НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Автоматизована система торгівлі нерухомістю та землею на базі RESO»

Виконав (-ла):
студент (-ка) IV курсу, групи ІТ-61
Савін Микола Олександрович _____

Керівник:
Асистент кафедри АУТС
Шинкевич Микола Костянтинович _____

Рецензент:
Доцент кафедри ОТ, кандидат технічних наук
Верба Олександр Андрійович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Савіну Миколі Олександровичу

1. Тема проєкту «Автоматизована система торгівлі нерухомістю та землею на базі RESO», керівник проєкту асистент Шинкевич Микола Констянтинович, затверджені наказом по університету від 07 травня 2020 р. №1081-с

2. Термін подання студентом проєкту 09.06.2020

3. Вихідні дані до проєкту

Мова програмування JavaScript, бібліотеки JavaScript ReactJs, NextJs, ExpressJs, Socket.IO, середовище програмування WebStorm, СУБД – PostgresSQL, платформа для реалізації – Google Cloud Platform.

4. Зміст пояснювальної записки

1. Вступ 2. Аналіз предметної області 3. Аналіз існуючих рішень 4. Аналіз вимог до програмного забезпечення 5. Аналіз сценаріїв використання системи 6. Вибір технологій розробки 7. Розробка системи 8. Тестування програмного забезпечення 9. Впровадження та використання розробленої системи

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

Діаграма варіантів використання, діаграма компонентів, діаграма модулів, діаграма структури бази даних

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вибір тематичного напрямку та узгодження теми дипломного проекту	22.02.2020	
2	Аналіз теоретичних матеріалів та вивчення предметної області	15.04.2020	
3	Розробка технічного завдання, вибір методів та засобів реалізації задачі	24.04.2020	
4	Огляд існуючих рішень з тематики роботи	27.04.2020	
5	Розробка структури прототипу та проектування системи	06.05.2020	
6	Реалізація проекту	20.05.2020	
7	Налагодження та перевірка програми	23.05.2020	
8	Оформлення пояснювальної записки	03.06.2020	
9	Передзахист дипломного проекту	04.06.2020	

Студент Микола САВІН

Керівник Микола ШИНКЕВИЧ

АНОТАЦІЯ

Савін М.О. Автоматизована система торгівлі нерухомістю та землею на базі RESO. КПІ ім. Ігоря Сікорського, Київ, 2020.

Ключові слова: Javascript, діаграма варіантів використання, діаграма пакетів, бібліотека ReactJs, бібліотека NextJs, бібліотека ExpressJs, стандарт RESO.

Основна частина документу викладена у пояснювальній записці, виконаній на 69 сторінках, та містить 3 рисунків та 46 таблиці.

Об’єктом розробки є система торгівлі нерухомістю та землею.

Мета розробки – створення системи торгівлі нерухомістю та землею з використанням стандарту RESO.

У дипломному проекті проведено аналіз існуючих рішень та на базі результатів розроблено систему торгівлі землею, дані якої відповідають стандарту RESO. Система є реалізованою на Google Cloud Platform, та має велику кількість тестів, які покривають основні сценарії її використання.

Отримані результати можуть бути корисними при створенні аналогічних чи подібних систем.

SUMMARY

Savin M. O. Automated trading system for non-land and land based on RESO. Igor Sikorsky KPI, Kyiv, 2020.

Keywords: Javascript, use-case diagram, packet diagram, library ReactJs, library NextJs, library ExpressJs, standard RESO.

The bulk of the document is outlined in the explanatory note, with 69 pages, and contains 3 figures and 46 tables.

The object of development is trading system for non-land and land.

The purpose of the development – creation of trading system for non-land and land based on RESO standard.

The graduation project has a thorough analysis of the existing decisions and based on these results developed system for land trading, with data using RESO standard. The system is deployed on the Google Cloud Platform, and has a large number of tests that cover the main scenarios of its use.

The obtained results can be useful in creation of analogic or similar systems.

Пояснювальна записка
до дипломного проєкту
на тему: «Автоматизована система торгівлі нерухомістю та землею на базі RESO»

Київ – 2020 року

ЗМІСТ

ВСТУП

Головною метою створення нового програмного забезпечення являється автоматизація процесів, які керуються людьми. Розвиток інформаційних технологій змінив світ, дав новий напрямок як і розвитку бізнесу, так і соціальним відношенням між людьми. Ще зовсім недавно облік більшості документів вівся у паперовому вигляді, а зараз ми вже отримуємо ідентифікаційну карту кожного громадянина у смартфоні, дані про якого зберігаються на добре захищених серверах. Власники, які змогли перевести власний бізнес, зараз відомі на весь світ, а інші так і залишились в минулому столітті.

Торгівля нерухомістю та землею не стала виключенням. Цей ринок існує ще з часів появи самого суспільства та відношень між людьми і завжди був найбільш стабільним та пропорційним до рівня життя людей у країні. Раніше для пошуку нових об’яв про нерухомість використовувались газети – але скільки пропозицій там могло вміститись – 5, 10?

Регулювання відношення між покупцем та продавцем нерухомістю стало основною задачею ринку землі у світі програмного забезпечення. Які дані потрібні покуплю для вибору тієї чи іншої будівлі, а які дані продавець може дати? З’явилось багато сервісів, які пропонують власне рішення даної проблеми, а тобто зберігають дані у форматі, який вважають найкращим. Тоді у відношенні між продавцем і покупцем появилась нова проблема – розбіжність та неточність знань про об’єкт нерухомості. Відвідуючи декілька сервісів, покупець може бачити багато інформації про будівлю, але вся вона може бути різною через проблему форму даних у цій предметній області.

Завданням цієї роботи є створення сервісу, який буде використовувати прийнятий світом формат даних про нерухомість. Так як цей ринок являється локальним, моє рішення буде обмежено кордонами України, але легко зможе бути розширене. У світі останніх подій та змін у законі України, централізація

усіх даних про землю та нерухомість до формату даних, який використовується у всьому світі, може бути корисним не тільки для українців, а й для іноземних громадян. Ще одним завданням є проходження всіх фаз створення реального програмного забезпечення: поява ідеї та її формування, створення вимог, створення програмних модулів, тестування, створення документації та експлуатація.

Для досягнення поставленої мети було використано форму зберігання даних про нерухомість розроблену RESO (Real Estate Standards Organization), яка являється найбільш актуальною в даних час та прийнятою у всьому світі, що і допоможе розвивати сервіс на світовому рівні.

Практичним завданням проекту є створення системи, яка буде зберігати дані про нерухомість у форматі, який відповідає вимогам світу, та дозволяє власникам створювати об’яви у одному місці.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування доцільності розробки

У сучасному представлені про мережу Інтернет є проблема у неактуальності даних так як деякі з них постійно повторюються.

Весь світ намагається вирішити дану проблему створенням стандартів для зберігання даних у певних предметних областях. У цій роботі пропонується рішення, яке буде використовувати стандарт моделі даних прийнятий всім світом. Однією з галузей, яка потребує використання такого стандарту є ринок нерухомості.

Історично склалось, що закон про продаж землі може появитись в Україні лише зараз. Дана робота не буде оцінювати його доцільність, але в будь-якому випадку розробка програмного забезпечення, яке буде зберігати новий потік даних про аграрні ділянки в єдиному форматі може допомогти розвитку України.

1.2 Аналіз предметної області

1.2.1 Нерухомість

Нерухомість - це власність, що складається з землі та будівель розміщеній на ній. Також до нерухомості відносять будь-які природні ресурси у межах земельної ділянки, таких як вода та корисні копалини. До речі у деяких країнах повітря також відносять до цих ресурсів.

Зазвичай нерухомість поділяють на 4 типи:

- житлова;
- комерційна;
- промислова;
- земельна.

Ця класифікація допомагає встановлювати правила власності та торгівлі нерухомістю, наприклад тільки зараз стає можливим передача земельної ділянки від одного власника до іншого, якщо це не спадщина.

До житлових об'єктів належать споруди для проживання вдома, такі як будинки, мобільні будинки та квартири.

Комерційна нерухомість включає структури, які використовуються для отримання доходу, такі як офіси, магазини, готелі, послуги та інші підприємства.

До промислових властивостей відносяться конструкції, що використовуються з ціллю, яка відрізняється від можливості проживання. Це в загальному фабрики, склади та науково-дослідні центри. Взагалі промисловий тип нерухомості використовується для виробництва товарів, а комерційний - для розподілу товарів.

Власність земельних ділянок включає в себе мало або взагалі ніяких споруд, таких як вільні землі, ферми, ранчо. Зазвичай такі ділянки використовують у аграрному ділі, лісництві, або ж вони просто стоять.

1.2.2 Ринок нерухомості

Ринок нерухомості - це всі об'єкти, доступні для продажу в певній місцевості. Через певні економічні зміни бувають випадки, коли ціна на нерухому власність одночасно збільшуються (або падає). Це те, що люди мають на увазі, коли кажуть, що ринок піднімається (або падає).

Ринок житла - це сегмент ринку нерухомості, який складається лише з житлової нерухомості. За тенденціями ринку житла уважно стежать, оскільки вони забезпечують загальний рівень добробуту.

Оскільки багато житлових об'єктів належать окремим сім'ям, якщо ринок житла працює добре, ми можемо припустити, що загальний добробут сім'ї у порядку, а отже і чиста ціна нерухомої власності зростає.

1.3 Цілі та задачі розробки

Основною метою даного проекту є збір та збереження усіх даних про нерухомість України у одному форматі. Дана система повинна допомагати власникам нерухомості будь-якого виду у публікації об’яв та знаходженні покупців, а для покупцям – у можливості знайти їх майбутню власність не шукаючи її на багатьох ресурсах.

Більшість моделей даних про нерухомість, які використовуються зараз в Україні не готові до відкриття ринку землі, який буде працювати з аграрними ділянками, а отже ціллю також являється універсальність.

1.4 Висновок до розділу

Ринок нерухомості потребує централізації даних. Тільки при таких умовах, можна організувати здорову конкуренцію між компаніями, які пропонують Вам свої програмні рішення для пошуку та продажу нерухомості.

Так як існує велика кількість видів нерухомості, дані, які будуть зберігатись, повинні бути готовими наприклад до переходу від одного виду до іншого.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Існуючі системи продажу нерухомості

Автоматизація ринку нерухомості не є новою ідеєю і завжди прогресує залежно від розвитку технологій. Велика кількість програмних рішень вже знайшла своїх користувачів, а отже корисним буде проаналізувати їх досвід.

2.1.1 ЛУН

ЛУН позиціонує себе як найбільший сайт з пошуку квартир в Україні. По факту даний сервіс являється агрегатором, який збирає інформацію з багатьох українських сервісів та подає у потрібному клієнтові форматі. Головною перевагою ЛУН є генерація даних способом збору інформації з інших сервісів, так як у такому випадку відпадає необхідність у створені форм для зберіганні інформації введеної користувачами. Також до переваг ЛУН можна добавити велику кількість фільтрів для пошуку необхідного об’єкту нерухомості – серед таких фільтрів є і карта. Також даний сервіс має стрічку новин, що допомагає користувачам бачити завжди актуальні події. Слід відзначити, що ЛУН підсумовує статистику по цінам на нерухомість кожного дня, що ще раз підкреслює про гарну підтримку та користь використовуваних алгоритмів. Серед недоліків я б відзначив вузька направленість сервісу та відсутня універсальність у даних. ЛУН спеціалізується лише на житловому виді нерухомості, а значить не має можливості слідкувати за комерційною, промисловою та земельною. Цілком можливо, що при бажані дана компанія могла б розширити сферу впливу, але через те, що універсальність у моделі даних відсутня потрібно буде перебудувати усю систему.

Основні переваги ЛУН:

– генерація даних за допомогою зовнішніх сервісів;
– кількість фільтрів для даних;

– актуальність та підтримка;
– алгоритми статистики.

Основні недоліки ЛУН:

– відсутня можливість створення власних об’яв;
– погана універсальність даних;
– вузька направленість.

2.1.2 Dom Ria

Dom Ria – яскравий приклад сервісів по збору інформації шляхом введення її користувачами. Даний сервіс має вузьку направленість, а отже збирає дані лише про житлову нерухомість. Серед переваг слід відзначити можливість викликати інспектора по нерухомості, який може допомогти вам оцінити ваше житло і виставити ціну, яка відповідає ринку. Це означає, що даний сервіс відійшов від простого збереження та розміщення даних до бізнес моделі, яка надає послуги зв’язані з сферою нерухомості. Серед недоліків слід відзначити маленьку кількість фільтрів для пошуку, що робить функцію знаходження потрібної власності більш важкою. Також я б відзначив погану модель даних – кількість рис, якими описують нерухомість являється надзвичайно малою і по факту нормальний опис дають лише фотографії, розміщені користувачами.

Основні переваги Dom Ria:

– генерація даних користувачами;
– бізнес модель компанії.

Основні недоліки Dom Ria:

– погана модель даних;
– маленька кількість фільтрів для пошуку;
– вузька направленість.

2.1.3 Zillow.com

Для того, щоб провести гарне дослідження існуючих рішень потрібно розглянути сервіс, який використовує RESO. Ринок нерухомості являється локальним і правильним є розглядання рішень, якими користуються в Україні, але жоден з таких сервісів ще не використовує даний стандарт, а тому розглянемо іноземну систему.

Zillow - це провідний сервіс пошуку нерухомості, присвячений наданню споживачам даних та знань про місце де знаходиться будинок, та з'єднанні їх з кращими місцевими професіоналами, які можуть допомогти наприклад у його оздоблені, садівництві чи ремонті.

Zillow служить повним життєвим циклом володіння та проживання в будинку: купівля, продаж, оренда, фінансування, реконструкція та інше. Він починається з живої бази даних Zillow, що містить понад 110 мільйонів американських будинків - включаючи будинки на продаж, будинки для оренди та будинки, які зараз не продаються, а також оціночні цінності для дому, оцінки вартості оренди та іншу інформацію, пов’язану з домом.

Це яскравий приклад програмного забезпечення світового рівня.

Серед недоліків слід відзначити все ту ж вузьку направленість – даний сервіс спеціалізується лише на житловій нерухомості.

Основні переваги Zillow:

- використання стандарту RESO;
- велика кількість фільтрів для пошуку;
- можливість пошуку робітників;
- можливість створення об’яв власноруч.

Основні недоліки Zillow:

- вузька направленість з даними про нерухомість.

2.2 Аналіз та порівняння

Розглянуті рішення показують, що сервіси, які зберігають дані про нерухомість мають велику різноманітність. Проаналізувавши ці застосунки можемо виділити основні недоліки, які потрібно урахувати при розроблені системи:

- вузька направленість з даними про нерухомість;
- використання стандарту;
- кількість фільтрів.

Врахувавши усі недоліки та переваги можна побудувати сервіс, який зможе надати користувачам можливості не враховані іншими розробниками.

2.3 Висновок до розділу

Українські існуючі рішення явно не відповідають світовому рівню. Керуючись досвідом попередників можна створити проект, який стане актуальним на місцевому ринку та швидко займе важливе місце у галузі нерухомості.

3 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Функціональні вимоги до системи

Опис функціональних вимог допомагає спроектувати програмне рішення правильно і в майбутньому не витрачати ресурси на переробку модулів, які не відповідають поставленим задачам. При аналізі існуючих програмних рішень було виділено основні переваги та недоліки, на базі яких і було створено вимоги до даної системи:

– наявність авторизації;

– наявність декількох рівнів доступу;

– можливість комплексного пошуку;

– можливість зміни даних;

– наявність блогу та статей з великою кількістю тегів для покращення результатів у пошукових ресурсах;

– можливість роботи з медіа-файлами;

– можливість роботи з картою, яка являється найкращим фільтром для пошуку за адресом;

– можливість адміністрування;

– наявність чату для спілкування між власником нерухомості та покупцем;

– база даних, яка відповідає стандартам моделі даних про нерухомість.

3.2 Нефункціональні вимоги до системи

Проаналізувавши досвід існуючих рішень можна виділити основні нефункціональні вимоги. Вони є ключовою деталлю у підтримці даного проекту у майбутньому, так як описують не функціонал, а життєздатність системи.

Основними нефункціональними вимогами є:

– розширюваність – так як дані, які будуть зберігатись не є повним описом предметної області нерухомості, система повинна бути здатною до розширюваності в будь-який момент часу;

– безпека – система повинна зберігати багато важливих користувацьких даних, а отже повинна бути добре захищена від можливих хакерських атак;

– відповідність законам України – так як ринок нерухомості та закони про нього постійно змінюються, система повинна відповідати останній версії законодавства та виключати функції, які переступають межу закону;

– SEO оптимізація – для того, щоб система стала відомою, вона повинна показувати гарні результати у сервісах пошуку. Для цього SEO теги системи повинні повністю відповідати контенту, який зберігається у сторінках;

– швидкість роботи – при пошуку нерухомості всередині системи потенційно буде використовуватись велика кількість даних, а отже всі запити повинні бути добре оптимізованими.

3.3 Висновки до розділу

На базі недоліків та переваг існуючих рішень було виділено основні функціональні та нефункціональні вимоги до системи. Це допоможе проекту бути кращим ніж аналоги та швидко зайняти своє місце на ринку. Також це зробить легшим майбутню підтримку та оптимізацію.

4 АНАЛІЗ СЦЕНАРІЇВ ВИКОРИСТАННЯ СИСТЕМИ

4.1 Взаємодія користувача з системою

Користувачі, які будуть працювати з системою, не завжди будуть мати технічний досвід, а отже потрібно виділити способи взаємодії, які будуть відповідати різним рівням знань.

4.1.1 Користувацький інтерфейс

Найбільш зручним способом для взаємодії користувача з системою є інтерфейс, який може бути відкритий у браузері. Використовуючи цей спосіб роботи, користувач не потребує якихось технічних знань і може працювати з даними системи просто за допомогою компонентів інтерфейсу зрозумілих будь-якій людині.

4.1.2 Взаємодія з використанням запитів

Будь-яка сучасна система повинна мати відкритий API для технічних користувачів. Це допомагає розширити використання системи, так як будь-який технічний спеціаліст зможе створити клієнтський інтерфейс для будь-якої платформи та використовувати ваші дані з посиланням на джерело. У такому випадку можлива централізація усіх даних про нерухомість на одному сервері, але з використанням великої кількості клієнтських інтерфейсів.

4.2 Сценарії використання системи

При побудові будь-якого проекту важливим є опис основних сценаріїв взаємодії користувача з програмним рішенням. Існує багато способів опису

сценаріїв, але діаграма використання являється найбільш швидким та ефективним рішенням.

На кресленику ІТ61.160БАК.004 Д1 наведена прецендентів даного проекту.

Таблиця 4.1 – Варіанти використання системи

Шифр варіанту використання	Назва варіанту використання
UC-1	Перегляд головної сторінки.
UC-1.1	Зміна мови інтерфейсу.
UC-2	Пошук статей.
UC-2.1	Перегляд результатів пошук.
UC-2.2	Зміна фільтрів пошуку
UC-2.3	Перегляд сторінки статті..
UC-3	Пошук об’яв про нерухомість.
UC-3.1	Перегляд результатів пошуку.
UC-3.2	Перегляд сторінки нерухомості.
UC-3.2.1	Зберегти результат.
UC-3.2.2	Написати редактору.
UC-3.3	Зміна фільтрів пошуку.
UC-4	Реєстрація.
UC-4.1	Вивід повідомлення про успішну реєстрацію.

Продовження таблиці 4.1

Шифр варіанту використання	Назва варіанту використання
UC-4.2	Вивід повідомлення про неуспішну реєстрацію.
UC-5	Авторизація панелі редакторів.
UC-5.1	Доступ до меню керування обліковим записом.
UC-5.1.1	Редагування даних облікового запису.
UC-5.2	Доступ до меню керування чатами.
UC-5.2.1	Перегляд усіх чатів редактора.
UC-5.2.2	Створення повідомлення.
UC-5.3	Доступ до меню керування нерухомістю.
UC-5.3.1	Створення нерухомості.
UC-5.3.2	Редагування нерухомості
UC-6	Авторизація панелі адміністраторів.
UC-6.1	Доступ до меню керування нерухомістю.
UC-6.1.1	Зняття нерухомості з ринку
UC-6.2	Доступ до меню керування користувачами.
UC-6.2.1	Блокування користувача.
UC-6.3	Доступ до меню керування статтями.
UC-6.3.1	Видалення статті.

У таблицях 4.2-3. приведені описи кожного з сценаріїв використання.

Опис сценарію використання з кодом UC-1 відображено у таблиці 4.2.

Таблиця 4.2 – Сценарій використання UC-1

Назва	Перегляд головної сторінки.
Опис	Користувач може переглядати головну сторінку.
Учасники	Користувач.
Початкові дані	
Результат	Відображена головна сторінка.
Порядок дій	1) Система надає посилання на головну сторінку; 2) Користувач натискає кнопку “Main page”; 3) Система відображає головну сторінку.

Опис сценарію використання з кодом UC-1.1 відображено у таблиці 4.3.

Таблиця 4.3 – Сценарій використання UC-1.1

Назва	Зміна мови інтерфейсу.
Опис	Користувач має змогу змінити мову інтерфейсу.
Учасники	Користувач.
Початкові дані	
Результат	Мова інтерфейсу змінена на обрану.
Порядок дій	1) Система надає кнопку вибору “Language”; 2) Користувач натискає на кнопку вибору;

Продовження таблиці 4.3

Порядок дій	3) Користувач обирає мову; 4) Система відображає сторінку з мовою.

Опис сценарію використання з кодом UC-2 відображено у таблиці 4.4.

Таблиця 4.4 – Сценарій використання UC-2

Назва	Пошук статей.
Опис	Користувачу надається можливість пошуку статей.
Учасники	Користувач.
Початкові дані	Користувач перейшов на сторінку пошуку.
Результат	Відображено знайдені статті.
Порядок дій	1) Система демонструє кнопку вводу; 2) Користувач вводить назву статті; 3) Система знаходить статті алгоритмом пошуку; 4) Система відображає знайдені статті.

Опис сценарію використання з кодом UC-2.1 відображено у таблиці 4.5.

Таблиця 4.5 – Сценарій використання UC-2.1

Назва	Перегляд результатів пошуку.
Опис	Користувач може переглядати результати пошуку.
Учасники	Користувач.
Початкові дані	Користувач ввів в пошук назву статті.

Продовження таблиці 4.5.

Результат	Результати пошуку переглянуті.
Порядок дій	1) Система надає усі знайдені статті; 2) Користувач переглядає короткий опис статей; 3) Користувач може обрати потрібну статтю.

Опис сценарію використання з кодом UC-2.2 відображено у таблиці 4.6.

Таблиця 4.6 – Сценарій використання UC-2.2

Назва	Зміна фільтрів пошуку.
Опис	Користувач може змінити фільтри пошуку.
Учасники	Користувач.
Початкові дані	Користувач задав пошук.
Результат	Фільтри пошуку змінені.
Порядок дій	1) Система надає усі знайдені статті; 2) Користувач переглядає короткий опис статей; 3) Користувач обирає потрібну йому статтю.

Опис сценарію використання з кодом UC-2.3 відображено у таблиці 4.7.

Таблиця 4.7 – Сценарій використання UC-2.3

Назва	Перегляд сторінки статті.
Опис	Користувачу надається можливість переглянути обрану статтю

Продовження таблиці 4.7.

Учасники	Користувач.
Початкові дані	Користувач обрав статтю.
Результат	Стаття переглянута.
Порядок дій	1) Система демонструє сторінку статті; 2) Користувач може переглядати дані статті.

Опис сценарію використання з кодом UC-3 відображено у таблиці 4.8.

Таблиця 4.8 – Сценарій використання UC-3

Назва	Пошук об’яв про нерухомість.
Опис	Користувачу надається можливість пошуку об’яв.
Учасники	Користувач.
Початкові дані	
Результат	Система відображає знайдені об’яви.
Порядок дій	1) Система демонструє кнопку вводу; 2) Користувач вводить назву чи опис об’яви; 3) Система знаходить об’яви алгоритмом пошуку; 4) Система відображає знайдені об’яви.

Опис сценарію використання з кодом UC-3.1 відображено у таблиці 4.9.

Таблиця 4.9 – Сценарій використання UC-3.1

Назва	Перегляд результатів пошуку.

Продовження таблиці 4.9.

Опис	Користувач переглядає результати пошуку об’яв.
Учасники	Користувач.
Початкові дані	Користувач ввів в поле пошуку назву об’яви чи її короткий опис.
Результат	Результати пошуку переглянуті.
Порядок дій	1) Система надає усі знайдені об’яви; 2) Користувач переглядає короткий опис наданих об’яв; 3) Користувач обирає необхідну нерухомість.

Опис сценарію використання з кодом UC-3.2 відображено у таблиці 4.10.

Таблиця 4.10 – Сценарій використання UC-3.2

Назва	Перегляд сторінки нерухомості.
Опис	Користувачу має змогу переглянути обрану нерухомість.
Учасники	Користувач.
Початкові дані	Користувач обрав об’яву нерухомості.
Результат	Об’ява нерухомості переглянута користувачем.
Порядок дій	1) Система ілюструє обрану сторінку нерухомості; 2) Користувач переглядає дані нерухомості та працює з інтерфейсом сторінки;

Опис сценарію використання з кодом UC-3.2.1 відображено у таблиці 4.11.

Таблиця 4.11 – Сценарій використання UC-3.2.1

Назва	Зберегти результат.
Опис	Користувачу надається можливість зберегти результат знайденої нерухомості.
Учасники	Користувач.
Початкові дані	Користувач переглянув об’яву нерухомості.
Результат	Знайдена нерухомість збережена користувачем.
Порядок дій	1) Система надає кнопку “Save result”; 2) Користувач натискає на кнопку “Save result”; 3) Система зберігає обрану нерухомість.

Опис сценарію використання з кодом UC-3.2.2 відображено у таблиці 4.12.

Таблиця 4.12 – Сценарій використання UC-3.2.2

Назва	Написати редактору
Опис	Користувач має змогу написати редактору
Учасники	Користувач.
Початкові дані	Користувач переглянув об’яву нерухомості.
Результат	Користувач встановив зв’язок з редактором.
Порядок дій	1) Система надає поле вводу і кнопку “Contact”; 2) Повідомлення надсилається редактору.

Опис сценарію використання з кодом UC-3.3 відображено у таблиці 4.13.

Таблиця 4.13 – Сценарій використання UC-3.3

Назва	Зміна фільтрів пошуку.
Опис	Користувач може змінити фільтри пошуку об’яв.
Учасники	Користувач.
Початкові дані	Користувач задав пошук об’яв нерухомості.
Результат	Фільтри пошуку об’яв змінені.
Порядок дій	1) Система надає фільтри пошуку об’яв; 2) Користувач обирає фільтри для змінення; 3) Користувач натискає кнопку “Change filtres”.

Опис сценарію використання з кодом UC-4 відображено у таблиці 4.14.

Таблиця 4.14 – Сценарій використання UC-4

Назва	Реєстрація.
Опис	Користувач має змогу зареєструватись.
Учасники	Користувач
Початкові дані	
Результат	Користувач зареєстрований або помилка.
Порядок дій	1) Система надає сторінку реєстрації; 2) Користувач заповнює поля вводу; 3) Користувач натискає кнопку “Sign in”;

Продовження таблиці 4.14

Порядок дій	4) Система додає користувача.

Опис сценарію використання з кодом UC-4.1 відображено у таблиці 4.15.

Таблиця 4.15 – Сценарій використання UC-4.1

Назва	Вивід повідомлення про успішну реєстрацію.
Опис	Повідомлення про успішну реєстрацію відображається користувачу.
Учасники	Користувач.
Початкові дані	Користувач ввів коректні дані в поля реєстрації.
Результат	Користувач отримав повідомлення про успішну реєстрацію.
Порядок дій	Система демонструє повідомлення про успішну реєстрацію.

Опис сценарію використання з кодом UC-4.2 відображено у таблиці 4.16.

Таблиця 4.16 – Сценарій використання UC-4.2

Назва	Вивід повідомлення про неуспішну реєстрацію.
Опис	Повідомлення про неуспішну реєстрацію відображається користувачу.
Учасники	Користувач.
Початкові дані	Користувач ввів некоректні дані в поля реєстрації.

Продовження таблиці 4.16

Результат	Користувач отримав повідомлення про неуспішну реєстрацію.
Порядок дій	Система демонструє повідомлення про неуспішну реєстрацію.

Опис сценарію використання з кодом UC-5 відображено у таблиці 4.17.

Таблиця 4.17 – Сценарій використання UC-5

Назва	Авторизація панелі редакторів.
Опис	Редактор авторизується у програмі.
Учасники	Редактор.
Початкові дані	Редактор знаходиться на сторінці авторизації.
Результат	Авторизація редактор у програмі відбулась.
Порядок дій	1) Програма показує сторінку авторизації редактора; 2) Редактор заповнює поля вводу; 3) На сторінці ілюстровано кнопку “Log in”; 4) Редактор натискає на кнопку “Log in”; 5) Програма авторизує редактора.

Опис сценарію використання з кодом UC-5.1.1 відображено у таблиці 4.18.

Таблиця 4.18 – Сценарій використання UC-5.1.1

Назва	Редагування даних облікового запису

Продовження таблиці 4.18

Опис	Редактор може редагувати дані свого облікового запису.
Учасники	Редактор
Початкові дані	Редактор авторизувався у програмі
Результат	Дані редактора редаговано.
Порядок дій	1) Програма показує обліковий запис; 2) Програма ілюструє кнопку “Edit”; 3) Редактор нажимає на кнопку “Edit”; 4) Редактор змінює дані в полях вводу; 5) Програма додає введені дані до таблиці бази даних про редактора.

Опис сценарію використання з кодом UC-5.2.1 відображено у таблиці 4.19.

Таблиця 4.19 – Сценарій використання UC-5.2.1

Назва	Перегляд усіх чатів редактора.
Опис	Редактор переглядає чати .
Учасники	Редактор.
Початкові дані	Редактор перейшов на сторінку чатів.
Результат	Усі чати редактора переглянуті.
Порядок дій	1) Програма демонструє сторінку чатів редактора; 2) Редактор переглядає усі чати у порядку від нових до створених раніше.

Опис сценарію використання з кодом UC-5.2.2 відображено у таблиці 4.20.

Таблиця 4.20 – Сценарій використання UC-5.2.2

Назва	Створення повідомлення.
Опис	Редактор має можливість створювати повідомлення.
Учасники	Редактор.
Початкові дані	Редактор обрав певний чат.
Результат	Повідомлення створено редактором.
Порядок дій	1) Програма надає поле для вводу повідомлення; 2) Редактор вводить текст повідомлення; 3) Редактор натискає на кнопку “Send”; 4) Система створює повідомлення в чаті.

Опис сценарію використання з кодом UC-5.3.1 відображено у таблиці 4.21.

Таблиця 4.21 – Сценарій використання UC-5.3.1

Назва	Створення нерухомості.
Опис	Редактор здатен створювати нерухомість.
Учасники	Редактор.
Початкові дані	Редактор перейшов на сторінку створення даних.
Результат	Програма додає нерухомість до бази даних.
Порядок дій	1) Програма надає поля вводу, кнопку додавання зображень “Image” і кнопку “Create”;

Продовження таблиці 4.21

Порядок дій	2) Редактор вносить в поля інформацію про нерухомість; 3) Редактор натискає на кнопку “Image”; 4) Редактор додає необхідні зображення; 5) Редактор натискає на кнопку “Create”; 6) Програма додає нерухомість до бази даних.

Опис сценарію використання з кодом UC-5.3.2 відображено у таблиці 4.22.

Таблиця 4.22 – Сценарій використання UC-5.3.2

Назва	Редагування нерухомості.
Опис	Редактор має можливість редагувати створену нерухомість.
Учасники	Редактор.
Початкові дані	Редактор обрав певну нерухомість.
Результат	Програма містить відредаговану нерухомість.
Порядок дій	1) Програма ілюструє сторінку редагування та кнопку збереження “Save”; 2) Редактор змінює дані нерухомості в полях вводу; 3) Редактор додає нові зображення; 4) Редактор натискає на кнопку збереження “Save”; 5) Програма вносить дані до таблиці бази даних про нерухомості;

Опис сценарію використання з кодом UC-5.3.3 відображен у таблиці 4.23.

Таблиця 4.23 – Сценарій використання UC-5.3.3

Назва	Видалення нерухомості.
Опис	Редактор здатен видаляти створену нерухомість.
Учасники	Редактор.
Початкові дані	Редактор обрав певну нерухомість.
Результат	Програма видаляє нерухомість з бази даних.
Порядок дій	1) Редактор натискає на кнопку “Delete”; 2) Програма видаляє нерухомість з бази даних.

Опис сценарію використання з кодом UC-6 відображено у таблиці 4.24.

Таблиця 4.24 – Сценарій використання UC-6

Назва	Авторизація панелі адміністраторів.
Опис	Адміністратор авторизується у програмі.
Учасники	Адміністратор.
Початкові дані	Адміністратор знаходиться на сторінці авторизації.
Результат	Авторизація адміністратора у програмі відбулась.
Порядок дій	1) Програма показує сторінку авторизації 2) Адміністратор заповнює поля вводу; 3) Адміністратор натискає на кнопку “Log in”; 4) Програма авторизує редактора.

Опис сценарію використання з кодом UC-6.1.1 відображено у таблиці 4.25.

Таблиця 4.25 – Сценарій використання UC-6.1.1

Назва	Зняття нерухомості з ринку.
Опис	Адміністратор знімає нерухомість з ринку продаж.
Учасники	Адміністратор.
Початкові дані	Адміністратор обрав певну нерухомість.
Результат	Нерухомість знята з ринку.
Порядок дій	1) На сторінці нерухомості програма ілюструє кнопку зняття з продаж; 2) Адміністратор натискає на кнопку зняття з продаж; 3) Програма вносить зміни в таблицю бази даних; 4) Програма не відображає обрану нерухомість у пошуку.

Опис сценарію використання з кодом UC-6.2.1 відображено у таблиці 4.26.

Таблиця 4.26 – Сценарій використання UC-6.2.1

Назва	Блокування користувача.
Опис	Адміністратор має можливість заблокувати користувача.
Учасники	Адміністратор.
Початкові дані	Адміністратор обрав певного користувача.

Продовження таблиці 4.26

Результат	Користувач не може користуватися даною програмою.
Порядок дій	1) Адміністратор зайшов на профіль певного користувача; 2) Система надає кнопку для блокування “Block user”; 3) Адміністратор натискає на кнопку блокування користувача «Block user”; 4) Програма вносить зміни в таблицю базу даних про користувача.

Опис сценарію використання з кодом UC-6.3.1 відображено у таблиці 4.27.

Таблиця 4.27 – Сценарій використання UC-6.3.1

Назва	Видалення статті.
Опис	Адміністратору надається можливість видалення статті.
Учасники	Адміністратор.
Початкові дані	Адміністратор обрав певну статтю.
Результат	Програма видалила обрану статтю з бази даних.
Порядок дій	1) Програма ілюструє сторінку редагування та надає кнопку видалення “Delete”; 2) Адміністратор натискає на кнопку “Delete”; 3) Програма видаляє статтю з бази даних.

Опис сценарію використання з кодом UC-6.3.2 відображено у таблиці 4.28.

Таблиця 4.28 – Сценарій використання UC-6.3.1

Назва	Створення статті.
Опис	Адміністратор здатен створювати статтю.
Учасники	Адміністратор.
Початкові дані	Адміністратор перейшов на сторінку створення статті.
Результат	Програма додала статтю до бази даних. Стаття доступна в пошуку.
Порядок дій	1) Програма надає поле вводу, кнопку додавання зображень “Image” і кнопку “Create”; 2) Адміністратор вносить в поле інформацію статті; 3) Адміністратор натискає на кнопку “Image”; 4) Адміністратор додає необхідні зображення; 5) Адміністратор натискає на кнопку “Create”; 6) Програма додає до бази даних.

4.3 Висновки до розділу

В цьому розділі було описано способи взаємодії користувача з системою. Для того, щоб описати можливі сценарії роботи було створено діаграму використання та доповнено її за допомогою таблиць.

5 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ

5.1 Вибір технологій розробки

При створені програмного забезпечення одною із основних задач є правильний вибір технологій.

Зазвичай проекти діляться на декілька частин, кожною з яких займається певна група розробників, кожна частина є незалежною та добре документованою, щоб при її використані не виникало проблем через незнання певних особливостей.

При виборі технологій слід опиратись саме на завдання, яке повинна виконувати система, адже в такому випадку можна виграти на наприклад на кількості пам’яті, яку використовує застосунок, чи на швидкості відправлення відповідей на запити до серверу. Також якщо немає можливості змінювати команду розробників, велику роль буде грати стек її технологій, адже перенавчання кожного члену команди на нову технологію може бути дорогим у коштах і часі.

Для розробки даної системи було вибрано такий стек технологій:

– мова програмування Javascript;

– Node.js для серверної сторони;

– фреймворк Express.js для створення API;

– фреймворк Next.js для генерації сторінок на серверній стороні та покращені SEO результатів у пошукових системах;

– бібліотека Knex.js для роботи з базою даних;

– React.js як бібліотека для побудови клієнтської сторони;

– бібліотека Socket.IO для роботи з web сокетами;

– бібліотека-фреймворк Bootstrap 4 для створення теми зовнішнього вигляду системи;

– фреймворки Mocha/Chai для створення інтеграційних тестів системи;

– база даних PostgresSQL;

– стандарт створений RESO для проектування моделі бази даних;

– веб сервіс Google Cloud Platoform для зберігання коду та менеджменту задач, розміщення бази даних та даної системи, збереження файлів.

5.1.1 Javascript

JavaScript – це мова програмування, яка працює з великою кількістю парадигм та є широковикористованою як і у браузерах, так і при написанні коду для серверної частини застосунків. [1]

Javascript має такі переваги:

– швидкість виконання;

– мультипарадигменність;

– велика спільнота розробників, що і створює підтримку при виникненні проблем;

– великий вибір фреймворків та бібліотек;

– можливість написанні серверної та клієнтської частини на одній мові.

У цьому проекті дана мова буде використана у всіх модулях та частинах застосунку – це полегшує розробку та підтримку системи у подальшому.

5.1.2 Node.js

Найпростіше визначення Node.js полягає в тому, що це середовище виконання Javascript коду на сервері. Це крос-платформний JavaScript з відкритим кодом, який допомагає у створенні мережевого додатку у режимі реального часу. [2]

Він пропонує розробникам керування подіями вводу-виводу та роботу з асинхронністю. Він також може інтерпретувати код JavaScript через двигун JavaScript V8 Google.

Основними перевагами Node.js є

– відкритість коду;
– велика спільнота;
– велика кількість фреймворків та бібліотек;
– швидкість.

5.1.3 Express.js

Express.js - це простий і гнучкий веб-фреймворк для серверів Node.js, що надає великий набір функцій та паттернів для створення легкопідтримануваного коду. Маючи в своєму розпорядженні безліч службових методів HTTP і проміжних оброблювачів, створити надійний API можна швидко і легко. [3]

5.1.4 Next.js

Next.js це фреймворк для статичних і оброблених сервером компонентів створених при використанні React.js. [4]

Основними перевагами Next.js є:

– розширюваність;
– написання css коду у Javascript;
– робота з мета тегами сторінки;
– велика підтримка.

Для знаходження користувачів, будь який сервіс використовує пошукові системи. Проблема усіх сторінок, створених на клієнтській стороні є відсутність SEO тегів, що опускає їх у самий низ відповідей на запроси користувачів у пошукових системах. Для того, щоб даний сервіс не втратив актуальність було прийнято рішення використовувати даний фреймворк, так як він дозволяє загружати сторінки з використанням React.js на сервері.

5.1.5 Knex.js

Knex.js - це конструктор запитів SQL, що включає провайдери для Postgres, MSSQL, MySQL, MariaDB, SQLite3, Oracle та Amazon Redshift, які призначених для гнучкості при переході з бази даних на іншу. [5] Він містить повнофункціональні конструктори запитів, підтримку транзакцій та можливість створення міграцій. Knex.js являється найкращим рішенням проблеми створення запитів до бази даних з Javascipt коду, так як позволяє створювати комплексі запити доволі швидко, але при цьому надає можливість не відходити від оригінального SQL коду.

5.1.6 React.js

React - це бібліотека для побудови інтерфейсів для роботи з користувачами. Вона надає можливість створення багаторазових компонентів інтерфейсу користувача, які зображують дані, що змінюються з часом. [6]

5.1.7 Socket.IO

Socket.IO – це бібліотека, яка дає можливість працювати з web сокетами високорівнево. Вона дозволяє користувачам спілкуватися в режимі реального часу двосторонньо та на основі певних подій. Вона працює на кожній платформі, веб-переглядачі чи пристрої, орієнтуючись однаково на надійність та швидкість. [7]

5.1.8 Bootstrap

Bootstrap – це безкоштовна бібліотека, яка по факту являє собою набір елементів для створення розмітки сайтів. Включає в себе готові компоненти для

створення інтерфейсу користувача: таблиці, шрифти, елементи форм, кнопки, замітки тощо. [8]

Особливості:

– при розмітці використовує сітку з 12 колонок, яка буде змінюватись залежно від розміру пристрою на якому переглядається застосунок;

– має можливість легко мігрувати з однієї теми на іншу. Якщо компанія проводить вирішила провести ребрендинг – достатньо буде просто замінити використовувану тему на нову;

– має велику кількість готових малюнків для кнопок.

Bootstrap був вибраний, щоб пришвидшити розмітку сторінок даної системи, так як без його використання розмітку для кожного пристрою потрібно робити окремо, що не тільки затримає розробку, а й ускладнить тестування.

5.1.9 Mocha/Chai

Mocha - це багатофункціональна бібліотека для тестування JavaScript коду, що працює на Node.js та у веб-переглядачі, що робить асинхронне тестування простим та цікавим. Тести Mocha виконуються послідовно, що дозволяє гнучко та точно звітувати, при цьому відображаючи невдалі та правильні тестові випадки. [9]

Chai – це бібліотека для порівняння результатів виконання функцій у Javascript. Зазвичай використовується з тестовими бібліотеками, щоб звіряти вірність виконання методів. [10]

5.1.10 PostgresSQL

PostgreSQL - це потужна об'єктно-реляційна база даних із відкритим кодом з більш ніж 30-річним активним розвитком, яка заслужила їй високу репутацію надійності функції та продуктивності. [11]

Вибір системи управління бази даних являється найбільш складним серед усіх технологій, так як при збереженні великої кількості даних, це може спричинити як і збереження так і витрату лишніх коштів. Від вибраної бази даних буде залежати швидкість опрацювання запитів, що в висновку може привести до втрати актуальності додатку.

PostgresSQL являється найкращим вибором серед усіх реляційних баз даних на даний момент, так як позволяє зберігати деякі дані у JSON форматі, має високу швидкість реакції та надійність. При виборі бази даних для даного проекту також потрібно було керуватись можливістю роботи з даними про координати та фігури, так як кожна нерухомість буде описуватись ними для покращення пошуку за допомогою карти, PostgresSQL має розширення PostGIS, яке позволяє зробити це. PostgresSQL має велику кількість операторів та алгоритмів для порівняння та знаходження даних, що дозволяє побудувати багатофункціональний пошуковий модуль.

5.1.11 Стандарт розроблений RESO

RESO створює відкриті стандарти, які сприяють інноваціям у сфері нерухомості. Організації-учасниці RESO спільно розробляють та впроваджують ці стандарти для створення ефективності: швидшої розробки та інтеграції програмного забезпечення, масштабованих інструментів, що перетинають ринки збуту, та більш точних, надійних та інформативних даних.

Організації з відкритими стандартами, як RESO, існують у більшості галузей для забезпечення прогресу технологій. Відомий приклад - W3C, який створює стандарти для всесвітньої павутини. Найпотужніші технологічні компанії світу погоджуються співпрацювати для створення спільної мережі, яка піднімає технологічні основи всіх конкурентів.

Перевагою цього є можливість отримувати дані, додатки, послуги та продукти багатьох конкуруючих компаній на одному смартфоні. У сфері нерухомості стандарти RESO створені для просування цього ж досвіду.

5.1.12 Google Cloud Platform

Google Cloud Platform – це платформа, яка надає набір хмарних служб, які виконуються на тій же самій інфраструктурі, яку Google використовує для своїх продуктів, призначених для кінцевих споживачів, таких як Google Search і YouTube. Крім інструментів для управління, також надається ряд модульних хмарних служб, таких як хмарні обчислення, зберігання даних, аналіз даних і машинне навчання. [12]

Дана платформа являється найкращим вибором для розміщення системи, так як вона пропонує розробникам можливість балансування запитів залежно від місця їх надходження. У нас час важливим являється доступ до сервісу з будь якого місця світу і тільки балансування запитів може допомогти пришвидшити відповідь наприклад на запит з Америки.

5.2 Висновки до розділу

В цьому розділі було оглянуто технології, які будуть використані при розробці даного програмного забезпечення. Всі технології було вибрано залежно від сучасних трендів у розробці та від задач поставлених у попередніх розділах.

6 РОЗРОБКА СИСТЕМИ

6.1 Структура модулів

Для того, щоб створювана система могла використовувати однакові функції у різних місцях, вона повинна бути поділена на шари, які в свою чергу будуть поділені на модулі. Найкращим способом такого розподілу являється діаграма пакетів, яка і є зображена на кресленику ІТ61.160БАК.004 ДЗ. Система має 4 шари, які взаємодіють між собою:

- шар представлення;
- шар контролерів;
- шар сервісів;
- шар доступу до даних.

Шар представлення являє собою клієнтський інтерфейс та функції взаємодії з ним. Він використовує дані повернені шаром контролерів та відправляє запити з даними, які надіслані користувачем.

Шар контролерів відповідає за відповіді на запити користувача з зовнішнього світу. Він приймає дані, приводить їх в формат зрозумілий системі та передає їх шару сервісів. Будь-який контролер повинен мати результат – ним може бути як помилка з HTTP/HTTPS кодом, так і дані, які будуть повернені користувачу.

Шар сервісів представляє з себе реалізацію всієї бізнес логіки у системі. Він працює з даними, змінює їх відповідно заданих параметрів, створює фільтри та запити до шару доступу до даних.

Шар доступу до даних зв’язує систему з базою даних. Він зберігає підключення до системи, приводить запити системи до мови SQL.

Важливою є валідація даних на кожному з шарів. Тільки в такому випадку система буде стійкою до помилок.

6.2 Структура проекту

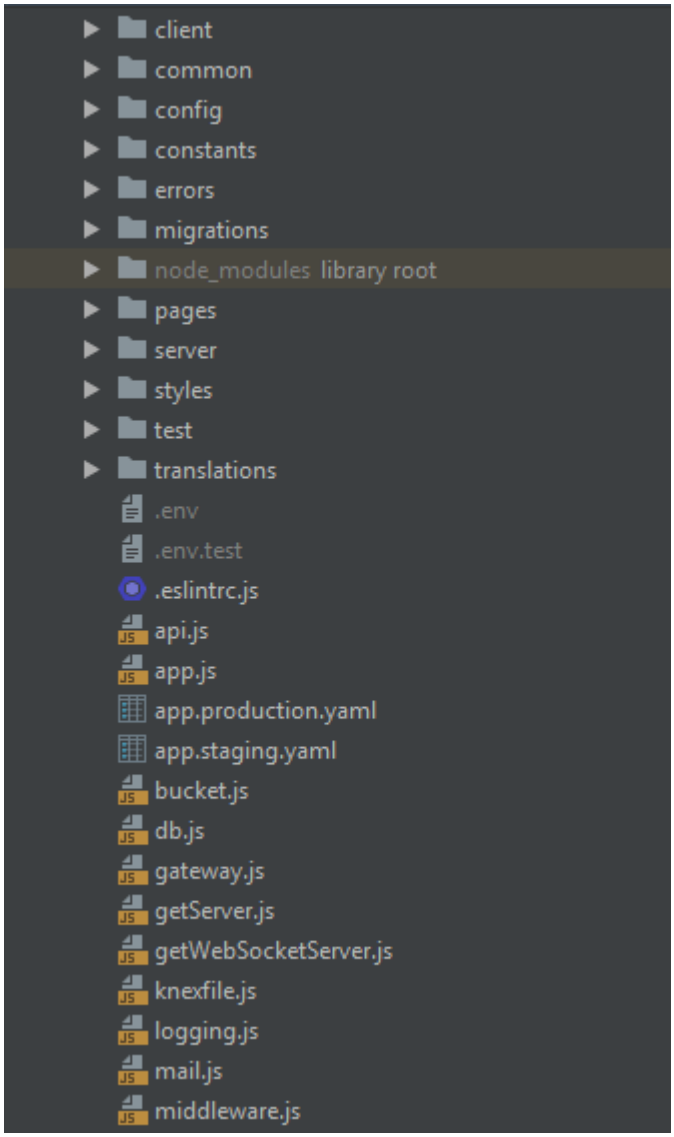

Рисунок 6.1 – структура проекту

Структура даного програмного продукту складається з наступних файлів та модулів:

– client – модуль, який описує зв’язок між всіма компонентами клієнтської сторони;

– common – модуль, який описує загальні функції, які використовуються у проекті;

– config – модуль конфігурацій та налаштування змінних зовнішнього середовища;

– constants – модуль з незмінними значеннями системи;

– errors – модуль, який описує класи помилок у системи;

– migrations – модуль, який описує міграції бази даних;

– node_modules – модуль зовнішніх бібліотек;

– pages – модуль з сторінками, які описані за допомогою React.js;

– server – модуль, який описує серверну сторону;

– styles – модуль, який описує стилі проекту;

– test – модуль тестів проекту;

– translations – модуль для підтримки багатьох мов у системі;

– .env – змінні середовища;

– .env.examle – приклад змінних, які потрібні для налаштування середовища;

– .eslintrc.js – файл, який описує правила редакції коду налаштовуючи Eslint;

– api.js - файл, який налаштовує клієнт до API серверу;

– app.js – головний файл, запускає та конфігурує усю систему;

– app.production.yaml – файл налаштування та підняття готової версії сервісу;

– app.staging.yaml – файл налаштування та підняття тестової версії сервісу;

– bucket.js – файл, який описує сховище для файлів, які будуть загружені у систему користувачами;

– db.js – файл, який описує базу даних та підключення до неї;

– gateway.js – файл, який описує клієнт для роботи з зовнішнім Braintree API, для проведення оплати у системі;

– getServer.js – файл, який запускає сервер системи;

– getWebSocketServer.js – файл, який запускає web сокет сервер системи

– knexfile.js – файл для налаштування бібліотеки knex;

– logging.js – файл для налаштування логування у системі;

– mail.js – файл, який описує клієнт для роботи з зовнішнім Mailjet API;

– middleware.js – файл, який зберігає проміжні шари для серверу.

Даний web-застосунок має складну систему роутингу запитів, так як форматом відповідей може бути як JSON об’єкт, так і згенерована HTML сторінка.

6.2.1 Серверний застосунок

Структура серверного застосунку, який будує обробники запитів для відповідей JSON формату зображена на рисунку 6.2:

– common – модуль, який відповідає за загальні функції, які використовуються на сервері;

– controllers – модуль, який відповідає за контроллери серверу;

– models – модуль, який описує модель серверу та являється абстракцією об’єктів бази даних;

– routes – модуль, який описує шляхи опрацювання запитів;

– services – модуль, який описує всі доступні сервіси на сервері, являється абстракцією усіх операцій незалежно від протоколу HTTP.

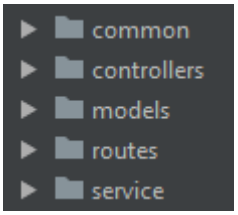

Рисунок 6.2 – структура серверного застосунку

6.2.2 Клієнтський застосунок

Структура застосунку, який будує обробники запитів для відповідей HTML формату зображена на рисунку 6.3:

a) [lang] – описує змінну частину шляху, яка відповідає за вибрану мову інтерфейсу;

б) _app.jsx – описує оболонку клієнтського додатку;

в) _document.jsx – описує документ;

г) _error.jsx – описує сторінку помилки, яка буде повернена, яка обробник помилок не буде знайдено;

д) index.tsx – описує головну сторінку додатку з використанням мови по замовчуванню;

1) accounts – модуль, який описує всі публічні сторінки для пошуку облікових записів;
2) blog – модуль, який описує всі публічні сторінки статей, які доступні в додатку;
3) invitations – модуль, який описує сторінку запрошення для конкретних користувачів при реєстрації;
4) listings – модуль, який описує обробник усі запитів для роботи з сторінками об’яв;
   - sales – модуль, який описує сторінки для роботи з об’явами про продаж нерухомості;
   - rentals – модуль, який описує сторінки для роботи з об’явами про оренду нерухомості;
5) manage – модуль, який описує сторінку керування обліковим записом користувача;
   - profile – модуль, який описує сторінку для роботи з даними користувача;
   - messages – модуль, який описує сторінку для роботи з чатами та повідомленнями користувача;
   - properties – модуль, який описує сторінку для роботи з об’єктами нерухомості, які є власністю користувача;
   - sales – модуль, який описує сторінку для роботи з об’явами про продажу нерухомості;

– rentals – модуль, який описує сторінку для роботи з об’явами про оренду нерухомості;

е) search – модуль, який відповідає за сторінки пошуку по системі;

ж) rentals – модуль, який відповідає за пошук об’яв про оренду;

з) sales – модуль, який відповідає за пошук об’яв про продажу;

и) about.jsx – сторінка, яка описує систему, розказує про що вона та наводить короткі відомості;

к) help.jsx – сторінка, яка містить інформацію про допомогу, контакті дані;

л) index.jsx – описує головну сторінку додатку;

м) sign_in.jsx – описує сторінку авторизації у додаток;

н) sign_up.jsx – описує сторінку реєстрації у додатку.

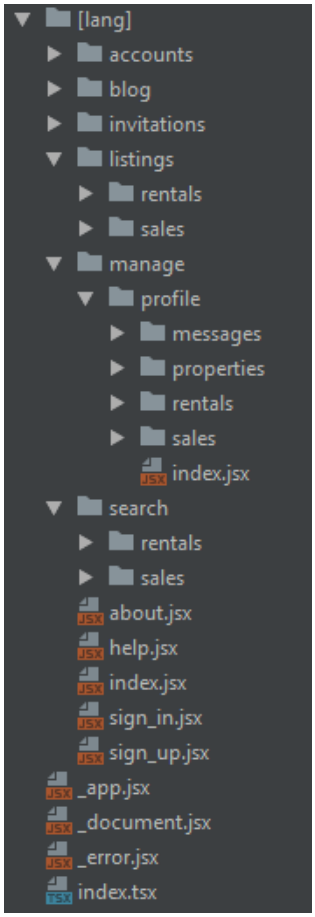

Рисунок 6.3 – структура клієнтського застосунку

6.3 Структура обробки запитів

Дана система має збірну систему опрацювання запитів, які надходять від клієнта. Відношення між програмними модулями є показаними на кресленику ІТ61.160БАК.004 Д2. В загальному застосунок піднімає три різних сервера:

– HTTP/HTTPS;
– Socket;
– PostgresSQL.

HTTP/HTTPS сервер служить з’єзднанням з клієнтом у режимі запит-відповідь, Socket сервер з’єднання допомагає обмінюватись з застосунком у режимі реального часу, а PostgresSQL сервер являється віддаленим місцем зберігання усіх баз даних та таблиць потрібних для опису об’єктів системи. В загальному це все і утворює один сервер з різними портами для підключення та рівнем захисту. Кожен сервер є створений за допомогою різних бібліотек та фреймворків.

Поєднання Next.js, Express.js та Socket.IO працює наступним чином. Серверний додаток очікує на HTTP або WebSocket запит від браузера або іншого клієнта. Отримавши запит, залежно від інформації в ньому про тип зв’язку буде обрано, який обробник почне опрацювання та який тип відповіді буде відправлено користувачу. Якщо даний запит відповідає протоколу WebSocket – то за його відповідь цілком буде відповідати сервер піднятий для опрацювання такого типу запитів – а саме Socket.IO. Відповідь у WebSocket каналі подається у форматі JSON у реальному часі, а отже може змінити контент сторінки користувача навіть без її перезавантаження. Якщо ж запит отриманий сервером відповідає протоколу HTTP – то його відповідь буде опрацьована сервером створеним за допомогою Express.js. В залежності від URL запиту, буде вибрано правильний обробник. У цій системі є два типи відповідей на HTTP запити – JSON об’єкт або згенерована на сервері HTML сторінка. Якщо обробником являється контроллер створений за допомогою Express.js – то відповіддю на

даний запит буде JSON об’єкт та різні допоміжні поля, наприклад зміна cookie браузеру. Якщо ж обробником являється контроллер Next.js – то відповіддю є сторінка згенерована за допомогою Next.js, який будує свою систему обробників залежно від структури директорії в якій находяться всі його сторінки.

6.4 Розробка бази даних

В цьому проекті було вирішено використовувати PostgresSQL, так як вона являється найкращим рішенням на ринку в даний момент. Система була розроблена гнучко і якщо в якийсь момент буде потрібен перехід до іншої СУБД, це може бути зроблено за допомогою зміни провайдера, який використовується у бібліотеці побудови запитів, так як усі міграції бази даних є описаними в коді. Для переносу усіх даних з однієї СУБД до іншої потрібно буде створити ще одну додаткову міграцію. PostgresSQL є реляційною СУБД, а отже діаграма бази даних повинна відповідати усім законам цього виду бази даних.

Діаграму бази даних наведено на креслинику ІТ61.160БАК.004 ДЗ, а опис, призначення та поля її таблиць зазначено у таблиці 6.1

Таблиця 6.1 – призначення таблиць бази даних

Назва відношення	Опис полів та даних
users	Таблиця зберігає дані про користувача з різними рівнями доступу, з наступними полями: ідентифікатор (id uuid pk), адресу пошти (email varchar not null), телефон (phone varchar not null), ім’я (firstName varchar not null), прізвище (lastName varchar not null), URL фото профілю (profileImageUrl varchar), дату народження (birthdate date not null), дату

Продовження таблиці 6.1

Назва відношення	Опис полів та даних
users	останньої авторизації (lastLoginAt datetime), роль (role enum not null), статус (status enum not null)
properties	Таблиця зберігає дані про об’єкти нерухомості з наступними полями: ідентифікатор (id uuid pk), тип (type enum not null), координати (coordinates json not null), адреса (address json not null), житлова площа (livingArea double), площа будівлі (buildingArea double), дані про паркування (parkingData json), рік в якому побудована (yearBuilt int), рік введення в експлуатацію (yearAltered int), кількість ванних кімнат (bathroomsTotal int), кількість спальних кімнат (bedroomsTotal int), дані про кімнати (rooms json), дані про меблі (appliances json), ідентифікатор власника (ownerId uuid not null).
appliances	Таблиця, яка є представленням меблів у системі. Зберігає наступні поля: назву (name varchar pk)
rental_listings	Таблиця, яка зберігає дані про об’яви у сфері оренди нерухомості. На одну нерухомість може бути тільки одна така об’ява, так як власник є тільки один. Зберігає наступні поля:

Продовження таблиці 6.1

Назва відношення	Опис полів та даних
rental_listings	ідентифікатор (uuid pk), дата публікації (publishedAt datetime), дата зміни даних (updatedAt datetime), дата створення (createdAt datetime not null), статус об’яви (status enum), медіа файли (media json), ціна (price double not null), опис (description varchar), посилання на зовнішній ресурс (externalUrl varchar), посилання на головну картинку (photoUrl varchar), ідентифікатор нерухомості (propertyId uuid not null), ідентифікатор користувача (publisherId uuid not null), який створив цю об’яву
sale_listings	Таблиця, яка зберігає дані про об’яви у сфері продажу нерухомості. На одну нерухомість може бути тільки одна така об’ява, так як власник є тільки один. Зберігає наступні поля: ідентифікатор (uuid pk), дата публікації (publishedAt datetime), дата зміни даних (updatedAt datetime), дата створення (createdAt datetime not null), статус об’яви (status enum), медіа файли (media json), ціна (price double not null), опис (description varchar), посилання на зовнішній ресурс (externalUrl varchar), посилання на головну картинку (photoUrl varchar),

Продовження таблиці 6.1

Назва відношення	Опис полів та даних
sale_listings	ідентифікатор нерухомості (propertyId uuid not null), ідентифікатор користувача (publisherId uuid not null), який створив цю об’яву
articles	Таблиця, яка описує всі статті, які можуть бути розміщені у блозі системи. Є добре оптимізованою за допомогою індексів для швидкого пошуку по тегам для підняття системи у пошукових ресурсах. При добавлені таблиці у систему, база даних автоматично працює з створеними індексами. Зберігає наступні поля: ідентифікатор статті (id uuid pk), заголовок (title varchar not null), короткий опис (shortDescription varchar), довгий опис (longDescription varchar), контент (content varchar), зберігає дані форматованого тексту, який при розміщені на сторінці браузера буде збагачений базовою стилізацією, дата публікації (publishedAt datetime), дата зміни даних (updatedAt datetime), дата створення (createdAt datetime not null), статус (status enum), посилання на головне фото (featurePhotoUrl varchar), набір тегів (tags json), мова, якою написана стаття (locale varchar), ідентифікатор користувача (publisherId uuid), який опублікував статтю

Продовження таблиці 6.1

Назва відношення	Опис полів та даних
chats	Таблиця, яка зберігає дані про всі чати створені у системі. Так як кожен чат є створеним з якоюсь ціллю – вона повинна зберігати і її, а отже присутні наступні поля: ідентифікатор (id uuid pk), дата створення (createdAt datetime not null), ідентифікатор об’яви про продаж нерухомості (saleListingId uuid), ідентифікатор об’яви про оренду нерухомості (rentalListingId uuid), ідентифікатор останнього повідомлення (lastMessageId uuid).
users_to_chats	Таблиця, яке реалізує відношення багато до багатьох між відношенням користувачів та відношенням чатів. Зберігає наступні поля: ідентифікатор (id int pk), ідентифікатор чату (chatId uuid), ідентифікатор користувача (userId uuid)
messages	Таблиця, яка зберігає дані про усі повідомлення створені в чатах. Зберігає наступні поля: ідентифікатор (id uuid pk), контент (content json not null), тип (type varchar not null), дата коли прочитане (seenAt datetime), дата коли надіслане (sentAt datetime not null), ідентифікатор чату (chatId uuid), ідентифікатор відправника (senderId uuid)

Продовження таблиці 6.1

Назва відношення	Опис полів та даних
favourite_sale_listings	Таблиця, яка зберігає дані про всі об’яви про продажу нерухомості, які були збережені тим чи іншим користувачем. По факту являється таблицею, яка реалізує зв'язок багато до багатьох між об’явами та користувачами для оптимізації пошуку та підбору схожих об’єктів. Зберігає наступні поля: ідентифікатор (id uuid pk), ідентифікатор об’яви про продажу нерухомості (saleListingId uuid), ідентифікатор користувача (userId uuid)
favourite_rental_listings	Таблиця, яка зберігає дані про всі об’яви про оренду нерухомості, які були збережені тим чи іншим користувачем. По факту являється таблицею, яка реалізує зв'язок багато до багатьох між об’явами та користувачами для оптимізації пошуку та підбору схожих об’єктів. Зберігає наступні поля: ідентифікатор (id uuid pk), ідентифікатор об’яви про продажу нерухомості (rentalListingId uuid), ідентифікатор користувача (userId uuid)

6.5 Висновки до розділу

У цьому розділі було описано усі можливі схеми взаємодії створеної системи з навколишнім світом.

Було описану усю структуру проекту та складну систему оброблення запитів. Для того щоб описати модель даних використану у системі було створено діаграму сутностей у базі даних та описано їх базові поля з типізацією, так як використовується СУБД з мовою запитів SQL.

7 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

7.1 Фази циклу створення програмного забезпечення

Розробка веб-сайту чи програмного забезпечення - складний процес, і неправильний крок на будь-якій стадії розробки програмного забезпечення може призведе до неминучих результатів як для якості продукту, так і для всього бізнесу. Він потребує працьовитості, відданості та досвіду від усіх учасників цього процесу та чіткого дотримання наступних фаз:

- планування;
- проектування;
- кодування;
- впровадження та інтеграція;
- тестування;
- встановлення та обслуговування.

Першу фазу часто називають фазою мозкового штурму, так як фахівці збирають вимоги та аналізують усі аспекти майбутнього програмного продукту. Розробники повинні розуміти вимоги клієнтів, а саме, чого саме вони хочуть і які проблеми можуть виникнути в процесі розробки. Цей етап передбачає спілкування між зацікавленими сторонами, командою проекту та користувачами.

Проектування програмного забезпечення є основним аспектом циклу розробки програмного забезпечення. Воно передбачає загальний дизайн продукту, а також структуру даних. Також на цій стадії планують базові рішення та проектують майбутню структуру файлової системи та відношення у ній.

Кодування – це основна фаза про програмістів. Зазвичай компанія призначає команду програмістів для певного проекту. Завдання розділяються на під-завдання, тому у кожного члена команди є своє незалежна частина. [13]

Під час четвертої фази циклу усі програми зв’язують між собою та провіряють їх роботу у реальному часі. Зазвичай програмне забезпечення

містить велику кількість програм, які потребують ретельної реалізації та покрокової інтеграції програмного продукту. Під час цього етапу розробки програмного забезпечення команда проекту перевіряє, чи працює програмний продукт на різних системах. У разі помилок тестери їх виправляють.

На п’ятому етапі програмне забезпечення направляється у відділ тестування. Робота тестувальників відіграє вирішальну роль для якості програмного забезпечення та його продуктивності. Перед запуском продукт потребує перевірки, яка включає тестування програмного забезпечення та рішення проблем, які знайдені тестерами. Коли відділ тестування переконався, що програмне забезпечення не містить помилок, він переходить до наступного етапу.

На шостому етапі програмне забезпечення передається клієнтам для встановлення на їх пристрої. Після встановлення, якщо клієнту потрібні будь-які модифікації, продукт підлягає процесу технічного обслуговування.

7.2 Тестування програмного забезпечення

В загальному виділяють два основних види тестів:

а) ручне;

б) автоматизоване;

1. unit-тестування;
2. інтеграційне.

Вибір потрібного виду тестів для проекту являється важливою задачею, так як написання коду для тестів може забрати багато часу, а відмова від них може спричинити велику проблему при підтримці продукту і привести до ще більших втратах як у часі так і у коштах. При дотримані загальних правил створений продукт може легко бути підтриманим у майбутньому.

7.2.1 Ручне тестування

Основною ціллю ручного тестування являється виявлення помилок на ранніх та пізніх етапах. При розробці даного програмного забезпечення було використано найбільш розповсюдженіший та дієвий спосіб ручного тестування, а саме – логування (logging). Налаштування даного процесу проходить з самого початку при створенні програмного продукту та налаштовується окремо для різних серверів, так як при піднятті їх на різних портах ми повинні бачити вивід для кожного з них окремо. Логування допомагає на ранніх етапах виявити помилки та виправити їх, на пізніх етапах воно записує дані про помилку отриману в конкретний момент часу, що допоможе у роботі з реальними користувачами та їх підтримкою. В даному проекті було використано декілька рівнів логування для збереження даних про всі помилки у системі та про важливу інформацію у виконанні окремих функцій. [14]

7.2.2 Unit-тестування

Unit-тестування являється одним з підвидів автоматизованого тестування та працює з нереальними даними та зовнішніми клієнтами. Усі дані в даному виді тестування є програмно згенерованими, а робота функцій, які працюють наприклад з базою даних є зімітованими. Такий вид тестування допомагає виявити помилку у алгоритмах проекту, а не у роботі з її зовнішніми ресурсами, що робить пошук помилки більш локалізованим. [15]

7.2.3 Інтеграційне тестування

Інтеграційне тестування являється одним із підвидів автоматизованого тестування. Такий вид тестів працює з реальними даними, які є згенеровані та добавлені у тестову базу даних. Усі зв’язки з зовнішніми ресурсами є

створеними, але з використанням тестових даних – тестових ключів та бази даних. Після кожного тесту, або групи тестів, змінені дані повинні бути повернені до початкової версії, щоб кожен тест міг виконуватись незалежно. При розробці цієї системи було використано даний вид тестів, для збереження часу на їх написання, але такому ж покритті роботи програми.

Кожному тесту повинен відповідати окремий сценарій. Всі опрацьовані сценарії, тестові випадки та їх результати є наведеними у таблицях 7.1-7.15.

Таблиця 7.1 – Тестовий випадок «Перевірка пошуку статей»

Функція	Пошук статей.
Результат функції	– сторінка з статтями відкрита; – на сторінці відображаються статті з бази даних повернені сервером; – результати відповідають наданим фільтрам.
Результат тесту	Пройшов.

Таблиця 7.2 – Тестовий випадок «Перевірка пошуку об’яв про нерухомість»

Функція	Пошук об’яв про нерухомість.
Результат функції	– сторінка з об’явами відкрита; – на сторінці відображаються об’яви з бази даних повернені сервером; – усі результати відповідають наданим фільтрам та сортуються відповідно до них.
Результат тесту	Пройшов.

Таблиця 7.3 – Тестовий випадок «Перевірка реєстрації користувача з правильними даними»

Функція	Реєстрація користувача з правильними даними
Результат функції	– користувач добавлений в базу даних; – користувач авторизований та cookies браузера створені; – користувач перенаправлений на сторінку редакції.
Результат тесту	Пройшов.

Таблиця 7.4 – Тестовий випадок «Перевірка реєстрації користувача з неправильними даними»

Функція	Реєстрація користувача з неправильними даними
Результат функції	– сервер повертає помилку про невірність даних; – користувач отримує повідомлення про невірність даних.
Результат тесту	Пройшов.

Таблиця 7.5 – Тестовий випадок «Перевірка авторизації користувача з правильними даними»

Функція	Авторизація користувача з правильними даними
Результат функції	– користувач авторизований та дані cookies в браузері створені; – дані про останню успішну авторизацію користувача змінені; – користувач перенаправлений на сторінку редакції.
Результат тесту	Пройшов.

Таблиця 7.6 – Тестовий випадок «Перевірка авторизації користувача з неправильними даними»

Функція	Авторизація користувача з неправильними даними
Результат функції	– сервер повертає помилку про невірність даних; – користувач отримує повідомлення про невірність даних.
Результат тесту	Пройшов.

Таблиця 7.7 – Тестовий випадок «Перевірка перегляду сторінки статті»

Функція	Знаходження та перегляд сторінки статті
Результат функції	– сервер знаходить потрібну статтю у базі даних та повертає її; – клієнт отримує дані статті, сторінка генерується та відображає їх.
Результат тесту	Пройшов.

Таблиця 7.8 – Тестовий випадок «Перевірка перегляду сторінки статті з невірно введеними даними»

Функція	Знаходження та перегляд сторінки статті з невірно введеними даними
Результат функції	– сервер повертає помилку, що такої статті немає в базі даних; – клієнт отримує дані про помилку та опрацьовує її залежно від коду помилки.
Результат тесту	Пройшов.

Таблиця 7.9 – Тестовий випадок «Перевірка перегляду сторінки об’яви про нерухомість»

Функція	Знаходження та перегляд сторінки об’яви про нерухомість
Результат функції	– сервер знаходить потрібну об’яву у базі даних та повертає її; – клієнт отримує дані про нерухомість.
Результат тесту	Пройшов.

Таблиця 7.10 – Тестовий випадок «Перевірка перегляду сторінки об’яви про нерухомість з невірно введеними даними»

Функція	Знаходження та перегляд сторінки об’яви про нерухомість з невірно введеними даними
Результат функції	– сервер не знаходить об’яву у базі даних та повертає помилку; – клієнт отримує помилку та опрацьовує її.
Результат тесту	Пройшов.

Таблиця 7.11 – Тестовий випадок «Перевірка зміна даних облікового запису»

Функція	Зміна даних облікового запису
Результат функції	– сервер опрацьовує запит, змінює рядок у базі даних та повертає новий; – клієнт отримує підтвердження, що дані було змінено.
Результат тесту	Пройшов.

Таблиця 7.12 – Тестовий випадок «Перевірка створення об’яви про нерухомість»

Функція	Створення об’яви про нерухомість
Результат функції	– сервер опрацьовує запит та створює новий запис об’яви про нерухомість у базі даних; – клієнт отримує підтвердження, що дані було збережено.
Результат тесту	Пройшов.

Таблиця 7.13 – Тестовий випадок «Перевірка видалення об’яви про нерухомість»

Функція	Видалення об’яви про нерухомість
Результат функції	– сервер опрацьовує запит та видаляє знайдений запис об’яви про нерухомість з базі даних; – клієнт отримує підтвердження, що дані було видалено.
Результат тесту	Пройшов.

Таблиця 7.14 – Тестовий випадок «Перевірка редакції об’яви про нерухомість»

Функція	Редакція об’яви про нерухомість
Результат функції	– сервер опрацьовує запит та редагує знайдений запис об’яви про нерухомість у базі даних; – клієнт отримує підтвердження, що дані було успішно відредаговано.
Результат тесту	Пройшов.

Таблиця 7.15 – Тестовий випадок «Перевірка створення статті»

Функція	Створення статті
Результат функції	– сервер опрацьовує запит та створює новий запис статті у базі даних; – клієнт отримує підтвердження, що статтю було збережено.
Результат тесту	Пройшов.

7.3 Висновки до розділу

У цьому розділі було описано сценарії та результати усіх тестів проведених при розробці системи. При будь-яких змінах у коді ці тести повинні бути проведені знову, щоб уникнути помилок в майбутньому.

8 ВПРОВАДЖЕННЯ ТА ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

8.1 Апаратні вимоги для використання системи

Даний проект може бут поділено на дві частини – серверну та клієнтську. Вимоги до серверної частини повинні бути виконані для реалізації цієї системи та можливості її відображення знаходження пошуковими ресурсами. Вимоги до клієнтської частики повинні бути виконані користувачами, які хочуть переглядати даний ресурс, так як при недостатній кількість пам’яті не всі її елементи можуть бути відображені.

Для розробки системи можуть використовуватись одні системні вимоги, але для підняття системи та впровадження її для роботи з користувачами рекомендується використовувати систему з гарною швидкістю.

Мінімальні системні вимоги до сервера для розробки описані у таблиці 8.1, а рекомендовані – у таблиці 8.2.

Таблиця 8.1 – мінімальні системні вимоги до апаратної частини сервера

Процесор	2-ядерний процесор з тактовою частотою 2 ГГц або краще.
Місце на диску	Залежить від кількості даних, яку хоче зберігати власник. Для початкового запуску потрібно 6 гігабайтів (велику кількість займають початково згенеровані дані для кращої роботи під час розробки)
Оперативна пам’ять	4 гігабайти.
Операційна система	Windows 10 або пізніша, macOS, Linux.

Таблиця 8.2 – рекомендовані системні вимоги до апаратної частини сервера

Процесор	4-ядерний процесор з тактовою частотою 2 ГГц або краще.
Місце на диску	Залежить від кількості даних, яку хоче зберігати власник. Для початкового запуску потрібно 8 гігабайтів
Оперативна пам’ять	8 гігабайти.
Операційна система	Windows 10 або пізніша, macOS, and Linux.

8.2 Інструкція з встановлення для розробки

Для розробки цього проекту потрібно встановити на систему Node.js 10 для підняття серверу та PostgresSQL для підняття серверу бази даних з розширенням Postgis для роботи з даними про координати. При запуску програми потрібно слідувати наступним інструкціям:

– в корневій папці проекту визвати команду «npm install» для встановлення усіх модулів;

– використовуючи файл .env.example створити свій .env файл з всіма змінними навколишнього середовища запрошеними проектом;

– в корневій папці проекту визвати команду «npm run start» для запуску системи;

– в корневій папці проекту визвати команду «npm run start:dev» для запуску системи в режимі розробки;

– в корневій папці проекту визвати команду «npm run test» для запуску наявних тестів.

8.3 Висновки до розділу

Інструкція для використання програми являється важливою складовою для її впровадження так як в подальшому вона може підтримуватись різними розробниками. Тільки за допомогою добре створеної документації можна легко добавити в команду нового члена, який зможе швидко та легко підійняти її для подальшої розробки у себе.

ВИСНОВКИ

Результатом виконання даного проекту є система для торгівлі нерухомістю, яка використовує стандарт RESO.

Спочатку було проаналізовано аналоги локального та світового ринку, виділено їх переваги та недоліки, проведено порівняння, а також визначено основні ідеї та функції, які є використані у цій системі. На базі цих даних було проведено аналіз вимог до системи – виділено чіткі функціональні та нефункціональні вимоги, які були дотримані при розробці. Залежно від вимог було вибрано технології, які можуть рішити поставлені задачі без втрат у швидкості та пам’яті. Система була створена та показала гарні результаті при проходженні тестів. На завершення було описано апаратні вимоги та описано хід реалізації системи.

Розроблене програмне забезпечення є готовим до використання та може бути розширене у будь-який момент, так як переймає досвід аналогів та використовує стандарт RESO.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mozilla. What is Javascript in 2020. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
2. Dan Rahmel. Advanced NodeJs! (Expert's Voice in Web Development) London, 2013. 412 p.
3. OpenJs. ExpressJs technology for api. New York, 2018 URL: https://www.guru99.com/node-js-express.html
4. Vercel. Rendering SSP pages using NodeJs. URL: https://nextjs.org/
5. Tom Holend. Knex as new query builder! (SQL Development). 2013 200 p.
6. Facebook Inc. Tutorial and first look on ReactJs. California 2018. URL: https://ru.reactjs.org/tutorial/tutorial.html
7. Open Collective. The best way to use web socket inside Node. 2015. URL: https://www.tutorialspoint.com/socket.io/index.htm
8. Patrick Hlaupe. Create your great theme with Bootstrap. 2015. URL: https://getbootstrap.com/
9. Yannick Croissant. Test your application with Javascript. Poland Varshaw 2016. URL: https://www.testim.io/blog/getesting-with-mocha-and-chai/
10. John Chasu. Backdone.js (Javascript unit testing). 2015. 220 p.
11. Tom Lane. The developers of postgresSQL. URL: https://wiki.postgresql.org/wiki/So,_you_want_to_be_a_developer%3F
12. Tony Smart. Google Cloud Platform (Tutorial for work with GCP). 2017, 321 p.
13. Основи теорії кодування. URL: https://studfile.net/preview/935560/
14. Manual testing for beginners. URL: https://studfile.net/preview/935560/
15. 8 benefits of unit testing. URL: https://dzone.com/articles/top-8-benefits-of-unit-testing