Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Дербеньов Гліб Сергійович,

студент групи АС-151

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

Модель автоматизованого укладання угоди купівлі-продажу нерухомості

Спеціальність:

121 – Інженерія програмного забезпечення

Спеціалізація:

Інженерія програмного забезпечення

Керівник:

Паулін Олег Миколайович*,*

доктор технічних наук, доцент

Одеса – 2020

**ЗМІСТ**

[ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ 4](#_Toc58789525)

[ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» 6](#_Toc58789526)

[АНОТАЦІЯ 7](#_Toc58789527)

[ВСТУП 8](#_Toc58789528)

[1 ВИМОГИ ДО ПРОГРАМНОЇ СИСТЕМИ 10](#_Toc58789529)

[1.1 Визначення бізнес-вимог 10](#_Toc58789530)

[1.2 Визначення функціональних вимог 11](#_Toc58789531)

[1.3 Визначення нефункціональних вимог 19](#_Toc58789532)

[1.4 Висновки до розділу 21](#_Toc58789533)

[2 ПЛАН ВИКОНАННЯ ПРОЕКТУ 22](#_Toc58789534)

[2.1 Оцінка тривалості розробки 22](#_Toc58789535)

[2.2 Розробка плану робіт проекту 26](#_Toc58789536)

[2.3 Висновки до розділу 30](#_Toc58789537)

[3 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ 31](#_Toc58789538)

[3.1 Проектування архітектури системи 31](#_Toc58789539)

[3.2 Проектування графічного інтерфейсу користувача 36](#_Toc58789540)

[3.3 Висновки до розділу 39](#_Toc58789541)

[4 РОЗРОБКА СИСТЕМИ 40](#_Toc58789542)

[4.1 Опис програмних технологій 40](#_Toc58789543)

[4.2 Опис програмних бібліотек 40](#_Toc58789544)

[4.3 Структури даних 40](#_Toc58789545)

[4.4 Інструкція з встановлення програмного продукту 49](#_Toc58789546)

[4.5 Інструкція з використання 49](#_Toc58789547)

[4.6 Висновки до розділу 49](#_Toc58789548)

[5 ТЕСТУВАННЯ ТА ВИПРОБУВАННЯ ПРОГРАМНОЇ СИСТЕМИ 50](#_Toc58789549)

[5.1 Модульне тестування 50](#_Toc58789550)

[5.2. Функціональне тестування 51](#_Toc58789551)

[5.3 Висновки до розділу 54](#_Toc58789552)

[6 ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ ПРОГРАМНОЇ СИСТЕМИ 55](#_Toc58789553)

[6.1 Експериментальне оцінювання результатів укладання угоди 55](#_Toc58789554)

[6.2 Висновки до розділу 57](#_Toc58789555)

[7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 58](#_Toc58789556)

[ВИСНОВКИ 60](#_Toc58789557)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 61](#_Toc58789558)

[ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ 62](#_Toc58789559)

[ДОДАТОК Б. ПИТАННЯ З ОХОРОНИ ПРАЦІ ТА БЕЗПЕКИ У НАДЗВИЧАЙНИХ СИТУАЦІЯХ 79](#_Toc58789560)

[Б.1 Організація та управління охороною праці 79](#_Toc58789561)

[Б.2 Визначення основних параметрів умов праці 81](#_Toc58789562)

[Б.3 Розрахунок заземлення 83](#_Toc58789563)

[Б.4 Визначення категорії пожежонебезпечного офісного приміщення 86](#_Toc58789564)

[Б.5 Розрахунок тривалості перебування людей на радіоактивно зараженій місцевості при встановленій дозі опромінення 88](#_Toc58789565)

[Б.6 Заходи з охорони праці та безпеки у надзвичайних ситуаціях 89](#_Toc58789566)

[Б.7 Безпека у суспільстві в умовах загрози COVID-19 90](#_Toc58789567)

Міністерство освіти і науки України

Одеський національний політехнічний університет

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

Спеціалізація: Інженерія програмного забезпечення

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри**

\_\_\_\_\_\_\_\_\_\_\_\_\_ Крісілов В. А.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

# **ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Дербеньова Гліба Сергійовича, група АС-151

1. Тема роботи: Модель автоматизованого укладання угоди купівлі-продажу нерухомості

Керівник роботи: Паулін Олег Миколайович, доктор технічних наук, доцент затверджені наказом ректора від «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_ р. № \_\_\_\_\_\_\_

1. Зміст роботи: вимоги до програмної системи, план виконання проекту, проектування програмної системи, розробка системи, тестування та випробування програмної системи, експериментальне оцінювання результатів укладання угоди
2. Перелік ілюстративного матеріалу: **слайд 1, слайд 2, …, слайд N (узгодити із презентацією)**
3. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання прийняв |
|  | доц. Москалюк А.Ю |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ р.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної роботи | Строк виконання | Примітка |
| 1 | Вимоги до програмної системи | 26.10.2020 | Виконав |
| 2 | План виконання проекту | 02.11.2020 | Виконав |
| 3 | Проектування програмної системи | 09.11.2020 | Виконав |
| 4 | Розробка системи | 30.11.2020 | Виконав |
| 5 | Тестування та випробування програмної системи | 07.12.2020 | Виконав |
| 6 | Визначення властивостей програмної системи | 14.12.2020 | Виконав |
| 7 | Охорона праці та безпека у надзвичайних ситуаціях | 21.12.2020 | Виконав |

**Здобувач вищої освіти**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г. С. Дербеньов

**Керівник роботи**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. М. Паулін

# **ЗАВДАННЯ НА РОЗРОБКУ РОЗДІЛУ «ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ»**

Дербеньова Гліба Сергійовича, група АС-151

Навчально-науковий інститут комп’ютерних систем

Кафедра системного програмного забезпечення

Тема роботи: Модель автоматизованого укладання угоди купівлі-продажу нерухомості

Зміст розділу:

1. Організація та управління охороною праці
2. Аналіз технологічних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях
3. Розрахунок заземлення
4. Визначення категорії пожежонебезпечного офісного приміщення
5. Розрахунок тривалості перебування людей на радіоактивно зараженій місцевості при встановленій дозі опромінення
6. Заходи з охорони праці та безпеки у надзвичайних ситуаціях
7. Безпека у суспільстві в умовах загрози COVID-19

Керівник роботи Консультант з охорони праці та БНС

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.М. Паулін \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Ю. Москалюк

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р. «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

# **АНОТАЦІЯ**

Робота присвячується можливості автоматизації реального процесу укладання угоди купівлі-продажу нерухомості. Аналізується процес укладання угоди і визначаються ті складові, які можливо автоматизувати. Будується модель вирішення основних проблем і завдань в даній області. Метою роботи є скорочення часу на укладання угоди. Завданнями є формування угоди з усіма її учасниками і об'єктом нерухомості, а також проектування і реалізація програмної системи з використанням побудованої моделі, мови програмування Javascript, мови моделювання UML, програмної платформи Node.js та NoSQL документо-орієнтованої системи керування базами даних MongoDB. Результатом є зменшення термінів укладання угоди в середньому до 30%.

Ключові слова: укладення угоди, автоматизація, модель, програмна система, Javascript, UML, Node.js, NoSQL, MongoDB.

**ABSTRACT**

The work is devoted to the possibility of automating the real process of concluding a real estate sales transaction. The process of concluding a deal is analyzed and the components that can be automated are identified. A model for solving the main problems and tasks in this area is being built. The purpose of the work is to reduce the time to conclude a deal. The tasks are to form a transaction with all its participants and the property, as well as design and implement a software system using the constructed model, Javascript programming language, UML modeling language, Node.js software platform and NoSQL document-oriented database management system MongoDB. The result is a reduction in the timing of the transaction by an average of 30%.

# **ВСТУП**

Розглядаючи ринок купівлі-продажу нерухомості як множину предметів, їх властивостей і відносин між ними, варто відзначити існування таких неавтоматизованих, трудомістких і ресурсовитратних процесів як підбір фахівців для укладення угоди, експертне підтвердження об'єкта нерухомості, визначення часток фахівців за участь в угоді. Прийнята в практиці схема протікання процесу укладання угоди купівлі-продажу нерухомості складається з таких етапів: пошук покупця; підготовка угоди; оформлення договору; реєстрація угоди.

Процес купівлі-продажу часто займає дуже багато часу, зобов'язуючи пройти по численних інстанціях для отримання відповідних документів та проведення операцій. Оцифрування даної задачі дозволила б автоматизувати і прискорити процес купівлі-продажу нерухомості.

З наукової точки зору роботу можна представити наступною формулою дослідження:

* актуальність: ринок нерухомості є важливим елементом економіки, який визначається людською потребою в наявності власної нерухомості (в Україні кількість укладених договорів купівлі-продажу квартир і будинків в 2019-му році перевищило 301 тисячу) [1][2];
* мета дослідження: скорочення часу на укладання угоди;
* вирішувані завдання: побудова моделі, формування угоди, проектування і реалізація програмної системи з використанням побудованої моделі;
* об'єкт дослідження: процес укладання угоди;
* предмет дослідження: технічні і юридичні засоби укладання угоди;
* методи дослідження: порівнювальний аналіз, експеримент;
* наукова новизна: вперше запропонована модель укладання угоди, що охоплює усі стадії купівлі-продажу нерухомості;
* практична значимість: реалізація ідеї дозволяє автоматизувати укладання угоди і отримати істотний виграш у затрачуваному часі;
* публікація: “BUILDING A MODEL OF REAL ESTATE SALES TRANSACTION”, «Modern scientific researches», issue No13, No.be13-032.

Моєю метою було покращення власних навичок у створенні програмного забезпечення та здобуття нових з урахуванням досвіду, набутого при навчанні та практиці, на прикладі створення цього програмного забезпечення в реальних умовах.

Потрібно було запропонувати свою власну модель автоматизованого укладання угоди купівлі-продажу нерухомості, провести проектування та розробити програмне забезпечення, яке вирішувало б низку проблем в реальному житті при купівлі-продажу нерухомості, провести необхідне тестування та експеримент та перевірити, чи виконана вимога досягнення початкової мети розробки. З точки зору охорони праці необхідно було проаналізувати робоче місце працівників, які працюватимуть з розробленим програмним забезпеченням, провести необхідні розрахунки та запропонувати заходи для покращення продуктивності праці та збільшення ефективності працівників.

# **1 ВИМОГИ ДО ПРОГРАМНОЇ СИСТЕМИ**

## **1.1 Визначення бізнес-вимог**

Кінцевому користувачу необхідно мати програму платформу, яка відтворює місце, де клієнти зможуть займатися укладанням угод купівлі-продажу об’єктів нерухомості.

Основний аспект, який є проблемою на сьогодні – це автоматизація системи забезпечення купівлі-продажу нерухомості.

Зацікавленими особами в даному випадку є: оцінювач, нотаріус, брокер, забудовник, архітектор, інвестор.

Для аналізу існуючих проблем користувачів та варіантів засобів для вирішення цих проблем зручно використовувати канву ціннісної пропозиції (див. рис. 1.1).

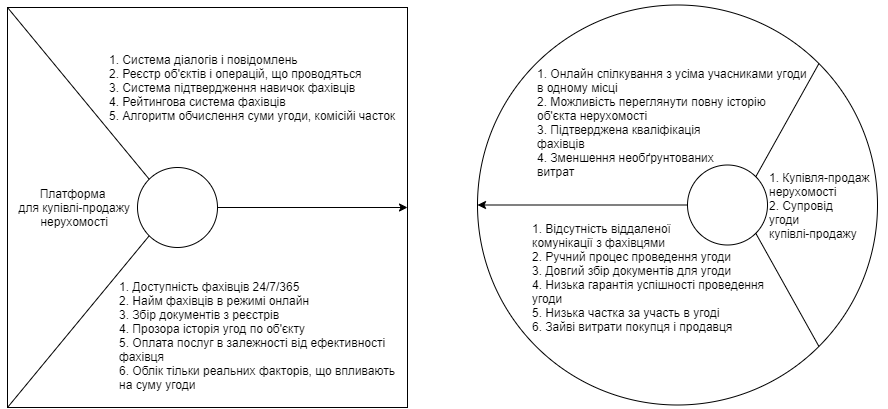


Рисунок 1.1 – Канва ціннісної пропозиції

Існуючі рішення, націлені на повну або часткову автоматизацію купівлі-продажу (нерухомості), не розглядають автоматизацію процесу укладення угоди (див. табл. 1.1).

Таблиця 1.1 – Порівняння існуючих рішень

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва рішення | Автоматизація ведення угод | Експертне підтвердження об'єкта нерухомості | Підбір фахівців | Розподіл часток | Ведення реєстру правочинів |
| Domik.ua | + | - | - | - | - |
| ЛУН | + | - | - | - | - |
| DOM.RIA | + | + | + | - | - |
| Propy | + | + | + | - | + |

Domik.ua є інформаційно-аналітичним ресурсом, присвяченим ринку нерухомості України та пов'язаних з ним сфер: фінансів, профільного законодавства, архітектури та дизайну.

ЛУН є компанією-розробником сервісів з пошуку нерухомості в 37-ми країнах світу (закордонна назва – Flatfy).

DOM.RIA є українським сайтом з продажу та оренди всіх типів нерухомості від приватних осіб, забудовників і агентств нерухомості.

Propy є міжнародною платформою, націленої на автоматизовану організацію угод купівлі-продажу нерухомості без посередників за допомогою технології.

З розгляду таблиці 1.1 видно, що важливим є автоматизація переведення даних об'єкта нерухомості в електронну форму (оцифрування); обліку експертної оцінки об'єкта нерухомості (одержуваної від оцінювача) при формуванні угоди; підбору фахівців, а також обліку, обчислення і розподілу їх часток; ведення реєстру.

## **1.2 Визначення функціональних вимог**

Вводиться поняття «угода» як процесу переходу прав власності на об'єкт нерухомості від однієї особи до іншої з подальшою оплатою. Особливістю угоди при роботі із нерухомістю є необхідність підписання попередніх угод між сторонами, підготовка паперів для укладення угоди, оплата послуг нотаріуса. В силу того, що законодавчо нотаріальне завірення має бути здійснене в нотаріальній конторі, далі розглядається можливість автоматизації процесу укладання угоди без зазначеного підпроцесу.

Запропонована модель є так званим цифровим (smart) контрактом [3]. Структура контракту представлена нижче (див. рис. 1)

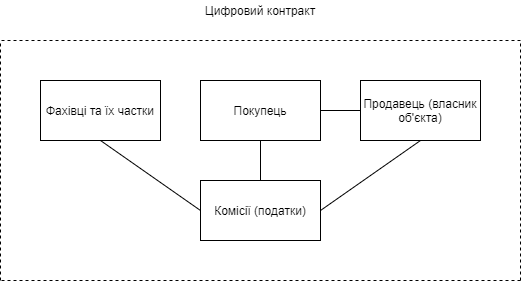


Рисунок 1.2 – Структура цифрового контракту

Представлена структура контракту складається з фундаментальних компонентів будь-якої угоди. Покупець – особа, яка бажає придбати об'єкт нерухомості. Продавець – власник об'єкту нерухомості. Фахівці – оцінювач, нотаріус, брокер. Частки фахівців за участь в угоді – суми винагороди за роботу. Комісії (податки) – суми до оплати.

Підбір фахівців представляється у вигляді використання даних реєстру правочинів про фахівців: якості їх роботи (рейтингу) і географічного розташування.

Для подальшого розуміння механізму обчислень і розподілів сум, які фігурують в угоді, введемо поняття «сума для покупця» і «сума для продавця» і вкажемо формули їх розрахунку в загальному вигляді.

(1.1)

де x – сума для покупця; y – базова вартість об'єкту нерухомості; y1 – база для нарахування податків (оціночна вартість об'єкту нерухомості); Нп – процентна ставка податків, які сплачує покупець; Набс – абсолютна ставка податків, які сплачує покупець; С – абсолютне значення суми часткою фахівців.

(1.2)

де z – сума для продавця; y – базова вартість об'єкту нерухомості; Кс – комісія брокера (у відсотках); Нс – ставка податків, які сплачує продавець (у відсотках).

З формул (1.1) (1.2) видно, що покупець платить за послуги фахівців. Отримані значення є вхідними даними для механізму розподілу.

Таким чином, при формуванні угоди модель покликана підібрати фахівців для проведення операції; розрахувати суми, відведені до виплати / отримання для того чи іншого учасника угоди, і розподілити їх між учасниками угоди; перевести права власності на об'єкт нерухомості від продавця до покупця; зробити запис про нову угоду в реєстрі.

На рис. 1.3 представлені функціональні вимоги до розроблюваної системи за допомогою діаграми варіантів використання мовою UML. Маємо двох акторів: Користувач та Система, між якими наведені 8 варіантів використання. Користувач – це узагальнення різного типу кінцевих користувачів, які користуватимуться програмною системою. В свою чергу розроблюваної програмної системи, яка виконує логіку, відповідну конкретному варіанту використання (прецеденту), – можливості модельованої системи (або частини її функціональності), завдяки якій користувач може отримати конкретний, вимірюваний і потрібний йому результат. Прецедент відповідає окремому сервісу системи, визначає один з варіантів її використання і описує типовий спосіб взаємодії Користувача з Системою.

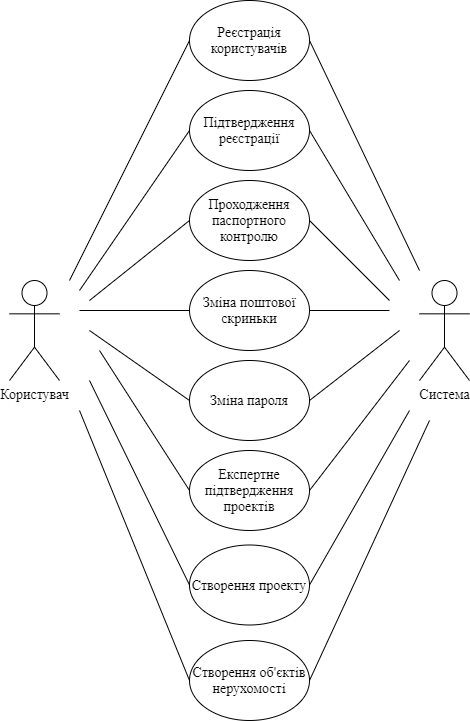


Рисунок 1.3 – Діаграма варіантів використання

З розгляду рис. 1.3 видно, що необхідно реалізувати такі прецеденти:

1. реєстрація користувачів;
2. підтвердження реєстрації;
3. проходження паспортного контролю (тобто становлення членом платформи) з вибором ролі за своїм фахом (брокер, нотаріус, забудовник, оцінювач, архітектор, інвестор) та підтвердження системою членства у платформі;
4. зміна поштової скриньки;
5. зміна пароля в двох варіантах (швидка зміна в особистому кабінеті та через пошту);
6. експертне підтвердження проектів платформи (експертна оцінка, виплата за роботу оцінювачів) з автоматичним підбором відповідного по регіону та спеціалізації експерта;
7. створення (оцифрування) об'єктів нерухомості;
8. створення (оцифрування) проект для забудови.

Нижче наданий більш детальний опис кожного прецеденту:

Прецедент «**Реєстрація користувачів**» відповідає за реєстрацію користувачів.

Мінімальні гарантії: дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно збережені в базу даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може реєструватися, та вводить свої дані.

2. Система завантажує дані та сповіщає користувача про успіх операції.

Альтернативні сценарії:

2.1 Дані невірні, система повідомляє про помилку.

Прецедент «**Підтвердження реєстрації**» відповідає за виконання підтвердження облікової запису вже існуючого користувача.

Мінімальні гарантії: дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно збережені в базу даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може завантажувати дані, та вводить свою пошту.

2 Система завантажує дані, відправляє лист та сповіщає користувача про успіх операції.

Альтернативні сценарії:

2.1 Пошта була введена невірно, система повідомляє про помилку.

Прецедент «**Проходження паспортного контролю**» відповідає за збір інформації про користувачів, а саме їх особисті паспортні дані.

Мінімальні гарантії: дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно збережені в базу даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може завантажувати дані, та вводить паспортні дані.

2. Система отримує дані користувача, перевіряє дані, після чого система повідомляє про успішне закінчення операції.

Альтернативні сценарії:

2.1 Дані введені невірно, система повідомляє про помилку.

Прецедент «**Зміна поштової скриньки**» відповідає за зміну поштової скриньки користувача.

Мінімальні гарантії: доступ до бази даних відсутній, дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно отримані з бази даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може переглядати дані, та тисне кнопку «Зміна поштової скриньки».

2. Система підключається до бази даних та отримує дані.

3. Користувач вводить нову пошту.

4. Система відображає всю збережену інформацію.

Альтернативні сценарії:

2.2 Доступ до бази даних відсутній, система повідомляє про помилку.

3.2 Введена користувачем пошта вже закріплена за іншим користувачем, система повідомляє про помилку.

Прецедент «**Зміна пароля**» відповідає за зміну діючого паролю користувача.

Мінімальні гарантії: доступ до бази даних відсутній, дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно отримані з бази даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може переглядати дані, та тисне кнопку «Зміна паролю».

2. Система підключається до бази даних та отримує дані.

3. Користувач вводить пароль.

4. Система відображає всю збережену інформацію.

Альтернативні сценарії:

2.2 Доступ до бази даних відсутній, система повідомляє про помилку.

Прецедент «**Експертне підтвердження проектів**» відповідає за перевірку проектів, що додаються користувачами, експертами-оцінювачами та надання дозволу на укладання угоди.

Мінімальні гарантії: доступ до бази даних відсутній, дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно отримані з бази даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може переглядати дані, перевіряє обраний проект та підтверджує коректність даних.

2. Система виконує цей запит та створює угоду.

Альтернативні сценарії:

2.2. Запит введений невірно, система повідомляє про помилку.

Прецедент «**Створення проекту**» відповідає за додавання нового проекту до бази даних.

Мінімальні гарантії: доступ до бази даних відсутній, дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно отримані з бази даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може переглядати дані, та вводить дані про проект та учасників угоди.

2. Система виконує цей запит та створює проект.

Альтернативні сценарії:

2.2. Запит введений невірно, система повідомляє про помилку.

Прецедент «**Створення об’єктів нерухомості**» за додавання об’єктів нерухомості у базу даних.

Мінімальні гарантії: доступ до бази даних відсутній, дані не отримані, система продовжує працювати.

Гарантії успіху: дані успішно отримані з бази даних.

Основний успішний сценарій:

1. Користувач переходить до меню, з якого може додати новий об’єкт нерухомості, та вводить дані про проект та учасників угоди.

2. Система оброблює дані та створює новий об’єкт.

Альтернативні сценарії:

3. Дані введені невірно, система повідомляє про помилку.

Тепер встановимо пріоритети для кожного прецеденту, використовуючи принцип MoSCow (див. табл. 1.2).

Таблиця 1.2 – Пріоритети функцій програмної системи

|  |  |
| --- | --- |
| Функція системи | Пріоритет |
| Реєстрація користувачів | M |
| Підтвердження реєстрації | M |
| Проходження паспортного контролю | M |
| Зміна поштової скриньки | W |
| Зміна пароля | S |
| Експертне підтвердження проектів | M |
| Створення проекту | C |
| Створення об’єктів нерухомості | M |

Як видно з таблиці 1.2, різні функції мають різні пріоритети. Пріоритет ‘M’ (must) – функція має найвищий пріоритет і повинна бути першочергово реалізована. Пріоритет ‘S’ (should) – важлива функція, але не з найвищою пріоритетністю, не має вирішального значення, але все одно є обов'язковою до виконання. Пріоритет ‘C’ (could) – бажана функція. Пріоритет ‘W’ (would) – найменш критична вимога.

## **Визначення нефункціональних вимог**

1. При створені програми повинен використовуватися інструментарій NPM.

2. Система повинна виконувати обробку версій нереляційної бази даних, що створена в системі управління базами даних MongoDB версії 4.0 або вище (4.2, 4.4).

3. Система повинна працювати в ОС Windows (7, 8, 10) та Ubuntu (16.04, 18.04, 20.04).

Також за стандартом ISO 9126, із зазначеними у ньому характеристиками та їх атрибутами, сформульовані такі нефункціональні вимоги до програмної системи (див. табл. 1.3).

Таблиця 1.3 – Атрибути характеристик

|  |  |  |
| --- | --- | --- |
| Характеристика по ISO 9126 | Властивість програмного забезпечення | Сценарії атрибутів якості |
| Функціональність | Здатність взаємодіяти | Система взаємодіє з базою даних кожен раз, коли це потрібно користувачу |
| Здатність взаємодіяти | Система успішно отримує дані з бази даних в 96% випадків |
| Безпека | Система здатна протистояти в 92% випадків потенційних загроз |

Продовження таблиці 1.3

|  |  |  |
| --- | --- | --- |
| Характеристика по ISO 9126 | Властивість програмного забезпечення | Сценарії атрибутів якості |
| Функціональність | Безпека | Дані бази даних не можуть бути змінені звичайними користувачами |
| Ефективність | Часові характеристики | Час реакції системи на дії користувача не довше 0.3 сек |
| У разі нормального навантаження система обробляє не менше 75 запитів користувачів в хвилину |
| Використання ресурсів | Додаток займає не більше 90 Мб на пристрої користувача |
| Додаток не споживає більше 5% роботи процесора і 10% оперативної пам'яті пристрою |
| Надійність | Відмовостійкість | Система продовжить роботу в 92% випадків після внутрішнього збою протягом 5 хвилин |

Продовження таблиці 1.3

|  |  |  |
| --- | --- | --- |
| Характеристика по ISO 9126 | Властивість програмного забезпечення | Сценарії атрибутів якості |
| Надійність | Відмовостійкость | Система здатна усунути 90% помилок усередині системи |
| Поновлюваність після порушення в роботі | Система продовжує роботу в 97% випадків при відсутності з’єднання з базою даних |
| Система збереже поточний стан в 92% ситуацій і відновить його при екстреному відключенні протягом хвилин |
| Переносимість | Адаптивність | Система дозволяє працювати на одному типі платформи зі зберіганням функцій |

## **1.4 Висновки до розділу**

В даному розділі була описана предметна область, виявлені проблеми, наявні в процесах предметної області, а також визначена мета проекту. Визначені функціональні вимоги до системи за діаграмою прецедентів. Прецеденти розподілені за пріоритетами за допомогою системи MoSCow. Також описані нефункціональні вимоги до системи, що розробляється.

# **2 ПЛАН ВИКОНАННЯ ПРОЕКТУ**

## **2.1 Оцінка тривалості розробки**

В цьому розділі буде визначена тривалість розробки за методом UCP (Use Case Points). Він складається з п’яти етапів та дозволяє визначити тривалість розробки в годинах на основі діаграми варіантів використання та сценаріїв.

Перший етап – «Оцінка акторів» – дає оцінку складності інтерфейсів системи. Розрізняють три типи акторів: простий, середній та складний.

Актор «Користувач» взаємодіє з системою за допомогою графічного інтерфейсу, тому його тип можна визначити як складний. Вага актора – 3.

Актор «Система» – це API системи, тому його тип можна визначити як простий. Після цього, треба обчислити нескоректовану оцінку акторів (UAW), тобто підрахувати кількість акторів кожного типу та помножити на відповідні вагові категорії.

*UAW* = 1 ∗ 3 + 1 ∗ 1 = 4 (2.1)

Другий етап – «Оцінка варіантів» – оцінює масштаб системи. Визначимо тип варіантів використання за кількістю неподільних операцій в ньому.

Варіант «Реєстрація користувачів» має 2 транзакції (ініціація варіанту використання, збереження результату).

Варіант «Підтвердження реєстрації» має 2 транзакції (ініціація варіанту використання, збереження результату).

Варіант «Проходження паспортного контролю» має 2 транзакції (ініціація варіанту використання, збереження результату).

Варіант «Зміна поштової скриньки» має 3 транзакції (ініціація варіанту використання, збереження результату, вивід результату).

Варіант «Зміна паролю» має 3 транзакції (ініціація варіанту використання, збереження результату, вивід результату).

Варіант «Експертне підтвердження проектів» має 2 транзакції (ініціація варіанту використання, збереження результату).

Варіант «Створення проекту» має 2 транзакції (ініціація варіанту використання, отримання результату).

Варіант «Створення об’єктів нерухомості» має 3 транзакції (ініціація варіанту використання, запит даних у користувача, отримання результату).

В таблиці 2.1 представлено оцінку складності варіантів використання.

Таблиця 2.1 – Оцінка складності варіантів використання системи

|  |  |  |  |
| --- | --- | --- | --- |
| Варіант використання | Кількість транзакцій | Складність | Вага |
| Реєстрація користувачів | 2 | Простий | 5 |
| Підтвердження реєстрації | 2 | Простий | 5 |
| Проходження паспортного контролю | 2 | Простий | 5 |
| Зміна поштової скриньки | 3 | Простий | 5 |
| Зміна паролю | 3 | Простий | 5 |
| Експертне підтвердження проектів | 2 | Простий | 5 |
| Створення проекту | 2 | Простий | 5 |
| Створення об’єктів нерухомості | 3 | Простий | 5 |

Для обчислення нескоректованої оцінки варіантів використання (*UUCW*) необхідно кількість варіантів використання кожного типу помножити на відповідні вагові коефіцієнти.

|  |  |
| --- | --- |
| *UUCW* = 8 ∗ 5 = 40 | (2.2) |

Виконується перша ітерація розробки, тому не потрібно враховувати можливість розділення результатів роботи в команді, так як показник *UCP* обчислюється як сума показників *UAW* та *UUCW* (формула 2.3). Ця оцінка показує складність системи, але без врахування технічних та зовнішніх факторів.

|  |  |  |
| --- | --- | --- |
| *UCP = UAW + UUCW* = 4 + 40 = 44 | (2.3) |  |

Третій етап – «Оцінка технічних факторів» – необхідний для оцінки складності архітектури системи. Оцінка виконується за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 – середній вплив, 5 – сильний вплив на розробку.

Таблиця 2.2 – Оцінка технічних факторів системи

|  |  |  |  |
| --- | --- | --- | --- |
| Фактор | Опис | Вага | Оцінка |
| Т1 | Розподіленість системи | 2 | 0 |
| Т2 | Час відгуку | 1 | 4 |
| Т3 | Ефективність кінцевого користувача | 1 | 3 |
| Т4 | Складність обробки | 1 | 1 |
| Т5 | Фокус на повторному використанні коду | 1 | 4 |
| Т6 | Простота інсталяції | 0,5 | 2 |
| Т7 | Простота використання | 0,5 | 4 |
| Т8 | Портативність | 2 | 4 |
| Т9 | Простота змін | 1 | 3 |
| Т10 | Паралельні обчислення | 1 | 0 |
| Т11 | Засоби захисту | 1 | 0 |
| Т12 | Доступ до третьої сторони | 1 | 0 |
| Т13 | Потреби в спеціальному навчанні | 1 | 1 |

Сума добутків вагових коефіцієнтів та оцінок для кожного з технічних факторів визначає показник TFactor.

|  |  |  |
| --- | --- | --- |
| *TFactor* = 2 \* 0 + 1 \* 4 + 1 \* 3 + 1 \* 1 + 1 \* 4 + 0,5 \* 2 + 0,5 \* 4 + 2 \* 4 + 1 \* 3 + 1 \* 0 + 1 \* 0 + 1 \* 0 + 1 \* 1 = 27 | (2.4) |  |

Оцінка технічного фактору обчислюється за формулою 2.5.

|  |  |
| --- | --- |
| TCF = 0.6 + (0.01 ∗ TFactor) = 0,6 + (0,01 ∗ 27) = 0,87 | (2.5) |

Четвертий етап – «Оцінка зовнішніх факторів» – дає коефіцієнт для організаційних ризиків при розробці. Оцінка виконується за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 – середній вплив, 5 – сильний вплив на розробку. Оцінку зовнішніх факторів представлено в таблиці 2.3.

Таблиця 2.3 – Оцінка зовнішніх факторів системи

|  |  |  |  |
| --- | --- | --- | --- |
| Фактор | Опис | Вага | Оцінка |
| F1 | Знайомство з процесом розробки | 1,5 | 2 |
| F2 | Досвід подібних проектів | 0,5 | 2 |
| F3 | Досвід об'єктно-орієнтованої розробки | 1 | 3 |
| F4 | Досвідченість провідного аналітика | 0,5 | 2 |
| F5 | Мотивація | 1 | 4 |
| F6 | Стабільність вимог | 2 | 4 |
| F7 | Часткова зайнятість працівників | -1 | 3 |
| F8 | Складність мови програмування | -1 | 3 |

Сума добутків вагових коефіцієнтів та оцінок для кожного з зовнішніх факторів визначає показник EFactor.

|  |  |  |
| --- | --- | --- |
| *EFactor* = 1,5 ∗ 2 + 0,5 ∗ 2 + 1 ∗ 3 + 0,5 ∗ 2 + 1 ∗ 4 + 2 ∗ 4 + (−1) ∗ 3 + (−1) ∗ 3 = 14 | (2.6) |  |

Оцінка зовнішнього фактору обчислюється за формулою 2.7.

|  |  |
| --- | --- |
| *EF* = 1.4 + (−0.03 ∗ *EFactor*) = 1,4 + (−0,03 ∗ 14) = 0,98 | (2.7) |

П’ятий етап – підраховуються результуючі оцінки. Скоректовані UCP обчислюються за формулою 2.8.

|  |  |
| --- | --- |
| = 44 ∗ 0,87 ∗ 0,98 = 37,5 | (2.8) |

Для визначення тривалості розробки необхідно з’ясувати, яка кількість робочих годин відповідає одному UCP. Для цього підраховується кількість факторів з множини F1 – F8, оцінки яких за абсолютним значенням перевищують

Таких факторів – 1. Так як результат менше 3, то для розрахунку тривалості розробки приймається, що одному UCP відповідає 20 робочих годин.

Підрахуємо тривалість розробки ТUCP.

*TUCP* = 37,5 \* 20 год = 750 год (2.9)

|  |  |
| --- | --- |
| Отже тривалість розробки дорівнює 750 робочих годин. | (2.9) |

## **2.2 Розробка плану робіт проекту**

Подуємо структурну декомпозицію робіт проекту (WBS – Work Breakdown Structure). Для цього необхідно виділити задачі, які мають бути виконані в проекті та відобразити їх ієрархію. Структурна декомпозиція робіт проекту представлена в таблиці 2.4.

Таблиця 2.4 – Структурна декомпозиція робіт

|  |  |
| --- | --- |
| Назва задачі | WBS |
| Проектування | 1 |
| Архітектура | 1.1 |
| Структура та організація класів | 1.2 |
| Графічний інтерфейс | 1.3 |
| Реалізація | 2 |
| ВВ «Реєстрація користувачів» | 2.1 |
| ВВ «Підтвердження реєстрації» | 2.2 |
| ВВ «Проходження паспортного контролю» | 2.3 |
| ВВ «Зміна поштової скриньки» | 2.4 |
| ВВ «Зміна пароля» | 2.5 |
| ВВ «Експертне підтвердження проектів» | 2.6 |
| ВВ «Створення проекту» | 2.7 |
| ВВ «Створення об’єктів нерухомості» | 2.8 |
| Тестування | 3 |
| ВВ «Реєстрація користувачів» | 3.1 |
| ВВ «Підтвердження реєстрації» | 3.2 |
| ВВ «Проходження паспортного контролю» | 3.3 |
| ВВ «Зміна поштової скриньки» | 3.4 |
| ВВ «Зміна пароля» | 3.5 |
| ВВ «Експертне підтвердження проектів» | 3.6 |
| ВВ «Створення проекту» | 3.7 |
| ВВ «Створення об’єктів нерухомості» | 3.8 |
| Інтеграційне тестування | 3.9 |

Для побудови плану проекту визначимо тривалості кінцевих задач та їх залежності одна від одної.

Тривалість всього проекту за UCP становить 750 годин. Спираючись на стандартний розподіл часу за WBS та власний досвід розподілимо доступні 750 годин між задачами (див. табл. 2.5).

Таблиця 2.5 – Визначення тривалості окремих задач та залежностей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WBS | Назва задачі | Розподіл | Час (год) | Попередники |
| 1 | Проектування | 36% | 270 | – |
| 1.1 | Архітектура | 10% | 75 | 1 |
| 1.2 | Структура та організація класів | 20% | 150 | 1.1 |
| 1.3 | Графічний інтерфейс | 6% | 45 | 1.2 |
| 2 | Реалізація | 54% | 405 | 1.3 |
| 2.1 | ВВ «Реєстрація користувачів» | 8% | 60 | 2 |
| 2.2 | ВВ «Підтвердження реєстрації» | 2% | 15 | 2.1 |
| 2.3 | ВВ «Проходження паспортного контролю» | 10% | 75 | 2.2 |
| 2.4 | ВВ «Зміна поштової скриньки» | 4% | 30 | 2.3 |
| 2.5 | ВВ «Зміна пароля» | 5% | 37,5 | 2.4 |
| 2.6 | ВВ «Експертне підтвердження проектів» | 12% | 90 | 2.5 |
| 2.7 | ВВ «Створення проекту» | 6% | 45 | 2.6 |
| 2.8 | ВВ «Створення об’єктів нерухомості» | 7% | 52,5 | 2.7 |
| 3 | Тестування | 10% | 75 | 2.8 |
| 3.1 | ВВ «Реєстрація користувачів» | 1% | 7,5 | 3 |
| 3.2 | ВВ «Підтвердження реєстрації» | 1% | 7,5 | 3.1 |
| 3.3 | ВВ «Проходження паспортного контролю» | 1% | 7,5 | 3.2 |

Продовження таблиці 2.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WBS | Назва задачі | Розподіл | Час (год) | Попередники |
| 3.4 | ВВ «Зміна поштової скриньки» | 1% | 7,5 | 3.3 |
| 3.5 | ВВ «Зміна пароля» | 1% | 7,5 | 3.4 |
| 3.6 | ВВ «Експертне підтвердження проектів» | 1% | 7,5 | 3.5 |
| 3.7 | ВВ «Створення проекту» | 1% | 7,5 | 3.6 |
| 3.8 | ВВ «Створення об’єктів нерухомості» | 1% | 7,5 | 3.7 |
| 3.9 | Інтеграційне тестування | 2% | 15 | 3.8 |

Проектування структури та організації класів може бути виконано тільки після проектування архітектури. Реалізація усіх функцій системи може початися після проектування. Тестування виконується над вже реалізованими функціями системи.

Для ілюстрації плану та графіка робіт з даного проекту використовуємо діаграму Ганта (див. рис. 2.1).

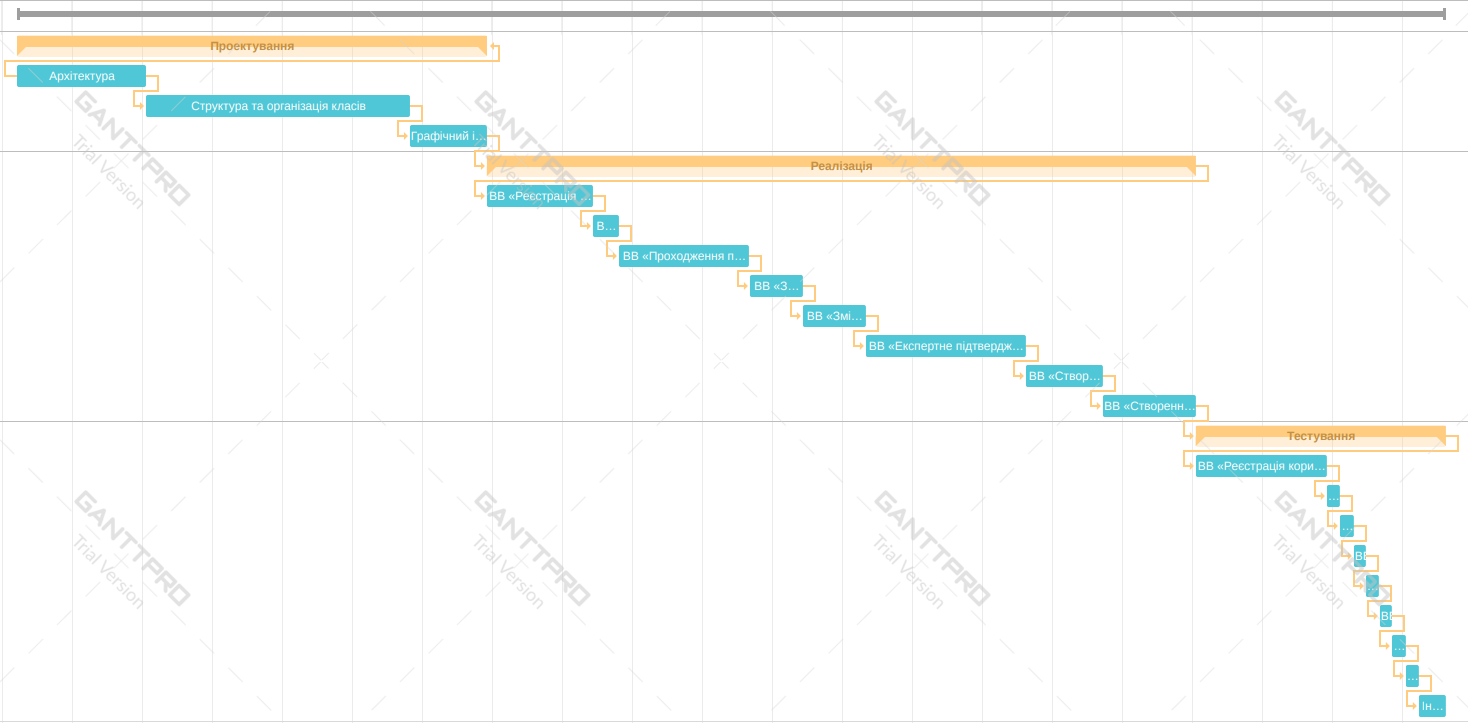


Рисунок 2.1 – Діаграма Ганта

Методами дослідження обрано порівнювальний аналіз та експеримент. Після виконаного плану розробки буде проведений експеримент. При експерименті буде проводитися замір часу на проведення ключових етапів (операцій) процесу купівлі-продажу нерухомості в реалізованій на підставі моделі програмної системи. В якості даних для порівняння при відповідному аналізі використовуватимуться витрати часу на пошук покупця, збір документів та їх перевірка сторонами угоди (підготовка угоди), оформлення угоди, реєстрацію угоди (перехід прав власності) та результуючого значення витраченого часу.

## **2.3 Висновки до розділу**

В даному розділі описано процес планування, під час якого визначено приблизну тривалість проекту методом UCP, а також створено план проекту та наданий опис процесу проведення експерименту, при якому вимірюватимуться часові показники результатів проведення ключових операцій купівлі-продажу в існуючих рішеннях та у реалізованій програмній системі. План проекту включає структурну декомпозицію робіт в проекті, розподіл часу та визначення залежностей між роботами. Був проведений аналіз ризиків в проекті. Ризики з найбільшим рангом були проаналізовані більш детально.

# **3 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ**

## **3.1 Проектування архітектури системи**

Предметна область – «Ринок купівлі-продажу нерухомості».

Так як програмне застосування представляє собою REST API застосування, то схема роботи застосування представлена у вигляді «запит – відповідь». При наявності запиту викликається відповідна функція-обробник, яка закріплена за обробку конкретного кінцевої точки API. Використовуємо концепцію MVC (Model-View-Controller) для поділу даних застосування, інтерфейсу користувача та керуючої логіки на 3 компоненти: модель, представлення та контролер [4].

В рамках розробки даної системи нижче буде наведена та розглянута концептуальна діаграма класів. Route – це клас, який визначає який контролер Users, Communities, Properties, Projects, PropertyContracts, ProjectContracts повинен бути викликаний, щоб виконати запит з API клієнта. В свою чергу User, Community, Property, Project, PropertyContract, ProjectContract – це моделі даних, до кожної з яких звертається відповідний контролер при обробці запиту і виконанні необхідної логіки.

Діаграма класів представлена на рис. 3.1.

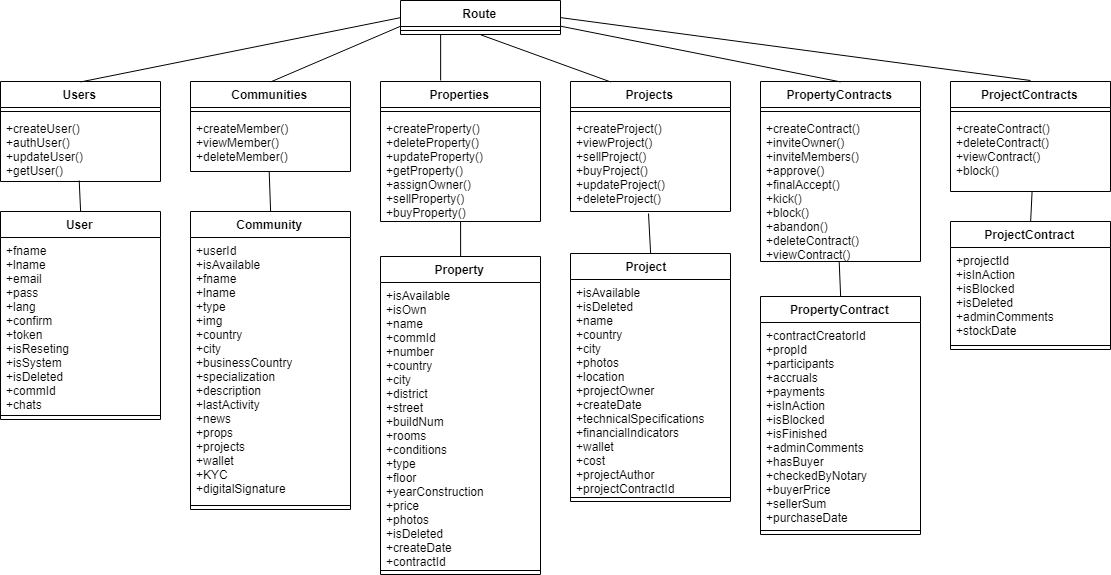


Рисунок 3.1 – Діаграмі класів проекту

Далі необхідно представити ключові функціональні процеси програмної системи у вигляді відповідних блок-схем алгоритму.

На рис. 3.2 представлена схема алгоритму «Реєстрація користувача».

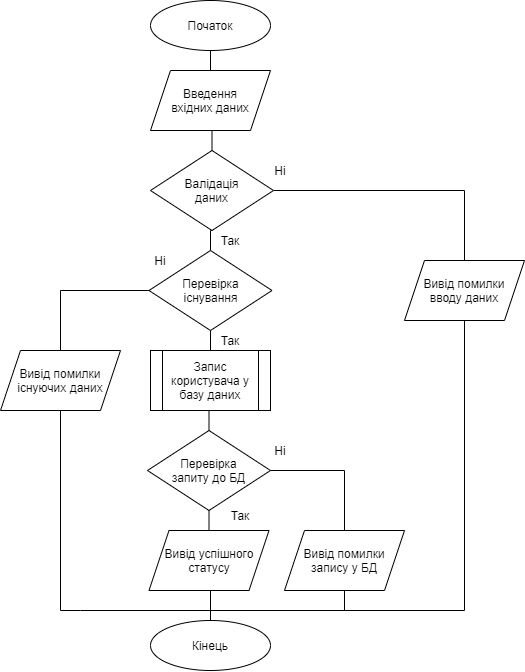


Рисунок 3.2 – Реєстрація користувача

На рис. 3.3 представлена схема алгоритму «Проходження паспортного контролю».

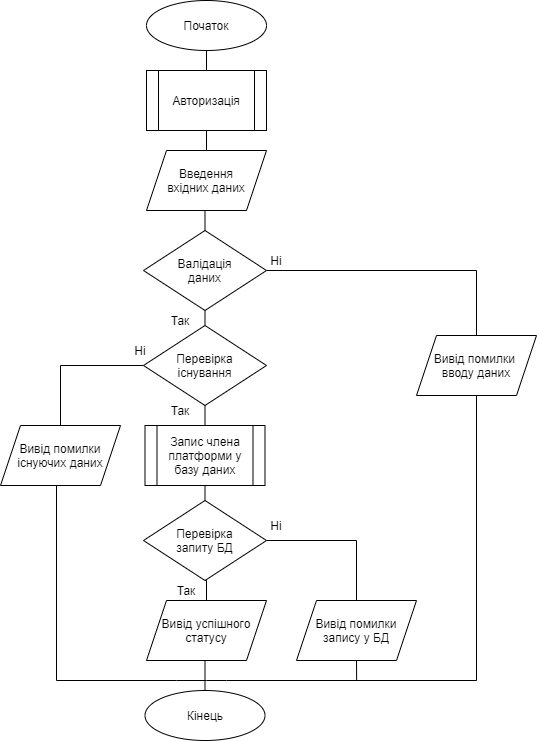


Рисунок 3.3 – Проходження паспортного контролю

На рис. 3.4 представлена схема алгоритму «Укладання угоди купівлі-продажу нерухомості»

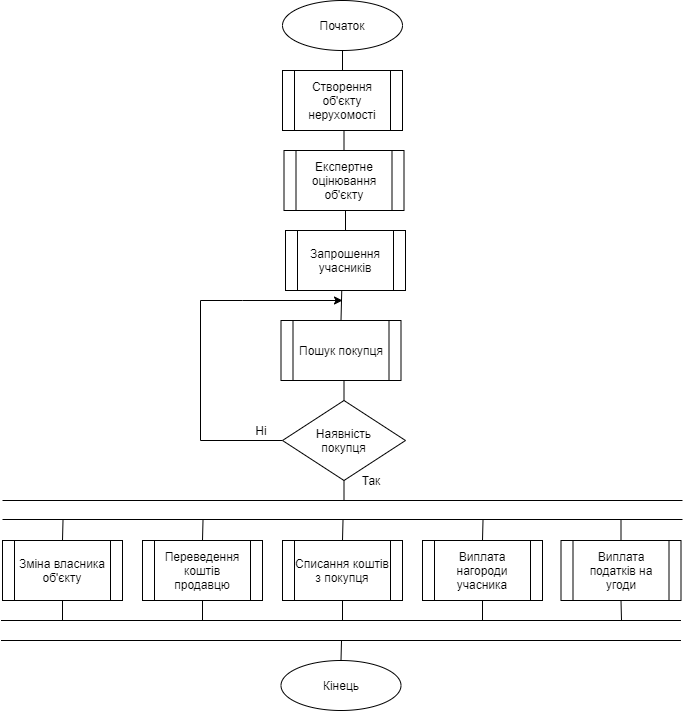


Рисунок 3.4 – Укладання угоди купівлі-продажу нерухомості

В таблиці. 3.1 описані методи класів проекту.

Таблиця 3.1 – Методи класів проекту

|  |  |
| --- | --- |
| Метод | Призначення |
| createUser() | Метод класу User, який створює нового користувача |
| authUser() | Метод класу User, який перевіряє дані користувача |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| Метод | Призначення |
| updateUser() | Метод класу User, який оновлює дані користувача |
| getUser() | Метод класу User, який повертає дані користувача |
| createMember() | Метод класу Community, який створює члена платформи |
| viewMember() | Метод класу Community, який повертає дані члена платформи |
| deleteMember() | Метод класу Community, який видаляє члена платформи |
| createProperty() | Метод класу Property, який створює об’єкт нерухомості |
| deleteProperty() | Метод класу Property, який видаляє об’єкт нерухомості |
| updateProperty() | Метод класу Property, який оновлює об’єкт нерухомості |
| getProperty() | Метод класу Property, який повертає дані про об’єкт нерухомості |
| assignOwner() | Метод класу Property, який визначає володаря об’єкту нерухомості |
| sellProperty() | Метод класу Property, який продає об’єкт нерухомості |
| buyProperty() | Метод класу Property, який купує об’єкт нерухомості |
| createProject() | Метод класу Project, який створює проект |
| viewProject() | Метод класу Project, який повертає дані про проект |
| sellProject() | Метод класу Project, який продає проект |
| buyProject() | Метод класу Project, який купує проект |
| updateProject() | Метод класу Project, який оновлює проект |
| deleteProject() | Метод класу Project, який видаляє проект |
| createContract() | Метод класу PropertyContract, який створює контракт |
| inviteOwner() | Метод класу PropertyContract, який запрошує володаря нерухомості |
| inviteMembers() | Метод класу PropertyContract, який запрошує проф. сторони |
| approve() | Метод класу PropertyContract, який підтверджує контракт |

Продовження таблиці 3.1

|  |  |  |
| --- | --- | --- |
| Метод | Призначення | |
| finalAccept() | Метод класу PropertyContract, який дає остаточне підтвердження | |
| kick() | | Метод класу PropertyContract, який видаляє члена контракту |
| block() | | Метод класу PropertyContract, який блокує контракт |
| abandon() | | Метод класу PropertyContract, який виводить з контракту |
| deleteContract() | | Метод класу PropertyContract, який видаляє контракт |
| viewContract() | | Метод класу PropertyContract, який повертає дані про контракт |
| createContract() | | Метод класу ProjectContract, який створює контракт |
| deleteContract() | | Метод класу ProjectContract, який видаляє контракт |
| viewContract() | | Метод класу ProjectContract, який повертає дані про контр. |
| block() | | Метод класу ProjectContract, який блокує контракт |

## **3.2 Проектування графічного інтерфейсу користувача**

При проектуванні графічного інтерфейсу користувача було звернено увагу на створення зрозумілого, зручного та звичного для користувача інтерфейсу. В якості інструменту для розробки інтерфейсу користувача використовуємо JavaScript-бібліотеку з відкритим вихідним кодом React.

Один з популярних підходів – це розміщення файлів в папках, згрупованих за функціональністю або маршрутом.

Проблем, пов'язаних з надмірною вкладеністю папок в React-проектах, може виникнути досить багато. Одна з них – це складність контролю щодо імпорту або поновлення цих імпортів при переміщенні файлів. В даному проекті немає вагомих підстав використовувати глибоку вкладеність папок, тому можемо обмежити себе максимум трьома рівнями вкладення в рамках проекту.

Реєстрація на платформі представлена на рис. 3.5.

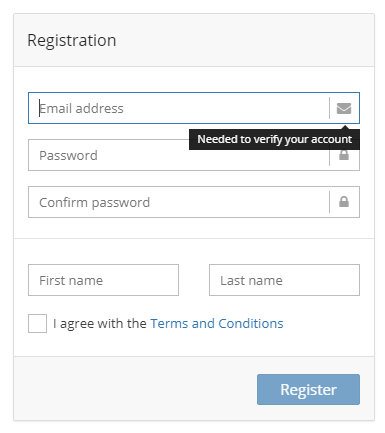


Рисунок 3.5 – Реєстрація на платформі

Додавання об’єкту нерухомості представлено на рис. 3.6.

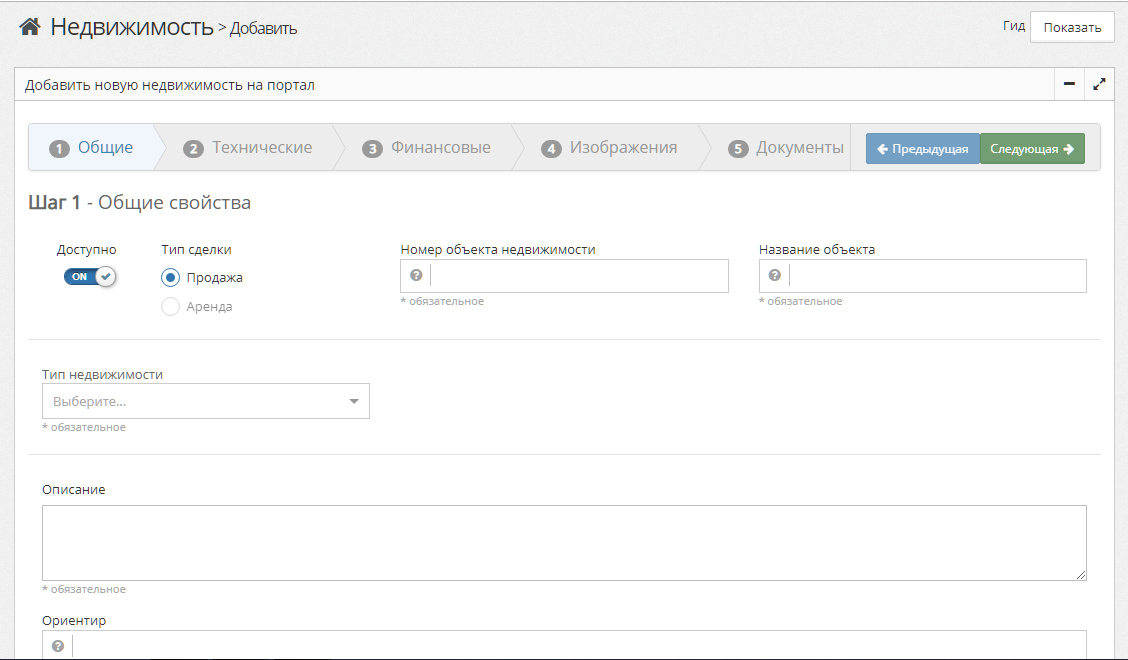


Рисунок 3.6 – Додавання об'єкту нерухомості

Лістинг об’єктів нерухомості представлений на рис. 3.7.

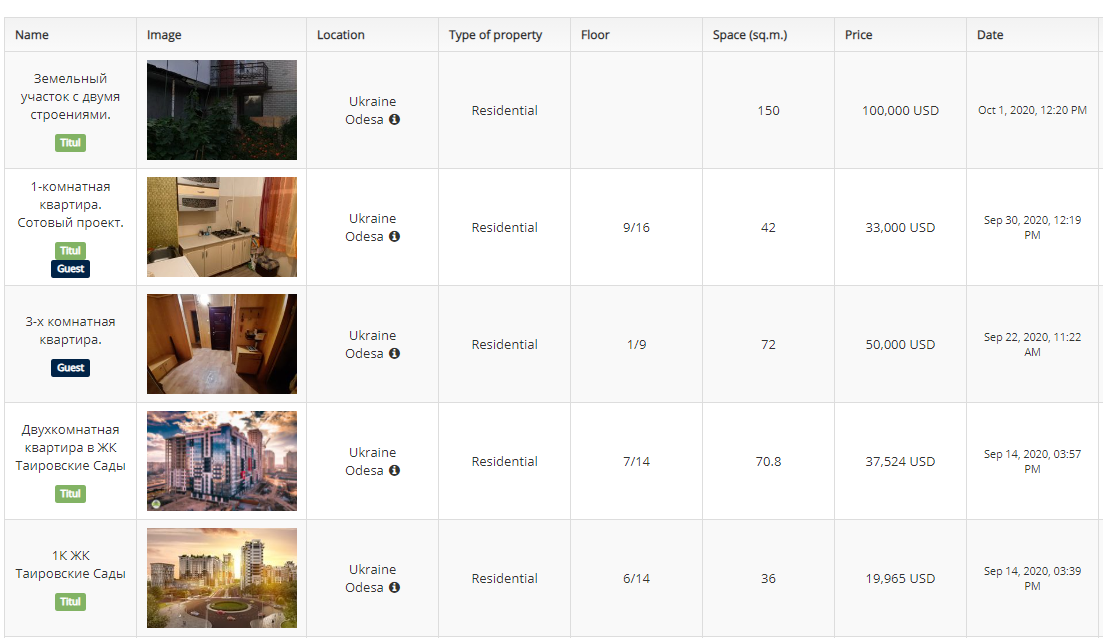


Рисунок 3.7 – Лістинг об'єктів нерухомості

Лістинг угод (контрактів), що укладаються, представлений на рис. 3.8.

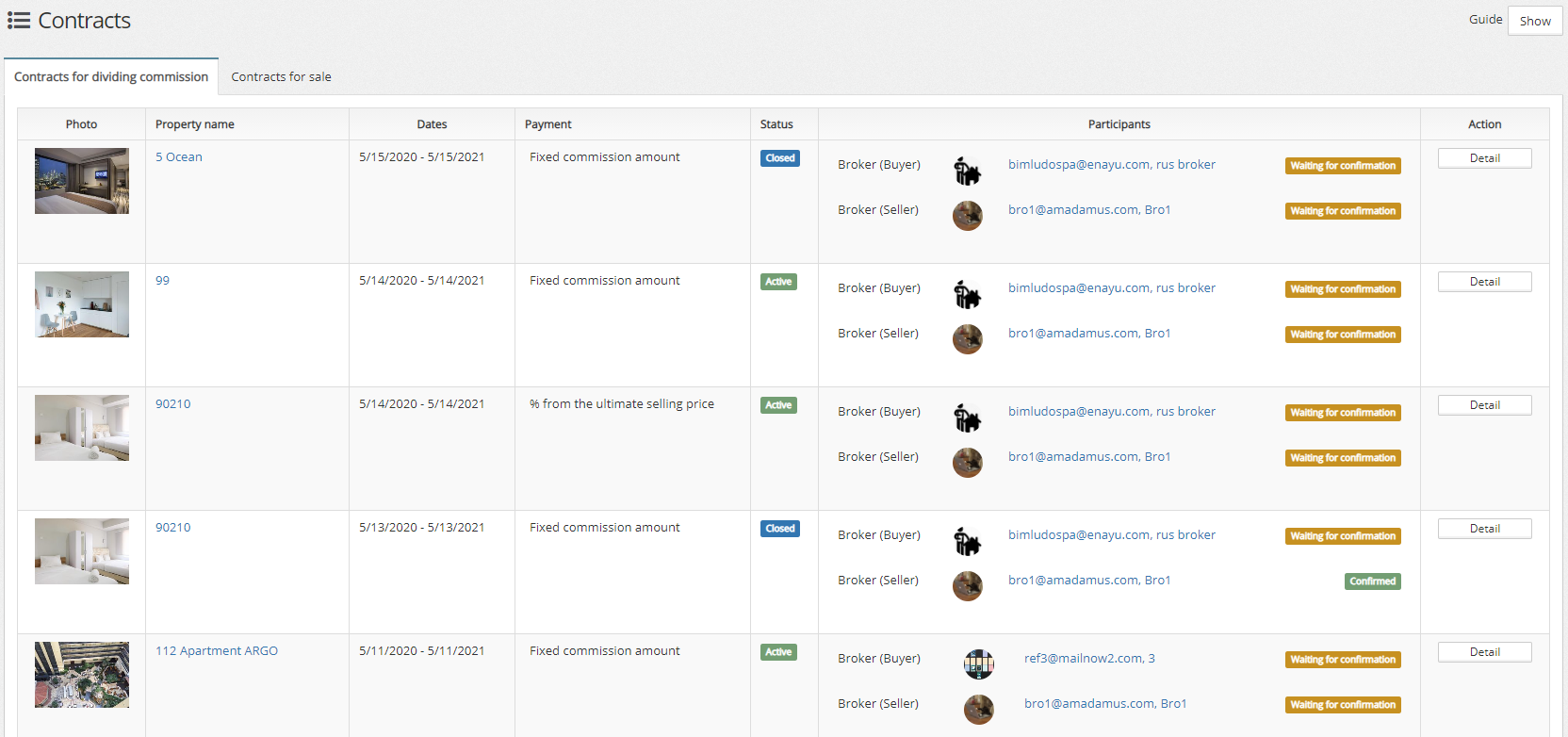


Рисунок 3.8 – Лістинг угод

Перегляд даних у базі даних представлений на рис. 3.9.

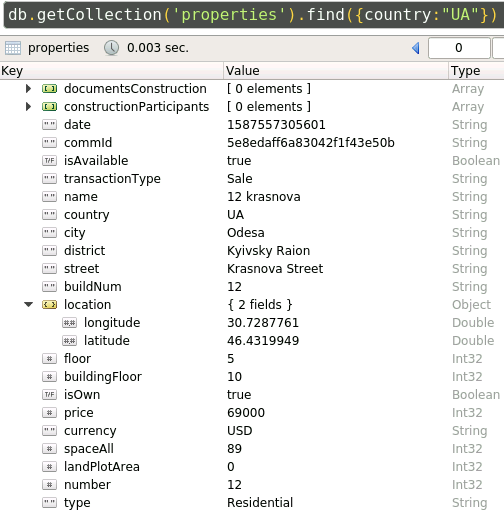


Рисунок 3.9 – Представлення даних у базі даних

## **3.3 Висновки до розділу**

В даному розділі виконано опис процесу проектування програмної системи за вимогами, що описані в першому розділі. Був розроблений архітектурний проект, а далі на його основі створено програмні класи системи. Описана структура вхідних та вихідних даних. Також надано опис проектування графічного інтерфейсу користувача та принципів, які використані при його створенні.

# **4 РОЗРОБКА СИСТЕМИ**

## **4.1 Опис програмних технологій**

Для реалізації програмної системи обрана мова програмування Javascript. Це динамічно типізована прототипна, скриптова мова програмування, що дозволяє без зайвих зусиль створювати сучасні програмні системи. Мова містить багато переваг, завдяки яким вона була обрана мною для реалізації програмної системи. Функціонал мови Javascript укладений у платформі Node.JS та фреймворку Express, які дозволяють побудувати REST API застосування [5].

В якості середовища розробки для написання коду на мові Javascript був використаний редактор Visual Studio Code. Це сучасний інструмент, що має багато зручних функцій, які полегшують процес написання коду, контролю його версій, тестування, та перевірку коректності.

В якості сховища даних є система управління базами даних MongoDB. Це документоорієнтована система управління базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць [6]. Класифікована як NoSQL, використовує JSON-подібні документи і схему бази даних. Написана на мові C++.

## **4.2 Опис програмних бібліотек**

В цьому проекті бібліотеки використовуються з глобального модулю NPM, такі як файлова система (fs), драйвер бази даних (mongoose), валідатори express-validator та ajv, захисник веб-застосувань helmet і cors, авторизаційна бібліотека jsonwebtoken, поштовий SMTP сервер nodemailer, модуль криптування crypto-js та інші. NPM – це менеджер пакетів, що входить у склад Node.js [7].

Сторонні бібліотеки не були використані при розробці даної програмної системи.

## **4.3 Структури даних**

Для виконання своїх функцій система працює з нереляційною базою даних MongoDB. Кожна колекція є файлом з розширенням «.bson», що містить дані цієї колекції. Кожний запит до колекції модифікує цей файл. MongoDB представляє документи JSON у двійково-кодованому форматі, який називається BSON за кадром. BSON розширює модель JSON для надання додаткових типів даних, впорядкованих полів, а також для ефективного кодування та декодування на різних мовах. Так як документи в MongoDB не мають чіткої схеми, тому структура документів в рамках однієї колекції нефіксована. Зокрема, колекція автоматично створюється тільки тоді, коли створюється перший документ. Але в рамках .js файлів створюються моделі документів.

Далі буде наданий список полів та методів, які існують в програмному застосуванні та в рамках документів колекцій у базі даних.

У таблиці 4.1 наведені атрибути таблиць (колекцій) проекту.

Таблиця 4.1 – Атрибути таблиць (колекцій) проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| User | \_id | ObjectId | Primary | Ідентифікатор користувача |
| User | fname | String |  | Ім'я користувача |
| User | lname | String |  | Прізвище користувача |
| User | email | String |  | Електрона пошта користувача |
| User | pass | String |  | Пароль користувача |
| User | lang | String |  | Мова користувача |
| User | confirm | Boolean |  | Прапор підтвердження реєстрації |
| User | token | String |  | Ключ доступу |
| User | isReseting | Boolean |  | Прапор зміни пароля |

Продовження таблиці 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| User | isSystem | Boolean |  | Прапор системного користувача |
| User | isDeleted | Boolean |  | Флаг видалення користувача |
| User | commId | ObjectId | Foreign | Ідентифікатор члена платформи |
| User | chats | Array |  | Масив чатів користувача |
| Community | \_id | ObjectId | Primary | Ідентифікатор члена платформи |
| Community | userId | ObjectId | Foreign | Ідентифікатор користувача |
| Community | isAvailable | Boolean |  | Прапор доступності члена платформи. |
| Community | fname | String |  | Ім'я члена платформи |
| Community | lname | String |  | Ім'я члена платформи |
| Community | type | Array |  | Фах члена платформи |
| Community | img | String |  | Фото члена платформи |
| Community | country | String |  | Країна члена платформи |
| Community | city | String |  | Місто члена платформи |

Продовження таблиці 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| Community | businessCountry | Array |  | Робочі країни члена платформи |
| Community | specialization | String |  | Спеціалізація члена платформи |
| Community | description | String |  | Опис члена платформи |
| Community | lastActivity | String |  | Остання активність члена платформи |
| Community | news | Array |  | Новини члена платформи |
| Community | props | Array |  | Об'єкти члена платформи |
| Community | projects | Array |  | Проекти члена платформи |
| Community | wallet | Object |  | Гаманець члена платформи |
| Community | KYC | Object |  | Паспортні дані члена платформи. |
| Community | digitalSignature | String |  | Цифровий підпис члена платформи |
| Property | \_id | ObjectId | Primary | Ідентифікатор об'єкту |
| Property | isAvailable | Boolean |  | Прапор доступності об'єкту |
| Property | isOwn | Boolean |  | Прапор володіння об'єктом |

Продовження таблиці 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| Property | name | String |  | Назва об'єкту |
| Property | commId | ObjectId | Foreign | Володар об'єкту |
| Property | number | Number |  | Номер об'єкту |
| Property | country | String |  | Країна об'єкту |
| Property | city | String |  | Місто об'єкту |
| Property | district | String |  | Район об'єкту |
| Property | street | String |  | Вулиця об'єкту |
| Property | buildNum | String |  | Номер дому об'єкту |
| Property | rooms | Number |  | Кількість кімнат об'єкту |
| Property | conditions | String |  | Умови об'єкту |
| Property | type | String |  | Тип об'єкту |
| Property | floor | Number |  | Поверх об'єкту |
| Property | yearConstruction | Number |  | Рік побудови об'єкту |

Продовження таблиці 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| Property | price | Number |  | Ціна об'єкту |
| Property | photos | Array |  | Фотографії об'єкту |
| Property | isDeleted | Boolean |  | Прапор видалення об'єкту |
| Property | createDate | String |  | Дата створення об'єкту |
| Property | contractId | ObjectId | Foreign | Ідентифікатор контракту |
| Project | \_id | ObjectId | Primary | Ідентифікатор проекту |
| Project | isAvailable | Boolean |  | Прапор доступності |
| Project | isDeleted | Boolean |  | Прапор видалення |
| Project | name | String |  | Назва проекту |
| Project | country | String |  | Країна проекту |
| Project | city | String |  | Місто проекту |
| Project | photos | Array |  | Фотографії проекту |
| Project | location | Object |  | Розташування |

Продовження таблиці 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| Project | projectOwner | ObjectId | Foreign | Володар проекту |
| Project | createDate | String |  | Дата створення |
| Project | technicalSpecifications | Object |  | Технічні специфікації |
| Project | financialIndicators | Object |  | Фінансові індикатори |
| Project | wallet | Object |  | Гаманець проект. |
| Project | cost | Number |  | Ціна проекту |
| Project | projectAuthor | ObjectId | Foreign | Автор проекту |
| Project | projectContractId | ObjectId | Foreign | Ідентифікатор контракту |
| PropertyContract | \_id | ObjectId | Primary | Ідентифікатор контракту |
| PropertyContract | contractCreatorId | ObjectId | Foreign | Творець контракту |
| PropertyContract | propId | ObjectId | Foreign | Ідентифікатор контракту |
| PropertyContract | participants | Array |  | Учасники контракту |
| PropertyContract | accruals | Array |  | Податки учасників |

Продовження таблиці 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| PropertyContract | payments | Array |  | Нагороди учасників |
| PropertyContract | isInAction | Boolean |  | Прапор дії контракту |
| PropertyContract | isBlocked | Boolean |  | Прапор блокування контракту |
| PropertyContract | isFinished | Boolean |  | Прапор завершення контракту |
| PropertyContract | adminComments | Array |  | Коментарі адміністрації |
| PropertyContract | hasBuyer | Boolean |  | Прапор покупця |
| PropertyContract | checkedByNotary | Boolean |  | Прапор перевірки |
| PropertyContract | buyerPrice | Number |  | Ціна (для покупця) |
| PropertyContract | sellerSum | Number |  | Сума (продавця) |
| PropertyContract | purchaseDate | String |  | Дата покупки |
| ProjectContract | \_id | ObjectId | Primary | Ідентифікатор контракту |
| ProjectContract | isInAction | Boolean |  | Прапор дії контракту |
| ProjectContract | isBlocked | Boolean |  | Прапор блокування |

Продовження таблиці 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Колекція | Атрибути | Тип | Ключ | Призначення |
| ProjectContract | isDeleted | Boolean |  | Прапор видалення |
| ProjectContract | adminComments | Array |  | Коментарі адміністрації |
| ProjectContract | stockDate | String |  | Дата підписання контракту |

Схема бази даних представлена на рис. 4.1.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4.1 – Схема бази даних

## **4.4 Інструкція з встановлення програмного продукту**

Для встановлення програмного продукту (серверної частини) необхідно мати його вихідний код та встановлену систему контроля версій git. Необхідно виконати команду git clone для клонування репозиторію на локальний комп’ютер або отримати готовий архів та розархівувати його. Необхідний розмір вільної пам’яті на ПК: 200 МБ. Додаткових дій по налаштуванню виконувати непотрібно. Серверна частина повинна бути розміщена на електронно-обчислювальній машині з можливістю виходу до мережі Інтернет.

## **4.5 Інструкція з використання**

Для встановлення програмного продукту необхідно мати встановлену на ПК платформу Node.js (10 версія і більше), модуль NPM (5.0 версія і більше) та СУБД MongoDB (4.0 версія та більше) [8].

База даних запускається командою mongod –dbpath=«шлях до місця розташування БД». За замовчування база даних прослуховується на порту 27017.

Серверна частина запускається командою node start у корньовій папці програмного продукта: HTTP сервер – порт 3000, HTTPS – порт 8000. Веб-сервер отримує команди API методів у вигляді запитів: http(s)://ip:port/«кінцева точка API»

З точки зору кінцевого користувача, для використання продукту можна використовувати веб-сайт, який наразі доступний за посиланням https://grem.capital (гостьова сторінка), https://cabinet.grem.capital (особистий кабінет).

## **4.6 Висновки до розділу**

В даному розділі описані програмні технології, що були використані при реалізації системи. Також описані використані бібліотеки.

Описано реалізацію функцій порівняння та об’єднання двох версій.

Створена інструкція з встановлення створеної програмної системи, а також інструкція з використання. Керуючись створеною інструкцією з встановлення система була встановлена на комп’ютер користувача. Також була перевірена її функціональність за інструкцією з використання.

# **5 ТЕСТУВАННЯ ТА ВИПРОБУВАННЯ ПРОГРАМНОЇ СИСТЕМИ**

## **5.1 Модульне тестування**

Виконаємо модульне тестування функцій системи «Видалення об’єкту нерухомості» та «Оновлення даних про користувача». Проведемо тестування методом чорного ящику. Це дозволить перевірити на коректність окремі модулі вихідного коду програми з відповідними керуючими даними.

В таблиці 5.1 описані класи еквівалентності, що використані при тестуванні функції реєстрації.

Таблиця 5.1 – Тестування видалення об’єкту нерухомості

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип | Опис параметрів | Очікуваний результат | Фактичний результат |
| 1 | вірний | Список с 6 власних об'єктів. Видалений 1 об'єкт. | 5 власних об'єктів | 5 власних об'єктів |
| 2 | вірний | Список с 2 власних об'єктів. Видалені 2 об'єкти. | 0 власних об'єктів | 0 власних об'єктів |
| 3 | вірний | Список с 5 власних об'єктів. Видалені 4 об'єктів. | 1 власний об'єкт | 1 власний об'єкт |
| 4 | невірний | Список с 1 власного об'єкту. Видалені 2 об'єкти. | Помилка вхідних даних | Помилка вхідних даних |
| 5 | вірний | Список с 6 власних об'єктів. Видалені 8 об'єктів. | Помилка вхідних даних | Помилка вхідних даних |

В таблиці 5.2 описані класи еквівалентності, що використані при тестуванні функції оновлення даних про користувача.

Таблиця 5.2 – Тестування оновлення даних про користувача

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип | Опис параметрів | Очікуваний результат | Фактичний результат |
| 1 | невірний | Користувач вводить букви в поле редагування дати народження | Помилка про невірні дані | Помилка про невірні дані |
| 2 | невірний | Користувач вводить число, менше за 1900 в дату народження | Помилка про невірний рік народження | Помилка про невірний рік народження |
| 3 | невірний | Користувач вводить число, більше за 2002 в дату народження | Помилка про невірний рік народження | Помилка про невірний рік народження |
| 4 | невірний | Користувач вводить цифри та букви в поле року народження | Помилка про невірний рік народження | Помилка про невірний рік народження |
| 5 | вірний | Користувач вводить число у діапазоні 1900-2002 в поле року народження | Повідомлення про успішну зміну року | Повідомлення про успішну зміну року |

## **5.2. Функціональне тестування**

Виконаємо функціональне тестування системи, яке необхідне для підтвердження того, що написана програма відповідає заявленим на початку проекту вимогам, тобто виконаємо її верифікацію. Це дозволить перевірити реалізацію функціональних вимог, тобто здатність ПЗ в певних умовах вирішувати завдання, потрібні користувачам. Функціональні вимоги визначають, що саме робить програмне забезпечення, які завдання воно вирішує [9].

В таблиці 5.3 описано тестування прецеденту «Реєстрація користувачів».

Таблиця 5.3 – Тестові варіанти прецеденту «Реєстрація користувачів»

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |
| Вибір в меню «Реєстрація» | Відображається вкладка, на якій зображені елементи, необхідні для реєстрації | Успішно |
| Введення коректних даних | Система починає завантаження даних | Успішно |
| Введення некоректних даних | Система відображає діалогове вікно з повідомленням про помилку | Успішно |
| Система завантажила дані | Система відображає діалогове вікно з повідомленням про успішне збереження даних до бази даних | Успішно |

В таблиці 5.4 описано тестування прецеденту «Підтвердження реєстрації».

Таблиця 5.4 – Тестові варіанти прецеденту «Підтвердження реєстрації»

|  |  |  |
| --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |
| Вибір в меню «Підтвердження» | Система відображає вікно з елементами, необхідними для підтвердження | Успішно |
| Введення коректних даних | Система починає завантаження | Успішно |
| Введення некоректних даних | Система відображає вікно з повідомленням про помилку | Успішно |
| Система завантажила дані | Система відображає вікно з повідомленням про збереження даних у базі даних | Успішно |

В таблиці 5.5 описано тестування прецеденту «Проходження паспортного контролю».

Таблиця 5.5 – Тестові варіанти прецеденту «Проходження паспортного контролю»

|  |  |  |  |
| --- | --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |  |
| Вибір в меню «Паспортний контроль» | Система відображає вікно з елементами, необхідними для паспортного контролю | Успішно |  |
| Введення коректних даних | Система починає завантаження даних | Успішно |  |
| Введення некоректних даних | Система відображає вікно з повідомленням про помилку | Успішно |  |
| Система завантажила дані | Система відображає вікно з повідомленням про збереження даних у базі даних | Успішно |  |

В таблиці 5.6 описано тестування прецеденту «Підтвердження члена платформи».

Таблиця 5.6 – Тестові варіанти прецеденту «Підтвердження члена платформи»

|  |  |  |  |
| --- | --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |  |
| Вибір в меню «Підтвердження» | Система відображає вікно з елементами, необхідними для підтвердження | Успішно |  |
| Введення коректного логіна | Система починає завантаження даних | Успішно |  |
| Введення некоректного логіна | Система відображає вікно з повідомленням про помилку | Успішно |  |

Продовження таблиці 5.6

|  |  |  |  |
| --- | --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |  |
| Система завантажила дані | Система відображає вікно з повідомленням про збереження даних у базі даних | Успішно |  |

В таблиці 5.7 описано тестування прецеденту «Експертне підтвердження».

Таблиця 5.7 – Тестові варіанти прецеденту «Експертне підтвердження»

|  |  |  |  |
| --- | --- | --- | --- |
| Дія | Очікуваний результат | Результат тестування |  |
| Вибір в меню «Підтвердження» | Система відображає вкладку, на якій можна підтвердити проект | Успішно |  |
| Натискання на кнопку «Підтвердження» з базою даних | Система отримує дані та сповіщає про отримання даних | Успішно |  |
| Натискання на кнопку «Підтвердження» без бази даних | Система відображає вікно з повідомленням про помилку | Успішно |  |
| Система завантажила дані | Система відображає вікно з повідомленням про збереження даних у базі даних | Успішно |  |

## **5.3 Висновки до розділу**

В даному розділі описано процес тестування реалізованої системи.

Було проведено функціональне тестування, яке показало, що система реалізована у відповідності до вимог користувача. Були використані різні набори даних для перевірки, що система успішно оброблює запити при коректних вхідних даних; та успішно видає помилку при введенні некоректних даних, або при наявності додаткових обмежень, імплементованих за спроектованою архітектурою програмної системи.

# **6 ВИЗНАЧЕННЯ ВЛАСТИВОСТЕЙ ПРОГРАМНОЇ СИСТЕМИ**

## **6.1 Експериментальне оцінювання результатів укладання угоди**

Згідно реальній практиці ринку [10], час на продаж квартири становить від 6 до 120 днів, а кількість укладених угод збільшується останні 6 років (див. рис. 6.1).

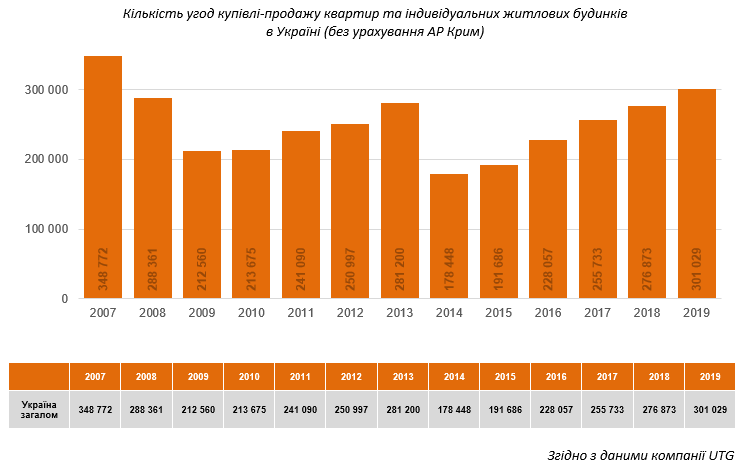


Рисунок 6.1 – Статистика по кількості угод купівлі-продажу нерухомості в Україні за 2007-2019 роки

Серед множини припущень щодо програмної системи необхідно виділити такі, які є критично важливими і мають вирішальне значення. Це означає, що їх спростування унеможливить використання програмної системи.

Сформуємо 3 гіпотези:

1. Користувачі додаватимуть об'єкти нерухомості до програмної системи;
2. Користувачі регулярно використовуватимуть програмну систему;
3. Користувачі будуть задоволені результатами експертної оцінки їх нерухомості фахівцями.

Детальніше опишемо кожну з гіпотез:

1. Потік нерухомості. Перевірка гіпотези здійснюється заповненням клієнтами змодельованої форми додавання об'єкта нерухомості, яка пропонує на підставі введених даних отримати приблизний час обробки заявки і орієнтовну вартість об'єкта нерухомості. Вимірюється кількість користувачів, які заповнили форму. Гіпотеза підтверджується, якщо не менше 65% користувачів заповнили форму. Гіпотеза – найвищої критичності, витрати на тестування – малі, надійність даних – висока, що витрачений час на вимір – середній;
2. Зацікавленість користувача в продукті. Перевірка гіпотези здійснюється тестуванням використання клієнта мінімально життєздатного продукту в повсякденному житті. Вимірюється кількість відвідувань сайту користувачем за день і проведений на сайті час. Гіпотеза підтверджується, якщо середня кількість відвідувань сайту на одного користувача не менше 2-ох разів або середній час, виділений користувачем на продукт, становить не менше 5% його добової активності в мережі. Гіпотеза – середньої критичності, витрати на тестування – середні, надійність даних – висока, витрачений час на вимір – середній;
3. Прийняття користувачами ціни. Перевірка гіпотези здійснюється запуском пілотного проекту з 20-ма користувачами і подальшим оцінюванням їх об'єктів нерухомості фахівцями платформи. Вимірюється кількість учасників, які згодні з наданою оцінювачами ціною об'єкта нерухомості. Гіпотеза підтверджується, якщо не менше 80% учасників задоволені отриманою ціною.

Ключові, критичні гіпотези перевірені.

Вимірюваною величиною є фактично витрачений час на укладання угоди купівлі-продажу нерухомості, тобто проходження усіх процедур від оцифрування об'єкту нерухомості до переходу прав власності на об'єкт нерухомості від продавця до покупця.

В результаті вимірів часу на проведення ключових етапів (операцій) в реалізованій на підставі моделі програмній системі отримані наступні дані (див. табл. 6.1).

Таблиця 6.1 – Порівняння отриманих результатів

|  |  |  |
| --- | --- | --- |
| Порівнюваний показник | Отримане значення (дні) | Реальне значення (дні) |
| Пошук покупця | 3-60 | 1-90. |
| Збір документів і їх перевірка сторонами угоди (підготовка угоди) | 2-5 | 3-10 |
| Оформлення угоди | 1 | 1 |
| Реєстрація угоди (перехід прав власності) | 3 | 3 |
| Разом (в середньому) | 40 | 57 |

Після зіставлення реальних та отриманих значень ключових показників бачимо середнє прискорення укладання угоди до 30%. Таким чином мета, поставлена в рамках цієї роботи, була досягнута.

## **6.2 Висновки до розділу**

В даному розділі було проведене експериментальне оцінювання результатів програмної системи, зокрема результатів укладання угоди, а також перевірка критичних гіпотез щодо програмної системи. Був отриманий результат, який свідчить про те, що мета розробки програмної системи на основі запропонованої моделі угоди, що є цифровим контрактом, в якому автоматизовані етапи підбору фахівців, експертного оцінювання об'єкта нерухомості, обчислення та розподілу часток фахівців, досягнута.

# **7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ**

Закон України про охорону праці визначає основні положення щодо реалізації конституційного права громадян на охорону їх життя і здоров'я в процесі трудової діяльності, регулює за участю відповідних державних органів відносини між власником підприємства, установи і організації або уповноваженим ним органом і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні [11].

Дипломна робота розглядає питання автоматизації реального процесу укладання угоди купівлі-продажу нерухомості. Їх виконання здійснюється працівниками галузі інженерії програмного забезпечення. В результаті регулярної роботи з комп'ютером, працівники IT-відділу наражаються на ряд небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики і ін.

Охорона праці – це система збереження життя і здоров'я працівників в процесі трудової діяльності, що включає в себе правові, соціально-економічні, організаційно-технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші заходи.

За рахунок наступних нормативних документів, затверджених спеціальним уповноваженим центральним органом виконавчої влади у галузі охорони здоров'я, забезпечується поліпшення безпеки, гігієни праці та виробничого середовища:

* ДСТУ 2293-99. Охорона праці. Терміни та визначення;
* НАПБ А.01.001-2015. Правила пожежної безпеки в Україні;
* ДСН 3.3.6.037-99. Санітарні норми виробничого шуму, ультразвуку та інфразвуку;
* НПАОП 0.00-1.28-10. Правила охорони праці при експлуатації ЕОМ;
* ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень;
* НАПБ Б.03.001-2204. Типові норми належності вогнегасників;
* ДБН В.2.5-28-2006. Природне і штучне освітлення.

Вивчення найважливіших заходів з охорони праці та безпеки у надзвичайних ситуаціях для працівників галузі інженерія програмного забезпечення наведено у Додатку Б та висвітлює наступні питання:

* організація та управління охороною праці на фірмі ТОВ «ГРУПА КОМПАНІЙ «ТИТУЛ»;
* вказівка заходів з покращення умов охорони праці;
* обчислення захисного заземлення;
* надзвичайні ситуації та шляхи їх запобігання.

Під час виконання роботи було проаналізовано робоче місце інженера-програміста в підрозділі IT-відділу. Були розглянуті питанні, що стосуються охорони праці на фірмі. Було розглянуто систему управління охороною праці, службу охорони праці, інструктажі, медогляд та професіональний відбір, які я пройшов під час працевлаштування, а також процес розслідування нещасних випадків. Були проаналізовані такі фактори, як мікрокліматичні умови праці, освітлення, іонізуюче випромінення, повітря робочої зони, пожежна безпека, електробезпека.

Також в рамках роботи було розраховано опір для системи заземлення на робочому місці у будівлі, а також визначена категорія пожежонебезпечного офісного приміщення та визначена тривалість перебування людей на радіоактивно зараженій місцевості при встановленій дозі опромінення.

В результаті розрахунків були запропоновані заходи для покращення продуктивності праці та збільшення ефективності працівників приміщення. В якості рекомендацій щодо поліпшення умов праці рекомендується забезпечити ефективне захисне заземлення електроприладів та покращити якість моніторів.

# **ВИСНОВКИ**

Технічна частина виконана в повному обсязі. Як результат автоматизації процес укладання угод з купівлі-продажу нерухомості був прискорений до 30%.

В даній роботі описані усі етапи процесу розробки програмної системи для забезпечення купівлі-продажу нерухомості.

В першому розділі була описана предметна область, виявлені проблеми, наявні в процесах предметної області, а також визначена мета проекту. Були визначені функціональні та нефункціональні вимоги до системи.

В другому розділі продемонстровано процес планування розробки, під час якого визначено приблизну тривалість проекту та створено план проекту. Також проаналізовано ризики, що присутні в проекті.

В третьому розділі виконано проектування програмної системи за вимогами. Спочатку була спроектована архітектура системи, потім на її основі створено програмні класи. Описана структура вхідних та вихідних даних. Також надано опис проектування графічного інтерфейсу користувача та принципів, які використані при його створені.

В четвертому розділі були описані програмні технології, що були застосовані при реалізації системи та використані бібліотеки.

В п’ятому розділі був процес тестування реалізованої системи. Було функціональне тестування, яке показало, що система реалізована у відповідності до вимог користувача.

В шостому розділі наданий опис властивостей програмної системи та результати експериментального оцінювання.

В сьомому розділі була виконана робота з питань охорони праці, зокрема виконане індивідуальне завдання задля забезпечення безпеки роботи замовника на його робочому місці під час роботи з даним програмним продуктом.

Мета, яка була поставлена на початковому етапі проектування, була досягнута. Всі варіанти використання були реалізовані.

# **ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Ukrainian Trade Guild: [Електронний ресурс]: Купівля житлової нерухомості в Україні за рік зросла на 8,7% - інфографіка – <https://utgcompany.com/stati_smi/kupivlya-zhytlovoyi-nerukhomosti-v-ukrayini-za-rik-zrosla-na-8-7-infografika/> (дата звернення 21.11.2020)

2. Королева А.М. Роль ринку нерухомості в економіці держави // Суспільство: політика, економіка, право – 2016. – №6 – С.71-73

3. Investopedia [Електронний ресурс]: Smart Contracts Definition – URL: <https://www.investopedia.com/terms/s/smart-contracts.asp> (дата звернення 20.11.2020)

4. Басс, Л. Архітектура програмного забезпечення на практиці [Текст] / Л. Басс, П. Клементс, Р. Кацман. – СПб.: Питер, 2006. – 576 с.

5. Ітан Браун. Веб-розробка із застосуванням Node і Express. Повноцінне використання стека JavaScript. // Санкт-Петербург: Питер, 2017. 336 с. ISBN 978-1-491-94930-6.

6. Sarah Mei Why You Should Never Use MongoDB [Електронний ресурс]: 2013

7. Херрон Д. «Node.js. Розробка серверних веб-додатків в JavaScript»: Пер. с англ. Слинкина А.А. - М.: ДМК Пресс, 2012. – 144 с

8. Kyle Simpson «You don’t know JS» [Серія книг] – 2014

9. Бейзер, Б. Тестування чорного ящика. Технології функціонального тестування програмного забезпечення і систем [Текст] / Б. Бейзер. – СПб.: Питер, 2004. – 320 с.

10. Секрети Ріелтора: [Електронний ресурс]: Скільки часу займає продаж квартири – URL: <https://kvartira-bez-agenta.ru/melochi/skolko-vremeni-zanimaet-prodazha-kvartiry> (дата звернення 21.11.2020)

11. Методичні вказівки для оформлення розділу «Охорона праці та безпека у надзвичайних ситуаціях" в Дипломному проекті» студентами інституту "Комп’ютерних систем» / Укладачі: А.Ю. Москалюк, С.Н. Бабюк,. В. М. Пуріч.- Одеса, ОНПУ.2018. - 21 с.