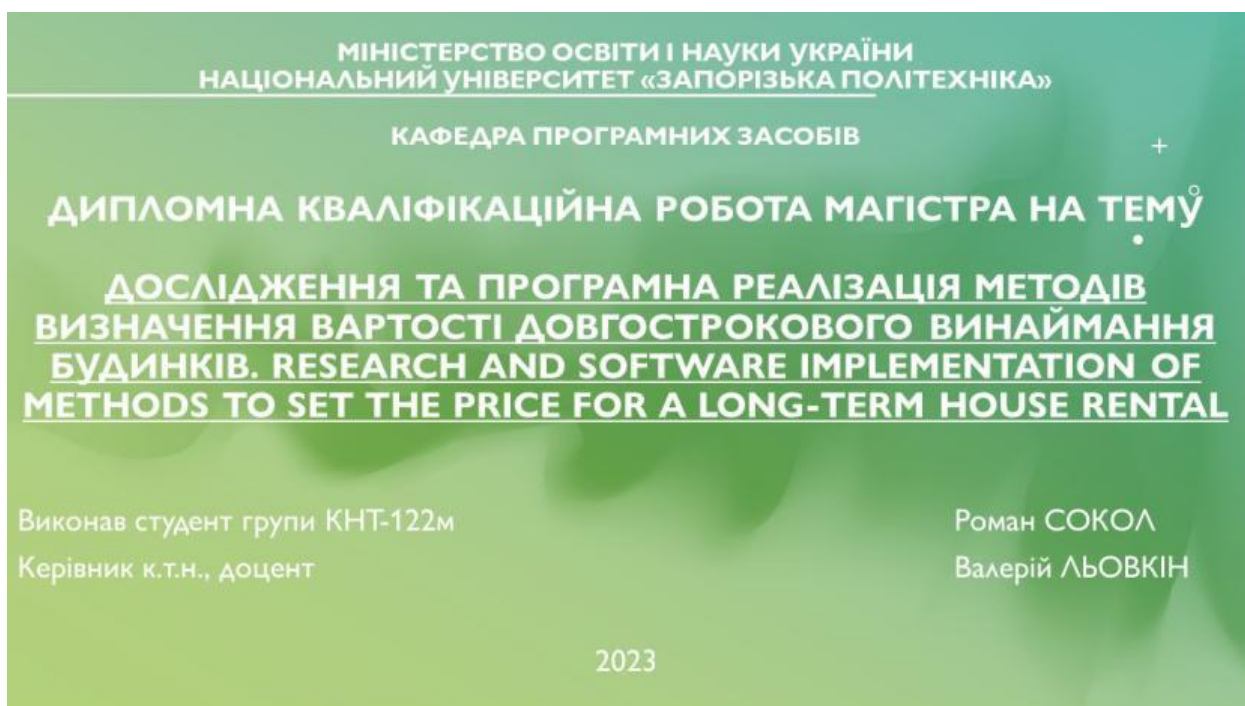


Рисунок В.1 – Слайд № 1 (Титульний слайд)

Об'єкт, предмет, мета роботи

- об'єкт роботи – процес довгострокового винаймання будинків;
- предмет роботи – методи визначення вартості довгострокового винаймання будинків;
- мета роботи – розробити програмне забезпечення довгострокового винаймання будинків для надання інформації для обґрунтованого прийняття рішень та підвищення взаємодії між власниками будинків та винаймачами.

2

Рисунок В.2 – Слайд № 2 (Об'єкт, предмет, мета роботи)

Завдання роботи

- проаналізувати аналоги програм підтримки довгострокового винаймання будинків;
- розробити метод визначення вартості довгострокового винаймання будинків;
- виконати проєктування програмного забезпечення підтримки довгострокового винаймання будинків;
- реалізувати програмне забезпечення підтримки довгострокового винаймання будинків.

3

Рисунок В.3 – Слайд № 3 (Завдання роботи)

Порівняння програмного забезпечення підтримки довгострокового винаймання будинків

Позиція порівняння	HousingAnywhere	Rightmove	Olx	Розробка
Пошук будинків на довгий строк	+	+	+	+
Фільтрування за типом будинку	+	–	+	+
Фільтрування за площею ділянки	–	–	+	+
Запит на відвідування будинку	–	+	–	+
Опис споруд на території будинку	–	–	–	+
Відгуки та оцінювання будинків	Відгуки про користувачів	–	–	Відгуки про будинки та користувачів
Консультування за будинком	+	+	–	+
Визначення вартості довгострокового винаймання будинків	–	–	–	+

4

Рисунок В.4 – Слайд № 4 (Порівняння програмного забезпечення підтримки довгострокового винаймання будинків)

Метод визначення вартості довгострокового винаймання будинків

- для реалізації методу потрібно створити вибірку даних, яка характеризує вартість винаймання будинків на певний момент з відповідним рівнем впливу тимчасових факторів для певного міста на основі характеристик:
 - вартість винаймання будинку на місяць;
 - загальні додаткові витрати;
 - кількість ванних кімнат;
 - кількість спальних кімнат;
 - кількість гаражів;
 - наявність басейну;
 - загальна площа власності;
 - площа земельної ділянки;
 - тип системи опалення;
- відкидання екземплярів з нехарактерною, нульовою вартістю, дублікатів;
- після нормування даних вибірки вона використовується для навчання моделі, яка має організовуватись на основі виділення вартості винаймання будинку на місяць як вихідної ознаки, а всіх інших – як вхідних: модель створюється на основі дерев рішень за допомогою методу Random Forest, після створення моделі та її навчання кожен новий екземпляр має формуватися на основі нормованих значень вхідних ознак.

5

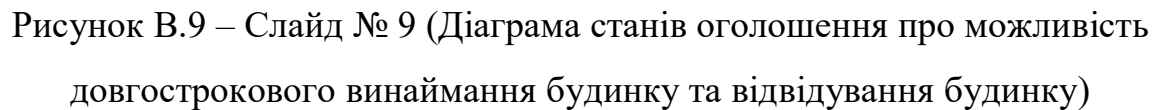
Рисунок В.5 – Слайд № 5 (Метод визначення вартості довгострокового винаймання будинків)

Вибір рішення для веброзробки

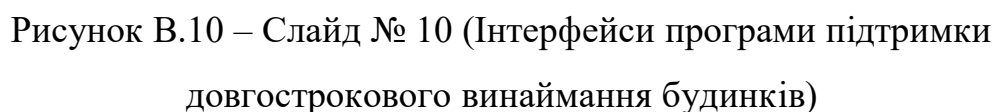
Позиція порівняння	Drupal	Django
Призначення	Типові рішення	Рішення з особливою функціональністю
Здатність до налаштувань	Середній рівень	Високий рівень
Система керування вмістом	Так	Ні, але має такі системи за необхідності
Фреймворк для розробки повного стеку	Ні	Так
Безпека	Середній рівень	Високий рівень
Відкритий доступ	Так	Так
Мова програмування	PHP	Python
Бібліотеки роботи з даними та машинного навчання	Середній рівень	Високий рівень

6

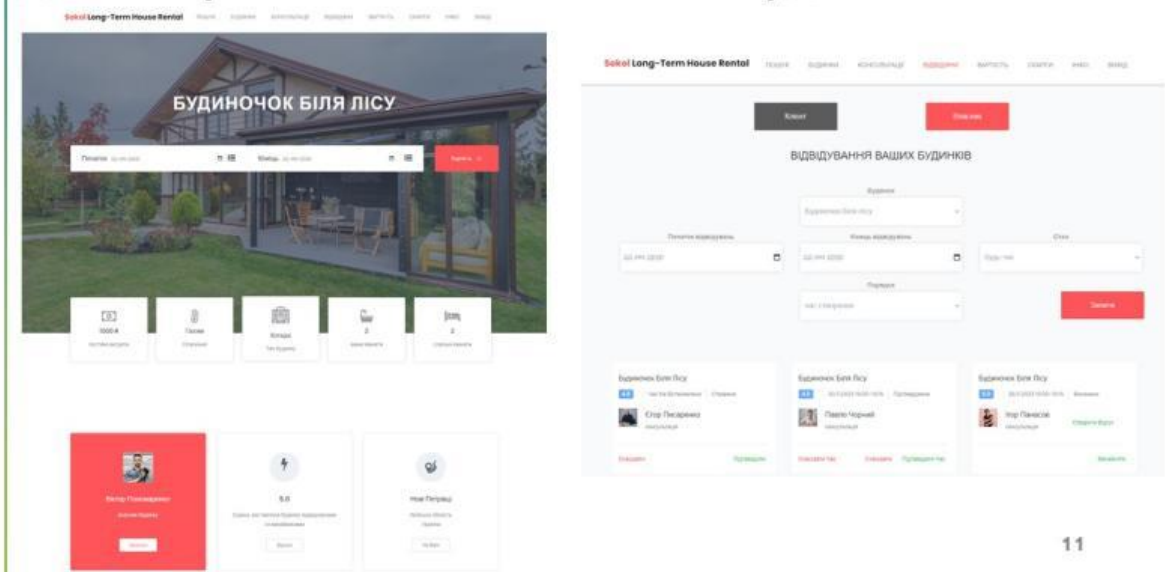
Рисунок В.6 – Слайд № 6 (Вибір рішення для веброзробки)



Інтерфейси програми підтримки довгострокового винаймання будинків



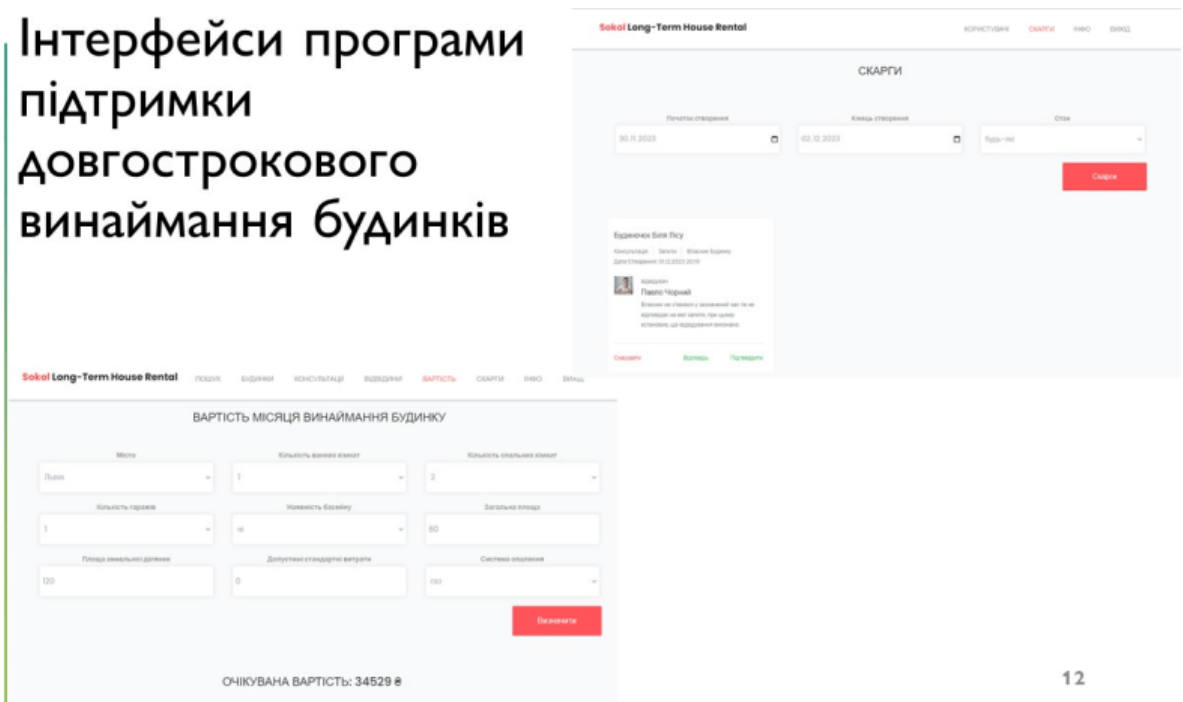
Інтерфейси програми підтримки довгострокового винаймання будинків



11

Рисунок В.11 – Слайд № 11 (Інтерфейси програми підтримки довгострокового винаймання будинків)

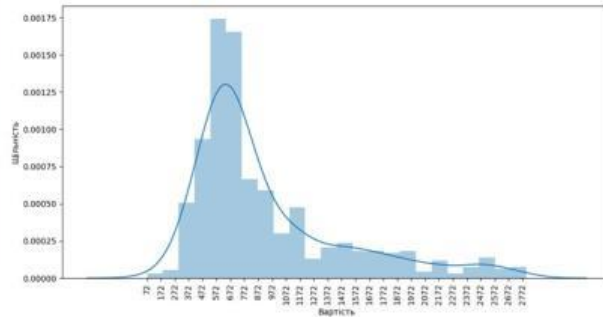
Інтерфейси програми підтримки довгострокового винаймання будинків



12

Рисунок В.12 – Слайд № 12 (Інтерфейси програми підтримки довгострокового винаймання будинків)

Експериментальне дослідження



Ознака вибірки даних	Коефіцієнт
Загальні додаткові витрати	0,53493
Кількість ванних кімнат	0,73813
Кількість спальних кімнат	0,52488
Загальна площа власності	0,54128
Площа тераси	0,20026
Площа земельної ділянки	0,56529
Рік будівництва	0,10797
Кількість гаражів	0,64654
Вид на море	0,20397
Наявність вітальні	0,13150
Наявність гардеробної	0,16888
Наявність басейну	0,32753
Наявність гаража або паркувального місця	0,10203
Можливість вимаймання з домашніми тваринами	-0,10588
Під'єднання до газопостачання	-0,11771
Наявність спортивного залу	0,05537
Наявність балкону	0,03177
Наявність кондиціонування приміщення	0,03194
Наявність кондиціонування повітря	0,04479
Наявність саду	0,28131
Наявність під'їзду підлоги	0,26869
Наявність ігрової кімнати або зони відпочинку	-0,02117
Наявність м'яблів	0,05743
Дуплекс	-0,00642
Тип системи опалення	0,38662
Наявність Wi-Fi	0,01170
Наявність зони барбію	0,12392

13

Рисунок В.13 – Слайд № 13 (Експериментальне дослідження)

Результати експериментального дослідження

Модель	Характеристики моделі	Середня абсолютна похибка	Модель	Характеристики моделі	Середня абсолютна похибка
Лінійна регресія	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки	0,0826669272	Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, тип системи опалення, 100 дерев	0,0705509
Лінійна регресія	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, загальна площа власності, площа земельної ділянки	0,0861494262	Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, 2000 дерев	0,0691457803
Метод опорних векторів	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки	0,0775624772	Random Forest	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, наявність басейну, загальна площа власності, площа земельної ділянки, тип системи опалення, 200 дерев	0,0687375081
Метод опорних векторів	Ознаки: загальні додаткові витрати, кількість ванних, спальних кімнат, гаражів, загальна площа власності, площа земельної ділянки	0,0793501855			

14

Рисунок В.14 – Слайд № 14 (Результати експериментального дослідження)

Висновки

- Розроблена програма призначена для підтримки довгострокового винаймання будинків, оголошення про яке створюється власником будинку, надаючи для цього інструменти пошуку оголошень, створення і визначення параметрів оголошень, створення запитів на відвідування будинку, консультування без відвідування, створення відгуків за результатами відвідування або винаймання, створення скарг на відвідувачів або власників будинків за необхідності та підтримки процесів вирішення конфліктів менеджерами. Розробка застосовує мову Python, базу даних на основі системи PostgreSQL, фреймворк Django.
- Наукова новизна роботи полягає в методі визначення вартості довгострокового винаймання будинків, який ґрунтується на створенні моделі прогнозування на основі методу Random Forest, даних відібраних інших будинків у заданому місті за визначеними ознаками, що дозволяє визначити очікувану місячну вартість винаймання будинку. Запропонований метод мав у результаті середню абсолютну похибку на 20,26 % меншу ніж у лінійної регресії та на 12,84 % меншу ніж у методу опорних векторів за результатами експериментального дослідження.

15

Рисунок В.15 – Слайд № 15 (Висновки)