

## АНОТАЦІЯ

На цій дипломній роботі було спроектовано та реалізовано веб систему для продажу та оренди земельних ділянок та нерухомості.

Система передбачає використання звичайними користувачами та рієлторськими компаніями. Для цього система має зручний користувацький інтерфейс та різні можливості для продаж чи здачу в оренду землі та нерухомості.

Що до використаних технологій та інструментів реалізації, розроблена веб система складається з двох частин:

- клієнтська частина, написана на базі бібліотеки React.TS, HTML розмітки, таблиць стилів Sass;
- серверна частина, написана на ASP.NET Core 5, з використанням мови програмування C# 8, та системи збереження та управління базами даних реляційної СУБД MS SQL Server.

Веб система розроблена згідно двошарової архітектури клієнт-сервер. Клієнт веб системи може експлуатуватися на будь-якій операційній системі: основною вимогою є наявність веб-оглядача. Для серверної частини веб системи необхідним є наявність бази даних SQL Server та веб-сервера IIS.

Загальний об'єм роботи 108 сторінок.

Також, окрім безпосередньо розробки веб системи, цей диплом містить економічну частину. У цій частині розраховано та визначено витрати для розробки даного продукту. Також було визначено потенційний дохід від продажу розробленої веб системи. Було обрано стратегію за якою буде проводитись розробка. Та визначено окупність даного програмного продукту.

## **ABSTRACT**

In this thesis, a web system for the sale and lease of land and real estate was developed and implemented.

The system is intended for use by ordinary users and real estate companies. To do this, the system has a user-friendly interface and various opportunities for sale or lease of land and real estate.

As for the technologies and implementation tools used, the developed web system consists of two parts:

- client part, written on the basis of React.TS library, HTML markup, Sass style sheets;
- server part written in ASP.NET Core 5, using the C # 8 programming language, and the system of storage and database management of relational DBMS MS SQL Server.

The web system is designed according to a two-tier client-server architecture. The client web system can run on any operating system: the main requirement is the presence of a web browser. The SQL Server database and the IIS web server are required for the server part of the web system.

The total volume of the work is 108 pages.

Also, in addition to the direct development of the web system, this diploma contains an economic part. In this part, the costs for the development of this product are calculated and determined. The potential revenue from the sale of the developed web system was also determined. The strategy according to which the development will be carried out was chosen. And the payback of this software product is determined.

## ЗМІСТ

|   |    |    |
|---|----|----|
| ВСТУП.....  | 9  |    |
| <br>РОЗДІЛ 1. АНАЛІЗ КОМП'ЮТЕРНИХ СИСТЕМ В ГАЛУЗІ ОРЕНДИ ТА ПРОДАЖУ НЕРУХОМОСТІ ТА ЗЕМЕЛЬНИХ ДІЛЯНОК .....                              |    | 10 |
| 1.1. Інформаційні системи на ринку купівлі-продажу та оренди нерухомості і рієлторська сфера.....                                       | 10 |    |
| 1.2. Системи-аналоги в Україні та світі. Оцінка конкурентів для веб системи для продажу та оренди земельних ділянок та нерухомості..... | 11 |    |
| 1.3. Висновки.....  | 14 |    |
| <br>РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБСИСТЕМИ ДЛЯ ПРОДАЖУ ТА ОРЕНДИ ЗЕМЕЛЬНИХ ДІЛЯНОК ТА НЕРУХОМОСТІ.....         |    | 16 |
| 2.1. Постановка задачі веб системи для продажу та оренди земельних ділянок та нерухомості.....  | 16 |    |
| 2.2. Інструменти реалізації веб системи для про продажу та оренди земельних ділянок та нерухомості.....                                 | 17 |    |
| 2.3. Специфікація вимог до веб системи з оренди та продажу земельних ділянок та нерухомості .....                                       | 18 |    |
| 2.3.1. Вступ.....   | 18 |    |
| 2.3.2. Загальний опис.....  | 19 |    |
| 2.3.3. Характеристики системи.....  | 21 |    |
| 2.3.4. Вимоги до зовнішніх інтерфейсів.....   | 23 |    |
| 2.3.5. Інші нефункціональні вимоги.....   | 24 |    |
| 2.3.6. Інші вимоги.....   | 25 |    |
| 2.4. Висновки.....  | 25 |    |
| <br>РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ ДЛЯ ПРОДАЖУ ТА ОРЕНДИ ЗЕМЕЛЬНИХ ДІЛЯНОК ТА НЕРУХОМОСТІ .....   |    | 26 |

|   |           |
|---|-----------|
| 3.1. Проектування архітектури та об'єктної моделі веб системи для продажу та оренди земельних ділянок та нерухомості..... | 26        |
| 3.1.1. Діаграма прецедентів .....   | 26        |
| 3.1.2. Діаграма послідовності .....   | 27        |
| 3.1.3. Діаграма діяльності.....   | 28        |
| 3.1.4. Діаграма класів.....   | 28        |
| 3.1.4. Діаграма інтерфейсів. ....   | 29        |
| 3.1.4. Архітектурна діаграма.....   | 30        |
| 3.1.5. Діаграма розгортання. ....   | 31        |
| 3.2. Проектування бази даних до веб системи для продажу та оренди земельних ділянок та нерухомості.....                   | 32        |
| 3.3 Проектування інтерфейсу користувача веб системи для продажу та оренди земельних ділянок та нерухомості. ....          | 36        |
| 3.4. Висновки. ....   | 37        |
| <b>РОЗДІЛ 4. РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ ДЛЯ ПРОДАЖУ ТА ОРЕНДИ ЗЕМЕЛЬНИХ ДІЛЯНОК ТА НЕРУХОМОСТІ .....</b>                       | <b>38</b> |
| 4.1. Реалізація серверної частини веб системи для продажу та оренди земельних ділянок та нерухомості. ....                | 38        |
| 4.2. Реалізація клієнтської частини веб системи для продажу та оренди земельних ділянок та нерухомості. ....              | 39        |
| 4.3. Опис роботи системи.....   | 40        |
| 4.4. Проведення тестових випадків для веб системи для продажу та оренди земельних ділянок та нерухомості. ....            | 44        |
| 4.4.1. Функціональне тестування.....  | 44        |
| 4.4.2. Тестування інтерфейсу.....   | 46        |
| 4.5. Висновки. ....   | 46        |

|  |     |
|--|-----|
| РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА.....  | 48  |
| 5.1. Економічна характеристика веб системи для продажу та оренди земельних ділянок та нерухомості.....   | 48  |
| 5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення веб системи для продажу та оренди земельних ділянок та нерухомості.....     | 48  |
| 5.3. Оцінювання та аналізування факторів внутрішнього та зовнішнього середовища для веб систем для продажу та оренди земельних ділянок та нерухомості..... | 49  |
| 5.4. Формування стратегічних альтернатив розвитку веб системи для продажу та оренди земельних ділянок та нерухомості.....                                  | 52  |
| 5.4.1. Стратегічні альтернативи першої групи.....  | 52  |
| 5.4.1. Стратегічні альтернативи другої групи.....  | 52  |
| 5.5. Бюджетування веб системи для продажу та оренди земельних ділянок та нерухомості.....  | 53  |
| 5.6. Висновки .....  | 59  |
| ВИСНОВКИ.....  | 60  |
| СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ .....   | 61  |
| ДОДАТКИ.....   | 63  |
| Додаток А. Додатки аналоги.....  | 63  |
| Додаток Б. UML діаграми .....  | 66  |
| Додаток В. Вайрфрейми та мокапи інтерфейсу.....  | 72  |
| Додаток Г. Інструкція користувача .....  | 75  |
| Додаток Д. Код .....   | 77  |
| Додаток Е. Скріншоти інтерфейсу користувача.....   | 100 |

## ВСТУП

Ринок нерухомості є одним із найбільших в світі. Кожного дня люди шукають собі житло для проживання, приміщення для свого бізнесу чи під інші потреби, або шукають земельну ділянку для забудови чи інших цілей.

Метою дипломної роботи «Веб система для продажу та оренди земельних ділянок та нерухомості» є розробка програмної системи яка надаватиме можливість здійснювати продаж та оренду земельних ділянок та нерухомості. Також дана програмна система може використовуватися для аналізу даних про оренду та продаж у рієлторських організація. Програмне забезпечення враховує особливості кожної окремої пропозиції, так і поточний станожної з земельних ділянок або нерухомості.

Розроблювана веб система орієнтована на використання компаніями з рієлтингу та простими користувачами для публікації своїх оголошень. Вона надає можливість зручного менеджменту та взаємодію з клієнтами. Це дозволить оптимізувати робочі процеси та зробити сервіс більш привабливим як для клієнта так і для працівників сервісу. Також веб система повинна внести свої зміни на ринку продажів та оренди земельних ділянок та нерухомості. Вона повинна зробити його більш прозорим, більш легко освоюваним для будь-якої пересічної людини. Щоб кожна людина могла з легкістю і з мінімальними зусиллями вирішити свої питання у цій області. Також дана веб система дозволить полегшити процес купівлі та оренди. Користуючись зручним, багатофункціональним і зрозумілим користувацьким інтерфейсом потенційний користувач має полегшити свої задачі у роботі з даним типом угод.

## РОЗДІЛ 1. АНАЛІЗ КОМП'ЮТЕРНИХ СИСТЕМ В ГАЛУЗІ ОРЕНДИ ТА ПРОДАЖУ НЕРУХОМОСТІ ТА ЗЕМЕЛЬНИХ ДІЛЯНОК

### 1.1. Інформаційні системи на ринку купівлі-продажу та оренди нерухомості і ріелторська сфера.

Ринок нерухомості відіграє важливу роль в економіці країни і в світовій економіці в цілому.

Важливу роль на ринку нерухомості та наданні цих послуг відіграють ресурси, де можна знайти потрібне оголошення. Пошук того що шукає людина займає найбільше часу. У сучасному світі люди часто з питаннями звертаються в інтернет. Веб-ресурс із всією необхідною інформацією, буде завжди дуже необхідний.

Загалом, найважливішою інформацією про нерухомість є його характеристики: загальна площа, житлова площа, площа земельної ділянки, різні комунікації, поверховість, тощо. Також важливим є можливість переглянути відкриті документи що до цієї нерухомості [1]. Корисною буде можливість зв'язатися з власником оголошення.

Щоб переглянути місце розташування нерухомості та дізнатися відстань до найближчих магазину, супермаркету, зупинки метро чи автобусної зупинки люди часто звертаються по допомогу до карт Google [2] чи аналогів. Інтеграція карт на сторінку оголошення був би приємним плюсом, який заощадить час для відвідувача сайту у пошуку місця розташування прочитаного об'єкта згідно оголошення.

Звісно нерухомість є одним з найважливіших сфер бізнесу і економіки. Нещодавно в Україні відкрили ринок землі. Донедавна у нашій країні було заборонено продавати землі господарського призначення, але після відкриття ринку землі в Україні відкриється новий ринок.

Дослідивши доступні веб сервіси у мережі Інтернет, було знайдено веб сайти, де продаж відбувається через аукціон, але така можливість доступна не всюди. Можливість аукціону є дуже корисною. У випадку відсутності, існує можливість

великої кількість угод укладених через попередні домовленості зі знайомими або навіть повністю закриті. В кращому випадку домовленості і торги відбуватимуться на словах або через веб-ресурси, які зовсім не призначені для такого виду товарів. Тому, для вирішення цієї проблеми, забезпечення вищого рівня прозорості та забезпечити користувачу зручність у робі з таким видом угод варто створити веб-ресурс, де можна переглянути детальну інформацію про земельну ділянку (її площину, розташування, оцінку землі, кадастрові карти, тощо), зв'язатися із власником, скласти угоду та провести оплату або здійснювати помісячну оплату у випадку оренди [3]. Додаткова корисною можливість аукціону, щоб кожен зацікавлений міг запропонувати свою ставку. Це дозволило б іще підвищити рівень прозорості, переможцем і відповідно новим власником має бути людина, яка найбільше зацікавлена у цій пропозиції.

## **1.2. Системи-аналоги в Україні та світі. Оцінка конкурентів для веб системи для продажу та оренди земельних ділянок та нерухомості.**

В інтернеті є доволі багато сайтів з оголошеннями. Але дуже мало з них призначені для продажу та оренди земельних ділянок та нерухомості.

Серед аукціонів з продажем земельних ділянок є українська універсальна біржа [3]. Найпопулярнішими майданчиками для продажів в Україні можна виділити олх [4] та Dom ria [5]. В світі є найпопулярнішими аукціонами є eBay [6] та amazon [7].

Що до сайтів по продажу земельних ділянок, зазвичай такі оголошення викладають на вище перелічених веб-платформах. Це пов'язано із популярністю веб ресурсів.

### **OLX [4].**

Одними з найпопулярніших ресурсів з оголошеннями, зокрема для пошуку нерухомості є олх (Додаток А.1). Це дуже популярна веб-платформа, де кожен охочий може викласти своє оголошення. А інші люди будуть переглядати його, відповідно цікавитися пропозицією і у випадку згоди сторін відбувається угода про продаж чогось, надання послуг і тд. На цьому веб-ресурсі є категорії нерухомості.

Але самі оголошення не передбачають великого опису і тому інформація є доволі обмеженою. І сам сайт передбачає лише викладання оголошення і подальший обмін контактами для комунікації клієнта з продавцем.

### **Українська універсальна біржа [3].**

Цей сайт являє собою земельний аукціон для продажу земельних ділянок державної та комунальної власності (Додаток А.2). На даному веб сайті існує можливість перегляду оголошень, з детальною інформацією про земельну ділянку. Цей сайт є вузькоспеціалізованим, передбачено багато нюансів пов'язаних з цією тематикою. Розробниками сайт позиціонується як зручне програмне забезпечення для купівлі земельних ділянок різного призначення. Сайт був розроблений по причині зняття мораторію на продаж землі в Україні. Сайт дозволяє розміщення оголошень з фото нерухомості, вартістю, коротким описом та контактами власника.

### **Dom ria [5].**

Dom ria це майданчик для купівлі та продажу нерухомості в Україні (Додаток А.3). Це його головна спеціалізація, тому це дозволяє створювати та переглядати дуже детальні оголошення, у межах цієї предметної області. Сайт орієнтований не велику кількість різних користувачів, ним можуть користуватися як приватні особи, так забудовники і різні агентства по нерухомості.

### **eBay [6].**

eBay це американська інтернет-компанія, яка є власником сайту для продажів eBay.com (Додаток А.4). Вона являє собою торговельний веб-сайт та онлайн майданчик для проведення аукціонів та звичайних продажів. На цьому сайті приватні та юридичні особи можуть здійснювати продаж та купівлю різноманітних товарів та послуг. Загалом, цей сайт має дуже велику кількість оголошеннЯ на найрізноманітнішу тематику і слугує називатися одним з найпопулярніших у світі.

На цьому сайті всі оголошення побудовані за схемою – фото, опис, способи оплати і можливість поставити ставку (у випадку аукціону). Що до вище згаданої тематики нерухомості і земельних ділянок, у потенційного користувача немає великих можливостей для опису свого оголошеннЯ у межах заданої тематики.

## AMAZON [7].

Amazon.com — один із перших інтернет-сервісів, орієнтованих на продаж реальних товарів масового попиту (Додаток А.5). Ця компанія є найбільшою у світі за обігом коштів. Компанія веде експансію на ринки інтернет-торгівлі в усьому світі. Amazon має окрім сайти для роздрібної торгівлі майже по всьому світу, що робить цю платформу однією з найпопулярніших у світі. Але цей ресурс є подібним до eBay і вони мають схожі недоліки. Потенційний користувач немає великих можливостей для опису свого оголошення у межах заданої тематики.

Порівняльна характеристика перерахованих веб систем (Таблиця 1.1).

Таблиця 1.1.

Порівняльна характеристика

|  | OLX | Українська<br>універсальна<br>біржа | Dom ria | eBay | AMAZON |
|--|-----|-------------------------------------|---------|------|--------|
| Аукціон  | -   | +                                   | -       | +    | +      |
| Детальний<br>опис згідно<br>предметної<br>області  | -   | +                                   | +       | -    | -      |
| Можливість<br>вести чат з<br>автором<br>оголошення | +   | -                                   | +       | +    | +      |
| Карти Google<br>на сторінці<br>оголошення          | +   | -                                   | +       | -    | -      |

## Продовження таблиці 1.1

|                                    |   |   |   |   |   |
|------------------------------------|---|---|---|---|---|
| Сервіси для менеджменту угод       | - | - | - | - | - |
| Сервіси для оплати угод            | + | - | - | - | - |
| Продаж та оренда земельних ділянок | - | + | - | - | - |

### **1.3. Висновки.**

Оглянувши вище перелічені системи, я побачив, що їх основний недолік полягає в тому, що більшість з них орієнтовані на велику кількість різноманітних оголошень.

Для забезпечення розвитку у продажі та оренді земельних ділянок та нерухомості, варто створити веб-ресурс, який буде поєднувати всі джерела інформації стосовно цієї предметної області. Тобто веб система повинна мати можливість перегляду декількох типів оголошень (оголошення з нерухомістю та земельних ділянок для забудови чи під господарські потреби).

Веб система повинна надавати наступні можливості: перегляд детальної інформацію про земельну ділянку (її площу, розташування, оцінку землі, кадастрові карти, тощо), зручний зв'язок із власником, можливість укладати угоди та проведення оплати або здійснювати помісячну оплату у випадку оренди. Додатковою корисною була б можливість аукціону, щоб кожен зацікавлений міг запропонувати свою ставку. Це дозволило б підвищити рівень прозорості, тому що власником об'єкта торгові і відповідно переможцем має бути людина, яка найбільше зацікавлена у цій пропозиції.

Також для адміністратора сайту, було б корисним надати можливість в менеджменті списку користувачів системи(додавання, видалення, редагування даних користувачів і списку користувачів). А також можливість менеджменту доступних оголошень на сайті, було б великим плюсом.

Що до оплачуваності розроблюваної системи - вона повинна бути безкоштовною для звичайних користувачів і оплачуваною для великих компаній. Головним потенційним користувачем системи є прості люди. Іншим потенційним користувачем мають стати рієлторські компанії, які мають використовувати переваги розроблюваної системи у своїх цілях.

## **РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ВЕБСИСТЕМИ ДЛЯ ПРОДАЖУ ТА ОРЕНДИ ЗЕМЕЛЬНИХ ДІЛЯНОК ТА НЕРУХОМОСТІ**

### **2.1. Постановка задачі веб системи для продажу та оренди земельних ділянок та нерухомості.**

Метою виконання дипломної роботи є розробка веб системи для продажу та оренди земельних ділянок та нерухомості.

Веб система повинна мати можливість перегляду декількох типів оголошень, які поділяються на оголошення з нерухомістю та земельних ділянок для забудови чи під господарські потреби. І в свою чергу повинні поділяються на оголошення про оренду і про продаж.

Веб система повинна надавати наступні можливості: перегляд детальної інформації про земельну ділянку, зв'язок із власником, можливість укладати угоди та проведення оплати або здійснювати помісячну оплату у випадку оренди. Додатково корисною є можливість аукціону.

Також для адміністратора сайту повинно бути надано можливість менеджменту списку користувачів системи та оголошень.

Система повинна сповіщати повідомленнями про досягнення критичних меж параметрів системи. Система повинна бути масштабованою та повинна складатися з окремих модулів.

Вхідні дані: відповідно до даної тематики, дані про сферу оренди та продажу земельних ділянок та нерухомості, технічна документація для .Net Core 5 [8] та мова програмування C#, для розробки клієнтської частини технічна документація по React.TS [10], документація бібліотеки стилів Bootstrap.

Вихідні дані: розроблена веб система для продажу та оренди земельних ділянок та нерухомості.

## **2.2. Інструменти реалізації веб системи для продажу та оренди земельних ділянок та нерухомості.**

Для розробки UML діаграм програмної системи було обрано draw.io [11]. Це зручний сервіс для побудови діаграм із великою кількістю функціоналу.

Розробка прототипу інтерфейсу користувача (мокапів та вейрфреймів) велася у середовищі Figma [12]. Обраний сервіс дозволяє зручно розробити всі необхідні рівні вейрфреймів та розробити мокапи для розроблюваної веб системи. Також сервіс передбачає створення високорівневих мокапів із об'єднанням їх у прототип, що дозволяє полегшити і покращити процес розробки програмного забезпечення. Figma має десктопну версію, її можна встановити на комп’ютер і зручно користуватися сервісом. Також існують інші версії цього сервісу, що робить його кросплатформенным. Та існує можливість збереження результатів роботи у хмару і синхронізації напрацювань на всіх робочих пристороях.

Для реалізації свого завдання для серверної частини було обрано мною мову програмування C# та платформу .Net [8]. Вона була обрана завдяки її універсалності та легкості у освоєнні для реалізації серверної частини розроблюваної веб системи.

Роботи з кодом на серверній частині проводилася у середовищі розробки Microsoft Visual Studio 2022 [14]. Це середовище було обране через дуже гарну сумісність з технологіями .Net та надає найкращі можливості для розробки.

Також на серверній частині використовується база даних для збереження даних. У межах даної дипломної роботи, використано реляційну систему управління базами даних Microsoft SQL Server [9]. Відповідно до тематики і поставлених задач, виникає потреба у реляційні базі даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

Клієнтську частину реалізовано на мові програмування TypeScript та бібліотеці React [10] та використано бібліотеки стилів react-bootstrap material-ui та інші, для написання власних стилів використано підхід написання sass. Мову програмування TypeScript [11] було обрану через кращу організацію роботи з цією мовою. У цій мові є багато доступних об’єктно орієнтованих переваг. Також ця

мова є строго типізованою. У порівняння з Java Script і його динамічною типізацією це є великою перевагою. Також для реалізації клієнтської частини було обрано функційний підхід, а не класовий. Таке рішення було зроблено через те, що це є більш новий підхід, він має велику кількість переваг у порівнянні з класовим підходом. Перш за все це хуки. Хуки дозволяють функційним компонентам мати доступ до станів і властивостей різних змінних та інших React особливостей. Через це, класові компоненти не потрібні. У цьому підході вдається дуже скоротити код і підвищити швидкість написання коду та покращити його якість. До того ж хуки є оптимізовані у самій бібліотеці реакту [9] і дозволяють пришвидшити роботу системи.

Робота з кодом на клієнтській частині відбувалася у середовищі Visual Studio Code [14]. Це середовище дозволяє вести комфортну розробку завдяки своїм широким можливостям. Широкі можливості середовища забезпечуються широким вибором додаткових пакетів і розширеннями для цього редактора. Також це середовище дуже добре поєднується з Microsoft Visual studio 2022 [15], бо ці середовища розроблені однією компанією і мають багато подібного.

## **2.3. Специфікація вимог до веб системи з оренди та продажу земельних ділянок та нерухомості [16].**

### **2.3.1. Вступ.**

#### **2.3.1.1. Призначення, мета.**

Ця веб система призначена для продажу та оренди земельних ділянок та нерухомості. Метою є розвиток ринку продажів та оренди земельних ділянок та нерухомості. Цільовими клієнтами є люди, які шукають собі житло і іншу нерухомість під свої потреби або земельну ділянку під забудови чи господарської діяльності. Також даний продукт може використовуватись рієлторськими компаніями для аналізу та розвитку ринку землі та нерухомості.

#### **2.3.1.2. Перспективи продукту.**

Створений веб-ресурс повинен поєднувати джерела інформації стосовно цієї предметної області, що буде вирішувати проблему пошуку необхідної інформації

стосовно оголошень у межах заданої предметної області. У перспективі підтримка проекту протягом довгого часу та можливість розширення функціоналі в залежності від потреб ринку.

### **2.3.2. Загальний опис.**

#### **2.3.2.1. Характеристики продукту.**

Веб система складається з декількох частин - клієнтської та серверної частини. Також можна відокреми у серверній частині частину з базою даних (Додаток Б.6). Ці елементи веб системи мають забезпечувати наступні функції:

Клієнтська частина має наступні функції:

- Створення оголошень про продаж чи оренду земельних ділянок чи нерухомості;
- Перегляд, фільтрація та сортування оголошень;
- Аукціон;
- Створення угод;
- Повна оплата замовлення або помісячна оплата у випадку оренди;
- Реєстрація та створення власного профіля користувача;
- Перегляд статистики.
- Підтримка локалізації інтерфейсу.
- Серверна частина має наступні функції:
- Обробка запитів від клієнтської частини;
- Доступ до даних з бази даних;
- База даних має наступні функції:
- Збереження даних, потрібних для роботи системи;
- Забезпечення цілісності даних;

#### **2.3.2.2. Класи користувачів та їх характеристики (Додаток Б.1).**

Розроблювана веб система передбачає 3 класи користувачів:

- Неавторизований користувач – може переглядати, фільтрувати оголошення. Також може переглянути коротку інформацію про веб систему. Має можливість авторизуватися у системі.

- Авторизований користувач – доступні можливості неавторизованого користувача та має можливість брати участь в аукціонах та може створювати нові оголошення про продаж чи оренду земельних ділянок чи нерухомості. У кожного зареєстрованого користувача є свій особистий кабінет. На сторінці перегляду списку угод він може переглянути які угоди він укладав, які проводив платіжки, може провести платіжку згідно укладених угод, може прийняти запит на створення угоди.
- Адміністратор – має можливості по запитах від клієнтів редагувати оголошення, може вносити зміни у роботу системи. Також головною особливістю ролі адміністратора є менеджмент облікових записів користувачів системи.

### **2.3.2.3. Середовище функціонування.**

Розроблювана веб система має дві частини. Відповідно для кожної частини існують свої вимоги.

- Для клієнської частини вимоги:
  - Веб-переглядач Google chrome не нижче версії 102.0.5005.61;
  - Стабільне інтернет з'єднання зі швидкістю 1 Mbit/c або краще;
- Для серверної частини необхідний серверний комп’ютер з наступними характеристиками:
  - Процесора Intel Core i7 / Xeon з кешем не менше 4Мб або аналогічні рішення на основі AMD або краще;
  - Оперативна пам’ять не менше 8 Гб;
  - Для збереження даних необхідно SSD диск на 2 терабайти і відповідно більше при збільшенні кількості користувачів;
  - Програмне забезпечення. Операційна система Windows 11 Pro. Microsoft .NET Core 6, а також Microsoft SQL Server 2019 для організації роботи бази даних та для управління базою даних;
  - Інтернет з’єднання 1Гбіт/c;

### **2.3.3. Характеристики системи..**

Детально ознайомитися із класами користувачів і випадками використання системи можна на діаграмі прецедентів (Додаток Б.1).

#### **2.3.3.1. Авторизація та реєстрація.**

**Опис:** Авторизація та аутентифікація користувача в системі.

**Пріоритет:** середній.

**Послідовності дія/відгук:**

Користувач натискає кнопку „увійти” відкривається модальне вікно для входу в систему. Користувач вводить логін та пароль у відповідні поля. Натискає кнопку увійти і входить в систему.

**Функціональні вимоги.**

REQ-1: Валідація даних, уведених в полях авторизації та реєстрації;

REQ-2: У випадку помилок при спробі увійти в систему відображати відповідні повідомлення;

REQ-3: Доступ до елементів системи відповідно до ролі користувача в системі;

REQ-4: Автоматичне приховання видимості даних входу.

#### **2.3.3.2. Створення нових оголошень.**

**Опис:** Створення нових оголошення по оренді або продажу нерухомості або земельних ділянок (Додаток Б.2).

**Пріоритет:** високий.

**Послідовності дія/відгук:** Користувач натискає кнопку для створення нового оголошення, вносить дані у відповідні поля, вибирає на сторінці у картах Google місце знаходження об'єкту оренди або продажу, завантажує фото об'єкта. Після внесення всієї необхідної інформації, користувач натискає кнопку для завершення створення нового оголошення. У разі успішного створення, оголошення з'явиться у списку доступних оголошень. У разі не удачі відобразиться відповідне вікно.

### **Функціональні вимоги.**

REQ-1: Валідація даних, уведених в полях форми створення нового оголошення;

REQ-2: У разі невдачі при створенні нового оголошення відобразиться відповідне вікно;

REQ-3: Карты Google на сторінці для задання адреси об'єку продажу чи оренди;

REQ-4: Можливість завантажувати фото об'єта продажу чи оренди;

REQ-5: Після успішного створення оголошення, можливість переглянути оголошення в особистому кабінеті;

REQ-3: Після успішного створення оголошення, можливість переглянути оголошення у списку всіх доступних оголошень на сайті;

#### **2.3.3.3. Перегляд списку оголошень.**

**Опис:** Перегляд списку оголошень із можливістю фільтрації та сортування оголошень.

**Пріоритет:** високий.

**Послідовності дія/відгук:** Користувач заходить на сторінку перегляду списку оголошень. Йому відображаються картки всіх оголошень доступних в системі. Користувач натискає відповідні кнопки для сортування і фільтрування списку оголошень. Відображаються відповідні зміни у списку.

### **Функціональні вимоги.**

REQ-1: Можливість перегляду списку оголошень без входу в систему;

REQ-2: Можливість сортувати список оголошень;

REQ-3: Можливість фільтрувати список оголошень;

REQ-4: Валідація даних, уведених в полях для сортування та фільтрації;

REQ-5: У разі введення неправильних даних у поля фільтрації чи сортування відобразити відповідне вікно;

#### **2.3.3.5 Створення угоди купівлі чи оренди згідно оголошення.**

**Опис:** Створення угоди на купівлю чи оренду об'єкта згідно оголошення.

**Пріоритет:** високий.

**Послідовності дія/відгук:** Користувач натискає кнопку для купівлі чи оренди на картці оголошення. З'являється відповідне вікно для підтвердження операції. Користувач підтверджує свої цілі на сторінці створення нової угоди.

#### **Функціональні вимоги.**

REQ-1: Можливість натиснення кнопки купівлі (створення угоди купівлі);

REQ-2: У разі не удачі при створенні нової угоди відобразиться відповідне вікно;

REQ-3: Валідація даних, уведених в полях форми для оплати угоди;

REQ-4: Після успішного створення угоди можливість переглянути угоду в особистому кабінеті;

#### **2.3.3.7. Менеджмент користувачів системи.**

**Опис:** Робота з аккаунтами користувачів системи.

**Пріоритет:** високий.

**Послідовності дія/відгук:** Адміністратор системи заходить у систему. При натисненні на вкладку з менеджментом користувачів системи адміністратор переходить на сторінку менеджменту користувачів системи. Відображається таблиця з користувачами та детальною інформацією про кожного. У адміністратора є можливість додати нового користувача, видалити чи змінити дані користувача.

#### **Функціональні вимоги.**

REQ-1: Можливість перегляду списку користувачів системи;

REQ-2: Можливість вносити зміни у список користувачів;

REQ-3: У разі не удачі при внесені змін у список користувачів відобразиться відповідне вікно.

#### **2.3.4. Вимоги до зовнішніх інтерфейсів.**

##### **2.3.4.1. Користувацькі інтерфейси.**

При вході на сайт користувачу відображається головна сторінка зі списком всіх оголошень. У верхній частині сторінки користувач може знайти кнопки для входу в систему (Додаток Е.1).

Увійшовши в систему користувач має можливість створювати нові оголошення, укладати угоди та переглядати свій особистий кабінет (Додаток Е.2).

Увійшовши в систему як адміністратор. Йому доступна вкладка менеджменту аккаунтів користувачів системи (Додаток Е.5). Та доступний доступ до всіх оголошень та модулів системи для перегляду та внесення змін.

Також користувацький інтерфейс повинен бути інтуїтивно зрозумілим та легко освоюваним новими користувачами системи. Також користувацький інтерфейс повинен підтримувати локалізацію.

#### **2.3.4.2. Програмні інтерфейси.**

Програмна система повинна підтримувати програмні інтеграції Google Services, Amazon Web Services, PayPal Web Services, Microsoft Identity.

#### **2.3.5. Інші нефункціональні вимоги.**

##### **2.3.5.1 Вимоги продуктивності.**

Система повинна відповідати наступним вимогам продуктивності:

Час на обробку запитів до сервера не повинен перевищувати 5 секунд;

Повинна бути можливість стабільної роботи до 2000 активних користувачів системи.

##### **2.3.5.2. Вимоги безпеки.**

Серверна частина повинна бути захищена від стороннього доступу і лише адміністратор повинен мати доступ до всіх модулів системи. Повинна забезпечуватись цілісність даних.

Для забезпечення безпеки між клієнтом і сервером, повинен використовуватися JWT-token. Він використовується для авторизації користувача в системі і забезпечує безпеку завдяки алгоритму асиметричного шифрування для передачі повідомлення з інформацією про користувача.

##### **2.3.5.3. Атрибути якості програмного продукту.**

Програмний продукт повинен бути мало зв'язним та легко піддаватися змінам для подальшої його підтримки у ході експлуатації. Щоб команда розробників на етапі підтримки не мала складнощів у випадку потреби додати підтримку нового функціоналу, чи змінені логіки системи.

### **2.3.6. Інші вимоги.**

Веб система повинна мати дозволи на доступ до інформації згідно заданої предметної області у країнах, де ця програмна система використовується.

### **2.4. Висновки.**

Після написання цього розділу дипломної роботи, було визначено мету та призначення розроблюваної системи. Визначено вхідні та вихідні дані. Та вибрано інструменти реалізації розроблюваної веб системи.

Було визначено основну бізнес-логіку програмної системи та визначено функціональні та нефункціональні вимоги. Було описано характеристики системи із функціональними умовами та послідовностями дія-відгук.

До найважливіших вимог можна віднести вимоги бізнес-логіки для реалізації головного сценарію використання веб системи. Це авторизація та аутентифікація в системі як звичайний користувач та створення та перегляд оголошення і створення угод згідно цих оголошень, оплата згідно угод.

## РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ ДЛЯ ПРОДАЖУ ТА ОРЕНДИ ЗЕМЕЛЬНИХ ДІЛЯНОК ТА НЕРУХОМОСТІ

### **3.1. Проектування архітектури та об'єктної моделі веб системи для продажу та оренди земельних ділянок та нерухомості.**

Для проектування розроблюваної веб системи на різних рівнях було використано UML діаграми [17]. У процесі проектування було визначено основні архітектурні рішення та було визначено структуру веб системи. Для основної бізнес-логіки було розроблені діаграма із детальним описом.

Проектування відбувалося в інструменті Draw.io [14].

Розроблені UML діаграм (Додаток Б):

- класів (class diagram);
- інтерфейсів (interface diagram);
- прецедентів (use case diagram) (Додаток Б.1);
- послідовності (sequence diagram) (Додаток Б.2);
- діяльності (activity diagram) (Додаток Б.3);
- кооперації (collaboration diagram) (Додаток Б.4);
- станів (statechart diagram) (Додаток Б.5);
- компонентів (component diagram) (Додаток Б.6);
- розгортання (deployment diagram);
- архітектурна діаграма (architecture diagram);

#### **3.1.1. Діаграма прецедентів (Додаток Б.2).**

На спроектованій діаграмі прецедентів існує три класи користувачів; звичайний користувач, адміністратор та неавторизований користувач. Відповідно у трьох ролей різні можливості в системі.

Опис діаграми.

Клієнт системи має наступні варіанти використання системи:

Зайти на головну сторінку. Переглянути список оголошень. Сортувати і фільтрувати їх. Знайти необхідне оголошення. Зв'язатися з автором оголошення. Створити запит на створення нової угоди купівлі або оренди. Після створення запиту відбуваються різні юридичні справи для передачі земельної ділянки чи нерухомості іншому власнику. Після завершення всіх юридичних справ. Власник оголошення робить підтвердження угоди. Користувач може побачити підтвердження угоди і оплатити зазначену суму на сторінці перегляду списку оголошень.

На сторінці оголошеннЯ користувач може зробити ставку, якщо це доступно і оголошення бере участь в аукціоні. Після зробленої ставки користувач може перейти на іншу сторінку. У випадку, якщо ставка виграє, користувач буде сповіщений про це у своєму особистому кабінеті.

Також користувач може створити угоду про оренду, по аналогії з створенням угоди про купівлю. Орендатору, у такому випадку доведеться оплачувати помісячну вартість оренди.

Авторизований користувач може створити нове оголошення. Для цього потрібно зайди на відповідну сторінку, увести дані оголошення та підтвердити створення.

Адміністратор системи має наступні варіанти використання системи:

- Менеджмент облікових записів користувачів. Менеджер може додавати, видаляти, змінювати облікові записи.
- Може керувати оголошеннями.

### **3.1.2. Діаграма послідовності (Додаток Б.2).**

Діаграма послідовності зображає послідовність дій користувачів системи задля досягнення якоїсь мети. Тобто являє собою деякий алгоритм із чітким описом послідовності виконання кожного етапу.

На спроектованій діаграмі описується послідовність дій користувача системи. На діаграмі присутні чотири лінії життя: користувач, клієнтська частина системи, API і база даних.

Для повної роботи з системою користувачу необхідно спершу авторизуватися в системі. У випадку, якщо користувач не має аккаунту в системі, його необхідно створити заповнивши відповідні форми у процесі авторизації.

Після успішної авторизації користувач може переглянути список оголошень та ознайомитися з ними, може переглянути детальну інформацію про оголошення. Також користувач може створити угоду відповідно до оголошення.

Авторизований користувач, як звичайний відвідувач, має можливість створювати нові оголошення.

У кінці роботи з системою користувач може вийти з системи.

### 3.1.3. Діаграма діяльності (Додаток Б.3).

На діаграмі діяльності зображені процеси роботи відвідувача системи. У даному випадку ця діаграма являє собою поведінковий граф. На діаграмі відображені процеси авторизації, перегляду оголошень, укладання угод та створення нових оголошень.

### 3.1.4. Діаграма класів.

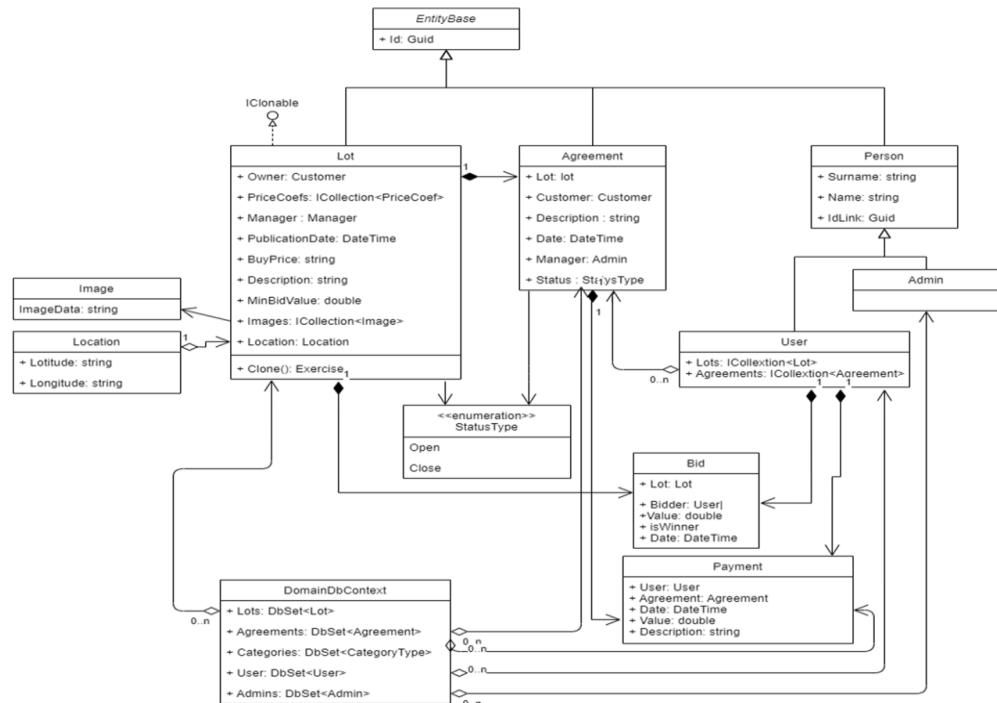


Рис. 3.1. Діаграма класів

На діаграмі класів зображені ієрархію класів на серверній частині системи. Головним класом у розроблюваній системі є клас Lot. Він містить всю потрібну інформацію про оголошення. Також клас Lot реалізує інтерфейс IClonable. Цей інтерфейс реалізує патерн прототип і дозволяє створювати велику кількість подібних оголошень, без великих затрат ресурсів. Допоміжними до цього класу є Image, Location, Status. Image - відповідає за збереження даних про фото оголошенні.

Location - відповідає за збереження адреси об'єкта з оголошенні. Agreement - являє собою угоду, укладену згідно оголошенні. Bid – ставка за лот від користувача системи. User і Admin користувачі системи.

DomainDbContext – контекст, який об'єднує всі класи для зручного доступу до них з інших рівнів системи. Також всі класи є нащадками від одного базового класу EntityBase, де визначено основні властивості для всіх класів.

Клас Payment має зв'язок композиції із класами Agreement і User, бо без цих класів Payment не може самостійно функціонувати. Payment посилається на ці класи. Без цих класів зникає сенс в оплаті, бо оплата без посилання за що оплачено і ким оплачена не має сенсу.

Bid теж має зв'язки композиції з класами Lot та User. Це пов'язано з тим що ставка повинна обов'язково посыпатися на оголошення і повинна посыпатися на власника ставки. Таким чином через клас Bid відбувається зв'язок між користувачем і оголошеннем.

Класи User і Admin є дочірніми класами від класу Person. Клас Person визначає основні властивості особистості - це прізвище, ім'я, номер телефону і т.д.

### **3.1.4. Діаграма інтерфейсів.**

Згідно діаграми інтерфейсів, у розроблюваній веб системі до більшості класів існують репозиторії для доступу даних з моделей.

Всі репозиторії унаслідуються від батьківського класу EntityRepository, де описано всі головні операції для доступу до даних. У батьківському інтерфейсі типізування відбувається через шаблон. Відповідно у методах визначених безпосередньо у дочірніх інтерфейсах є чітко визначеніми типи даних.

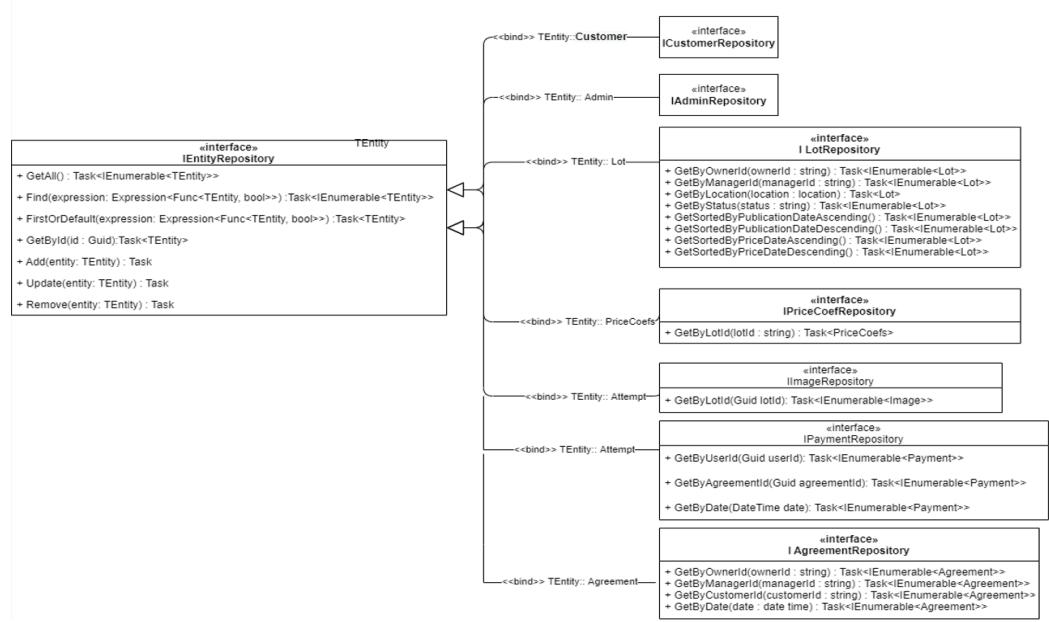


Рис. 3.2. Діаграма інтерфейсів

Інтерфейси для роботи з моделями визначені на рівні Data.Contract. Цей рівень задає інтерфейси для роботи з контекстом даних. Релізація інтерфейсів з цього рівня відбувається на рівні Data.

### 3.1.4. Архітектурна діаграма.

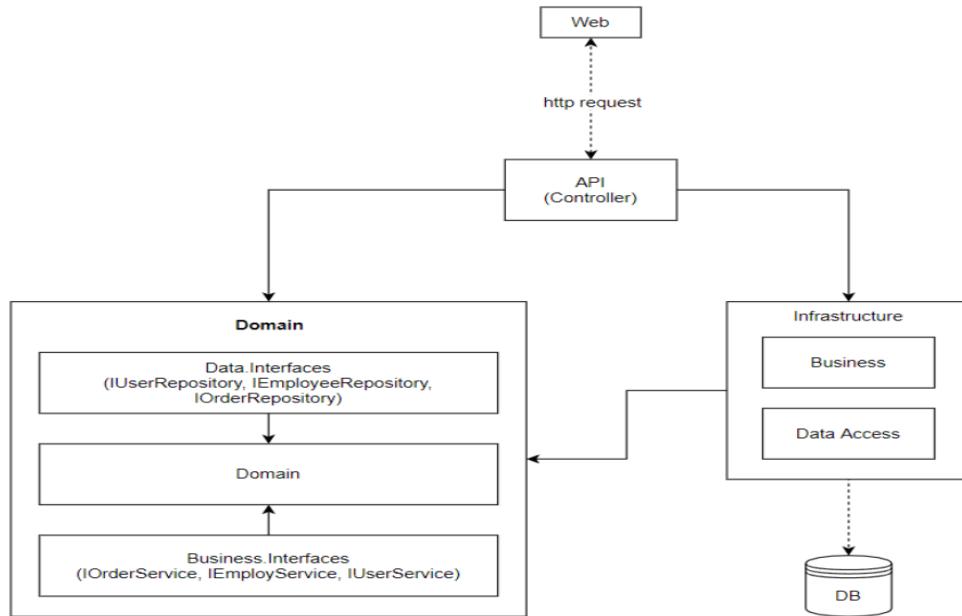


Рис. 3.3. Архітектурна діаграма

У процесі розробки серверної частини було використано onion архітектуру для кращої структуризації серверної частини. На найнижчому рівні знаходяться моделі. У них зберігається вся інформація системи. Для доступу до даних існує

наступний рівень доступу до даних. Він реалізований репозиторіями із методами доступу до моделей. Для зручності використання, репозиторії об'єднано в одиницю використання. Це архітектурний патерн проєктування для полегшення роботи з репозиторіями. Також цей патерн дозволяє спростити впровадження залежностей до репозиторіїв у сервісах. Бізнес логіка додатку знаходитьться на рівні бізнесу. У ньому відбуваються логічні операції прийняті з вищих рівнів і відбувається використання репозиторіїв з одиниці використання. На найвищому рівні є кінцеві точки, так звані контроллери. У них приймаються запити від клієнтської частини і викликаються функції з рівня бізнес логіки [17].

### 3.1.5. Діаграма розгортання.

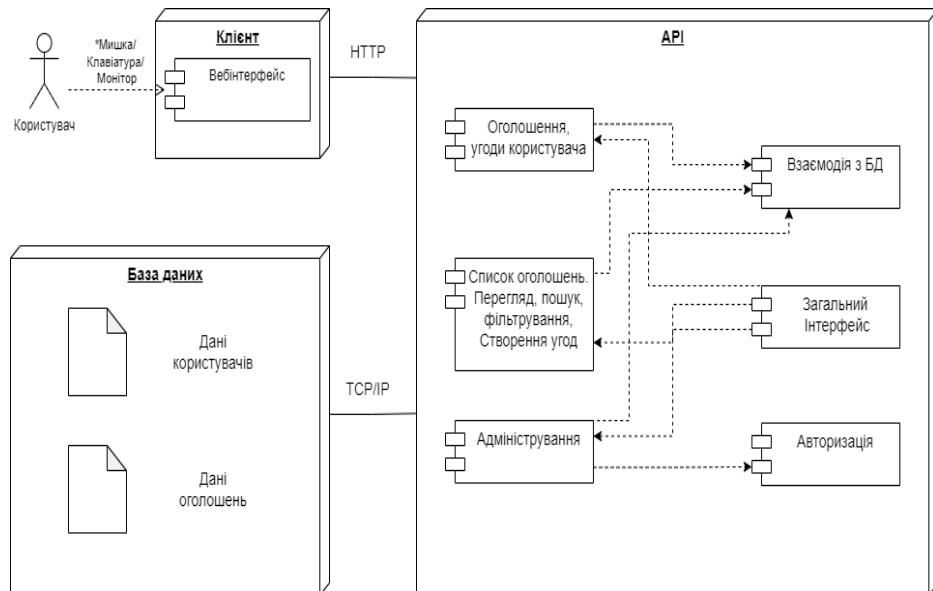


Рис. 3.4. Діаграма розгортання

На діаграмі розгортання відображено основні компоненти, які будуть працювати у системі. Згідно діаграми, у системі повинні бути 3 основних модуля:

Клієнт (задає веб інтерфейс для користувача);

API (працює із запитами від клієнту і працює з доступом до бази даних);

База даних (є сховищем для даних системи).

Відповідно вище перераховані модулі мають іще додаткові підмодулі. У API модулі, існують:

- підмодуль оголошень та угод;

- підмодуль взаємодії з базою даних
- підмодуль оголошень, який дозволяє зручно працювати із списками оголошень
- підмодуль адміністрування

### 3.2. Проектування бази даних до веб системи для продажу та оренди земельних ділянок та нерухомості.

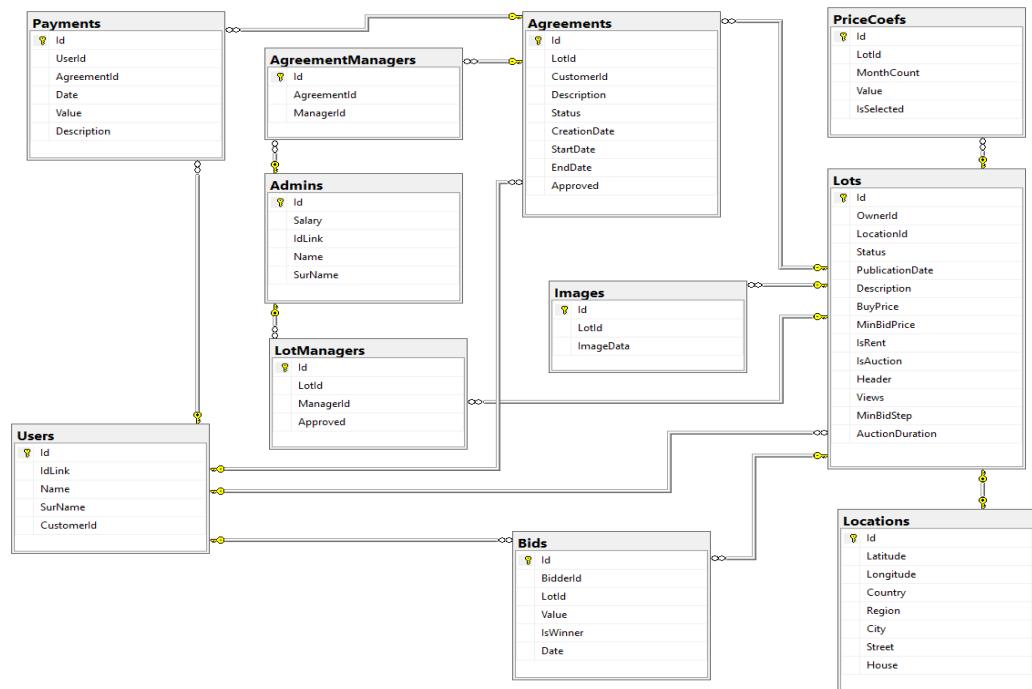


Рис. 3.5. Діаграма бази даних

Відповідно до тематики і поставлених задач, виникає потреба у реляційній базі даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня. Для збереження даних у системі, було обрано реляційну базу даних Microsoft SQL Server 2019 [19]. Згідно заданої тематики цей тип баз даних дуже добре підходить під задані задачі. Опис бази даних.

Таблиця Lot – відповідає за збереження даних про оголошення.

Містить такі поля:

Id – унікальний ідентифікатор оголошення в системі;

OwnerId(FK) – ідентифікатор власника оголошення (посилання на таблицю User);

Location(FK) – місцезнаходження об'єкта з оголошення (посилання на таблицю Location);

Status – статус оголошення;

Publication Date – дата створення оголошення;

Header – заголовок оголошення;

Description – опис оголошення;

BuyPrice – вартість об'єкту з оголошення;

MinBidValue – мінімальна ставка на аукціоні;

MinBidStep – мінімальний крок підняття ставки на аукціоні;

AuctionDuration – тривалість аукціону;

isRent – прапорець, який означає можливість оренди;

isAuction – прапорець, який означає можливість торгів;

Views – число переглядів.

Таблиця LotManager – забезпечує зв'язок багато до багатьох для таблиць Lot та Admin

Містить такі поля:

Id - унікальний ідентифікатор в системі;

LotId - ідентифікатор оголошення (посилання на таблицю Lot);

ManagerId - ідентифікатор менеджера (посилання на таблицю Admin);

Approved – прапорець, який є вказівником що оголошення дозволене менеджером.

Таблиця PriceCoef – відповідає за збереження даних про вартість оренди в залежності від тривалості оренди.

Містить такі поля:

Id – унікальний ідентифікатор в системі;

LotId(FK) – ідентифікатор оголошення (посилання на таблицю Lot);

DaysCount – кількість днів;

Value – вартість оренди.

Таблиця Image – відповідає за збереження даних про фото оголошення.

Містить такі поля:

**Id** – унікальний ідентифікатор в системі;

**LotId(FK)** – ідентифікатор оголошення (посилання на таблицю Lot);

**ImageData** – дані фото у форматі Base64.

Таблиця Agreement – відповідає за збереження даних про угоди.

Містить такі поля:

**Id** – унікальний ідентифікатор угоди в системі;

**CustomerId(FK)** – ідентифікатор клієнта (посилання на таблицю User);

**Status** – статус угоди;

**Description** – опис оголошення;

**CreationDate** – дата створення угоди і початку ;

**StartDate** – дата початку дії угоди;

**EndDate** – дата закінчення дії угоди;

**Approved** – поле відповідальне за прийняття угоди у випадку прийняття приймає true і у випадку неприйняття false.

Таблиця AgreementManager – забезпечує зв'язок багато до багатьох для таблиць Lot та Admin

Містить такі поля:

**Id** - унікальний ідентифікатор в системі;

**AgreementId** - ідентифікатор угоди (посилання на таблицю Agreement);

**ManagerId** - ідентифікатор менеджера (посилання на таблицю Admin);

Таблиця Bid – відповідає за збереження даних ставки користувачів.

Містить такі поля:

**Id** – унікальний ідентифікатор в системі;

**BidderId(FK)** – ідентифікатор власника зробленої ставки (посилання на таблицю User);

**Value** – розмір ставки;

**IsWinner** – пропорець, який означає чи ставка виграла;

**Date** – дата ставки.

Таблиця User – відповідає за збереження даних про користувачів системи.

Містить такі поля:

Id – унікальний ідентифікатор в системі;

IdLink – ідентифікатор персони (посилання на таблицю бази даних Identity).

Name – ім’я;

SurName – прізвище.

Таблиця Admin – відповідає за збереження даних про адміністраторів системи.

Містить такі поля:

Id – унікальний ідентифікатор в системі;

IdLink – ідентифікатор персони (посилання на таблицю бази даних Identity).

Name – ім’я;

SurName – прізвище;

Salary – поле із сумою заробітньої плати адміністратора системи.

Таблиця Location – відповідає за збереження даних про розташування.

Містить такі поля:

Id – унікальний ідентифікатор в системі;

Latitude – широта;

Longitude – довгота;

Country – країна;

Region – регіон;

City – місто;

Street – вулиця;

House – номер будинку.

Таблиця Person – відповідає за збереження даних про персону.

Містить такі поля:

Id – унікальний ідентифікатор в системі;

PassWord – пароль;

PhoneNumber – номер телефону;

Email – адреса електронної пошти.

Таблиця Payment – відповідає за збереження даних про оплату.

Містить такі поля:

Id – унікальний ідентифікатор в системі;

UserId – ідентифікатор персони (посилання на таблицю User);

AgreementId – ідентифікатор угоди (посилання на таблицю Agreement)

Description – опис оплати;

Date – дата проведення оплати;

Value – розмір оплати.

Для менеджменту користувачів системи додано другу базу даних під назвою LandSellingIdentity. В її основі використана бібліотека аутентифікація Microsoft Identity [21]. Це підхід значно спростило роботу з користувачами системи, зокрема значно полегшив реалізацію аутентифікації користувача в системі.

### **3.3 Проектування інтерфейсу користувача веб системи для продажу та оренди земельних ділянок та нерухомості.**

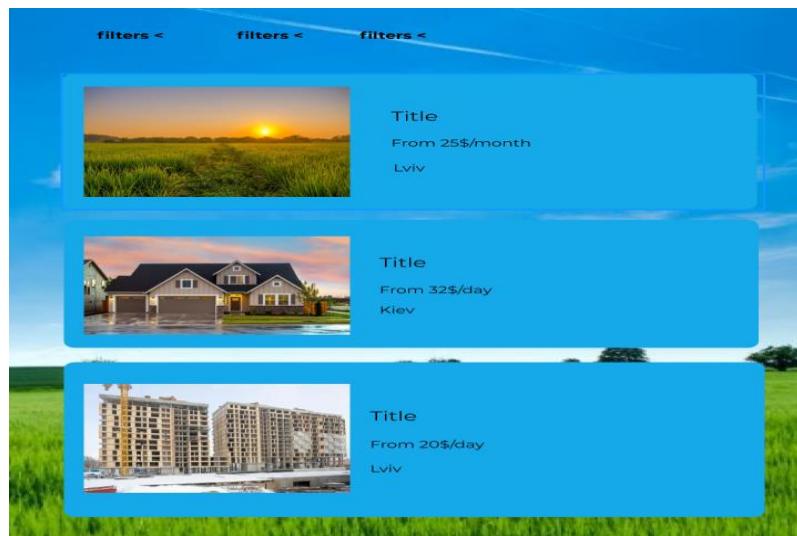


Рис. 3.6. Мокап сторінки перегляду списку оголошень

Відповідно до попередньо проведених оцінок предметної області, виникла необхідність у наступному списку сторінок:

- Головна сторінка із списком оголошень, із можливістю фільтрування та сортування оголошень;

- Сторінка створення нових оголошень;
- Сторінка перегляду детальної інформації оголошення;
- Сторінка створення угоди згідно оголошення;
- Модальні вікна для авторизації та реєстрації;
- Сторінка перегляду списку користувачів системи;
- Сторінка редагування/додавання користувачів.

Для вище перерахованого списку сторінок було розроблено вейрфрейми та мокапи (Додаток В) для візуалізації інтерфейсу користувача на процесі проєктування.

Для розробки було використано інструмент Figma [12].

Під час написання клієнтської частини та інтерфейсу користувача розробка велася на основі створених мокапів. Також використовувалися графічні бібліотеки та інші інструменти для полегшення роботи з інтерфейсом.

### **3.4. Висновки.**

У межах цього розділу було розроблено та спроектовано UML діаграми розроблюваної веб системи. Розроблені діаграми допоможуть на етапі написання програмного коду і заощадять багато часу, так як було спроектовано програмно систему на різних рівнях і у різних випадках. Було розроблено основу архітектури системи та спроектовано основні рішення для реалізації заданої системи відповідно предметній області. Кожну діаграму було описано та пояснено основні архітектурні рішення на них.

Також для збереження даних системи, було спроектовано діаграму бази даних. Було визначено основні таблиці та зв'язки між ними. Також було описано кожну сутність, її призначення і кожен зв'язок у таблиці.

Для полегшення розробки інтерфейсу користувача було спроектовано вейрфреми та мокапи. Для розробки інтерфейсу користувача використовувався інструмент Figma [12].

Для проєктування UML діаграм використовувалася десктопна версія інструменту Draw.io [14].

## РОЗДІЛ 4. РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ ДЛЯ ПРОДАЖУ ТА ОРЕНДИ ЗЕМЕЛЬНИХ ДІЛЯНОК ТА НЕРУХОМОСТІ

### 4.1. Реалізація серверної частини веб системи для продажу та оренди земельних ділянок та нерухомості.

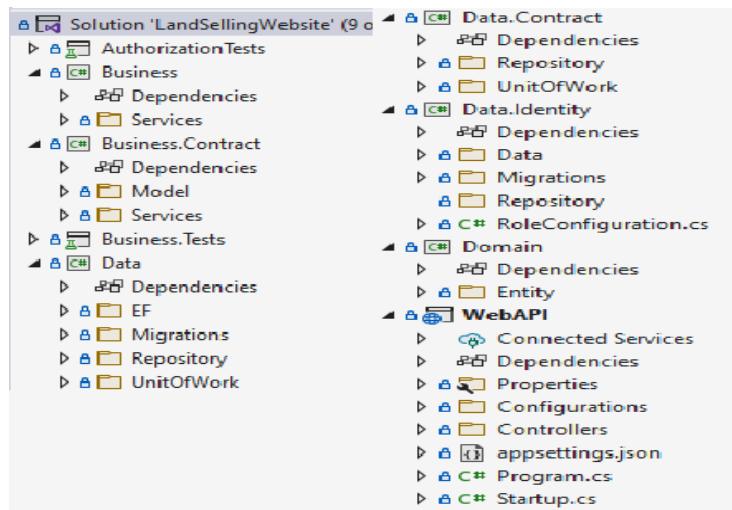


Рис. 4.1. Структура серверної частини веб системи

Серверна частина системи складається з декількох проектів, які знаходяться в одному рішенні. Це пов'язано з тим, що серверна частина має багатошарову архітектуру. Таке рішення добре підходить під поставлені задачі і дозволяє дуже добре розділити код на різні рівні і досягти гарної і зрозумілої ієрархії. На серверній частині використовується дві бази даних. LandSellingContext використовується для збереження даних про оголошення і менеджмент цих оголошень. LandSellingIdentityContext використовується для збереження даних про користувачів системи.

На найнижчому рівні знаходиться проект Domain. У ньому описано моделі. На основі цих моделей будується база даних для збереження інформації у системі. Кожна модель має свій набір властивостей і є нащадком від базової моделі. У базовій моделі описано головні і спільні властивості для всіх моделей.

Для доступу до моделей існує рівень Data. На цьому рівні описано класи і методи, які реалізують доступ до моделей через контекст бази даних(репозиторії).

Також на цьому рівні існує клас одиниця роботи. Це архітектурний патерн програмування для збереження посилань на репозиторії. Таке рішення дозволяє використати впровадження залежностей у бізнес-логіці додатка. Також на рівні Data існує контекст бази даних для роботи з оголошениями.

Рівень Data.Contract задає інтерфейс, який повинен реалізовувати рівень Data. Рівень Data наче підписується на рівень „контракту” і повинен його реалізовувати.

У проекті Data.Identity міститься контекст бази даних для роботи з користувачами системи.

Над рівнем Data існує рівень Business. Це рівень бізнес-логіки. У ньому прописано всю логіку для реалізації CRUD операцій та інших необхідних логічних операцій для роботи веб системи.

Рівень бізнес-логіки реалізує інтерфейси рівня Business.Contract. На цьому рівні в інтерфейсах описуються методи, необхідні на рівні бізнес-логіки. Також на цьому рівні визначені моделі представлення. Вони слугують допоміжними моделями для рівня бізнес логіки і використовуються коли для функціоналу не потрібні зайні властивості з „моделей оригіналів” на рівні Domain.

На найвищому рівні у рішенні серверної частини знаходиться рівень кінцевих точок. Цей рівень має назву WebAPI. На цьому рівні знаходяться всі контролери, які приймають запити від клієнтської частини. На цьому рівні в контроллерах відбувається виклик функцій з рівня бізнес-логіки. Також на цьому рівні знаходяться файли конфігурації проекту. До файлів конфігурації відносяться: файли для мап між моделями з рівня Domain та моделей визначених на рівні бізнес-логіки, файли для ініціалізації впровадження залежностей, файли з заголовками безпеки та інші файли конфігурації.

#### **4.2. Реалізація клієнтської частини веб системи для продажу та оренди земельних ділянок та нерухомості.**

Розробка клієнтської частини почалася з написання головної сторінки із списком оголошень (Додаток Е.1) та сторінка створення нового оголошення (Додаток Е.2). Для цього було підключено відповідні бібліотеки для використання

сторонніх API та було розроблено інтерфейс користувача відповідно до заздалегідь розроблених вейрфреймів та мокапів. Було написано логіку для створення запитів до серверної частини. Для цього всього було написано відповідні реакт-компоненти для реалізації поставлених завдань.

Після реалізації головних сторінок, було продовжено реалізацію інших сторінок, по аналогії до початкових. Для взаємодії з кінцевими точками на серверній частині і відповідно бізнес логікою, було написано компоненти-сервіси, які реалізують різні за типами HTTP запити.

В системі присутні два класи користувачів: адміністратор та звичайний користувач. Для цього було реалізовано контроль ролями на клієнтській та серверній частинах.

#### 4.3. Опис роботи системи.

При вході на клієнтську частину користувача зустрічає головна сторінка зі списком доступних оголошень про оренду чи продаж земельних ділянок чи нерухомості (Додаток Е.1). На цьому вікні користувач може ознайомитися із доступними оголошеннями та відсортувати список за різними критеріями.

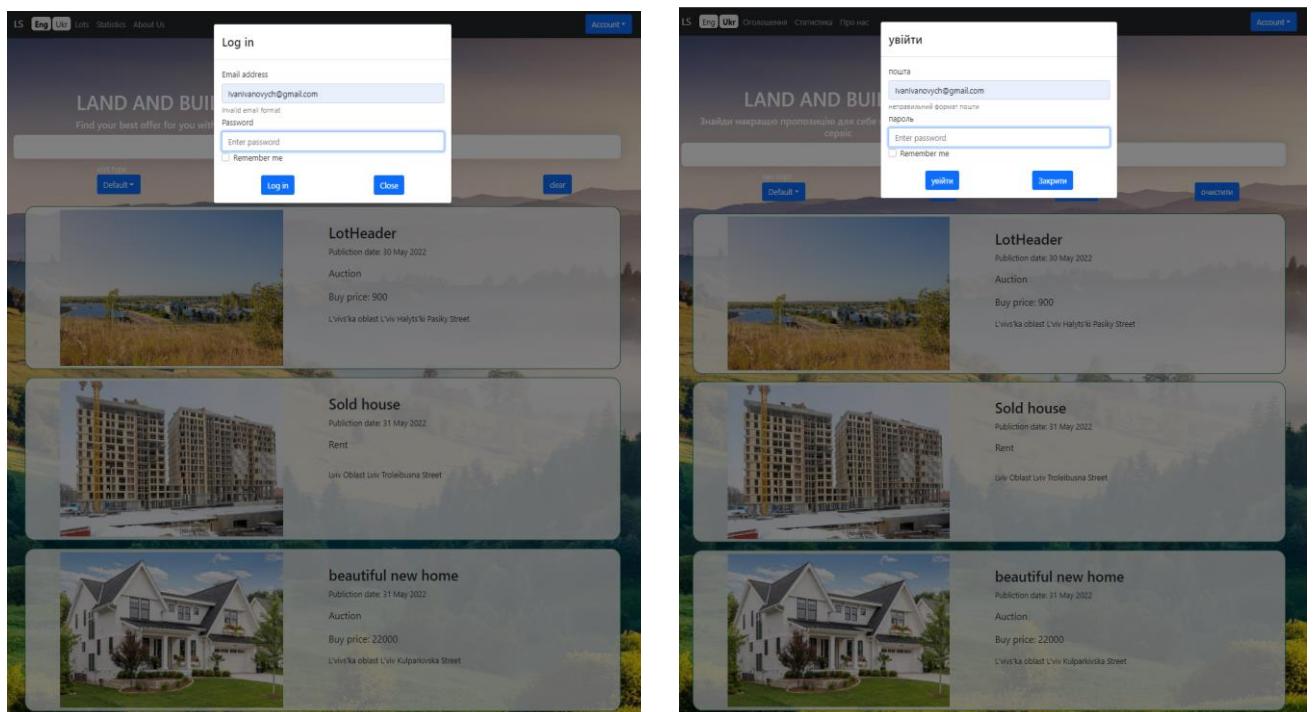


Рис. 4.2. Головна сторінка з відкритим вікном входу в систему

При натисненні на будь-яке оголошення користувач переходить на сторінку з детальним описом цього оголошення (Додаток Е.3). На цій сторінці користувач може ознайомитися з детальною інформацією про вибране оголошення.

Якщо користувач авторизований у системі, йому доступні можливості зв'язатися з автором оголошення, поставити ставку якщо це оголошення бере участь в торгах, також можна створити запит на створення угоди на основі оголошення.

Для авторизації необхідно вибрати відповідний пункт у верхній частині сторінки. Під час авторизації необхідно ввести свій логін та пароль. У випадку якщо користувач не має свого облікового запису в системі - необхідно зареєструватися у відповідному вікні або додано новий аккаунт може адміністратор сайту.

Після успішної авторизації користувачу як звичайному користувачу доступні можливості ставити ставки у торгах, створювати нові оголошення, укладати угоди на основі оголошень, робити оплату по укладених угодах. Після успішної авторизації користувачу як адміна, йому доступні додаткові можливості менеджменту облікових записів користувачів (Додаток Е.5) та доступний менеджмент оголошень.

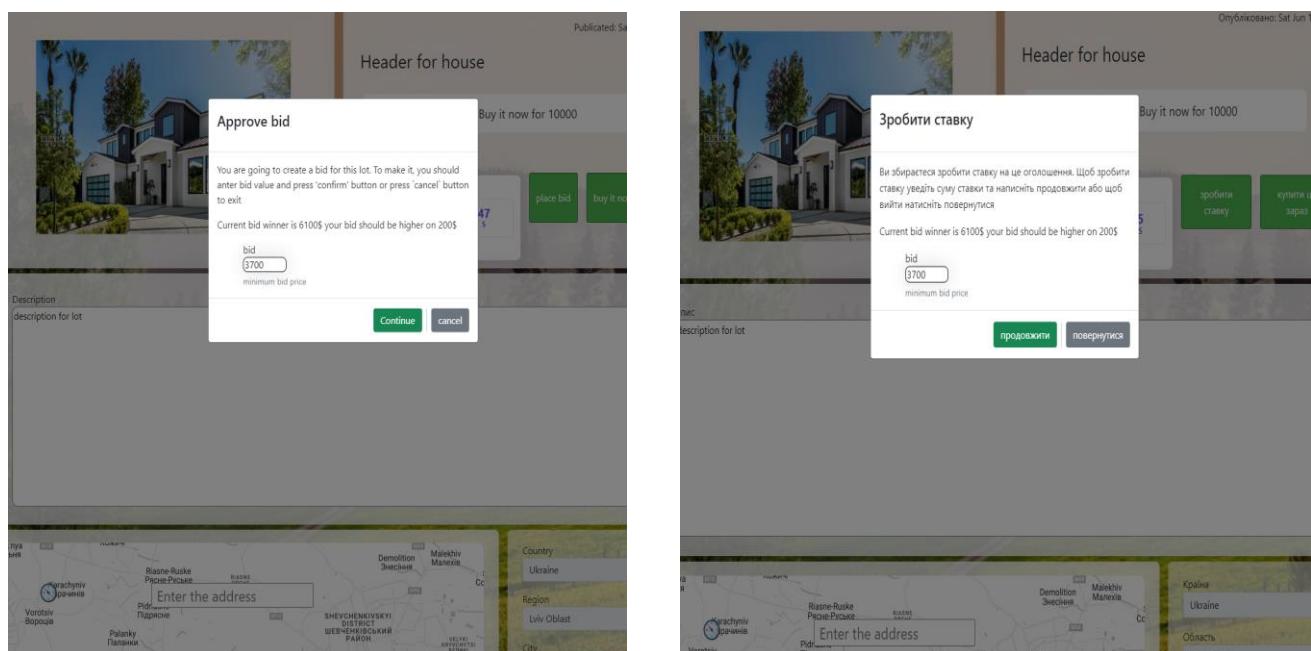


Рис. 4.3. Можливість зробити ставку

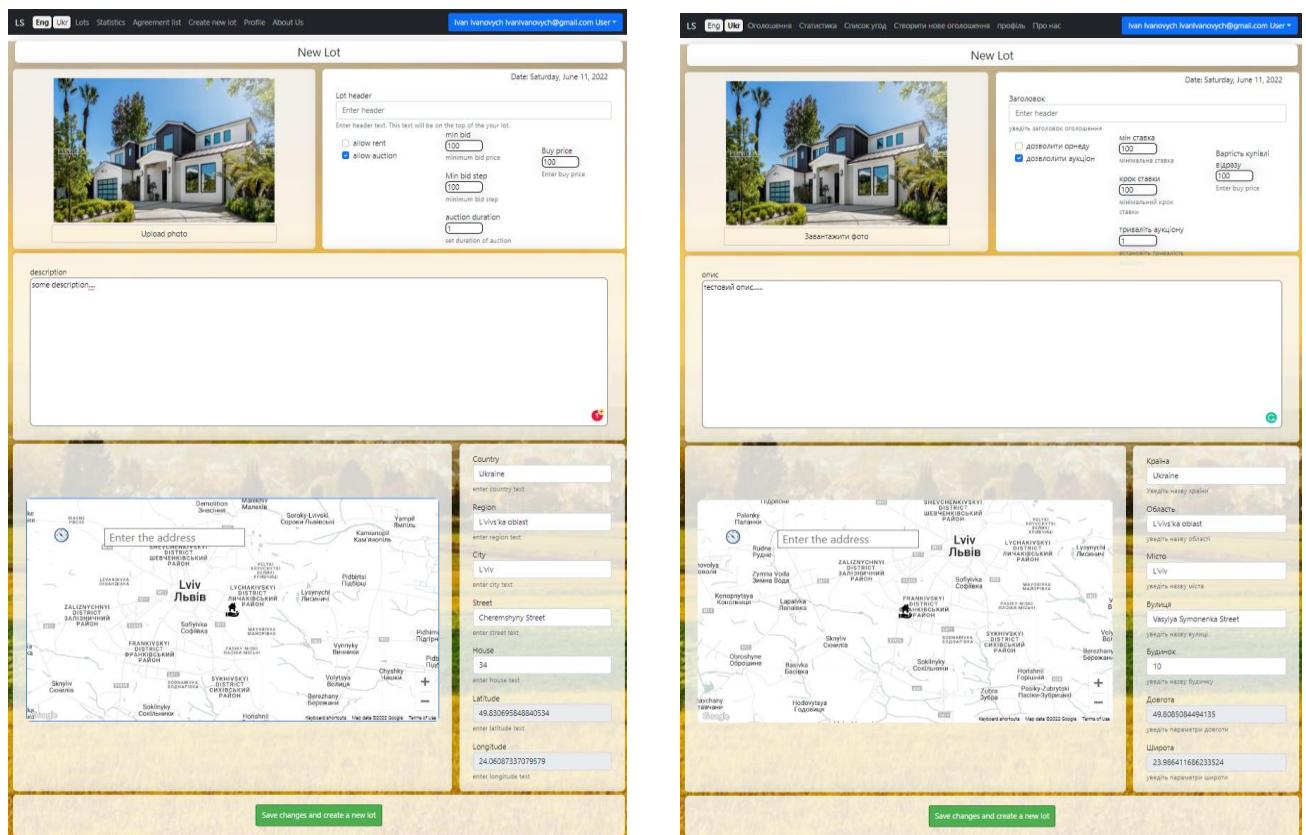


Рис. 4.4. Створення нового оголошення

Для створення нових оголошень (Додаток Е.2), авторизованому користувачу необхідно вибрати лінк “створити оголошення” на верхній частині сторінки, після чого відбувається перехід на сторінку створення нового оголошення. На цій сторінці користувач вносить дані про оголошення: фото об’єкта, назву, опис, задає параметри аукціону або оренди та вносить адресу об’єкта. Що до адреси, користувач може вносити адресу самостійно або може вибрати місцезнаходження на карті Google. Користувач підтверджує своє оголошення. Як результат відображається повідомлення про успішність створення нового оголошення і нове оголошення відображається у списку доступних оголошень.

Для укладання угоди, авторизованому користувачу необхідно натиснути на створення угоди оренди або на кнопку купівлі на сторінці перегляду оголошення. Після цього буде сформовано запит на створення нової угоди. Після збору відповідних документів та домовленостей з автором оголошення, автор має можливість прийняти запит на створення угоди. Як результат клієнту відображається угода у його списку угод.

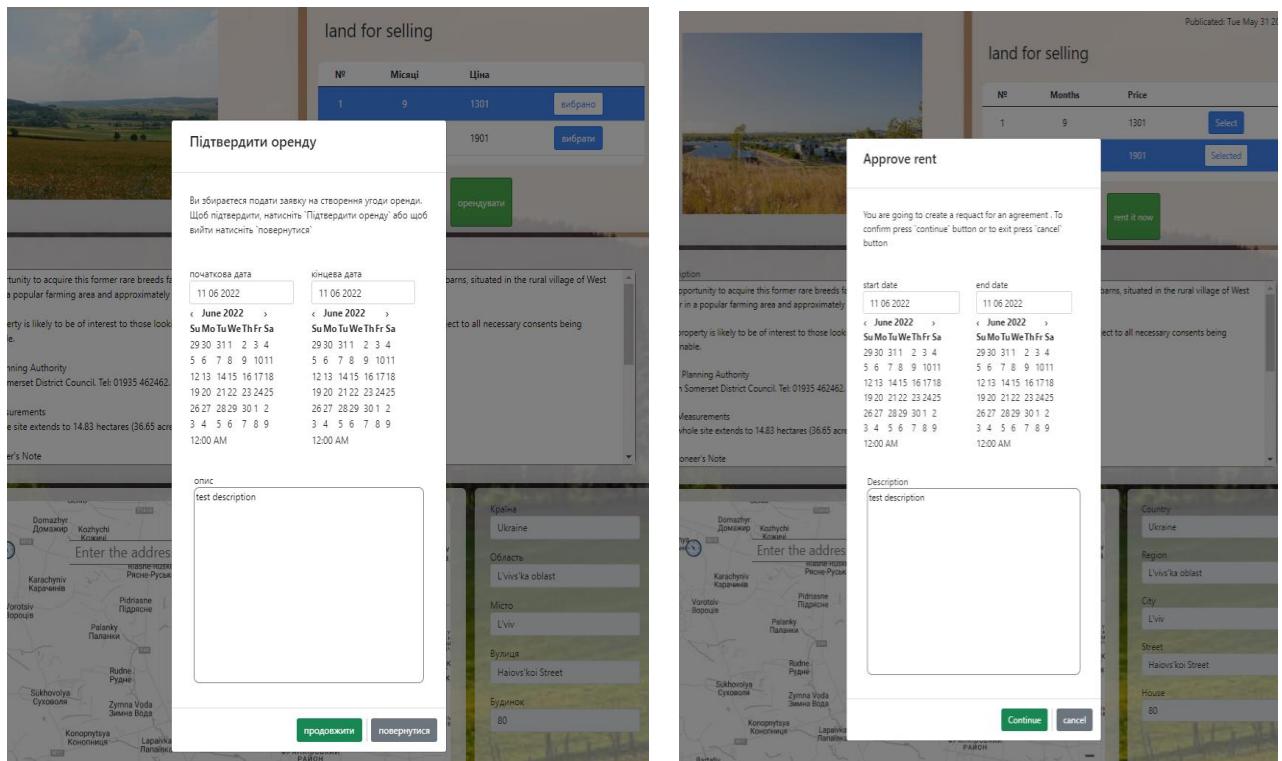


Рис. 4.5. Створення запиту на створення угоди оренди

Якщо перейти за посиланням у верхній частині робочого вікна та вибрати список укладених угод. Користувачу доступна можливість ознайомитися з угодою та оплатити або зробити запит на скасування угоди (Додаток Е.7).

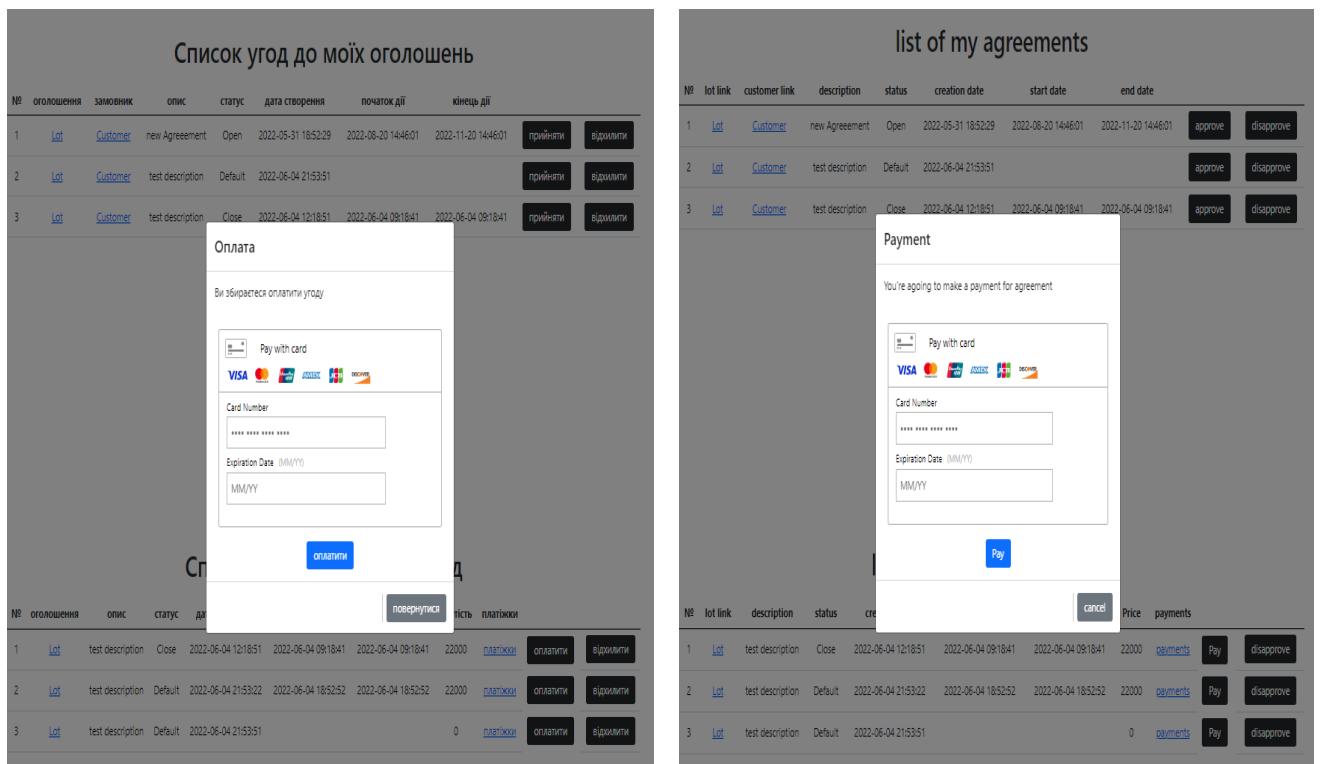


Рис. 4.6. Оплата згідно угоди

Для оплати, згідно угоди, необхідно вибрати відповідну угоду та натиснути кнопку «оплатити». Після натискання з'явиться відповідне вікно для оплати. Користувач вносить дані своєї банківської картки та підтверджує оплату. Після проходження оплати користувачу відобразиться вікно про успішність проведення операції.

#### **4.4. Проведення тестових випадків для веб системи для продажу та оренди земельних ділянок та нерухомості.**

Для тестування веб системи для продажу та оренди земельних ділянок та нерухомості було використано функціональне тестування, тестування продуктивності засобами Visual Studio та Task Manager, також було проведено ручне тестування користувацького інтерфейсу.

##### **4.4.1. Функціональне тестування.**

Юніт-тести було написано для серверної частини розробленої веб системи. Тестами було покрито бізнес шар серверної частини. Було написано декілька варіантів використання із різними тестовими даними (Таблиця 4.1.)

Таблиця 4.1.

Варіанти використання для функціонального тестування

| Варіанти використання                                  | Тестові випадки | Тестові дані |
|--|-----------------|--------------|
| Створення нових оголошень                              | 1               | 3            |
| Редагування, видалення, отримання даних про оголошення | 1               | 4            |
| Укладання угод на оренду або купівлю                   | 1               | 3            |
| Завантаження, кодування та декодування фото            | 1               | 5            |
| Створення нової ставки (можливість роботи аукціону)    | 1               | 3            |
| Проведення оплати                                      | 1               | 3            |
| Разом  | 6               | 21           |

Усі юніт-тести було написано у окремому проекті під назвою Business.Tests.

Більшість тестів було написано по шаблону AAA із використанням бібліотеки Moq для тестування поведінки та для тестування стану (створення стабів та моків), також було використано бібліотеки Xunit або Nunit.

Структура та опис тестів. Оскільки відбувається тестування високого рівня бізнес логіки, спочатку потрібно ініціалізувати роботу нижчих рівнів у системі. Відповідно до шаблону сперу відбувається ініціалізація нижчих рівнів. Спочатку ініціалізується ріень даних (репозиторії та об'єднання реопзиторів в одиницю роботи), потім створюється новий екземпляр класу сервісу.

```
[SetUp]
0 references
public void SetUp(string longitude, string latitude)
{
    //Arrange
    ILocationRepository stubLocationRepository =
        Mock.Of<ILocationRepository>(lr => lr.Add(It.IsAny<Location>()) == Task.CompletedTask &&
                                         lr.GetByLongitudeAndLatitude(longitude, latitude) == Task.FromResult(locationId));

    ILotRepository stubLotRepository =
        Mock.Of<ILotRepository>(lr => lr.Add(It.IsAny<Lot>()) == Task.CompletedTask &&
                                         lr.GetById(It.IsAny<Guid>()) == Task.FromResult(lot) &&
                                         lr.GetAll() == Task.FromResult(lots));

    ILotUnitOfWork unitOfWork =
        Mock.Of<ILotUnitOfWork>(of => of.ImageRepository == null &&
                                         of.LotRepository == stubLotRepository &&
                                         of.LocationRepository == stubLocationRepository &&
                                         of.UserRepository == null &&
                                         of.AdminRepository == null &&
                                         of.BidRepository == null &&
                                         of.PriceCoeRepository == null &&
                                         of.AgreementRepository == null);

    lotService = new LotService(mapper, unitOfWork);
}
```

Рис. 4.7. Приклад ініціалізації у тесті

Після ініціалізації нижчих рівнів і рівня бізнес-логіки відбувається етап дій. На цьому етапі відбувається виклик методу з рівня бізнес локіки.

```
[Test]
0 references
public void GetAll()
{
    // Act
    var result = lotService.GetAll();

    // Assert
    Assert.That(result.Result as IEnumerable<UpdateLotDTO>, Is.EqualTo(lotDTOs));
}
```

Рис. 4.8. Приклад виклику тестового метода та перевірка на правильність результатів

Останнім етапом є перевірка правильності отриманих результатів після виклику метода з рівня бізнес-логіки. Для цього викликається відповідний статичний метод для порівняння або інших операцій з області видимості класу Assert.

#### **4.4.2. Тестування інтерфейсу.**

Було проведено тестування користувальського інтерфейсу. Також протестовано головний і відповідно найважливіший функціонал клієнської частини веб системи (Таблиця 4.2.). Тобто тестування покриває головний сценарій корисутувача для використання системи від входу в систему, перегляду списку оголошень, вибір оголошення, участь в торгах і укладання угод на основі оголошень, оплата угод та вихід з системи.

Таблиця 4.2.

Варіанти використання для тестування інтерфейсу

| Варіант використання  | Тестові дані |
|---|--------------|
| Перевірка екрану профілю                                      | 4            |
| Створення нових оголошень                                     | 5            |
| Перевірка можливості сортування та перегляду списку оголошень | 2            |
| Перегляд детальної інформації стосовно оголошенні             | 3            |
| Укладання угод  | 2            |
| Перегляд та редагування профіля користувача                   | 3            |
| Перегляд списку укладених угод                                | 2            |
| Перегляд списку створених оголошень                           | 3            |
| Створення нової ставки  | 1            |
| Оплата угоди  | 2            |
| Разом   | 27           |

#### **4.5. Висновки.**

У цьому розділі було реалізовано розроблювану веб систему для продажу та оренди нерухомості та земельних ділянок.

Написано код для реалізації серверної та клієнтської частин веб системи. На серверній частині було написано базу даних за підходом „спочатку код”, відповідно було написано рівень Domain і на основі цього рівня і встановлених залежностей у файлі контексту було згенеровано базу даних.

Було реалізовано архітектурні рішення та проектні рішення розроблені у попередніх етапах розробки для двох частин системи. Також було описано роботу програми та її основні елементи.

Для стабільності роботи було проведено тестування серверної та клієнтської частин веб системи. Для серверної частини було написано Unit-тести а для клієнтської частини було проведено ручне тестування Також на цьому етапі було розроблено інструкцію користувача (Додаток Г).

## РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

### **5.1. Економічна характеристика веб системи для продажу та оренди земельних ділянок та нерухомості.**

Метою цієї бакалаврської роботи є створення веб системи для продажу та оренди земельних ділянок та нерухомості із використанням кабінетів користувачів, створенням оголошень, створенням угод, надаванням послуг продажу, аукціону чи оренди. Ця веб система повинна надавати простоту у використанні та інформативність для користувачів.

Серед найпопулярніших методів пошуку нерухомості чи земельних ділянок є пошук на сайтах з оголошеннями. Дана веб система дозволяє створювати інформативні оголошення відповідно до заданої предметної області.

Проект створений у вигляді сайту, який можна відвідати у будь-якому веб переглядачі. Тому цей веб ресурс повинен мати популярність серед користувачів різних пристройів.

### **5.2. Інформаційне забезпечення та формування гіпотези щодо потреби розроблення веб системи для продажу та оренди земельних ділянок та нерухомості.**

Продаж та оренда нерухомості – це мабудь один з найдавніших ринків у світі. Іще здавна одні люди шукали собі житло, а інші продавали за винагороду. Актуальність цієї предметної області завжди буде високою. Для побудови нерухомості потрібні земельні ділянки. Інформацію про це важко знайти в інтернеті. Подібні оголошення зазвичай знаходяться на сайтах, які є дуже універсальними і не дозволяють у повній мірі описати оголошення. Зазвичай користувачу недоступно додавання кадастрових карт до оголошення, оцінку землі та інших важливих характеристик.

Для купівлі земель господарського призначення, в Україні дуже мала кількість веб ресурсів для пошуку даного типу оголошень. Ринок землі є новим для

України. Донедавна було заборонено продавати землі господарського призначення. Тепер після зняття заборони у нас відкривається новий ринок. І дане програмне забезпечення повинне забезпечити зручний досвід для користувача у пошуку оголошень даної предметної області та відповідно створення угод на основі оголошень.

Дана програмна система повинна об'єднати доступний функціонал з сайтів-аналогів та повинна надавати кращий сервіс для користувачів. До найголовнішого функціоналу можна віднести: створення оголошень про купівлю чи продаж будинків чи іншої нерухомості, а також земельних ділянок під забудову та земельних ділянок під господарське призначення. Серед найбільш схожих по функціоналу програм були мною порівняні веб системи: olx, Українську універсальна біржа, dom ria, eBay, amazon. Всі перелічені мають частину функціоналу розробленої веб системи. Але у них існують деякі недоліки, які описані у попередніх розділах.

### **5.3. Оцінювання та аналізування факторів внутрішнього та зовнішнього середовища для веб систем для продажу та оренди земельних ділянок та нерухомості.**

Цей розділ має на меті показати оцінку та аналіз компонентів зовнішнього та внутрішнього середовищ групою експертів.

Зовнішні компоненти мають оцінки в межах [-5; 5]. Межі шкали відображають найбільший позитивний або негативний вплив середовищ на даний проект.

Внутрішні проекти мають оцінки в межах [0; 5]. Нижня межа представляє занедбаність, відсутність чи поганий стан елемента. Верхня межа демонструє високий рівень організації.

Кожен компонент має свій певний фактор вагомості, порахований за допомогою певних коефіцієнтів. Сума компонентів дорівнює одиниці. Зважений рівень впливу факторів можна розрахувати як добуток впливу елемента в балах та рівня вагомості. Далі наведено результати обрахунків (Таблиця 5.1).

Таблиця 5.1.

Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ

| Фактори   | Середня експертна оцінка, бали | Середня вагомість факторів | Зважений рівень впливу, бали |
|---|--------------------------------|----------------------------|------------------------------|
| <i>Фактори зовнішнього середовища</i>             |                                |                            |                              |
| Споживачі   | 5                              | 0,11                       | 0,55                         |
| Постачальники                                     | 4                              | 0,1                        | 0,4                          |
| Конкуренти  | -3                             | 0,1                        | -0,3                         |
| Державні органи влади                             | 2                              | 0,05                       | 0,1                          |
| Інфраструктура                                    | 3                              | 0,06                       | 0,18                         |
| Законодавчі акти                                  | -2                             | 0,1                        | -0,1                         |
| Профспілки, партії та інші громадські організації | 2                              | 0,05                       | 0,1                          |
| Система економічних відносин в державі            | -2                             | 0,06                       | -0,06                        |
| Організації-сусіди                                | 2                              | 0,01                       | 0,02                         |
| Міжнародні події                                  | 1                              | 0,01                       | 0,01                         |
| Міжнародне оточення                               | 1                              | 0,03                       | 0,03                         |
| Науково-технічний прогрес                         | 3                              | 0,07                       | 0,21                         |
| Політичні обставини                               | 3                              | 0,06                       | 0,18                         |
| Соціально-культурні обставини                     | 2                              | 0,05                       | 0,1                          |
| Рівень техніки та технологій                      | 4                              | 0,04                       | 0,16                         |
| Особливості міжнародних економічних відносин      | 3                              | 0,02                       | 0,06                         |
| Стан економіки                                    | 5                              | 0,08                       | 0,4                          |
| Загальна сума                                     | 33                             | 1                          | 2,04                         |

## Продовження таблиці 5.1

| <i>Фактори внутрішнього середовища</i> |    |      |      |
|--|----|------|------|
| Цілі                                   | 5  | 0,11 | 0,55 |
| Структура                              | 4  | 0,16 | 0,64 |
| Завдання                               | 4  | 0,07 | 0,28 |
| Технологія                             | 4  | 0,2  | 0,8  |
| Працівники                             | 3  | 0,21 | 0,63 |
| Ресурси                                | 3  | 0,25 | 0,75 |
| Загальна сума                          | 23 | 1    | 3,65 |

Із результатів проведеного експертного оцінювання впливу факторів зовнішнього середовища можна зробити наступні висновки:

Найбільш позитивний вплив мають споживач та постачальник. Це пов'язано з тим що споживач диктує цінову політику і він робить попит на дану продукцію. Тобто в залежності від оцінок споживача буде залежати успішність реалізації даного продукту. Стан економіки теж є дуже важливим. В залежності від цінового благополуччя споживачів буде залежати вірогідність успішної реалізації даного продукту. Також важливим можна вважати рівень техніки та технологій. Важливість цих чинників випливає з залежності розробки продукту від використовуваних технологій. Старіші технології важче використовуються і на них важче знайти відповідного спеціаліста – відповідно дорожча розробка програмного продукту. Що є дуже важливим при розробці.

Що до негативних факторів, найбільш негативним можна вважати законодавчі акти та систему економічних відносин в державі.

Оцінюючи фактори внутрішнього середовища, найбільший вплив мають цілі, структура та технології. Найменший вплив мають працівники та ресурси.

Порівнюючи фактори зовнішнього та внутрішнього середовищ, фактори внутрішнього середовища мають значно вищий рівень впливу на проект ніж фактори зовнішнього середовища. Тому в ході розробки та просування продукту слід приділити значну увагу саме факторам внутрішнього середовища.

## 5.4. Формування стратегічних альтернатив розвитку веб системи для продажу та оренди земельних ділянок та нерухомості.

### 5.4.1. Стратегічні альтернативи першої групи.

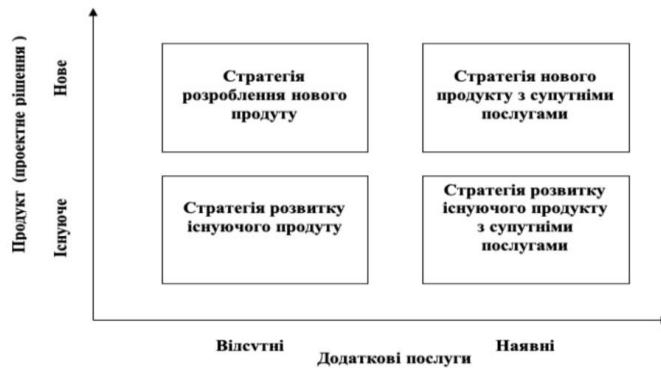


Рис. 5.1. Стратегічні альтернативи першої групи

Серед наведених стратегічних альтернатив першої групи (Рис. 5.1) було обрано стратегію нового продукту з супутніми послугами. Ця стратегія представляє собою розроблення нового програмного забезпечення. Також цей підхід передбачає пропонування нових додаткових послуг.

### 5.4.1. Стратегічні альтернативи другої групи.

Серед наведених стратегічних альтернатив другої групи (Рис. 5.2) було обрано стратегію розвитку продукту. Даної стратегії була обрана бо у межах даної проєкту планується створювати новий програмний продукт для існуючого сегменту ринку нерухомості та нововідкритого сегменту ринку земельних ділянок. Хоч цей стратегічний план є досить ризикованим, але на ринку існують перспективи потреби даного програмного забезпечення.



Рис. 5.2. Стратегічні альтернативи другої групи

**5.5.Бюджетування веб системи для продажу та оренди земельних ділянок та нерухомості.**

Таблиця 5.2.

Бюджет витрат матеріалів та комплектуючих виробів

| Назва матеріалів та комплектуючих | Марка, тип, модель              | Фактична кількість, шт. | Ціна за одиницю, грн. | Разом, грн. |
|-----------------------------------|---------------------------------|-------------------------|-----------------------|-------------|
| Комп'ютер                         | Процесор i5-10500               | 1                       | 4500,00               | 14998       |
|                                   | Оперативна пам'ять Kingston 8GB | 2                       | 800,00                |             |
|                                   | Материнська плата ASUS          | 1                       | 1500,00               |             |
|                                   | SSD Samsung 860EVO              | 1                       | 2699,00               |             |
|                                   | Відеокарта RX 570               | 1                       | 4500,00               |             |
|                                   | Інші пристрой                   | 5                       | 999,00                |             |
| Монітор                           | Samsung Odyssey G5 32``         | 1                       | 9999,00               | 9999,00     |
| Монітор                           | Samsung LF27G35TF 27``          | 1                       | 8000,00               | 8000,00     |
| Периферійні пристрої              | Миш A4Tech                      | 1                       | 250,00                | 2599,00     |
|                                   | Клавіатура A4Tech               | 1                       | 350,00                |             |
|                                   | Принтер Canon MP250             | 1                       | 1999,00               |             |
| Wifi-роутер                       | TP-LINK Archer A5               | 1                       | 1500,00               | 1500,00     |
| Разом:                            | —                               | 17                      | 37096,00              | 37096,00    |

Формула для визначення річної суми амортизації:

$$AB = PB - LB/T,$$

де AB – річна сума амортизаційних відрахувань, PB – первісна вартість, LB – ліквідаційна вартість, T – термін корисного використання об'єкта

$$Na = 100\% / T ,$$

де  $Н_а$  – норма амортизаційних відрахувань.

Термін корисного використання ноутбука, Wi-Fi роутера, монітора на 4 роки

Для комп’ютера:  $ПВ = 14\ 998$  грн,  $ЛВ = 6\ 000$  грн.

$$AB = (17\ 000 - 6000)/4 = 2\ 249,5 \text{ грн/рік.}$$

$$Н_а = 100\% / 4 = 25\%$$

Для монітора:  $ПВ = 9\ 999$  грн,  $ЛВ = 7\ 000$  грн.

$$AB = (9999 - 7000)/4 = 749,75 \text{ грн/рік.}$$

$$Н_а = 100\% / 4 = 25\%$$

Для периферійних пристроїв:  $ПВ = 2599$  грн,  $ЛВ = 500$  грн.

$$AB = (2599 - 500)/4 = 525 \text{ грн/рік.}$$

$$Н_а = 100\% / 4 = 25\%$$

Для роутера:  $ПВ = 1500$  грн,  $ЛВ = 1000$  грн

$$AB = (1500 - 1000)/4 = 125 \text{ грн/рік.}$$

$$Н_а = 100\% / 4 = 25\%$$

Таблиця 5.3.

**Бюджет витрат на оплату праці**

| Посада,<br>спеціаль-<br>ність    | Кількість<br>працівників,<br>осіб | Час<br>роботи,<br>дні | Денна заробітна<br>плата<br>працівників, грн. | Сума витрат<br>на оплату<br>праці, грн. |
|----------------------------------|-----------------------------------|-----------------------|---|---|
| <i>Основна заробітна плата</i>   |                                   |                       |   |   |
| Розробник ПЗ                     | 1                                 | 60                    | 850,00  | 51000,00                                |
| Тестер                           | 1                                 | 10                    | 350,00  | 3500,00                                 |
| Дизайнер                         | 1                                 | 10                    | 500,00  | 5000,00                                 |
| Разом                            | 3                                 | 80                    | 1700,00                                       | 56700,00                                |
| <i>Додаткова заробітна плата</i> |                                   |                       |   |   |
| Тестер                           | 1                                 | 6                     | 500   | 3 000,00                                |
| Дизайнер                         | 1                                 | 6                     | 833   | 5 000,00                                |
| Разом                            | 2                                 | 12                    | 1333  | 8 000,00                                |

Таблиця 5.4.

## Бюджет обов'язкових відрахувань та податків

| Посада,<br>спеціаль-<br>ність | Сума<br>основної<br>заробітної<br>плати | Сума<br>додаткової<br>заробітної<br>плати | Разом<br>витрат на<br>оплату<br>праці | Сума<br>єдиного<br>внеску на<br>соціальне<br>страхування<br>(22%), грн. | Сума<br>податку з<br>доходів<br>фізичних<br>осіб (18%),<br>грн. |
|-------------------------------|---|---|---------------------------------------|---|---|
| Програміст                    | 51 000,00                               |   | 51 000,00                             | 11 220,00   | 9180,0  |
| Тестувальник                  | 3 500,00                                | 3 000,00                                  | 6 500,00                              | 1430,00   | 1 170,0   |
| Дизайнер                      | 5 000,00                                | 5 000,00                                  | 10 000,00                             | 2200,00   | 1800,0  |
| Разом:                        | 56 700,00                               | 8 000,00                                  | 64 700,00                             | 14 850,00   | 12 150,0  |

Таблиця 5.5.

## Бюджет загальновиробничих витрат

| Статті витрат                                      | Сума, грн. |
|--|------------|
| <i>Змінні загальновиробничі витрати, у т.ч.:</i>   | —          |
| - заробітна плата допоміжного персоналу;           | 6 500,00   |
| - витрати на МШП;                                  | 400,00     |
| - витрати на електроенергію та технологічні цілі;  | 3 000,00   |
| Разом змінних витрат:                              | 9 900,00   |
| <i>Постійні загальновиробничі витрати, у т.ч.:</i> | —          |
| - комунальні послуги;                              | 4 500,00   |
| - витрати на оренду;                               | 8 000,00   |
| - витрати на ремонт;                               | 4 800,00   |
| - інші постійні витрати;                           | 1 500,00   |
| Разом постійних витрат:                            | 11 800,00  |
| <i>Разом загальновиробничих витрат:</i>            | 30 600,00  |

Таблиця 5.6.

## Бюджет адміністративних витрат та витрат на збут

| Статті витрат                                  | Сума, грн. |
|--|------------|
| <i>Адміністративні витрати, у т.ч.:</i>        | —          |
| - заробітна плата адміністративного персоналу; | 10700,00   |
| - витрати на МШП;                              | 500,00     |
| - витрати на сплату податків і зборів;         | 1000,00    |
| - знос адміністративного обладнання;           | 700,00     |
| Разом адміністративних витрат:                 | 12 900,00  |
| <i>Витрати на збут, у т.ч.:</i>                | —          |
| - заробітна плата менеджерів зі збути;         | 15000,00   |
| - витрати на гарантійний ремонт;               | 0,00       |
| - витрати на гарантійне обслуговування;        | 0,00       |
| - витрати на налагодження і експлуатацію;      | 40 000,00  |
| - витрати на рекламу;                          | 12000,00   |
| Разом витрат на збут:                          | 67 000,00  |

Таблиця 5.7.

## Зведений кошторис витрат на розробку проектного рішення

| Статті витрат                                    | Одиниці виміру | Фактична кількість, шт. | Ціна одиниці, грн. | Разом, грн. |
|--|----------------|-------------------------|--------------------|-------------|
| Сировина і матеріали                             | шт             | -                       | -                  | -           |
| Купівельні напівфабрикати та комплектуючі вироби | шт             | 17                      | -                  | 37 096,00   |
| Зворотні відходи (вирах.)                        | грн            | -                       | -                  | -           |
| Паливо та електроенергія на технологічні цілі    | грн            | 1000                    | 1,68               | 1680,00     |
| Основна заробітна плата                          | грн            | 3                       | -                  | 56700,00    |
| Додаткова заробітна плата                        | грн            | 2                       | -                  | 8 000,00    |

## Продовження таблиці 5.7

|  |     |    |   |            |
|--|-----|----|---|------------|
| Відрахування на соціальне страхування            | грн | 3  | - | 14 850,00  |
| Витрати на утримання й експлуатацію устаткування | грн | -  | - | -          |
| Загальновиробничі витрати, у.т.ч.:               |     |    |   |            |
| - змінні;  | грн | 3  | - | 9 900,00   |
| - постійні;                                      | грн | 4  | - | 11 800,00  |
| <i>Разом виробничих витрат:</i>                  | грн | 7  | - | 30 600,00  |
| Адміністративні витрати                          | грн | 4  | - | 12 900,00  |
| Витрати на збут                                  | грн | 3  | - | 67 000,00  |
| Інші операційні витрати                          | грн | -  | - | -          |
| <i>Разом виробничих і операційних витрат:</i>    | грн | 16 | - | 110 500,00 |

Ціна програмного продукту визначається за нижченаведеною формулою із врахуванням рентабельності виробництва 55%.

Формула для розрахунку ціни продукту:

$\Pi = СБ * Р + СБ$ , де:

$\Pi$  – ціна одинці продукту, грн.;

СБ – собівартість продукту, грн.;

Р – рентабельність виробництва, %.

Ціна продукту:

СБ = 110 500,00 грн.;

Р = 55%;

$$\Pi = 110\ 500,00 * 0,55 + 110\ 500,00 = 171\ 275,00 \text{ (грн.)}$$

$$\text{Ціна з ПДВ} = \Pi * 1,2 = 193\ 375,00$$

$$\text{Кількість реалізованої продукції} = 2$$

$$\text{Дохід від реалізації продукції} = \text{Ціна з ПДВ} * \text{кількість реалізованої продукції} = 210\ 180,00 * 2 = 386\ 750,00 \text{ (грн.)}$$

$\text{ПДВ} = \text{Дохід від реалізації продукції} / 6 = 64\ 458,33 \text{ (грн.)}$

$\text{Чистий дохід від реалізації продукції} = \text{Дохід від реалізації продукції} - \text{ПДВ} = 386\ 750,00 - 64\ 458,33 = 322\ 291,67 \text{ (грн.)}$

$\text{Валовий прибуток} = \text{Чистий дохід від реалізації продукції} - \text{Собівартість реалізованої продукції} = 322\ 291,67 - 177\ 796,00 = 144\ 495,67 \text{ (грн.)}$

$\text{Фінансовий результат від операційної діяльності} = \text{Валовий прибуток} - \text{Адміністративні витрати} - \text{Витрати на збут} - \text{Інші операційні витрати} = 144\ 495,67 - 12\ 900,00 - 12000,00 = 119\ 595,67 \text{ (грн.)}$

$\text{Податок на прибуток} = \text{Фінансовий результат від операційної діяльності} * \text{розмір податку на прибуток} = 119\ 595,67 * 0,18 = 21\ 527,22 \text{ (грн.)}$

$\text{Чистий прибуток (збиток)} = \text{Фінансовий результат від операційної діяльності} - \text{Податок на прибуток} = 35\ 400,00 - 6\ 372,00 = 98\ 068,45 \text{ (грн.)}$

Таблиця 5.8.

Бюджет фінансових результатів

| Показники                                       | Сума, грн. |
|---|------------|
| Дохід від реалізації продукції(обсяг 1)         | 386 750,00 |
| Податок на додану вартість                      | 64 458,33  |
| Чистий дохід від реалізації продукції           | 322 291,67 |
| Собівартість реалізованої продукції             | 177 796,00 |
| Валовий прибуток                                | 144 495,67 |
| <i>Операційні витрати:</i>                      |            |
| - адміністративні витрати                       | 12 900,00  |
| - витрати на збут                               | 67 000,00  |
| - інші операційні витрати;                      | 0,00       |
| Фінансовий результат від операційної діяльності | 119 595,67 |
| Податок на прибуток (18%)                       | 21 527,22  |
| Чистий прибуток (збиток)                        | 98 068,45  |

## 5.6. Висновки

У цьому розділі було проведено економічну оцінку доцільності розробки, економічну вигоду, витрати та доходи у під час реалізації веб системи для продажу та оренди нерухомості та земельних ділянок. Було порівняно розроблювану систему з подібними на ринку і було прийнято стратегії для розвитку проєкту та було переглянуто стратегічні альтернативи розвитку продукту. Було обраховано бюджетування, необхідне для початку і проведення розробки продукту. Пораховано витрати на техніку, персонал та команду розробки.

Під час виконання економічної частини, було виконано усі необхідні розрахунки. У результаті проведення розрахунків рентабельності визначено реалізацію двох одиниць цього продукту і становитиме 386 750,00 (грн.). При такому доході, чистий дохід становитиме 98 068,45 (грн.). У результаті розрахунків можна зробити висновок, що вагому роль у доцільності розробки програмного забезпечення є чистий дохід від його реалізації. На це найбільше впливає кінцева вартість реалізованого продукту та сума витрат.

Опираючись на розрахунки, можна зробити висновок, що розробка та реалізація даної веб системи для продажу та оренди нерухомості та земельних ділянок є економічно вигідною. Програмний продукт враховує потреби ринку і є оптимальним рішенням, хоч водночас є вузькоспеціалізованим і з вибраною стратегією нового продукту можуть виникнути складнощі із продажами продукту. Але завдяки гнучкості розроблюваної програмної системи є можливість у майбутньому розширити функціонал системи під потреби ринку, що дає можливість подолати всі потенційні проблеми і успішно тримати продукт на ринку.

## ВИСНОВКИ

Під час бакалаврської кваліфікаційної роботи було розроблено веб систему для продажу та оренди нерухомості та земельних ділянок.

У ході розробки, на етапі аналізу предметної області було визначено доцільність та мету реалізації даної системи. Було проаналізовано проєкти-аналоги, які зараз доступні на ринку.

Під час постановки задачі було вибрано засоби для реалізації проєкту. Було розділено систему на два частини: серверну та клієнтську. Визначено та специфіковано вимоги до розроблюваної системи.

На етапі проєктування було спроектовано структуру, поведінку та взаємодію елементів системи. Розроблено UML діаграми. Спроектовано та визначено основні архітектурні рішення. Спроектовано базу даних. Розроблено вейфрейми та мокапи для клієнтської частини додатка.

На етапі реалізації було написано код, який повинен виконувати поставлені завдання, специфіковані та визначені на етапі проєктування. Реалізовано клієнтську та серверну частини веб системи. Проведено тестування на серверній та клієнтських частинах.

Також було обраховано економічну вигідність розробки та реалізації даної веб системи. У результаті обрахунків можна зробити висновок що дана веб система є економічно вигідною.

Після виконання бакалаврської кваліфікаційної роботи я отримав досвід розробки складних систем. Оскільки було проведено всі етапи життєвого циклу програмного забезпечення – це дає цілісну картину створення програмного забезпечення. Оскільки було проведено роботу із серверною та клієнтською частинами проєкту це дає можливість вести розробку програмних систем у всьому наборі технологій, необхідному для створення подібних веб систем.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Як спрацювали Прозорро. Продажі і що буде далі. [Електронний ресурс] – Режим доступу до ресурсу: <https://biz.nv.ua/ukr/experts/prozorro-prodazhi-shcho-prodayut-ta-kupuyut-pributki-i-rozvitok-shcho-dali-novini-ukrajini-50100000.html> (2022).
2. Карты Google [Електронний ресурс] – Режим доступу до ресурсу: <https://www.google.com.ua/maps/@50.4851493,30.4721233,14z?hl=ua> (2022).
3. Українська універсальна біржа [https://sale.uub.com.ua/filter/land-parcel?&page=1&rows=10&sidx=dateModified&sord=desc&filter\\_type=&action=Y&type\\_filter=land-parcel&status=all](https://sale.uub.com.ua/filter/land-parcel?&page=1&rows=10&sidx=dateModified&sord=desc&filter_type=&action=Y&type_filter=land-parcel&status=all) (2022).
4. olx нерухомість [Електронний ресурс] – Режим доступу до ресурсу: <https://www.olx.ua/d/uk/nedvizhimost/> (2022).
5. dom ria [Електронний ресурс] – Режим доступу до ресурсу: <https://dom.ria.com/uk/> (2022).
6. ebay [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ebay.com/> (2022).
7. amazon [Електронний ресурс] – Режим доступу до ресурсу: <https://www.amazon.com/> (2022).
8. Richter J. CLR via C# / Jeffrey Richter. – Redmond, Washington 98052-6399: Microsoft prePress, 2012. – 813 с. – (A Division of Microsoft Corporation). – (Full Coverage of Multicore Programming; вип. 4).
9. SQL Server Tutorial [Електронний ресурс] – 2022. – Режим доступу до ресурсу: <https://www.sqlservertutorial.net/> (2022).
10. React JavaScript-бібліотека для створення користувачьких інтерфейсів [Електронний ресурс] // FaceBookOpenSource. – 2022. – Режим доступу до ресурсу: <https://uk.reactjs.org/> (2022).

- 11.Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale 1st Edition / Boris Cherny., 2019. – 342 с. – (Programming TypeScript).
- 12.Draw.io [Електронний ресурс] Режим доступу до ресурсу:  
<https://www.draw.io/> (2022).
- 14.Figma [Електронний ресурс] Режим доступу до ресурсу: <https://www.figma.com/> (2022).
- 15.VC Code [Електронний ресурс] – Режим доступу до ресурсу:  
<https://code.visualstudio.com/> (2022).
- 16.MS Visual Studio [Електронний ресурс] – Режим доступу до ресурсу:  
<https://visualstudio.microsoft.com/> (2022).
- 17.Грицюк Ю. Аналіз вимог до програмного забезпечення / Юрій Грицюк. – Львів: Львівська політехніка, 2018. – 456 с.
- 18.Booch G. The Unified Modeling Language User Guide / G. Booch, J. Rumbaugh, I. Jacobson., 2005. – 407 с. – (Addison Wesley). – (3nd Edition).
- 19.Software Architecture — The Onion Architecture [Електронний ресурс] // Shivendra Odean. – 2019. – Режим доступу до ресурсу:  
<https://medium.com/@shivendraodean/software-architecture-the-onion-architecture-1b235bec1dec>.
- 20.Левус Є. Вступ до інженерії програмного забезпечення / Є. Левус, Н. Мельник. – Львів: Львівська політехніка, 2018. – 248 с.
- 21.MS SQL Server [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.microsoft.com/ru-ru/sql-server/sql-server-2019> (2022).
- 22.MS Identity Web authentication library [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/azure/active-directory/develop/microsoft-identity-web> (2022).

## ДОДАТКИ

### Додаток А. Додатки аналоги.

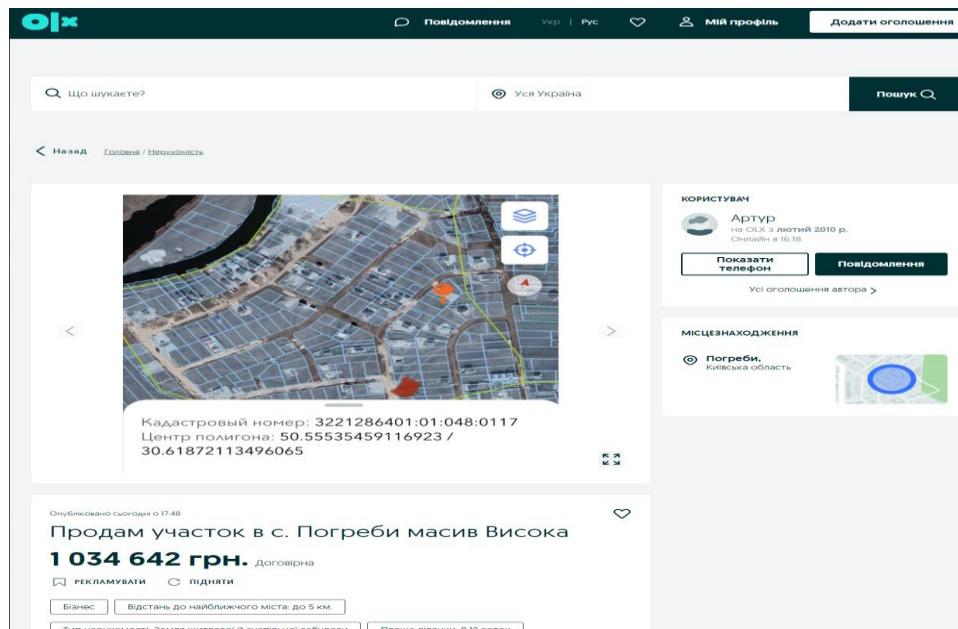


Рис. А.1. Сторінка сайту OLX

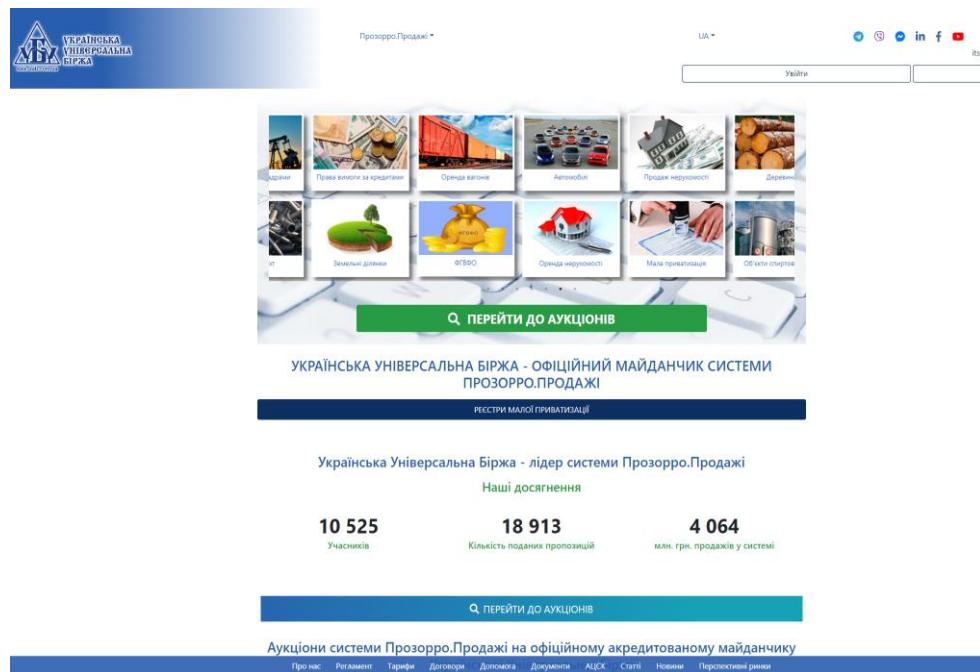


Рис. А.2. Сторінка сайту Rielator

**Інші 1к квартири в місті Рівне**  
78 пропозицій

**Довготривала оренда 1к квартири на вул. Макарова 44**

Рівне · район Ювілейний · Макарова вулиця

**4 000 грн · 125 \$**

**Перевірено по відеодзвінку**  
Звіт від 18 тра

- ✓ Вторинне житло
- ✓ 1 кімната
- ✓ 5 поверх з 9
- ✓ Пропозиція від посередника
- ✓ 3 опаленням
- ✓ З ремонтом
- ✓ 3 меблями

2 спальніх місця  
Загальна площа 35 м<sup>2</sup>  
Додатково платя по лічильникам і за опалення

**Поділитись** **В обране**

**ПЕРЕВІРЕНІЙ РІЕЛТОР**  
**Інна** · На сайті близько 24 годин тому  
**(067) XXX XXX**  
**Написати в чат**

**9** Високий рейтинг ріелтора

- ✓ Працює з DOM.RIA 5+ років
- ✓ Продає 8 об'єктів нерухомості

Рис. А.3. Сторінка сайту Dom.Ria

**ebay** Shop by category  All Categories  Advanced

Back to previous page | Listed in category: Business & Industrial > Modular & Prefabricated Buildings

**Magic House Steel Frame Prefab House Resort Huts Triangle House Wood Log Cabin**

Condition: New

Square Meters: 34 Square meters

Quantity: 1 Last one

Price: US \$36,000.00

**Buy It Now** **Add to cart** **Add to Watchlist**

30-day Returns 22 Watchers

Shipping: US \$10,000.00 Economy Shipping from Greater China to worldwide | [see details](#)

International shipment of items may be subject to customs processing and additional charges.

Located in: Shanghai, China

Delivery: Estimated between Mon, Jul 18 and Mon, Sep 12

This item has an extended handling time and a delivery estimate greater than 30 business days. Please allow additional time if international shipping is subject to customs processing.

Returns: 30 days return | Buyer pays for return shipping | [see details](#)

Payments:

**Similar sponsored items**

- Modular Converted Container 60m2 Holiday Home Portable
- Converted Shipping Container 21ft Holiday Home Portable
- Modular Converted Container 90m2 Holiday Home Portable
- Holiday Home portable cabin,building,modular,Converted
- Modular Converted Container 80ft Holiday Home Portable

**Feedback on our suggestions**

Рис. А.4. Сторінка сайту eBay

amazon Deliver to Ukraine All live+house Hello, Sign in Account & Lists Returns & Orders Cart

All Today's Deals Customer Service Registry Gift Cards Sell

JCs Wildlife Barn Owl Nesting Box: Do It Yourself Assembly Kit - Build Your Own Barn Owl Bo... ★★★★★ 17 \$147.99 prime Sponsored

Patio, Lawn & Garden > Outdoor Storage & Housing



ECOHOUSEMART Moon House 33' DIAM Dome FRAMING KIT D33H-986, Dark brown  
Brand: ECOHOUSEMART

Brand ECOHOUSEMART  
Color Dark brown  
Material Wood, Alloy Steel  
Item 396 x 396 x 252 inches  
Dimensions LxWxH  
Style A dome

Share Email Facebook Twitter Pinterest Have one to sell? Sell on Amazon

**About this item**

- The dome frame diameter: 32'10" (10 m)
- The dome frame is pre-sized for 2 floors
- Height of the dome frame: 20' 11" (6.375 m)
- THIS IS NOT A COMPLETE HOUSE. YOU ARE BUYING A DOME FRAMING KIT ONLY!
- STAMPED STRUCTURAL DRAWINGS ARE NOT INCLUDED AND MAY BE PRODUCED AT ADDITIONAL COST

Garden Shed Outdoor Storage... \$249.99 Sponsored

### Brands related to this category on Amazon

Sponsored



**KidKraft**  
Made for Make Believe  
[Shop KidKraft >](#)



**Alvantor®**  
Pop Up Bubble Tents for 2022  
[Shop Alvantor >](#)

### Have a question?

Find answers in product info, Q&As, reviews

Type your question or keyword

### Product Description

Рис. А.5. Сторінка сайту AMAZON

## Додаток Б. UML діаграми

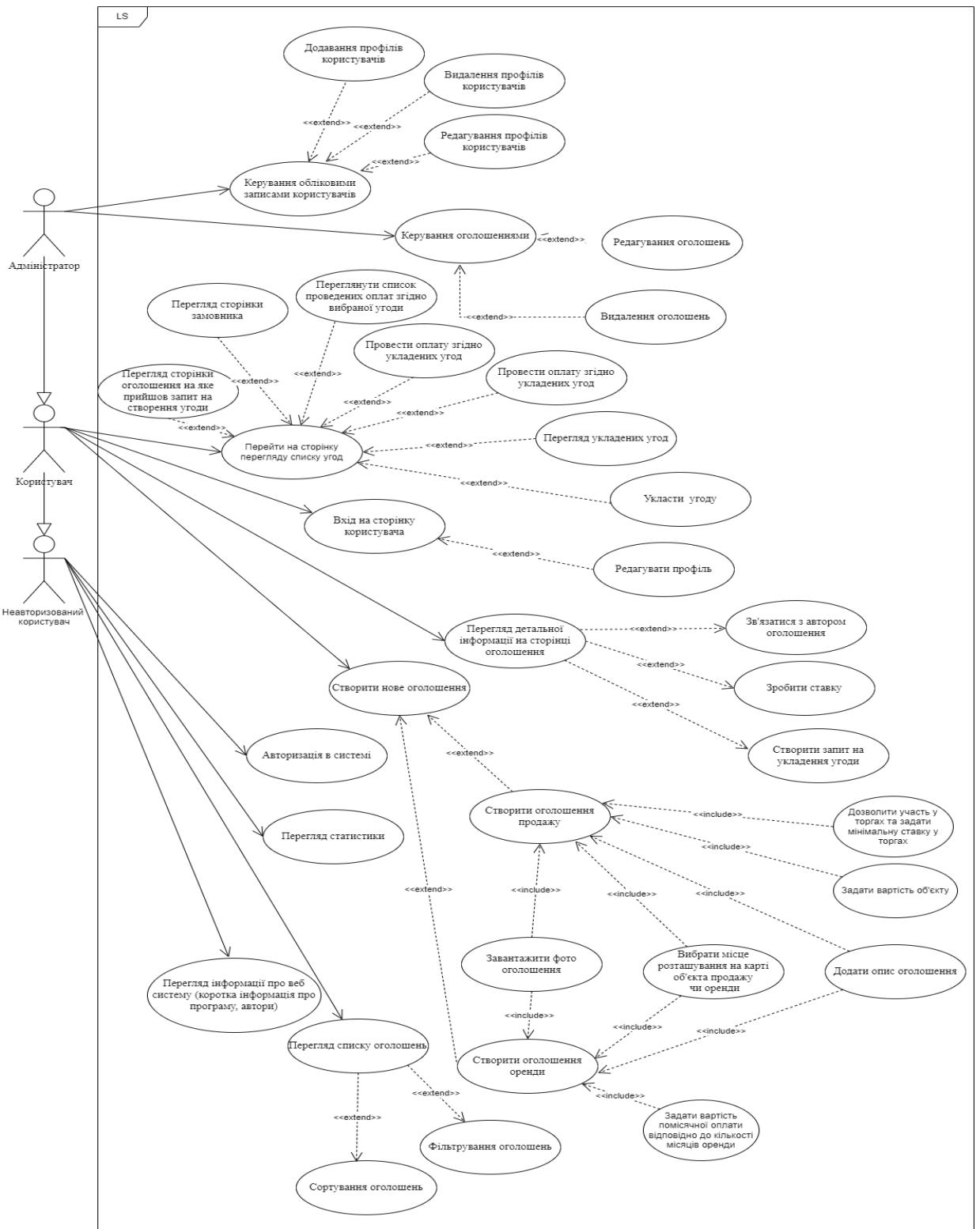


Рис. Б.1. Діаграма прецедентів

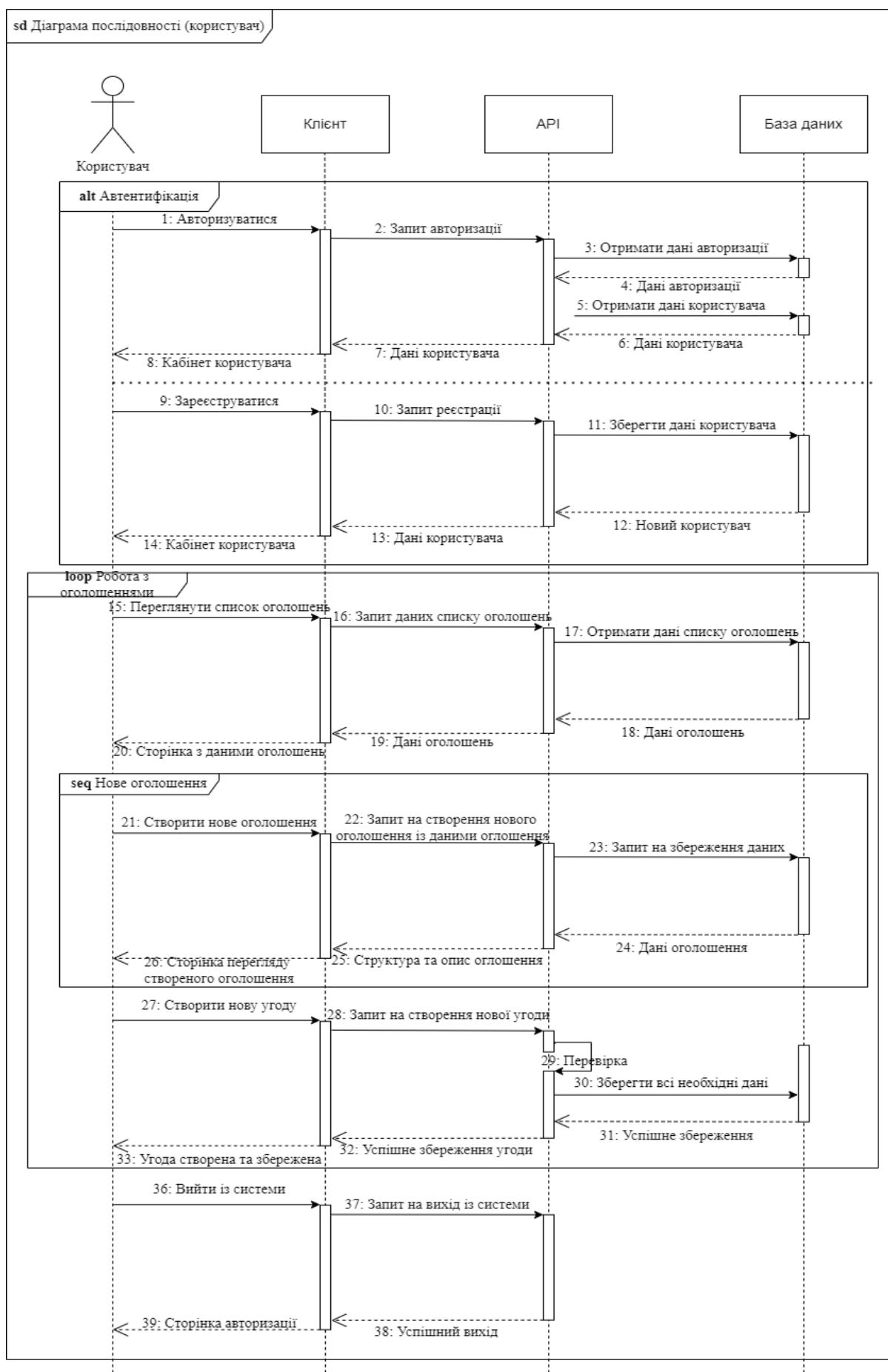


Рис. Б.2. Діаграма послідовності

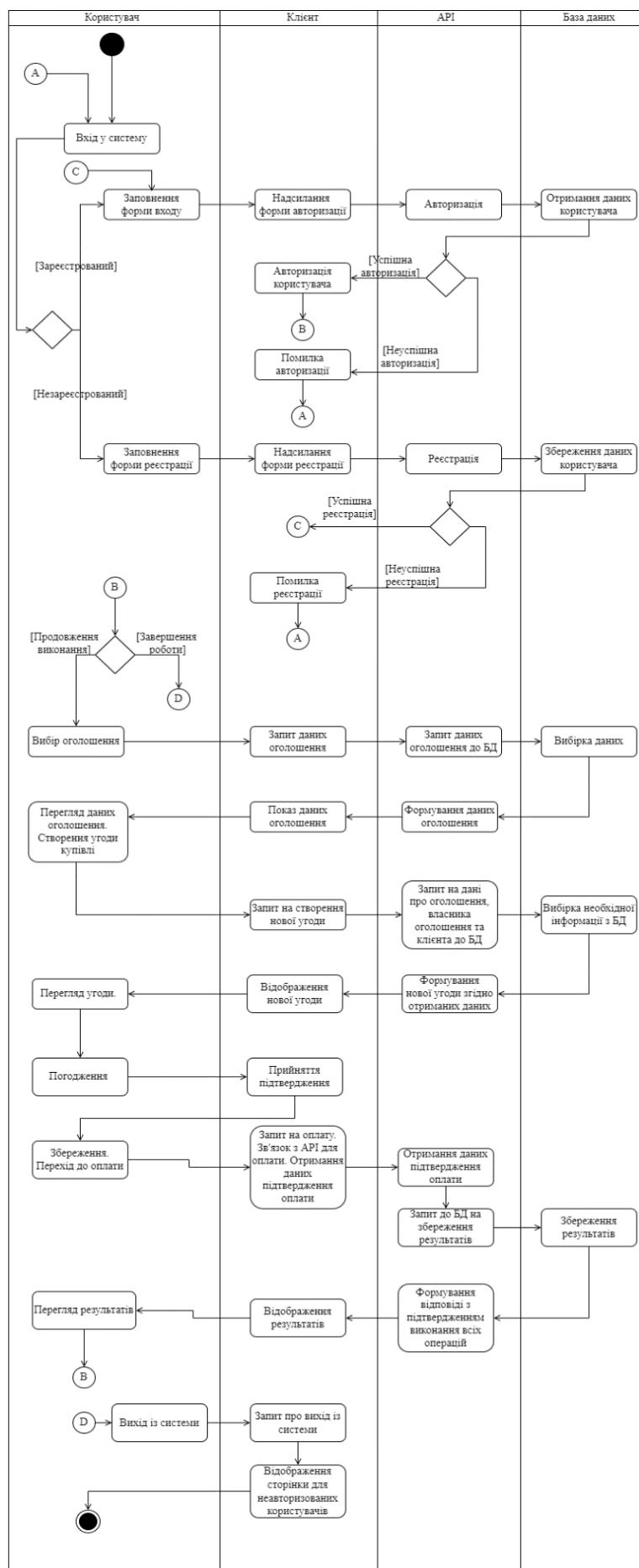


Рис. Б.3. Діаграма діяльності

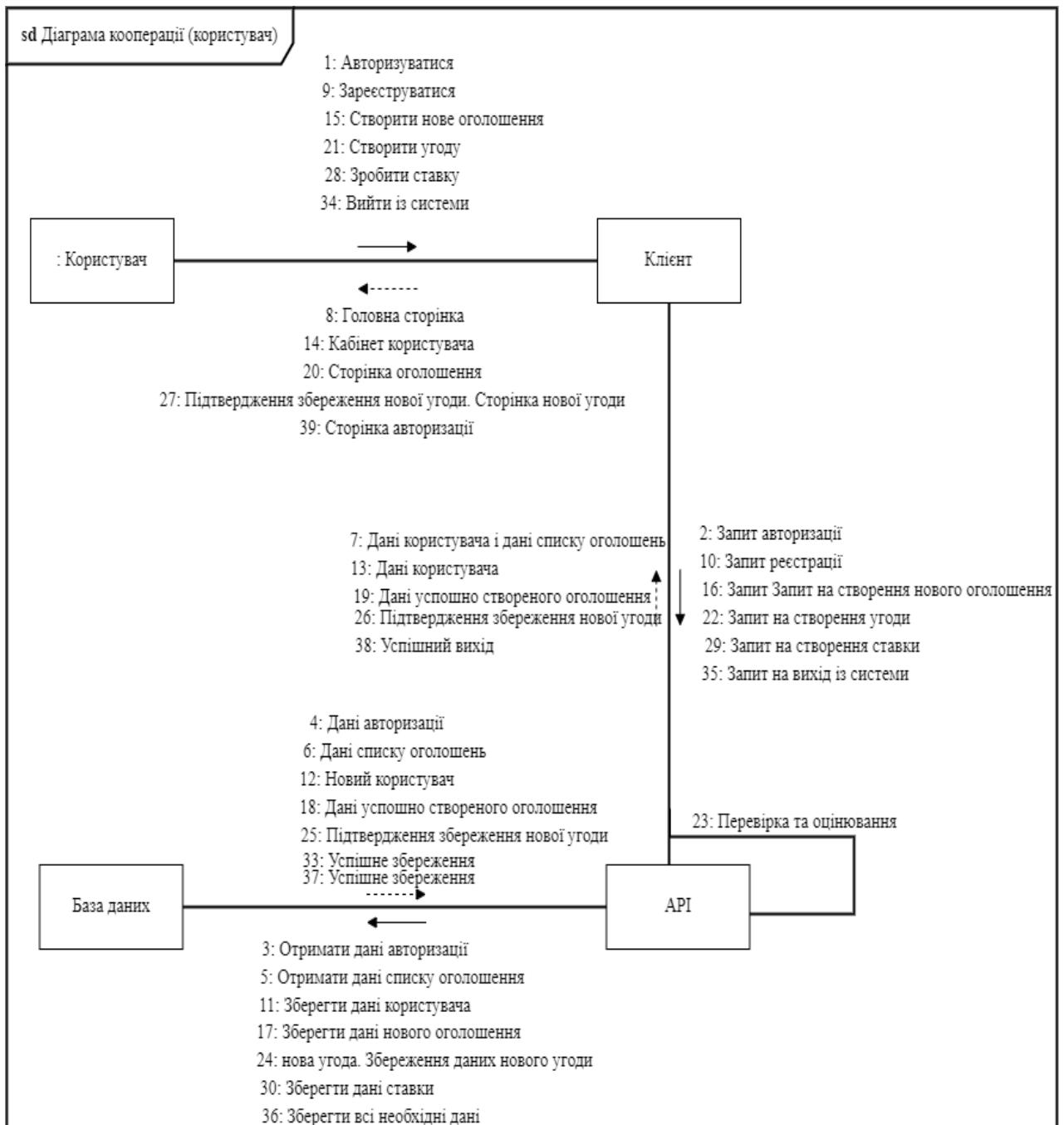


Рис. Б.4. Діаграма кооперації

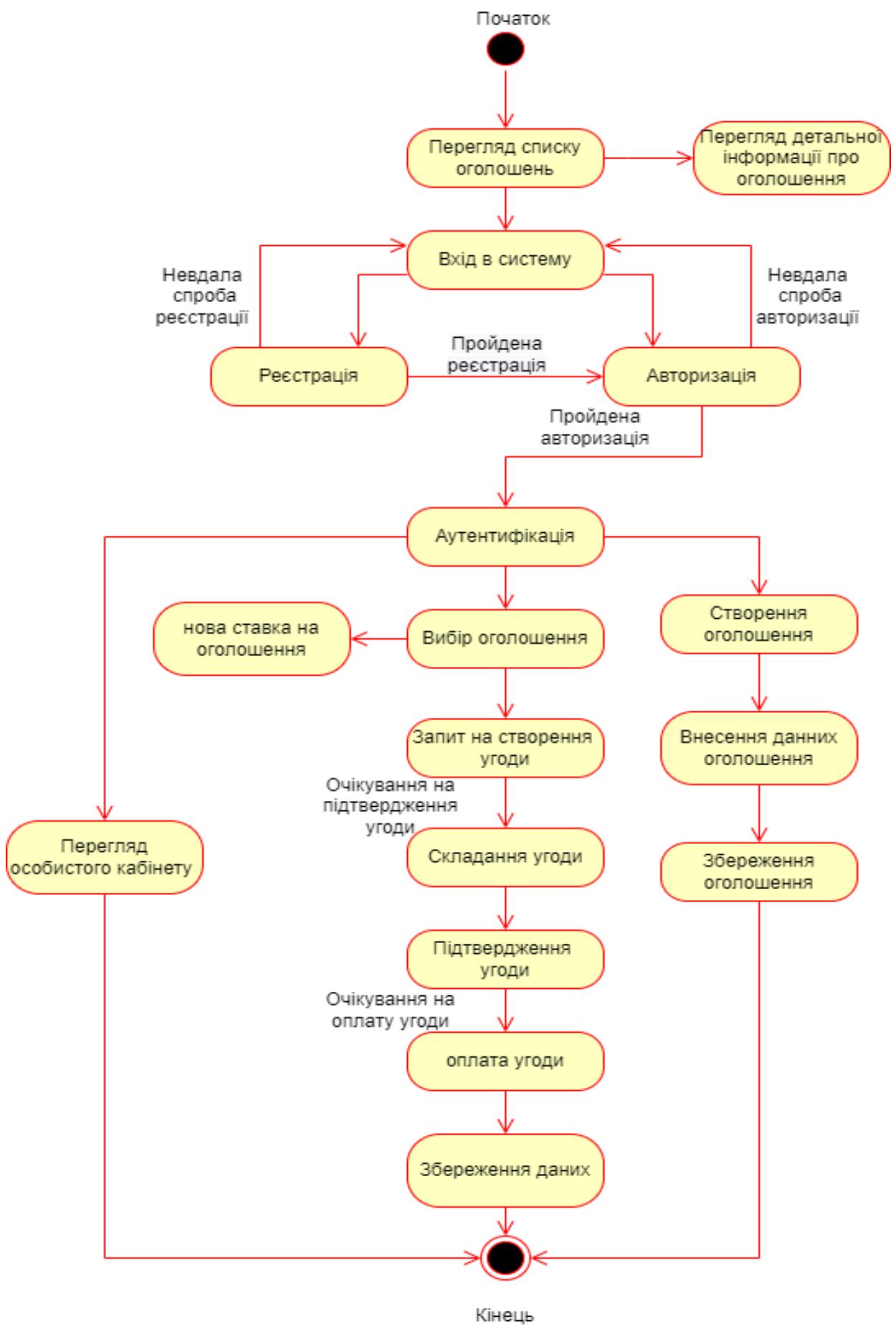


Рис. Б.5. Діаграма станів

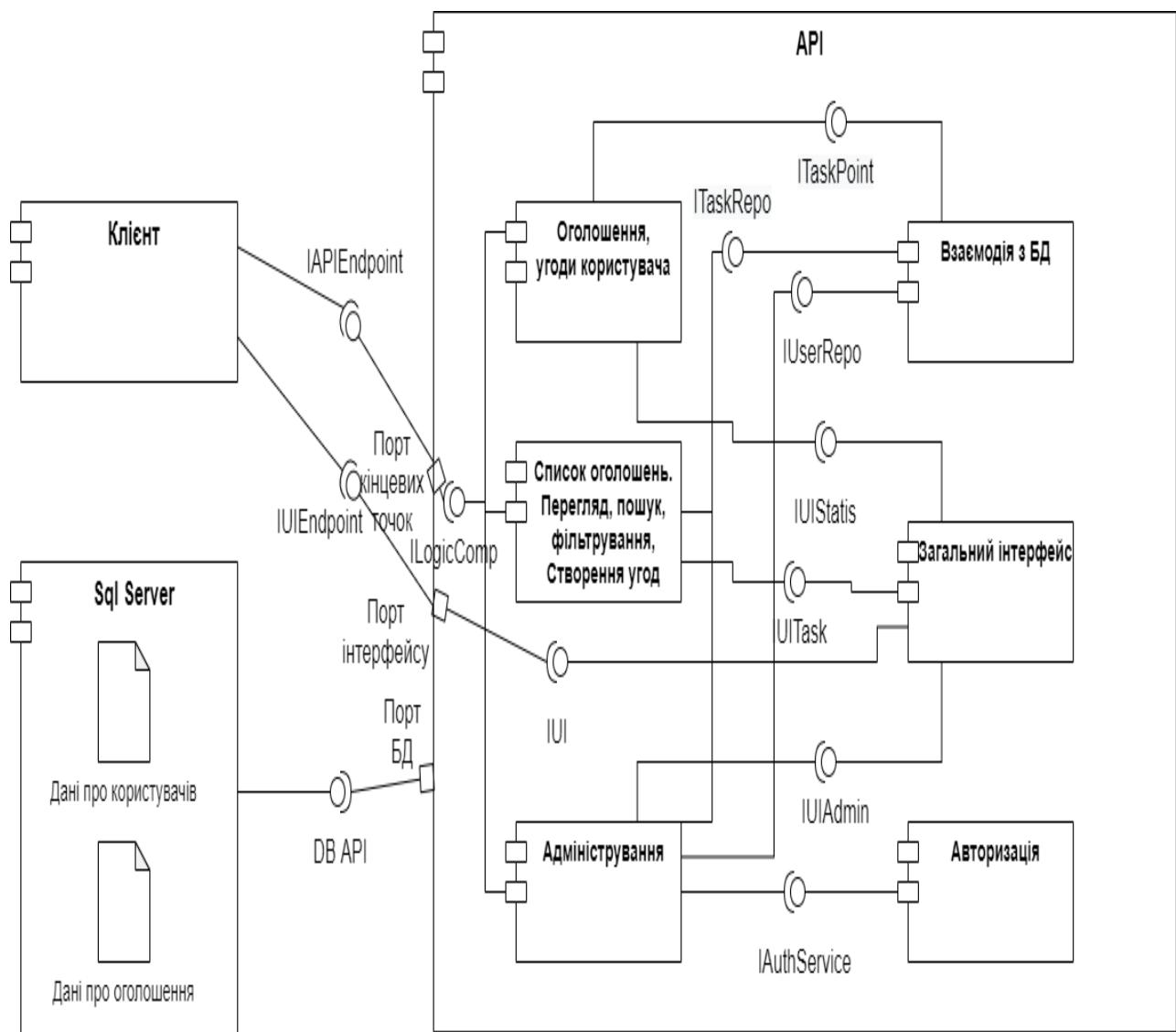


Рис. Б.6. Діаграма компонентів

## Додаток В. Вайрфрейми та мокапи інтерфейсу

The screenshot shows a web-based booking interface for a house in Lviv, Ukraine. At the top, there's a navigation bar with a logo, the text "LandSelling", and links for "RENT", "SOLD", "LOTS", "RULES", and "ACCOUNT". Below the navigation bar is a large image of a two-story house with a dark roof and light-colored siding, set against a backdrop of a cloudy sky at sunset. To the right of the image, the address "Lviv, Lucasha strit, 2" is displayed next to a location pin icon. Below the address is a rating of "3.0" with five yellow stars. Underneath the rating are four grey boxes representing different stay durations: "30+ days", "8+ days", "3-7 days", and "1-2 days". Below these boxes are price options: "17\$ / day", "22\$ / day", "25\$ / day", and "28 / day". A large grey callout box contains descriptive text: "This is a medium size home for every live. This house contain bathroom, kitchen, toilet and other opportunities for 5 person". At the bottom, there are two input fields for "start data" (containing "13-06-2021") and "end data" (containing "13-06-2021"), followed by a prominent yellow "BOOK" button.

Рис. В.1. Мокап сторінки найму

The screenshot shows a web-based application interface for managing users. At the top, there is a navigation bar with the logo 'LandSelling' and links for RENT, SOLD, LOTS, RULES, USERS, and SIGN OUT. Below the navigation bar, the title 'SYSTEM USERS LIST' is displayed in bold capital letters. A table with five columns (No, First name, Last name, Position, Email) is shown, but it is currently empty. Below this, the title 'ADD SYSTEM USER' is displayed in bold capital letters. Below the title, there are four input fields labeled 'First name', 'Last name', 'Email', and 'Password', each with a corresponding text input box. At the bottom of the form area, there is a dark button labeled 'CREATE ACCOUNT'.

Рис. В.2. Мокап сторінки перегляду списку користувачів

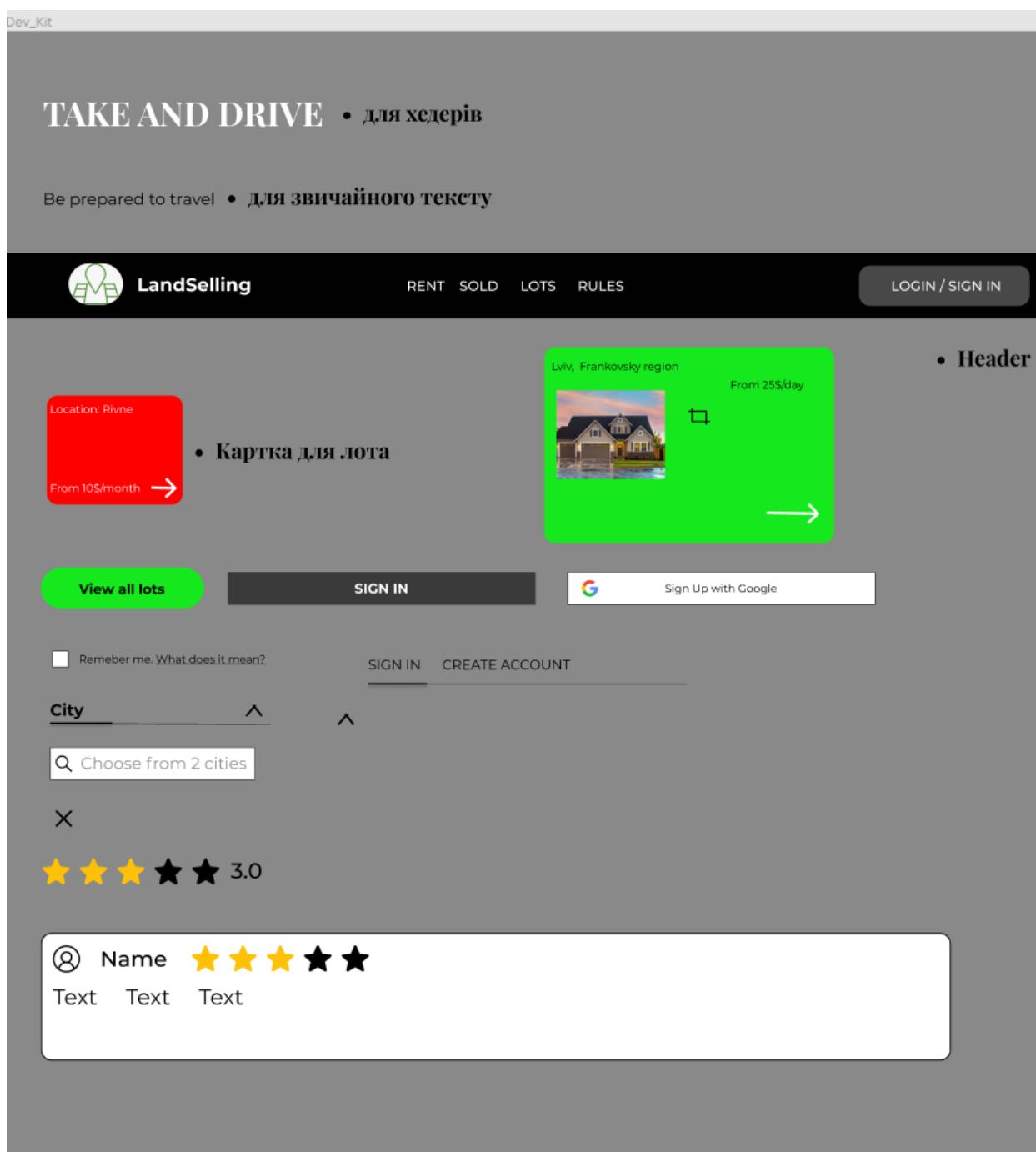


Рис. В.3. UI kit

## Додаток Г. Інструкція користувача

### 1. Компоненти програмного забезпечення

Розроблювана веб система хоститься в середовищі Azure DevOps. Відповідні всі необхідні компоненти для її роботи розміщені у цьому середовищі. Проект має серверну частину, написану на C# 9.0 у середовищі MS Visual Studio 2022. Для збереження даних на серверній частині використовується база даних MS SQL Server. Клієнтська частина написана з використанням бібліотеки React на мові програмування TypeScript у середовищі програмування MS Visual Studio Code.

### 2. Встановлення програмного забезпечення

Для запуску веб системи необхідно відкрити веб переглядач та ввести доменне ім'я LandSelling. У результаті правильного введення доменного імені відобразиться головна сторінка веб системи.

### 3. Базові функції ПЗ

Вхід в систему. Щоб увійти в систему потрібно натиснути кнопку для входу в систему. Відобразиться модальне вікно із полями пошти та паролю. Користувач вводить дані і натискає кнопку підтвердження. Відбується вход в систему.

Створення нових оголошень про оренду чи продаж земельних ділянок чи нерухомості. Для цього необхідно увійти в систему. Далі вибрati у верхній частині сторінки посилання на створення нового оголошення. Відбудеться перехід на сторінку створення нового оголошення. Користувач вводить дані і натискає кнопку для створення нового оголошення. Оголошення зберігається в системі. І як результат відображається відповідне сповіщення і якщо перейти на сторінку зі списком оголошень, новостворене оголошення з'явиться у списку.

Можливість аукціону по продажу земельних ділянок чи нерухомості. Для цього необхідно увійти в систему, далі можна переходити на сторінки оголошень і на оголошеннях з доступною такою можливістю можна зробити ставку.

Перегляд оголошень може виконати будь-який користувач, для цього не потрібен вхід в систему. Також будь-який користувач може використати функцію фільтрування та сторування списку оголошенні для пролегшення пошуну необхідного оголошення. Ця функція доступна на верхній частині головної сторінки. Для перегляду детальної інформації стосовно замовлення достатньо перейти на головну сторінку і вибрати зі списку будь-яке оголошення. При виборі оголошення відбудеться перехід на сторінку з детальною інформацією про оголошення.

Можливість укладати угоди що до продажу чи оренди. Для цього спершу необхідно бути авторизованим у системі. Авторизованому користувачу доступні кнопки створення угод на сторінках оголошень. При натисненні на одну з цих кнопок відбувається створення запиту на створення угоди. В залежності від типу угоди користувачу доведеться вносити додаткові дані для створення заявки. Після надсилання заявки на створення угоди, якщо власник оголошення домовиться з клієнтом – вони можуть укласти угоду.

Можливість проведення оплати згідно укладених угод. Для цього спершу необхідно бути авторизованим у системі. Авторизованому користувачу в особистому кабінеті доступний список угод. Список складається з двох частин, поділених за типом. Перший тип запити на власні оголошенні. І другий тип це власні угоди. Відповідно вибрали власну угоду із переліку, яка є прийнята власником оголошенні, користувач може здійснити оплату. Для цього необхідно внести дані своєї карти і підтвердити оплату. Після завершення операції відобразиться відповідне вікно із результатом.

#### **4. Аналіз помилок та можливих проблем**

У випадку неправильної роботи веб системи необхідно дотримуватися підказок, які відображаються у випадку проблем у роботі системи.

Можна перевірити правильність налаштування середовища роботи.

## Додаток Д. Код

### **LandSellingContext.cs**

```

using Domain.Entity;
using Domain.Entity.LotManagement; //User
using Domain.Entity.LotManagement.AgreementManagement;
using Microsoft.EntityFrameworkCore;

namespace Data.EF
{
    public class LandSellingContext : DbContext
    {
        public
            LandSellingContext(DbContextOptions<LandSellingContext> options)
            : base(options)
        {
        }

        protected override void
OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);
            optionsBuilder.UseLazyLoadingProxies();

            optionsBuilder.UseSqlServer(
                x => x.UseNetTopologySuite());
        }

        public virtual DbSet<User> Users { get; set; }
        public virtual DbSet<Admin> Admins { get; set; }
        public virtual DbSet<Location> Locations { get; set; }
        public virtual DbSet<Lot> Lots { get; set; }
        public virtual DbSet<PriceCoef> PriceCoefs { get; set; }
    }

    public virtual DbSet<Image> Images { get; set; }
    public virtual DbSet<Bid> Bids { get; set; }
    public virtual DbSet<Agreement> Agreements { get; set; }
    public virtual DbSet<Payment> Payments { get; set; }
    public virtual DbSet<LotManager> LotManagers { get; set; }

    public virtual DbSet<AgreementManager>
AgreementManagers { get; set; }

    protected override void
OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        //Admin
        modelBuilder.Entity<Admin>()
            .HasIndex(i => i.Id)
            .IsUnique(true);

        //Location
        modelBuilder.Entity<Location>(entity =>
        {
            entity.HasIndex(i => i.Id)
            .IsUnique();
        });

        //Lot
        modelBuilder.Entity<Lot>(entity =>
        {
            entity.HasIndex(i => i.Id)
            .IsUnique();

            entityhasOne(p => p.Lot)
            .WithOne(b => b.Location)
            .IsRequired(true)
            .HasForeignKey<Lot>(l => l.LocationId);
        });

        //Image
        modelBuilder.Entity<Image>(entity =>
        {
            entity.HasIndex(i => i.Id)
            .IsUnique();

            entityhasOne(p => p.Owner)
            .WithMany(b => b.Lots)
            .IsRequired(true)
            .HasForeignKey(k => k.OwnerId);
        });

        //LotManager
        modelBuilder.Entity<LotManager>(entity =>
        {
            entity.HasIndex(i => i.Id)
            .IsUnique();

            entityhasOne(p => p.Lot)
            .WithMany(b => b.LotManagers)
            .IsRequired(true)
            .HasForeignKey(k => k.ManagerId);

            entityhasOne(p => p.Manager)
        });
    }
}

```



## LotUnitOfWork.cs

```

private IAgreementManagerRepository _agreementManagerRepository;

public LotUnitOfWork(LandSellingContext dbContext)
{
    _dbContext = dbContext;
}

public ILotRepository LotRepository => _lotRepository
?=? new LotRepository(_dbContext);

public IAgreementRepository AgreementRepository =>
_agreementRepository ??= new AgreementRepository(_dbContext);

public IBidRepository BidRepository =>
_bidRepository ??= new BidRepository(_dbContext);

public IImageRepository ImageRepository =>
_imageRepository ??= new ImageRepository(_dbContext);

public ILocationRepository LocationRepository =>
_locationRepository ??= new LocationRepository(_dbContext);

public IPaymentRepository PaymentRepository =>
_paymentRepository ??= new PaymentRepository(_dbContext);

public IPriceCoefRepository PriceCoefRepository =>
_priceCoefRepository ??= new PriceCoefRepository(_dbContext);

public IUserRepository UserRepository =>
_userRepository ??= new UserRepository(_dbContext);

public IAdminRepository AdminRepository =>
_adminRepository ??= new AdminRepository(_dbContext);

public ILotManagerRepository LotManagerRepository =>
_lotManagerRepository ??= new LotManagerRepository(_dbContext);

public IAgreementManagerRepository AgreementManagerRepository =>
_agreementManagerRepository ??= new AgreementManagerRepository(_dbContext);

public async Task<int> Save() => await
_dbContext.SaveChangesAsync();
}

}

namespace Data.UnitOfWork
{
    public class LotUnitOfWork : ILotUnitOfWork
    {
        private readonly LandSellingContext _dbContext;

        private ILotRepository _lotRepository;
        private IAgreementRepository _agreementRepository;

        private IBidRepository _bidRepository;

        private IImageRepository _imageRepository;
        private ILocationRepository _locationRepository;
        private IPaymentRepository _paymentRepository;
        private IPriceCoefRepository _priceCoefRepository;

        private IUserRepository _userRepository;
        private IAdminRepository _adminRepository;

        private ILotManagerRepository
_lotManagerRepository;
    }
}

```

## LotService.cs

```

using System.Linq;
using System.Threading.Tasks;

namespace Business.Services.LotManagement
{
    public class LotService : ILotService
    {
        private IMapper _mapper;
        private readonly ILotUnitOfWork _unitOfWork;

        public LotService(IMapper mapper, ILotUnitOfWork
unitOfWork)

```

```

using AutoMapper;
using Business.Contract.Model.LotManagement;
using Business.Contract.Model.LotManagement.Lot;
using Business.Contract.Services.LotManagement;
using Data.Contract.UnitOfWork;
using Domain.Entity;
using Domain.Entity.Constants;
using Domain.Entity.LotManagement;
using System;
using System.Collections.Generic;

```

```

{
    _mapper = mapper;
    _unitOfWork = unitOfWork;
}

public async Task<Guid> Create(CreateLotDTO
createLot, Guid ownerIdLink)
{
    Lot newLot = _mapper.Map<Lot>(createLot);
    newLot.Status = Domain.Entity.Constants.State.Open;
    newLot.OwnerId = await _unitOfWork.UserRepository.GetByIdLink(ownerIdLink).Id;
    newLot.PublicationDate = DateTime.Now;

    Location location = new Location()
    {
        Latitude = createLot.Location.Latitude,
        Longitude = createLot.Location.Longitude,
        Country = createLot.Location.Country,
        Region = createLot.Location.Region,
        City = createLot.Location.City,
        Street = createLot.Location.Street
    };

    newLot.LocationId = await _unitOfWork.LocationRepository.Add(location);

    await _unitOfWork.LotRepository.Add(newLot);
    Guid lotId = await _unitOfWork.LotRepository.GetByLocationId(newLot.LocationId);
    return lotId;
}

public async Task Delete(Guid id)
{
    var lot = await _unitOfWork.LotRepository.GetById(id);
    await _unitOfWork.LotRepository.Remove(lot);
    await _unitOfWork.Save();
}

public async Task Update(UpdateLotDTO updateLot,
Guid lotId)
{
    Lot newLot = _mapper.Map<Lot>(updateLot);
    newLot.Id = lotId;
    await _unitOfWork.LotRepository.Update(newLot);
    await _unitOfWork.Save();
}

public async Task Viewed(Guid lotId)
{
    Lot lot = await _unitOfWork.LotRepository.GetById(lotId);
    lot.Views++;
    await _unitOfWork.LotRepository.Update(lot);
    await _unitOfWork.Save();
}

public async Task Approve(Guid lotId)
{
    LotManager lotManager = await _unitOfWork.LotManagerRepository.GetByLotId(lotId);
    lotManager.Approved = true;
    await _unitOfWork.LotManagerRepository.Update(lotManager);
    await _unitOfWork.Save();
}

public async Task<ReturnLotDTO> GetById(Guid
lotId)
{
    Lot lot = await _unitOfWork.LotRepository.GetBy
Id(lotId);
    Location location = await _unitOfWork.Location
Repository.GetById(lot.LocationId);
    LocationDTO locationDTO = _mapper.Map<Location
DTO>(location);
    ReturnLotDTO lotDTO = _mapper.Map<ReturnLot
DTO>(lot);
    lotDTO.Location = locationDTO;
    return lotDTO;
}

public async Task<IEnumerable<ReturnLotDTO>>
GetMy(Guid ownerIdLink)
{
    var ownerId = await _unitOfWork.UserRepository
.GetByIdLink(ownerIdLink).Id;
    IEnumerable<Lot> lots = await _unitOfWork.Lot
Repository.GetByOwnerId(ownerId);
    List<ReturnLotDTO> lotDTOS = new
List<ReturnLotDTO>();
    foreach (Lot lot in lots)
    {
        ReturnLotDTO lotDTO = _mapper.Map<ReturnLot
DTO>(lot);
        Location location = await _unitOfWork.Location
Repository.GetById(lot.LocationId);
        LocationDTO locationDTO = _mapper.Map<Locati
DTO>(location);
    }
}

```

```

lotDTO.Location = locationDTO;
lotDTOs.Add(lotDTO);
}
return lotDTOs;
}

public async Task<IEnumerable<ReturnLotDTO>>
GetAll()
{
    IEnumerable<Lot> lots = await _unitOfWork.LotRepository.GetAll();
    List<ReturnLotDTO> lotDTOs = new List<ReturnSimpleLotDTO>();
    foreach (Lot lot in lots)
    {
        ReturnLotDTO lotDTO = _mapper.Map<ReturnLotDTO>(lot);
        Location location = await _unitOfWork.LocationRepository.GetById(lot.LocationId);
        LocationDTO locationDTO = _mapper.Map<LocationDTO>(location);
        lotDTO.Location = locationDTO;
        lotDTOs.Add(lotDTO);
    }
    return lotDTOs;
}

public Task<IEnumerable<ReturnSimpleLotDTO>> Get(GetLotOptionsDTO getLotOptionsDTO)
{
    GetLotOptions getLotOptions = _mapper.Map<GetLotOptions>(getLotOptionsDTO);
    IEnumerable<Lot> lots = await _unitOfWork.LotRepository.GetByLotType(getLotOptions.LotType);
    if (getLotOptions.State == State.Open)
    {
        lots = lots.Where(l => l.Status == State.Open);
    }
    else if (getLotOptions.State == State.Close)
    {
        lots = lots.Where(l => l.Status == State.Close);
    }
    if (getLotOptions.SortType == SortType.ByCostRaising)
    {
        lots = lots.OrderBy(l => l.BuyPrice);
    }
    else if (getLotOptions.SortType == SortType.ByCostDescending)
    {
        lots = lots.OrderByDescending(l => l.BuyPrice);
    }
    List<ReturnSimpleLotDTO> lotDTOs = new List<ReturnSimpleLotDTO>();
    foreach (Lot lot in lots)
    {
        ReturnSimpleLotDTO lotDTO = _mapper.Map<ReturnSimpleLotDTO>(lot);
        Location location = await _unitOfWork.LocationRepository.GetById(lot.LocationId);
        LocationDTO locationDTO = _mapper.Map<LocationDTO>(location);
        if (locationDTO.Country == null)
            locationDTO.Country = "";
        if (locationDTO.City == null)
            locationDTO.City = "";
        if (locationDTO.House == null)
            locationDTO.House = "";
        if (locationDTO.Street == null)
            locationDTO.Street = "";
        if (locationDTO.Region == null)
            locationDTO.Region = "";
        lotDTO.Location = locationDTO;
        lotDTOs.Add(lotDTO);
    }
    return lotDTOs;
}

public async Task<int> GetViewsByLotId(Guid lotId)
{
    return await _unitOfWork.LotRepository.GetViewsByLotId(lotId);
}

public async Task Take(Guid lotId, Guid managerIdLink)
{
    var lotManager = new LotManager();
    lotManager.LotId = lotId;
    lotManager.ManagerId = await _unitOfWork.UserRepository.GetByIdLink(managerIdLink).Id;
}

```

```

        await
    }
    _unitOfWork.LotManagerRepository.Add(lotManager);
    await _unitOfWork.Save();
}
}

```

## LotController.cs

```

using Business.Contract.Model.LotManagement;
using Business.Contract.Model.LotManagement.Lot;
using Business.Contract.Services.LotManagement;
using Domain.Entity.Constants;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Threading.Tasks;

namespace WebAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class LotController : BaseController
    {
        private readonly ILotService _lotService;

        public LotController(ILotService lotService)
        {
            _lotService = lotService;
        }

        [HttpPost]
        [Route("[action]")]
        [Authorize(AuthenticationSchemes = "Bearer")]
        public async Task<ActionResult>
Create(CreateLotDTO newLot)
        {
            try
            {
                return Ok(await _lotService.Create(newLot,
GetUserId()));
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [HttpPost]
        [Route("[action]")]
        [Authorize(AuthenticationSchemes = "Bearer")]
        public async Task<ActionResult> Take(Guid lotId)
        {
try
{
    var IdLink = GetUserId();
    await _lotService.Take(lotId, IdLink);
    return Ok("lot taken by manager idLink: " +
IdLink);
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}

[HttpPut]
[Route("[action]")]
[Authorize(AuthenticationSchemes = "Bearer")]
public async Task<ActionResult>
Update(UpdateLotDTO newLot, Guid lotId)
{
try
{
    await _lotService.Update(newLot, lotId);
    return Ok(lotId + " lot updated");
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}

[HttpPut]
[Route("[action]")]
[AllowAnonymous]
public async Task<ActionResult> Viewed(Guid lotId)
{
try
{
    await _lotService.Viewed(lotId);
    return Ok(lotId + "lot updated");
}
catch (Exception ex)
{
    return BadRequest(ex.Message);
}
}
}

```

```

        }

    [HttpPost]
    [Route("[action]")]
    [Authorize(AuthenticationSchemes = "Bearer")]
    public async Task<ActionResult> Approve(Guid lotId)
    {
        try
        {
            await _lotService.Approve(lotId);
            return Ok(lotId + " lot approved");
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [HttpDelete]
    [Route("[action]")]
    [Authorize(AuthenticationSchemes = "Bearer")]
    public async Task<ActionResult> Delete(Guid lotId)
    {
        try
        {
            await _lotService.Delete(lotId);
            return Ok(lotId + " lot deleted");
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [HttpGet]
    [Route("[action]")]
    [AllowAnonymous]
    public async Task<ActionResult> GetById(Guid lotId)
    {
        try
        {
            return Ok(await _lotService.GetById(lotId));
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [HttpGet]
    [Route("[action]")]
    [AllowAnonymous]
    public async Task<ActionResult> GetMy(Guid lotId)
    {
        try
        {
            return Ok(await _lotService.GetMy(GetUserId()));
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [HttpGet]
    [Route("[action]")]
    [AllowAnonymous]
    public async Task<ActionResult> GetAll()
    {
        try
        {
            return Ok(await _lotService.GetAll());
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [HttpPost]
    [Route("[action]")]
    [AllowAnonymous]
    public async Task<ActionResult> Get(GetLotOptionsDTO getLotOptionsDTO)
    {
        try
        {
            return Ok(await _lotService.Get(getLotOptionsDTO));
        }
        catch (Exception ex)
        {
            return BadRequest(ex.Message);
        }
    }

    [HttpGet]
    [Route("[action]")]
    [AllowAnonymous]
    public async Task<ActionResult> GetViewsByLotId(Guid lotId)
    {
        try
        {
    
```

```
        return Ok(await _lotService.GetViewsByLotId(lotId));  
    }  
    catch (Exception ex)  
    {  
        return BadRequest(ex.Message);  
    }  
}
```

## newLot.tsx

```
import React, { useState, useEffect, useCallback } from "react";
import {
  Container,
  Col,
  Row,
  Button,
  InputGroup,
  FormControl,
} from "react-bootstrap";
// import { BootstrapTable, TableHeaderColumn } from "react-bootstrap-table";
import style from "./NewLot.module.sass";
import saveLot from "./Services/SaveLot";
import savePriceCoef from "./Services/SavePriceCoef";
import { useHistory } from "react-router-dom";
import UploadImage from "./Component/UploadImage/UploadImage";
import SaveImages from "./Services/SaveImages";
import TextData from "../../../../Assets/jsonData/TextData/NewLotPage.json";
import Spinner from "react-bootstrap/spinner";
import MapComponent from "../../../../Components/Map/MapComponent";
import LinkConfig from "../../../../Assets/jsonData/LinkConfig/LinkConfig.json";
import { MapContextProvider } from "../../../../Components/Map/useMapContext";

export default function NewLot() {
  let history = useHistory();

  const [dataStatus, saveDataStatus] = useState({
    isLoading: false,
    requests: null,
    inProgress: null,
  });

  const [lotId, setLotId] = useState(null);

  useEffect(() => {
    console.log("lotId, ", lotId);
  }, [lotId]);
}
```

```
const [lotData, setLotData] = useState({  
  header: "",  
  description: "",  
  buyPrice: 100,  
  isRent: false,  
  isAuction: false,  
  minBidPrice: 100,  
  minBidStep: 100,  
  auctionDuration: 1,  
  location: {  
    latitude: 0,  
    longitude: 0,  
    country: "",  
    region: "",  
    city: "",  
    street: "",  
    house: "",  
  },  
});  
  
const handleSetLocationFunction = (value) => {  
  setLotData((prev) => ({ ...prev, location: value }));  
};  
  
const handleSetLocation = useCallback((value) => {  
  handleSetLocationFunction(value);  
}, []);  
  
const handleInputChangeFunction = (event) => {  
  const { name, value } = event.target;  
  console.log(name, value);  
  setLotData((prev) => ({ ...prev, [name]: value }));  
};  
  
const handleInputChangeLocation = (event) => {  
  const { name, value } = event.target;  
  console.log(name, value);  
  setLotData((prev) => ({  
    ...prev,  
    location: {  
      ...prev.location,  
      [name]: value,  
    },  
  }));  
};
```





```

    style={{ display: "flex", flexDirection:
"column" }}}

>
<label>Min bid</label>
<input
  className={style.input_style}
  type="number"
  min="100"
  max="50000000000"
  id="flexCheckDefault"
  step="100"
  name="minBidPrice"
  value={lotData.minBidPrice}
  onChange={(e) => {
    handleInputChange(e);
  }}
/>

<small id="emailHelp" class="form-text text-muted">
  minimum bid price
</small>
</div>
<div
  className={style.text_input}
  style={{ display: "flex", flexDirection:
"column" }}>
  <label>Min bid step</label>
  <input
    className={style.input_style}
    type="number"
    min="100"
    max="50000000000"
    id="flexCheckDefault"
    step="100"
    name="minBidStep"
    value={lotData.minBidStep}
    onChange={(e) => {
      handleInputChange(e);
    }}
/>

<small id="emailHelp" class="form-text text-muted">
  minimum bid step
</small>
</div>
<div
  className={style.text_input}
  style={{ display: "flex", flexDirection:
"column" }}>
  <label>auction duration</label>
  <input
    className={style.input_style}
    type="number"
    min="1"
    max="100"
    id="flexCheckDefault"
    step="1"
    name="auctionDuration"
    value={lotData.auctionDuration}
    onChange={(e) => {
      handleInputChange(e);
    }}
/>
<small id="emailHelp" class="form-text text-muted">
  set duration of auction
</small>
</div>
</div>
)}
</Col>
<Col>
  {lotData.isAuction && (
    <div
      className={style.text_input}
      style={{ display: "flex", flexDirection:
"column" }}>
      <label>Buy Price</label>
      <input
        className={style.input_style}
        type="number"
        min="100"
        max="50000000000"
        id="flexCheckDefault"
        step="100"
        name="buyPrice"
        value={lotData.buyPrice}
        onChange={(e) => {
          handleInputChange(e);
        }}
      />
      <small id="emailHelp" class="form-text text-muted">
        Enter buy price
      </small>
    </div>
  )}
</Col>
</Row>
</div>
</Col>

```

```

</Row>
<Row>
  {lotData.isRent && (
    <Col className={style.container_style}>
      <Container>
        <Row className="HeaderText">
          <div>{TextData.Services}</div>
        </Row>
        <Row>
          <div className={style.price_coefs_style}>
            {/* <BootstrapTable
              data={priceCoefs}
              bodyStyle={{ border: "none" }}
              tableStyle={{ border: "none" }}
              headerStyle={{ border: "none !important" }}
              version="4"
            >
              <TableHeaderColumn width="100" isKey
                dataField="id">
                ID
              </TableHeaderColumn>
              <TableHeaderColumn width="100"
                dataField="days">
                Months
              </TableHeaderColumn>
              <TableHeaderColumn width="100"
                dataField="cost">
                Cost
              </TableHeaderColumn>
            </BootstrapTable> */}
            </div>
          </Row>
          <Row>
            <Col>
              <InputGroup className="form-control">
                <InputGroup.Text>Months</InputGroup.Text>
              </InputGroup>
            </Col>
            <Col>
              <FormControl
                onChange={(e) =>
                  daysClickHandler(e.target.value)}
                type="number"
                min="1"
                max="100"
                aria-label="Months (Dyration for rent)"
              />
            </Col>
            <Col>
              <InputGroup className="form-control">
                <InputGroup.Text>Cost</InputGroup.Text>
              </InputGroup>
            </Col>
          </Row>
        <Row>
          <Col>
            <FormControl
              onChange={(e) =>
                costClickHandler(e.target.value)}
              type="number"
              min="1"
              max="10000"
              step="100"
              aria-label="Cost (Cost of rent dyring this
                period)"
            />
          </Col>
        </Row>
        <Row>
          <Col>
            <Button variant="primary"
              onClick={addNewPriceCoef}>
              Add price coef
            </Button>
          </Col>
        </Row>
      </Container>
    </Col>
  )}

  {/* description */}
  <Row>
    <Col className={style.container_style}>
      <div className={style.description_col_style}>
        <label>{TextData.Description}</label>
        <textarea
          className={style.description_area_style}
          name="description"
          value={lotData.description}
          onChange={(e) => {
            handleInputChange(e);
          }}>
        </textarea>
      </div>
    </Col>
  </Row>
  <Row>
    <Col className={style.container_style}>
      <div className={style.map_col_style}>
        <MapContextProvider>
          <MapComponent
            isNewLot={true}
            handleSetLocation={handleSetLocation}
            lotLocation={lotData.location}
            center={{ lat: lotData.location.latitude,
                      lng: lotData.location.longitude,
                    }}>
        </MapContextProvider>
      </div>
    </Col>
  </Row>

```

```

</div>
</Col>
<Col className={style.text_input_col}>
  <div className={style.text_input}>
    <label>Country</label>
    <input
      type="header"
      class="form-control"
      id="exampleInputEmail1"
      aria-describedby="headerHelp"
      placeholder="Enter Country"
      name="country"
      value={lotData.location.country}
      onChange={(e) => {
        handleInputChangeLocation(e);
      }}
    />
    <small id="emailHelp" class="form-text text-muted">
      Enter country text.
    </small>
  </div>
  <div className={style.text_input}>
    <label>Region</label>
    <input
      type="header"
      class="form-control"
      id="exampleInputEmail1"
      aria-describedby="headerHelp"
      placeholder="Enter region"
      name="region"
      value={lotData.location.region}
      onChange={(e) => {
        handleInputChangeLocation(e);
      }}
    />
    <small id="emailHelp" class="form-text text-muted">
      Enter region text.
    </small>
  </div>
  <div className={style.text_input}>
    <label>City</label>
    <input
      type="header"
      class="form-control"
      id="exampleInputEmail1"
      aria-describedby="headerHelp"
      placeholder="Enter city"
      name="city"
      value={lotData.location.city}
      onChange={(e) => {
        handleInputChangeLocation(e);
      }}
    />
    <small id="emailHelp" class="form-text text-muted">
      Enter city text.
    </small>
  </div>
  <div className={style.text_input}>
    <label>Latitude</label>
    <input
      type="header"
      class="form-control"
      id="exampleInputEmail1"
      aria-describedby="headerHelp"
      placeholder="Enter latitude"
      name="latitude"
      value={lotData.location.latitude}
      onChange={(e) => {
        handleInputChangeLocation(e);
      }}
    />
    <small id="emailHelp" class="form-text text-muted">
      Enter latitude text.
    </small>
  </div>
  <div className={style.text_input}>
    <label>Longitude</label>
    <input
      type="header"
      class="form-control"
      id="exampleInputEmail1"
      aria-describedby="headerHelp"
      placeholder="Enter longitude"
      name="longitude"
      value={lotData.location.longitude}
      onChange={(e) => {
        handleInputChangeLocation(e);
      }}
    />
    <small id="emailHelp" class="form-text text-muted">
      Enter longitude text.
    </small>
  </div>

```

```

        aria-describedby="headerHelp"
        placeholder="Enter house number"
        name="latitude"
        value={lotData.location.latitude}
        disabled
      />
      <small id="emailHelp" class="form-text text-
muted">
        Enter street text.
      </small>
    </div>
    <div className={style.text_input}>
      <label>Longitude</label>
      <input
        type="header"
        class="form-control"
        id="exampleInputEmail1"
        aria-describedby="headerHelp"
        placeholder="Enter house number"
        name="longitude"
        value={lotData.location.longitude}
        disabled
      />
      <small id="emailHelp" class="form-text text-
muted">
        street text.
      </small>
    </div>
  </Col>
</Row>
<Row className={style.save_row_style}>
  <Col className={style.container_style}>
    <Button
      className={style.lot_button}
      variant="primary"
      onClick={newLot}
      style={{ backgroundColor: "#4CAF50",
borderColor: "#2f6d31" }}
    >
      {TextData.Create}
    </Button>
  </Col>
</Row>
</Container>
)
</div>
);
}

```

## LotList.tsx

```

import React, { useState, useEffect } from "react";
import { useHistory } from "react-router-dom";
import {
  Container,
  Col,
  Row,
  Button,
  DropdownButton,
  Dropdown,
} from "react-bootstrap";
import Spinner from "react-bootstrap/spinner";

import { SimpleLot } from "../../Components/Types/Lot";
import LoadLotsService from "./Services/LoadLotsService";
import LotCardDeck from "../../Components/LotCard/LotCardDeck/LotCardDeck";
import TextData from "../../Assets/jsonData/TextData/LotList.json";
import style from "./LotListStyle.module.sass";
import { Trans } from "react-i18next";

interface LotListProps {}

function LotList(props: LotListProps) {
  let history = useHistory();

  const [dataLoading, setDataLoading] = useState({
    isLoading: true,
    requests: null,
    inProgress: null,
  });

  const back = () => {
    history.push({
      pathname: "/home",
    });
  };

  const clearSortTypes = () => {
    setSelectedParam({
      lotType: "All",
      sortType: "Default",
      state: "Default",
    });
  };

  const [lots, setLots] = useState<SimpleLot[]>();

  const SortTypes: string[] = ["Default", "ByCostRaising",
"ByCostDescending"];

```

```

        ) : (
      <Container>
        <div className={style.present_col}>
          <div className={style.description}>
            <h1>
              <Trans i18nKey="LandBuilding">LAND AND
              BUILDING</Trans>
            </h1>
            <h5>
              <Trans i18nKey="BestOffer">
                Find your best offer for you with our service
              </Trans>
            </h5>
          </div>
        </div>
        <Row className={style.lot_list_header_text}>
          <div>
            <Trans i18nKey="LotsList">List of lots</Trans>
          </div>
        </Row>
        <div className={style.top_row_options_style}>
          <div className={style.sort_col_style}>
            <label className={style.sort_label_style}>
              <Trans i18nKey="SortType">sort type</Trans>
            </label>
            <DropdownButton
              className={style.drop_down_button}
              title={selectedParams.sortType}
              onSelect={setSortTypeHandler}
            >
              {SortTypes.map((item, id) => (
                <Dropdown.Item key={id} eventKey={item}>
                  {item}
                </Dropdown.Item>
              )))
            </DropdownButton>
          </div>
          <div style={{ width: "6rem" }}>
            <label className={style.sort_label_style}>
              <Trans i18nKey="LotType">lot type</Trans>
            </label>
            <DropdownButton
              className={style.drop_down_button}
              title={selectedParams.lotType}
              onSelect={setLotTypeHandler}
            >
              {LotTypes.map((item, id) => (
                <Dropdown.Item key={id} eventKey={item}>
                  {item}
                </Dropdown.Item>
              )))
            </DropdownButton>
          </div>
        </div>
      <div style={{ width: "6rem" }}>
        <label className={style.sort_label_style}>
          <Trans i18nKey="SortType">sort type</Trans>
        </label>
        <DropdownButton
          className={style.drop_down_button}
          title={selectedParams.sortType}
          onSelect={setSortTypeHandler}
        >
          {SortTypes.map((item, id) => (
            <Dropdown.Item key={id} eventKey={item}>
              {item}
            </Dropdown.Item>
          )))
        </DropdownButton>
      </div>
    </div>
  );
}

const LotTypes: string[] = ["All", "Rent", "Auction"];
const States: string[] = ["Default", "Open", "Close"];
const [selectedParams, setSelectedParam] = useState<{
  lotType: string;
  sortType: string;
  state: string;
}>({
  lotType: "All",
  sortType: "Default",
  state: "Default",
});

const setSortTypeHandler = (value: any) => {
  setSelectedParam((prev) => ({
    ...prev,
    sortType: value,
  }));
};

const setLotTypeHandler = (value: any) => {
  setSelectedParam((prev) => ({
    ...prev,
    lotType: value,
  }));
};

const setStateHandler = (value: any) => {
  setSelectedParam((prev) => ({
    ...prev,
    state: value,
  }));
};

useEffect(() => {
  LoadLotsService({
    selectedParams: selectedParams,
    setLots: setLots,
    setDataLoading: setDataLoading,
  });
}, [selectedParams]);

return (
  <div className={style.lotlist_page_background}>
    <Container>
      {dataLoading.isLoading ? (
        <Spinner animation="border" role="status">
          <span className="visually-hidden">Loading...</span>
        </Spinner>
      )
    )
  )
);

```

```

        </DropdownButton>
    </div>
    <div>
        <label className={style.sort_label_style}>
            <Trans i18nKey="StateType">state type</Trans>
        </label>
        <DropdownButton
            className={style.drop_down_button}
            title={selectedParams.state}
            onSelect={setStateHandler}
        >
            {States.map((item, id) => (
                <Dropdown.Item key={id} eventKey={item}>
                    {item}
                </Dropdown.Item>
            )))
        </DropdownButton>
    </div>
    <div style={{ display: "flex", justifyContent:
        "bottom" }}>
        <Button
            style={{ width: "5rem", height: "2.5rem", marginTop: "1.5rem", variant="primary", onClick={clearSortTypes} }>
            <Trans i18nKey="Clear">clear</Trans>
        </Button>
    </div>
    <div>
        <Col>{lots && <LotCardDeck lots={lots} />}</Col>
        </Container>
    <}>
        </Container>
    </div>
);
}

export default LotList;

```

## LotView.tsx

```

/* eslint-disable react-hooks/exhaustive-deps */
import React from "react";
import { useCallback, useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import Spinner from "react-bootstrap/spinner";
import { Container, Col, Row } from "react-bootstrap";

```

```

import { Trans } from "react-i18next";
import LoadDetailLotInfoService from "./Services/LoadDetailLotInfoService";
import UpdateLotViewsService from "./Services/UpdateLotViewsService";
import LoadImagesService from "../../../../Components/LotCard/LotCard/Service/LoadImagesService";
import { LotImage } from "../../../../Components/Types/LotImage";
import { DetailedLot } from "../../../../Components/Types/Lot";
import { PriceCoef } from "../../../../Components/Types/PriceCoef";
import { Bid } from "../../../../Components/Types/Bid";
import LotImageCarousel from "../../../../Components/Image/ImageCarousel/LotImageCarousel";
import MapComponent from "../../../../Components/Map/MapComponent";
import AuctionLot from "./Components/BuyLot/BuyLot";
import RentLot from "./Components/RentLot/RentLot";
import LoadBidsService from "./Services/LoadBidsService";
import LoadPriceCoefService from "./Services/LoadPriceCoefService";
import style from "./LotView.module.sass";
interface LotViewProps {}

function LotView(props: LotViewProps) {
    const params: { id: string } = useParams();
    const [date, setDateFunction] = useState<string>();
    const [remainingTime, setRemainingTime] = useState<string>("");
    const setDate = useCallback((arg: string) => {
        setDateFunction(arg);
    }, []);
    let lotInitialState = {
        id: "",
        ownerId: "",
        status: "",
        header: "",
        description: "",
        views: 0,
        publicationDate: "",
        buyPrice: 0,
        minBidPrice: 0,
        minBidStep: 0,
    };

```

```

auctionDuration: 0,
isRent: false,
isAuction: false,
location: {
  latitude: 0,
  longitude: 0,
  country: "",
  region: "",
  city: "",
  street: "",
  house: "",
},
ownerInfo: {
  Name: "",
  Surname: "",
  Role: "",
  Email: "",
  PhoneNumber: "",
},
images: [],
priceCoefs: [],
bids: [],
};

const [priceCoef, setPriceCoef] =
useState<PriceCoef[]>([]);

const [lot, setLot] = useState<DetailedLot>(lotInitialState);

const [lotInfo, setLotInfo] =
useState<DetailedLot>(lotInitialState);

const [dataLoading, setDataLoading] = useState({
  isLoading: true,
  requests: null,
  inProgress: null,
});

useEffect(() => {
  LoadDetailLotInfoService({
    lotId: params.id,
    setLotInfo: setLot,
    setDataLoading: setDataLoading,
  });
}, []);

useEffect(() => {
  if (lot.id !== "") {
    setLotInfo(lot);
    UpdateLotViewsService({ lotId: lot.id });
  }
});

LoadImagesService({
  lotId: lot.id,
  setImageArray: (arg: LotImage[]) => {
    setLotInfo((prev) => ({ ...prev, images: arg }));
  },
});

if (lot.isRent === true) {
  LoadPriceCoefService({
    lotId: lot.id,
    setPriceCoef: setPriceCoef,
  });
}

if (lot.isAuction === true) {
  LoadBidsService({
    lotId: lot.id,
    setBids: (arg: Bid[]) => {
      setLotInfo((prev) => ({ ...prev, bids: arg }));
    },
  });
}

setDataLoading((prev: any) => ({
  ...prev,
  isLoading: false,
  requests: false,
  inProgress: false,
}));


useEffect(() => {
  let tempArray: PriceCoef[] = [];
  for (let i = 0; i < priceCoef.length; i++) {
    let item: PriceCoef = priceCoef[i];
    item.number = i + 1;
    tempArray.push(item);
  }
  setLotInfo((prev) => ({ ...prev, priceCoefs: tempArray }));
}, [priceCoef]);

useEffect(() => {
  if (lotInfo.publicationDate) {
    setDate(new Date(lotInfo.publicationDate).toDateString());
  }
});

let tempTime = new Date(lotInfo.publicationDate);
tempTime.setDate(tempTime.getDate() +
  lotInfo.auctionDuration);
setRemainingTime(tempTime.toISOString());
}

console.log("lotInfo", lotInfo);

```

```

    }, [lotInfo.publicationDate]);
    <Trans i18nKey="LotViewDescription">Description</Trans>
    return (
      <div className={style.lotview_page_background}>
        <Container>
          {dataLoading.isLoading ? (
            <Spinner animation="border" role="status">
              <span className="visually-hidden">Loading...</span>
            </Spinner>
          ) : (
            <Container>
              {/* Header row */}
              <div>
                <div className={style.header_text}>
                  <Trans i18nKey="LotViewHeader">lot</Trans>
                </div>
              </div>
              {/* Image download and set main properties */}
              <Row>
                <Col className={style.container_style}>
                  <div className={style.center_image_carousel}>
                    <LotImageCarousel imgArray={lotInfo.images} />
                  </div>
                </Col>
                <Col className={style.lot_info_column}>
                  <label style={{ justifyContent: "right" }} className={style.date_style}>
                    {"Published: " + date}
                  </label>
                  <label className={style.text_header}>{lotInfo.header}</label>
                  {lotInfo.isAuction && (
                    <AuctionLot lotInfo={lotInfo} remainingTime={remainingTime}/>
                  )}
                  {lotInfo.isRent && (
                    <RentLot lotInfo={lotInfo} />
                  )}
                </Col>
              </Row>
              {/* description */}
              <Row>
                <Col className={style.container_style}>
                  <div className={style.description_col_style}>
                    <label>
                      i18nKey="LotViewCountry">Country</Trans>
                    </label>
                    <input type="header" className="form-control" id="exampleInputEmail1" aria-describedby="headerHelp" placeholder="Country" name="country" value={lotInfo.location.country} disabled/>
                  </div>
                  <div className={style.text_input}>
                    <label>
                      i18nKey="LotViewRegion">Region</Trans>
                    </label>
                    <input type="header" className="form-control" id="exampleInputEmail1" aria-describedby="headerHelp" placeholder="region" />
                  </div>
                </Col>
              </Row>
            </Container>
          ) : (
            <div>
              <div>
                <label>
                  i18nKey="LotViewDescription">Description</Trans>
                </label>
                <textarea className={style.description_area_style} name="description" value={lotInfo.description} disabled/>
              </div>
            </div>
          )
        </Container>
      </div>
    )
  )
}

const LotView = ({ lotInfo }) => {
  const [dataLoading, setDataLoading] = useState(false);
  const [remainingTime, setRemainingTime] = useState(null);
  const [isAuction, setIsAuction] = useState(false);
  const [isRent, setIsRent] = useState(false);

  useEffect(() => {
    const interval = setInterval(() => {
      const now = new Date();
      const publicationDate = new Date(lotInfo.publicationDate);
      const timeLeft = publicationDate - now;
      const daysLeft = Math.floor(timeLeft / (1000 * 60 * 60 * 24));
      const hoursLeft = Math.floor((timeLeft % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
      const minutesLeft = Math.floor((timeLeft % (1000 * 60 * 60)) / (1000 * 60));
      const secondsLeft = Math.floor((timeLeft % (1000 * 60)) / 1000);
      setRemainingTime(` ${daysLeft}d ${hoursLeft}h ${minutesLeft}m ${secondsLeft}s`);
    }, 1000);
    return () => clearInterval(interval);
  }, []);

  const handleSetLocation = () => {
    const location = {
      latitude: lotInfo.location.latitude,
      longitude: lotInfo.location.longitude,
    };
    const coordinates = JSON.stringify(location);
    localStorage.setItem("location", coordinates);
  };

  return (
    <div>
      <div>
        <div>
          <div>
            <div>
              <div>
                <div>
                  <div>
                    <div>
                      <div>
                        <div>
                          <div>
                            <div>
                              <div>
                                <div>
                                  <div>
                                    <div>
                                      <div>
                                        <div>
                                          <div>
                                            <div>
                                              <div>
                                                <div>
                                                  <div>
                                                    <div>
                                                      <div>
                                                        <div>
                                                          <div>
                                                            <div>
                                                              <div>
                                                                <div>
                                                                  <div>
                                                                    <div>
                                                                      <div>
                                                                        <div>
                                                                          <div>
                                                                            <div>
                                                                              <div>
                                                                                <div>
                                                                                  <div>
                                                                                    <div>
                                                                                      <div>
                                                                                        <div>
                                                                                          <div>
                                                                                            <div>
                                                                                              <div>
                                                                                                <div>
                                                                                                  <div>
                                                                                                    <div>
                                                                                                      <div>
                                                                                                        <div>
                                                                                                          <div>
                                                                                                            <div>
                                                                                                              <div>
                                                                                                                <div>
                                                                                                                  <div>
                                                                                                                    <div>
                                                                                                                      <div>
                                                                                                                        <div>
                                                                                                                          <div>
                                                                                                                            <div>
                                                                                                                              <div>
                                                                                                                                <div>
                                                                                                                                  <div>
                                                                                                                                    <div>
                                                                                                                                      <div>
                                                                                                                                        <div>
                                                                                                                                          <div>
                                                                                                                                            <div>
                                                                                              </div>
                            </div>
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    )
  );
};

export default LotView;

```

```

name="region"                               </Col>
value={lotInfo.location.region}             </Row>
disabled                                     <Row className={style.save_row_style}>
/>                                         <Col className={style.container_style}>
</div>                                       <label>
<div className={style.text_input}>                <Trans
<label> i18nKey="LotViewViews">Views</Trans>
<Trans i18nKey="LotViewCity">City</Trans>
</label>                                     {lotInfo.views}
<input type="header"                         </label>
className="form-control"                     </Col>
id="exampleInputEmail1"                     </Row>
aria-describedby="headerHelp"               </Container>
placeholder="city"                         )}
name="city"                                 </Container>
value={lotInfo.location.city}               </div>
disabled                                     );
/>                                         )
</div>                                       );
<div className={style.text_input}>                export default LotView;
<label>
<Trans i18nKey="LotViewStreet">Street</Trans>
</label>
<input type="header"                         import React, { useState } from "react";
className="form-control"                     import { Modal, Button } from "react-bootstrap";
id="exampleInputEmail1"                     import { Trans } from "react-i18next";
aria-describedby="headerHelp"               import CountdownTimer from
placeholder="street"                        "./../../../../../Components/TimeCounter/CountdownTimer";
name="street"                                import { CreateAgreement } from
value={lotInfo.location.street}            "./../../../../../Components/Types/Agreement";
disabled                                     import BadRequest from
/>                                         "./../../../../../Components/Message/BadRequest";
</div>                                       import { DetailedLot } from
<div className={style.text_input}>                "./../../../../../Components/Types/Lot";
<label>
<Trans i18nKey="LotViewHouse">House</Trans>
</label>
<input type="header"                         import CreateNewAgreementService from
className="form-control"                     "./../../Services/CreateNewAgreementService";
id="exampleInputEmail1"                     import style from "./BuyLotStyle.module.sass";
aria-describedby="headerHelp"               import { CreateBid } from
placeholder="house number"                 "./../../../../../Components/Types/Bid";
name="house"                                import PlaceBidService from
value={lotInfo.location.house}             "./../../Services/PlaceBidService";
disabled                                     interface BuyLotProps {
/>                                         lotInfo: DetailedLot;
</div>                                       remainingTime: string;
                                         }
                                         }

                                         function BuyLot(props: BuyLotProps) {
                                         const [goodRequest, setGoodRequest] = useState<{

```

## BuyLot.tsx

```

import React, { useState } from "react";
import { Modal, Button } from "react-bootstrap";
import { Trans } from "react-i18next";

import CountdownTimer from
"./../../../../../Components/TimeCounter/CountdownTimer";
import { CreateAgreement } from
"./../../../../../Components/Types/Agreement";

import BadRequest from
"./../../../../../Components/Message/BadRequest";

import { DetailedLot } from
"./../../../../../Components/Types/Lot";
import CreateNewAgreementService from
"./../../Services/CreateNewAgreementService";

import style from "./BuyLotStyle.module.sass";
import { CreateBid } from
"./../../../../../Components/Types/Bid";
import PlaceBidService from
"./../../Services/PlaceBidService";

interface BuyLotProps {
  lotInfo: DetailedLot;
  remainingTime: string;
}

function BuyLot(props: BuyLotProps) {
  const [goodRequest, setGoodRequest] = useState<{

```

```

show: boolean;                                         };
message: string;
} > ({ show: false, message: "" });
const [badRequest, setBadRequest] = useState<{
  show: boolean;
  message: string;
}> ({ show: false, message: "" });

const [showWindow, setShowWindow] = useState<{
  bid: boolean;
  buy: boolean;
}> ({ bid: false, buy: false });

const [bid, setBid] = useState<CreateBid>({
  lotId: props.lotInfo.id,
  value: props.lotInfo.bids
  ? props.lotInfo.bids[props.lotInfo.bids.length - 1].value +
    props.lotInfo.minBidStep
  : props.lotInfo.minBidPrice,
});
const [expired, setExpired] = useState<boolean>(false);
useState<CreateAgreement>({
  lotId: props.lotInfo.id,
  description: "test description",
  startDate: new Date().toISOString(),
  endDate: new Date().toISOString(),
});
const BuyLot = () => {
  console.log(agreement);
  CreateNewAgreementService({ agreement: agreement });
};

const PlaceBid = () => {
  var res = PlaceBidService({ bid: bid });
  if (res) {
    setGoodRequest((prev) => ({
      ...prev,
      show: res,
      message: "bid created",
    }));
  } else {
    setBadRequest((prev) => ({
      ...prev,
      show: res,
      message: "something went wrong",
    }));
  }
  setShowWindow((prev) => ({ ...prev, bid: false }));
}

  return (
    <div>
      <BadRequest show={badRequest.show}>
        text={badRequest.message} />
      <BadRequest show={goodRequest.show}>
        text={badRequest.message} />
      <Modal
        style={{ display: "flex", marginTop: "10%" }}
        show={showWindow.bid}
        getOpenState={(e: any) => {
          setShowWindow((prev) => ({ ...prev, bid: e }));
        }}
        tabIndex="-1"
      >
        <Modal.Header>
          <Modal.Title style={{ justifyContent: "center" }}>
            <div className={style.modal_header}>
              <p>
                <Trans i18nKey="LotViewBodyApproveBid">BodyApproveBid</Trans>
                </p>
              </div>
            </Modal.Title>
            <Modal.Header>
              <Modal.Body>
                <p className={style.modal_text}>
                  <Trans i18nKey="LotViewBodyApproveBidDescription">
                    LotViewBodyApproveBidDescription
                  </Trans>
                  </p>
                  <div>
                    <p>
                      {props.lotInfo.bids
                        ? `Current bid winner is ${props.lotInfo.bids[props.lotInfo.bids.length - 1].value
                          ? `${props.lotInfo.bids[props.lotInfo.bids.length - 1].value
                            ? `Your bid should be higher than ${props.lotInfo.minBidStep}$` :
                            `Minimum bid: ${props.lotInfo.minBidPrice}$`}
                          : ``}
                        : ``}
                    </p>
                  </div>
                </Modal.Body>
              </Modal.Header>
            </Modal>
          <div className={style.bid_input_container}>
            <label>bid</label>
            <input
              className={style.input_style}
              type="number"
            >
          </div>
        </Modal>
      </div>
    );
}

```

```

min={
    props.lotInfo.bids
    ? props.lotInfo.bids[props.lotInfo.bids.length -
1].value
        : props.lotInfo.minBidPrice
    }
max="50000000000"
id="flexCheckDefault"
step={props.lotInfo.minBidStep}
name="minBidPrice"
value={bid.value}
onChange={(e) => {
    setBid((prev) => ({
        ...prev,
        value: Number(e.target.value),
    }));
}}
/>
<small id="emailHelp" className="form-text text-
muted">
    minimum bid price
</small>
</div>
</div>
</Modal.Body>
<Modal.Footer>
    <Button
        variant="success"
        onClick={() => {
            PlaceBid();
        }}
    >
        <Trans i18nKey="FooterContinue">FooterContinue</Trans>
        <Button>
            <div className="vr" />
            <Button
                variant="secondary"
                onClick={() => {
                    setShowWindow((prev) => ({ ...prev, bid: false }));
                }}
            >
                <Trans i18nKey="FooterCancel">FooterCancel</Trans>
                <Button>
                    </Button>
                </Modal.Footer>
            </Modal>
            <div className={style.auction_text_style}>
                <label className={style.text_style}>
                    {props.lotInfo.bids
                    ? "Highest bid is " +
                    props.lotInfo.bids[props.lotInfo.bids.length - 1].value
                        : "Minimum bid: " + props.lotInfo.minBidPrice}
                    {" or Buy it now for " + props.lotInfo.buyPrice}
                </label>
            <div>
                Pick</label>
                <div className={style.auction_options_style}>
                    <div className={style.bid_container}>
                        <CountdownTimer
                            targetDate={new Date(props.remainingTime)}
                            setExpired={setExpired}
                        />
                        <label className={style.bids_count_style}>
                            {props.lotInfo.bids
                                ? props.lotInfo.bids.length + ". bids"
                                    : "0 bids"}
                        </label>
                    </div>
                    <Button
                        className={style.button_style}
                        variant="primary"
                        onClick={() => {
                            setShowWindow((prev) => ({ ...prev, bid: true }));
                        }}
                        style={{
                            backgroundColor: "#4CAF50",
                            borderColor: "#2f6d31",
                        }}
                        disabled={expired ? true : false}
                    >
                        <Trans i18nKey="LotViewPlaceBid">PlaceBid</Trans>
                    </Button>
                    <Button
                        className={style.button_style}
                        variant="primary"
                        onClick={BuyLot}
                        style={{
                            backgroundColor: "#4CAF50",
                            borderColor: "#2f6d31",
                        }}
                        disabled={expired ? true : false}
                    >
                        <Trans i18nKey="LotViewBuyNow">BuyNow</Trans>
                    </Button>
                </div>
            </div>
        </div>
    </div>
);
```

```

        }

    export default BuyLot;
}

const [showRentWindow, setShowRentWindow] = useState<boolean>(false);
const [selectedPriceCoefIdState, setSelectedPriceCoefIdState] =
    useState<string>("");

const RentLot = () => {
    selectedPriceCoefIdState !== undefined &&
        SelectPriceCoefService({ priceCoefId: selectedPriceCoefIdState });

    console.log(agreement);
    CreateNewAgreementService({ agreement: agreement });
};

return (
    <div>
        <BadRequest show={badRequest.show} text={badRequest.message} />
        <Modal style={{ display: "flex", marginTop: "10%" }} show={showRentWindow} getOpenState={(e: any) => setShowRentWindow(e)} tabIndex="-1">
            <Modal.Header>
                <Modal.Title style={{ justifyContent: "center" }}>
                    <div className={style.modal_header}>
                        <p>
                            <Trans i18nKey="LotViewApproveRent">Approve rent</Trans>
                        </p>
                    </div>
                </Modal.Title>
                <Modal.Header>
                    <Modal.Body>
                        <p className={style.modal_text}>
                            <Trans i18nKey="LotViewBodyApprove">Body Approve</Trans>
                        </p>
                        <div className={style.date_style}>
                            <div className={style.date_picker_style}>
                                <Trans i18nKey="LotViewBodyStartDate">StartDate</Trans>
                                    <Datetime value={moment(agreement.startDate).format("DD-MM-YYYY")}>
                                        onChange={(value: any) => {
                                            setAgreement((prev) => ({
                                                ...prev,
                                                startDate: value.format(),
                                            }));
                                        }}
                                    </Datetime>
                            </div>
                        </div>
                    </Modal.Body>
                </Modal.Header>
            </Modal>
        </div>
    </div>
);
}

```

## RentLot.tsx

```

import React, { useState } from "react";
import { Modal, Table, Button } from "react-bootstrap";
import moment from "moment";
import Datetime from "react-datetime";

import TheaderList from "../Table/TheaderList";
import Tbody from "../Table/Tbody";
import { Trans } from "react-i18next";

import { CreateAgreement } from "../../../../Components/Types/Agreement";
import CreateNewAgreementService from "../../../../Services/CreateNewAgreementService";
import SelectPriceCoefService from "../../../../Services/SelectPriceCoefService";
import { DetailedLot } from "../../../../Components/Types/Lot";
import BadRequest from "../../../../Components/Message/BadRequest";
import style from "./RentLotStyle.module.sass";

interface RentLotProps {
    lotInfo: DetailedLot;
}

function RentLot(props: RentLotProps) {
    const [badRequest, setBadRequest] = useState<{
        show: boolean;
        message: string;
    }>({
        show: false,
        message: "",
    });

    const [agreement, setAgreement] = useState<CreateAgreement>({
        lotId: props.lotInfo.id,
        description: "test description",
        startDate: new Date("2022-08-01T11:55:03.030Z").toISOString(),
        endDate: new Date("2022-12-06T11:55:03.030Z").toISOString(),
    });
}

```

```

        }
    />
</div>
<div className={style.date_picker_style}>
    <Trans
        i18nKey="LotViewBodyEndDate">EndDate</Trans>
        <Datetime
            value={moment(agreement.endDate).format("DD
MM YYYY")}>
            onChange={(value: any) => {
                setAgreement((prev) => ({
                    ...prev,
                    endDate: value.format(),
                }));
            }}
        />
    </div>
    </div>
    <div className={style.description_col_style}>
        <label>
            <Trans
                i18nKey="LotViewDescription">Description</Trans>
                </label>
                <textarea
                    className={style.description_area_style}
                    name="description"
                    value={agreement.description}
                    onChange={(e: any) => {
                        setAgreement((prev) => ({
                            ...prev,
                            description: e.value,
                        }));
                    }}
                />
            </div>
            <Modal.Body>
                <Modal.Footer>
                    <Button
                        variant="success"
                        onClick={() => {
                            RentLot();
                            setShowRentWindow(!showRentWindow);
                        }}
                    />
                </Trans
                    i18nKey="FooterContinue">FooterContinue</Trans>
                    </Button>
                    <div className="vr" />
                    <Button
                        variant="secondary"
                        onClick={() => {
                            setShowRentWindow(!showRentWindow);
                            setAgreement((prev) => ({
                                ...prev,
                                endDate: value.format(),
                            }));
                        }}
                    />
                <Trans
                    i18nKey="FooterCancel">FooterCancel</Trans>
                    <Button
                        variant="primary"
                        onClick={() => {
                            setShowRentWindow(!showRentWindow);
                            setAgreement((prev) => ({
                                ...prev,
                                endDate: value.format(),
                            }));
                        }}
                    />
                </Trans
                    i18nKey="LotViewRentNow">RentNow</Trans>
                    <Table responsive>
                        <HeaderList />
                        <Tbody
                            bodyData={props.lotInfo.priceCoefs}
                            setSelectedPriceCoefIdState={(arg: string) =>
                                setSelectedPriceCoefIdState(arg)
                            }
                        />
                    </Table>
                </div>
                <div className={style.rent_button_container}>
                    <Button
                        variant="primary"
                        onClick={() => {
                            if (selectedPriceCoefIdState === "") {
                                setBadRequest({
                                    show: true,
                                    message: "select Price Coef "
                                });
                            }
                            setShowRentWindow(true);
                        }}
                        style={{
                            backgroundColor: "#4CAF50",
                            borderColor: "#2f6d31",
                        }}
                        disabled={selectedPriceCoefIdState === "" ? true : false}
                    />
                </div>
            </Trans
                i18nKey="LotViewRentNow">RentNow</Trans>
                </Button>
            </div>
        </div>
    );
}

export default RentLot;

```

## Додаток Е. Скріншоти інтерфейсу користувача

LS Eng Ukr Lots Statistics Agreement list Create new lot Profile About Us Ivan Ivanovych IvanIvanovych@gmail.com User ▾

### LAND AND BUILDING

Find your best offer for you with our service

List of lots

sort type Default ▾      lot type All ▾      state type Default ▾      clear

**LotHeader**  
Publication date: 30 May 2022  
Auction  
Buy price: 900  
L'viv's'ka oblast L'viv Halyts'ki Pasiky Street

**Sold house**  
Publication date: 31 May 2022  
Rent  
Lviv Oblast Lviv Troleibusna Street

**beautiful new home**  
Publication date: 31 May 2022  
Auction  
Buy price: 22000  
L'viv's'ka oblast L'viv Kulparkivska Street

**land for selling**  
Publication date: 31 May 2022  
Rent  
L'viv's'ka oblast L'viv Haiovs'koi Street

Previous

Рис. Е.1. Сторінка зі списком оголошень

LS 2022 Home About Us Ivan Ivanovich IvanIvanovich@gmail.com User ▾

### New Lot



Date: Tuesday, May 31, 2022

**Lot Header**  
Sold house  
Enter header text. This text will be on the top of your lot.

allow rent  
 allow auction

**Description**

This home is currently under construction at 1394 W Watson Lane and will soon be available for move-in at Anderson Farms Collection. Contact Ivory Homes today to lock in your options!

😊 📲



Enter the address

**Country**  
Ukraine  
Enter country text.

**Region**  
Lvivska oblast  
Enter region text.

**City**  
Lviv  
Enter city text.

**Street**  
Viacheslava Chornovola Avenue  
Enter street text.

**House**  
45a  
Enter house text.

**Latitude**  
49.85160370694793  
Enter latitude text.

**Longitude**  
24.02292727976227  
Enter longitude text.



LandSelling  
All rights reserved © 2021

**Navigation**

- [Lots](#)
- [About us](#)
- [Rules](#)

**Social media:**

- [YouTube: LandSelling.ua](#)
- [Instagram: @LandSelling.ua](#)
- [Facebook: LandSelling](#)
- [Telegram: LandSelling](#)

**Contact us:**

+380 959 171 229

Рис. E.2. Сторінка створення нового оголошення

LS Eng Ukr Lots Statistics Agreement list Create new lot Profile About Us Ivan Ivanovich Ivanivanovich@gmail.com User ▾

## Lot



Published: Tue May 31 2022

**beautiful new home**

Highest bid is 3000 or Buy it now for 22000

Your Pick

**Expired!!!**

This auction is expired, you can not place a bid.

place bid    buy it now

6. bids

Description

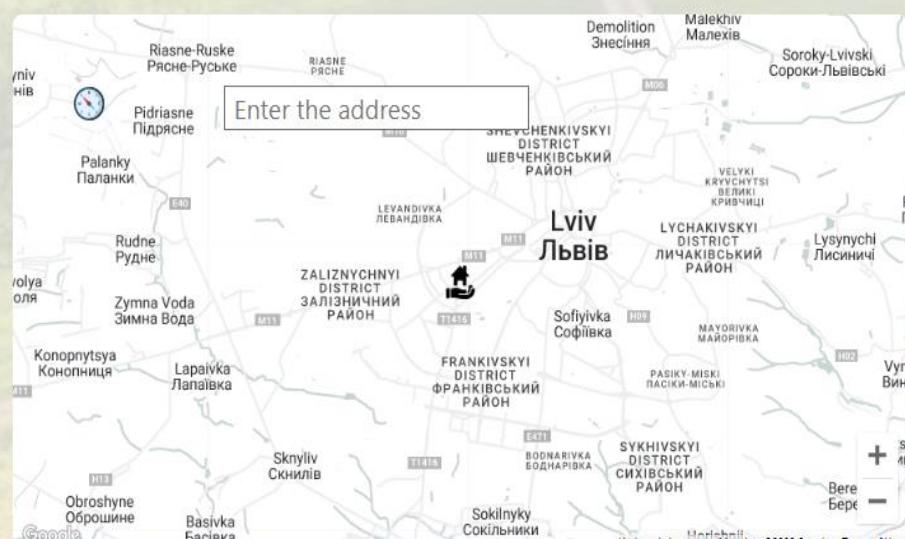
Green Program

Ivory Green

Now Building

This home is currently under construction at 1397 W Watson Lane and will soon be available for move-in at Anderson Farms Collection. Contact Ivory Homes today to lock in your options!

One of the most exciting parts about this charming neighborhood is the future park going in across the street. The park will house pavilions, basketball courts, soccer fields, and baseball diamonds. It is sure to be a great addition to the community, and it will be perfect for those warm Utah summers.



Enter the address

Country: Ukraine

Region: L'viv's'ka oblast

City: L'viv

Street: Kulparkivska Street

House: 37

Views: 37

Рис. Е.3. Сторінка перегляду оголошення з аукціоном

LS Eng Ukr Lots Statistics Agreement list Create new lot Profile About Us Ivan Ivanovych Ivanivanovych@gmail.com User ▾

### Lot



[Previous](#) [Next](#)

Published: Sat Jun 11 2022

| № | Months | Price |          |
|---|--------|-------|----------|
| 1 | 0      | 901   | Selected |
| 2 | 0      | 2501  | Select   |

[rent it now](#)

---

Description

Description for rent

Country: Ukraine

Region: Lvivs'ka oblast

City: Lviv

Street: Hlynyanskyi Trakt Street

House: 88



Enter the address

Views: 5



Navigation

- [Lots](#)
- [About Us](#)
- [Rules](#)

Social media

- [YouTube: LandSelling.ua](#)
- [Instagram: @LandSelling.ua](#)
- [Facebook: LandSelling](#)
- [Telegram: LandSelling](#)

Contact us

+380 959 171 229

Рис. Е.4. Сторінка перегляду оголошення з орендою

The screenshot shows a user management interface. At the top, there's a navigation bar with links for Home, Statistics, Lots, Users, and About Us. On the right, it shows the current user as 'Valentyn Burets' with an email of 'kingandgames28@gmail.com' and a role of 'Admin'. Below the navigation is a title 'User List'. A table displays three users with columns for First Name, Last Name, Email, and Role. The users are: Ivan Ivanovich (Email: ivanivanovich@gmail.com, Role: User), Stepan Stepkin (Email: step@gmail.com, Role: User), and Valentyn Burets (Email: kingandgames28@gmail.com, Role: Admin). Below the table are pagination controls: 'Rows per page: 10', '1-3 of 3', and navigation arrows. A large 'Add user' button is centered at the bottom.

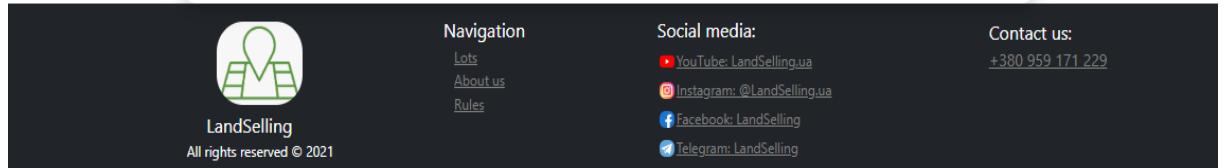


Рис. Е.5. Сторінка перегляду списку користувачів системи

This screenshot shows a modal window titled 'Add user' overlaid on the User List page. The modal contains fields for 'First name: \*' (Ivan), 'Last name: \*' (Ivanovich), 'email: \*' (ivanivanovich@gmail.com), and a 'Choose Role' dropdown set to 'User'. There's also a 'password: \*' field and a large 'Add user' button at the bottom. The background shows the same user list and footer as the previous screenshot.

Рис. Е.6. Сторінка додавання нового користувача

The screenshot shows a web-based management system. At the top, there is a table listing three agreements:

| 1 | Lot | Customer | test description | Default | 2022-06-03 19:28:51 |                     |                     |
|---|-----|----------|------------------|---------|---------------------|---------------------|---------------------|
| 2 | Lot | Customer | new Agreement    | Open    | 2022-05-31 18:52:29 | 2022-08-20 14:46:01 | 2022-11-20 14:46:01 |
| 3 | Lot | Customer | test description | Close   | 2022-06-04 12:18:51 | 2022-06-04 09:18:41 | 2022-06-04 09:18:41 |

Below the table, there is a modal window titled "Оплата" (Payment) with the following content:

Ви зираєтеся оплатити угоду

Pay with card

VISA MasterCard AMEX JCB DISCOVER

Card Number: 4111 1111 1111 1111 VISA

Expiration Date (MM/YY): 11 / |

Buttons: "оплатити" (Pay), "повернутися" (Return)

At the bottom of the page, there is a footer with the LandSelling logo, navigation links (Lots, About Us, Rules), social media links (YouTube, Instagram, Facebook, Telegram), and contact information (Contact us: +380 959 171 429).

Рис. Е.7. Сторінка додавання нового користувача

The screenshot shows the "Who we are?" section of the website. It features two versions of the same illustration of a diverse group of people cheering.

**Left Version (Ukrainian):**

- Header: "Хто ми?"
- Text: "Ми – команда молодих професіоналів, які хочуть розробити нову та унікальну продукцію, яка не залишить вас бездумки."
- Image: An illustration of a diverse group of people cheering.
- Section: "Наша головна ціль"
- Text: "Наши головные цели – это получить максимальную выручку от разработки этого сайта, чтобы использовать его и наилучшую в новых и более сложных проектах. Каждый из нас хочет развиваться в этом направлении и развивать все лучшие и лучшие проекты."

**Right Version (English):**

- Header: "Who we are?"
- Text: "We are a team of young professionals who want to develop new and unique products that will not leave you indifferent."
- Image: An illustration of a diverse group of people cheering.
- Section: "Our main goal"
- Text: "Our main goal is to gain maximum skills in the development of this site to apply them in the future in new and even better projects. Each of us wants to develop in this direction in the future and develop better and better projects."

Рис. Е.8. Сторінка з описом команди

LS 2022 Home About Us Account ▾

What is "LandSelling"?

**Our team**

Used instruments

Valentyn Burets  
Front-end and back end developer

Navigation  
[Lots](#)  
[About us](#)  
[Rules](#)

Social media:  
[YouTube: LandSelling.ua](#)  
[Instagram: @LandSelling.ua](#)  
[Facebook: LandSelling](#)  
[Telegram: LandSelling](#)

Contact us:  
+380 959 171 229

All rights reserved © 2021

Рис. Е.9. Сторінка з автором

LS 2022 Home About Us Account ▾

What is "LandSelling"?

Our team

**Used instruments**

**Front-end**

React (also known as React.js or ReactJS) is an open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality

**Back-end**

The .NET Framework (pronounced as "dot net") is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library called Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (in contrast to a hardware environment) named the Common Language Runtime (CLR). The CLR is an application virtual machine that provides services such as security, memory management, and exception handling. As such, computer code written using .NET Framework is called "managed code". FCL and CLR together constitute the .NET Framework.

Рис. Е.10. Сторінка з описом використаних технологій

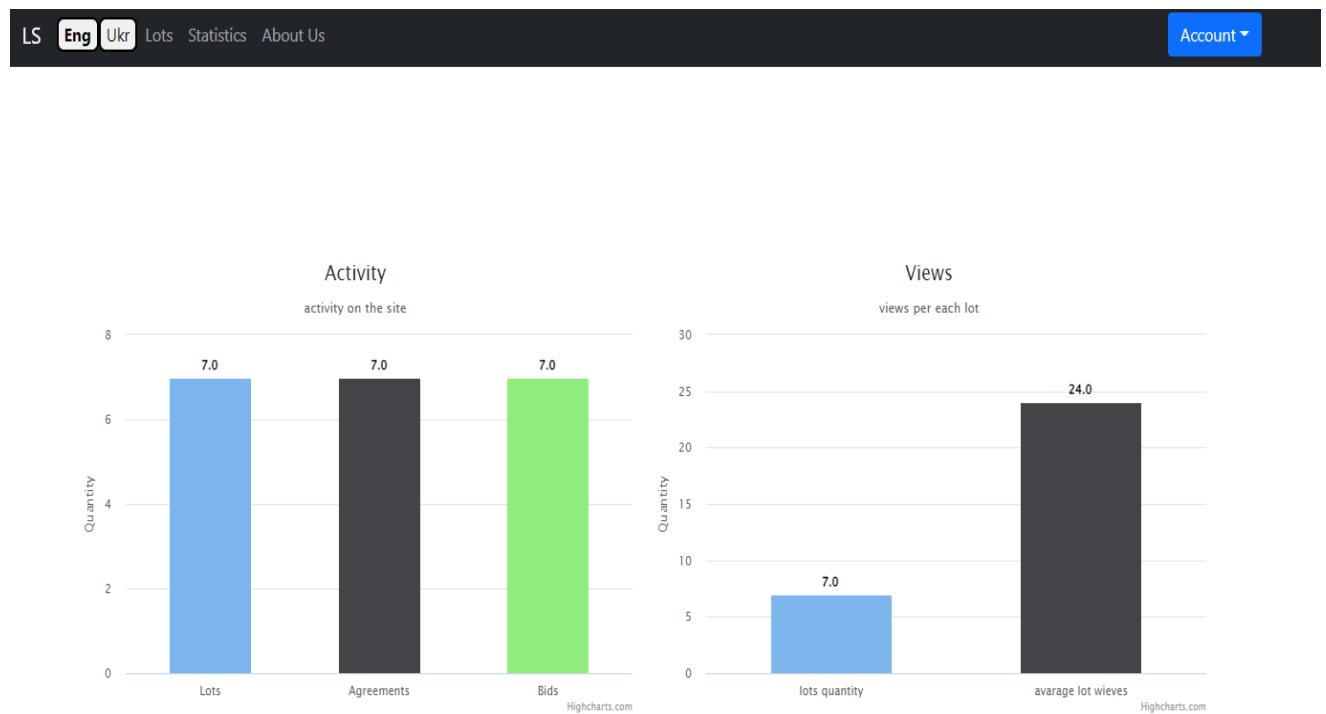


Рис. Е.11. Сторінка зі статистикою

This screenshot shows a table of land lots managed on the platform. Each row contains information about a specific lot, including its publication date, buy price, minimum bid price, minimum bid step, auction duration, whether it's for rent or auction, and its location.

|   | Publication date    | buy price | min bid price | min bid step | auction duration | is rent | is auction | location                  | Actions   |
|---|---------------------|-----------|---------------|--------------|------------------|---------|------------|---------------------------|---|
| 5 | 2022-05-30 23:37:18 | 900       | 200           | 100          | 1                | -       | +          | Lvivs'ka oblast Lviv 4    | <button>Update</button> <button>Delete</button> |
| 2 | 2022-05-31 14:33:17 | 100       | 100           | 100          | 1                | +       | -          | Lviv Oblast Lviv 7        | <button>Update</button> <button>Delete</button> |
| 7 | 2022-05-31 22:30:47 | 22000     | 1200          | 300          | 6                | -       | +          | Lvivs'ka oblast Lviv 37   | <button>Update</button> <button>Delete</button> |
|   | 2022-05-31 22:37:14 | 100       | 100           | 100          | 1                | +       | -          | Lvivs'ka oblast Lviv 80   | <button>Update</button> <button>Delete</button> |
|   | 2022-06-04 21:40:25 | 7000      | 1200          | 700          | 20               | -       | +          | Львівська область Львів 8 | <button>Update</button> <button>Delete</button> |
| 7 | 2022-06-11 16:20:24 | 10000     | 3700          | 200          | 14               | -       | +          | Lviv Oblast Lviv          | <button>Update</button> <button>Delete</button> |
|   | 2022-06-11 16:21:18 | 100       | 100           | 100          | 1                | +       | -          | Lvivs'ka oblast Lviv 88   | <button>Update</button> <button>Delete</button> |



Рис. Е.12. Сторінка менеджменту списку оголошень

LS Eng Ukr Lots Statistics Agreement list Create new lot Profile About Us Ivan Ivanovich Ivanivanovych@gmail.com User ▾

## list of my agreements

| Nº | lot link            | customer link            | description      | status  | creation date       | start date          | end date            | Price |                         |                            |
|----|---------------------|--------------------------|------------------|---------|---------------------|---------------------|---------------------|-------|-------------------------|----------------------------|
| 1  | <a href="#">Lot</a> | <a href="#">Customer</a> | new Agreement    | Open    | 2022-05-31 18:52:29 | 2022-08-20 14:46:01 | 2022-11-20 14:46:01 | 900   | <a href="#">approve</a> | <a href="#">disapprove</a> |
| 2  | <a href="#">Lot</a> | <a href="#">Customer</a> | test description | Default | 2022-06-04 21:53:51 |                     |                     | 0     | <a href="#">approve</a> | <a href="#">disapprove</a> |
| 3  | <a href="#">Lot</a> | <a href="#">Customer</a> | test description | Close   | 2022-06-04 12:18:51 | 2022-06-04 09:18:41 | 2022-06-04 09:18:41 | 22000 | <a href="#">approve</a> | <a href="#">disapprove</a> |
| 4  | <a href="#">Lot</a> | <a href="#">Customer</a> | test description | Default | 2022-06-11 16:40:18 | 2022-06-11 13:40:11 | 2022-06-11 13:40:11 | 10000 | <a href="#">approve</a> | <a href="#">disapprove</a> |
| 5  | <a href="#">Lot</a> | <a href="#">Customer</a> |                  | Default | 2022-06-11 16:40:59 |                     |                     | 0     | <a href="#">approve</a> | <a href="#">disapprove</a> |

## list of agreements requests

| Nº | lot link            | description      | status  | creation date       | start date          | end date            | Price | payments                 |                     |                            |
|----|---------------------|------------------|---------|---------------------|---------------------|---------------------|-------|--------------------------|---------------------|----------------------------|
| 1  | <a href="#">Lot</a> | test description | Close   | 2022-06-04 12:18:51 | 2022-06-04 09:18:41 | 2022-06-04 09:18:41 | 22000 | <a href="#">payments</a> | <a href="#">Pay</a> | <a href="#">disapprove</a> |
| 2  | <a href="#">Lot</a> | test description | Default | 2022-06-04 21:53:22 | 2022-06-04 18:52:52 | 2022-06-04 18:52:52 | 22000 | <a href="#">payments</a> | <a href="#">Pay</a> | <a href="#">disapprove</a> |
| 3  | <a href="#">Lot</a> | test description | Default | 2022-06-04 21:53:51 |                     |                     | 0     | <a href="#">payments</a> | <a href="#">Pay</a> | <a href="#">disapprove</a> |
| 4  | <a href="#">Lot</a> | test description | Default | 2022-06-11 21:36:51 | 2022-06-11 18:36:50 | 2022-06-11 18:36:50 | 10000 | <a href="#">payments</a> | <a href="#">Pay</a> | <a href="#">disapprove</a> |
| 5  | <a href="#">Lot</a> | test description | Default | 2022-06-11 21:41:15 | 2022-06-11 18:41:11 | 2022-06-11 18:41:11 | 10000 | <a href="#">payments</a> | <a href="#">Pay</a> | <a href="#">disapprove</a> |

  
**LandSelling**  
 All rights reserved

[Navigation](#)  
[Lots](#)  
[About Us](#)  
[Rules](#)

[Social media](#)  
[YouTube: LandSelling.ua](#)  
[Instagram: @LandSelling.ua](#)  
[Facebook: LandSelling](#)  
[Telegram: LandSelling](#)

[Contact us](#)  
 +380 959 171 229

Рис. E.13. Сторінка менеджменту списку угод