Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи управління каталогом нерухомості

(тема)

Виконав:
студент II курсу, групи СПм-19-1
Касапов В.О.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: ст. викл. Носик А.М
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Касапову Владиславу Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи управління каталогом нерухомості

затверджена наказом по університету від " 30 " жовтня 2020 р. № 1486 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи 1) платформа реалізації серверної частини додатку .NET
2) Платформа реалізації клієнтської частини додатку Angular 10
3) СУБД MSSQL

4. Перелік питань, що потрібно опрацювати в роботі
1) аналіз систем управління каталогом нерухомості
2) моделювання системи та вибір програмного забезпечення
3) побудова моделі веб-сервісу
4) розробка веб-сервісу для доступу к каталогам нерухомості

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційні матеріали. Слайди – 9 арк. Ф. А4

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз системи управління каталогом	03.11.20-09.11.20	
2	Моделювання системи та вибір	10.11.20-17.11.20	
	програмного забезпечення		
3	Побудова моделі веб-сервісу	18.11.20-23.11.20	
	та форми навчального процесу		
4	Розробка веб-сервісу для учасників	24.11.20-01.12.20	
	віддалених навчальних курсів		
5	Оформлення результатів атестаційної роботи	02.12.20-07.12.20	
6	Подання атестаційної роботи керівникові та	08.12.20-09.12.20	
	її попередній захист		

Дата видачі завдання 02 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Носик А.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 73 с., 20 рис., 3 табл., 1 додаток, 13 джерел.

ВЕБ-СЕРВІС, ІНТЕРНЕТ, УПРАВЛІННЯ КАТАЛОГОМ НЕРУХОМОСТІ, МОДЕЛЬ, СЕРВЕР, ІНФОРМАЦІЙНА СИСТЕМА, HTML, C#, ANGULAR, CMS.

Метою атестаційної роботи є аналіз сучасних методів і систем управління каталогом нерухомості та розробка веб-сервісу для доступу до каталогів нерухомості.

У ході виконання атестаційної роботи було проаналізовано вимоги до методів і систем управління каталогом нерухомості, сучасні технології та засоби для реалазції веб-застосунку системи.

Наукова новизна результатів роботи полягає в доповненнях методів проектування та автоматизації систем управління каталогом інфораційних об’єктів у вигляді веб-сервісу. Покращено методи розробки архітектури бази даних для більш швидкого розширення інформаційних об’єктів системи.

Практичне значення одержаних результатів полягає в запропонованих технологічних рішеннях удосконалення процесів побудови інформаційних систем, які можут бути використані розробниками програмного забезпечення при розробці веб-застосунків та при моделюванні систем.

ABSTRACT

Master’s thesis: 73 pages, 20 figures, 3 tables, 1 appendices, 13 sources.

WEB SERVICE, INTERNET, REAL ESTATE CATALOG MANAGEMENT, MODEL, SERVER, INFORMATION SYSTEM, HTML, C #, ANGULAR, CMS.

The purpose of the certification work is the analysis of modern technologies of grouping and search of real estate and the development of a web service for access to real estate catalogs.

In the course of the attestation work, the requirements for real estate catalog management systems, modern technologies and tools for implementing the system's web application were analyzed.

The scientific novelty of the obtained research results is the development of theoretical and methodological and practical recommendations for improving complex information systems, namely methods of developing flexible and easy to expand systems that will partially reduce the time to supplement the information objects of the system.

The practical significance of the obtained results lies in the proposed technological solutions to improve the processes of building information systems, which can be used by software developers in the development of web applications and system modeling.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

ІС – інформаційна система

СУБД – система управління базами даних

Фреймворк – інфраструктура програмних рішень, що полегшує розробку складних систем.

Angular (зазвичай так називають фреймворк Angular 2 або Angular 2+, тобто вищі версії) – написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET;

CSS – Cascading Style Sheets – спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних

DOM – це незалежний від платформи і мови програмний інтерфейс, що дозволяє програмам і скриптам отримати доступ до вмісту HTML-документів

HTML – Hyper Text Markup Language – мова розмітки гіпертекстових документів, яка є стандартом для використання у Інтернеті;

MVC – модель-вигляд-контролер (англ., Model-view-controller).

SQL – Structured Query Language – декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних

SPA – Single Page Application – підхід проектування веб-сайтів, який функціонує на основі єдиного документу розмітки та використовує AJAX-запити для завантаження контенту;

ВСТУП

Сучасні тенденції демонструють, що набуває популярність автоматизація усіх бізнесс процессів, один с таких процессів це швидкий доступ до каталогу певних об’єктів для подальшої роботи с ними, що суттэво зменшить час затрачених на пошук інформації про цей об’єкт.

На фоні великого ринку нерухомості з’явилася потреба в автоматизації пошуку та групування нерухомості для подальшого її продажу, оцінки, оновлення та формування певних юридичних документів зв’язаних з нею.

Одним з найбільш ефективних методів швидкого доступу до таких каталогів це веб-застосунок за допомогою якого користувач матиме швидкий доступ до набору нерухомості та будь-якій інформації про нерухомість

Метою атестаційної роботи є аналіз сучасних методів і систем управління каталогом нерухомості та розробка веб-сервісу для доступу до каталогів нерухомості.Об’єкт дослідження – процес управління катогом нерухомості. Предмет досліджэня є методи та засоби розробки веб-сервісу для реалізації системи управління каталогом нерухомості .

Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналіз вимог системи управління каталогом нерухомості;
- змоделювати систему та вибрати оптимальне програмне забезпечення;
- розробити та побудувати модель веб-сервісу;
- побудувати архітектуру бази даних;
- розробити веб-сервіс.

Наукова новизна результатів роботи полягає в доповненнях принципів проектування та автоматизації систем управління каталогом інфораційних об’єктів у вигляді веб-сервісу. Покращено методи розробки архітектури бази даних для більш швидкого розширення інформаційних об’єктів системи.

Практичне значення отриманих результатів у створенні веб-застосунку системи управління каталогом нерухомості. Результати розробленої

технології створення веб-додатків для управління каталогом інформаційних об’єків можут бути використані розробниками програмного забезпечення, як методичні рекомендації для розробки проектів відповідного спрямування.

Результати роботи опубліковані на міжнародних науково-технічних конференціях.

1 РОЛЬ СИСТЕМИ УПРАВЛІННЯ КАТАЛОГОМ НЕРУХОМОСТІ В СУЧАСНОМУ СВІТІ

1.1 Основні поняття та особисті риси системи управління каталогом нерухомості

Каталогізація нерухомості – це роботи по однаковому поданням, збиранню, класифікації, ідентифікації, кодування, реєстрації, обробки, зберігання і розподілу інформації про продукції, що випускається..

Нерухомість має тенденцію до безперервної зміни певної інформації, що приводить до динамічної зміни критеріїв пошуку та класифікації. Ефективна робота системи каталогізації не може спиратися тільки на емпіричний досвід і інтуїцію розробників, вона потребує системного підходу до процесу, побудови ефективних зв’язків між підсистемами та компонентами системи, розробку швидкої та надійної бази даних.

При проектуванні системи управління каталогом нерухомості з метою формування найбільш повного уявлення про систему необхідно розглядати ряд моделей не тільки з погляду поточних вимог і можливостей, але і перспектив розвитку. Можливі наступні проекції складних систем: структурна, функціональна, інформаційно-технологічна та еволюційна (рис. 1.1).

Функціональна модель розбиває складну систему на компоненти за функціональними ознаками. Наприклад: пошук нерухомості, списки нерухомості, карта. Більшість комерційних пакетів управління каталогом нерухомості розробляється саме на основі функціональних моделей.

Інформаційно-технологічна модель може відноситися як до визначення потоків інформації в системі, так і до технологій, що використовуються у процесі роботи з нерухомістю і технічній побудові системи [1].

Еволюційна модель відбиває розвиток системи в часі, а також перспективи розвитку системи в майбутньому.

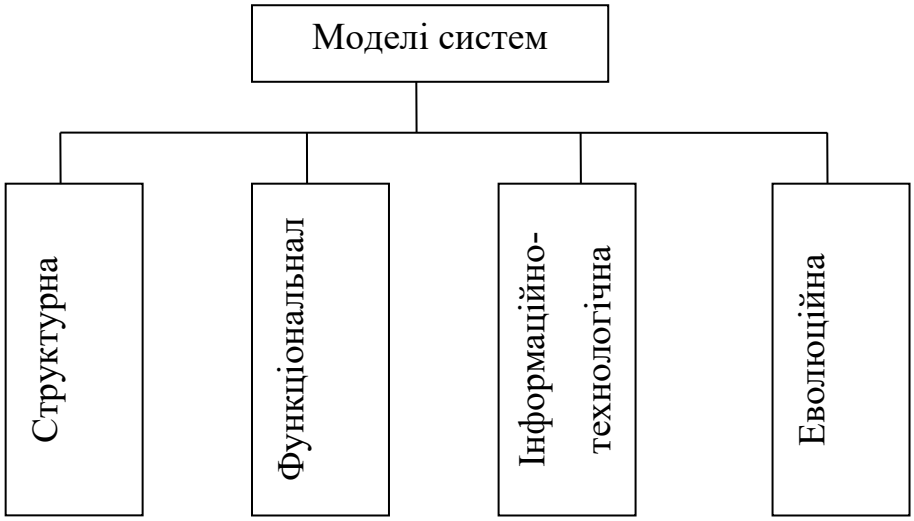

Рисунок 1.1 – Основні типи моделей складних систем

Система каталогізації нерухомості, ґрунтуючись на загальній теорії систем, може також інтерпретуватися як відкрита система. Цей факт має прямий вплив на всі проекції системи. Наприклад, стосовно інформаційно-технологічної проекції системи каталогізації нерухомості це буде означати, що система повинна мати можливість самоадаптації до змін зовнішнього середовища.

Системи каталогізації нерухомості бувають різного ступеня складності і різної орієнтації. Ієрархію системи каталогізації нерухомості можна досить умовно представити у вигляді піраміди з різними рівнями (рис. 1.2).

На першому рівні піраміди знаходяться засоби розробки каталогу нерухомості що забезпечують можливість розробки каталогу нерухомості на основі спеціалізованих середовищ програмування

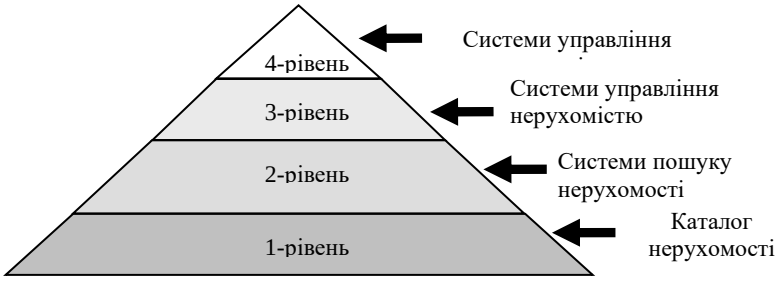

Рисунок 1.2 – Класифікація систем управління каталогом нерухомості

На другому рівні розташовуються системи пошуку нерухомості, які дозволяють шукати нерухомість за певними ознаками та характеристиками. Такі системи являють собою спеціалізовані бази даних (БД) разом із механізмами пошуку по ключових словах, агрегування інформації про нерухомість, документообігу та ін.

На третьому рівні знаходяться системи управління нерухомістю, що дозволяють керувати інформацією про нерухомість: редагування нерухомості, реєструвати користувачів і їх права доступу, на основі прав доступу користувача редагувати інформацію про нерухомість, збирати і зберігати інформацію про дії користувачів щодо редагування інформацію нерухомості.

На верхньому рівні піраміди розташовуються системи управління нерухомістю та документообігом, що поєднують у собі систему керування процесом каталогізації нерухомості та редагування інформації і систему керування документами зв’язаних з нерухомістю. Такі системи використовуються для швидкого створення, розгортання та управління більшою частиною контенту зв’язаною з нерухомістю. Як правило, у сучасних системах під контентом розуміють документи та загальну інформацію нерухомості. Об’єктний підхід та ідея інформаційного об’єкту є ключовими для розбудови сучасної системи роботи с нерухомістю . За цим підходом система поділяється на окремі самостійні модулі, фрагменти, теми й дрібніші дидактичні одиниці, кожна з яких має нести певний обсяг інформації і виконувати свою функцію у процесі управління нерухомістю. Умовно таку дидактичну одиницю саме і називають інформаційним об’єктом.

У процесі каталогізації для кожного об’єкта формується індивідуальна послідовність інформаційних об’єктів. Набір інформаційних об’єктів залежить від типу, району та власників нерухомості. Передбачається можливість редагування інформації про нерухомість та створення інформаційних блоків таких як оцінка вартості, пільги та скарги.

База даних електронного каталогу нерухомості (далі – електронний каталог) містить абстрактні моделі реального світу і подає їх як визначену систему класифікації об’єктів та явищ. В електронному каталозі забезпечено однозначну інтерпретацію абстрактних моделей комп’ютерними системами та їх користувачами, створено умови для розподіленого вироблення, поширення та використання даних. Електронний каталог є списком нерухомості класифікованим за певною інформацією та вказаних на мапі

Сучасне формування та використання даних о нерухомості потребує чіткого порядку класифікації об’єктів, їх властивостей і відношень. Один із засобів вирішення цієї проблеми – формування відповідних каталогів та класифікаторів об’єктів і явищ як необхідних складових метаданих для наборів даних нерухомості. Застосування єдиної структури, яка може використовуватися для пошуку певних категорій нерухомості, як у каталозі так і на мапі вирішує цю проблему [2].

Каталогізація об’єктів та їх властивостей має на меті систематизацію інформації для різних предметних сфер застосування даних на основі єдиної структури, єдиних правил і вимог до системи кодування, опису об’єктів, їх властивостей, операцій з об’єктами і просторових відношень між об’єктами.

У електронних каталогах визначені типи, операції з ними, характеристики та зв’язки, що подаються у вигляди, достатніх для переробки даних в придатну для використання користувачем інформацію. Такий каталог - це інструмент, який можна використовувати для редагування, розширення та перегляду інформації в зручному вигляді.

Електронний каталог призначений для уніфікації опису структури і складу наборів даних з метою підвищення якості й ефективності використання даних на всіх етапах створення, супроводження та використання баз даних у системі управління каталогом нерухомості

Структура каталогу, склад об'єктів та типи даних для інформації об'єкту регламентуються у період збирання даних, створення каталогу та вибору напряму діяльності користувачів цього каталогу. На етапі

формування баз даних нерухомості каталог фактично задає концептуальну модель наборів даних, оскільки містить вичерпну інформацію про склад атрибутів, типи даних та можливі значень, а також про атрибутивні відношення, що задають зв’язки між об’єктами різних типів [3].

Електронний каталог має важливе значення для оцінювання якості наборів геопросторових даних, адже він становить інформаційну базу для програм тестування відповідності наборів визначеним у каталозі системам класифікації та кодування даних і доменам значень атрибутів.

1.2 Галузі застосування систем управління каталогом нерухомості

З базами нерухомості працюють велика кількість компаній та державних установ. Тому назвати всі галузі застосування системи досить складно, головні з них:

- оцінка вартості нерухомості;
- рієлторська діяльність;
- продаж нерухомості;
- реєстрація нерухомого майна ;
- містобудівна інспекція;
- оподаткування нерухомості;
- надання житлово-комунальних послуг;
- проведення будівельних робіт;
- облік прав на нерухоме майно;

Значення наукових і технічних проблем систему управління каталогом нерухомості полягає в забезпеченні інформацією, контролі й підтримці прийняття управлінських рішень у сферах планування та проектування, автоматизації процесів потребуючих взаємодії великого обсягу взаємопов’язаної інформації.

Класифікація та каталогізація інформації важлива як для аналізу існуючого в системі інформаційного обміну, так і для організації його в

системі, що створюється. За допомогою класифікації можна виявити об’єкти, де надмір або недостатньо інформації.

На сьогоднішній день при щодня зростаючих обсягах збереженої інформації про нерухомість різних типів користувачі відчувають труднощі при підборі необхідних їм даних.

Навіть в умовах однієї великої організації пошук інформації про нерухомість ускладнюється тим, що вона зберігається на різних носіях. Крім того, організації часто не мають узагальненого каталогу архівних даних зможливістю зручного пошуку і попереднього перегляду інформації про нерухомість без необхідність звернення до архівного носія інформації. Якщо розглядати цю проблему на регіональному рівні, то вона посилюється територіальної віддаленістю власників інформації, частою необізнаністю зацікавлених організацій про вже наявну інформацію в організаціях цього регіону і труднощами підтримки даних в актуальному стані на поточний момент часу.

1.3 Постановка задач

При правильному підході до розробки будь-яких систем безпосередньо перед моделюванням потрібно виділити основні вимоги до проекту. У випадку з розробкою системи управління каталогом нерухомості необхідно зрозуміти, які завдання ставляться перед системою, яким чином інтегрувати її в процес управління і які ресурси і можливі витрати на проектування і підтримку. Але перш за все, важливо проаналізувати стан даних, на базі яких будується система, а саме виділити вихідні дані, необхідні для розробки та кінцеві можливості продукту.

Розібравши сучасні потреби сервісів та бізнесу який використовує каталоги нерухомості, можна зробити висновок, що потребуються такий функціонал системи, показаний на (рис. 1.3).

Користувач має можливість отримати доступ до захищеної бази

нерухомості. Він має доступ до даних на мапі, створити свій список нерухомості використовуючи базу нерухомості, має можливість редагувати інформацію о нерухомості та використовувати систему для пошуку інформації про будь-який об’єкт нерухомості що міститься у базі. Важливим завданням розробника є організація функціоналу системи саме таким чином.

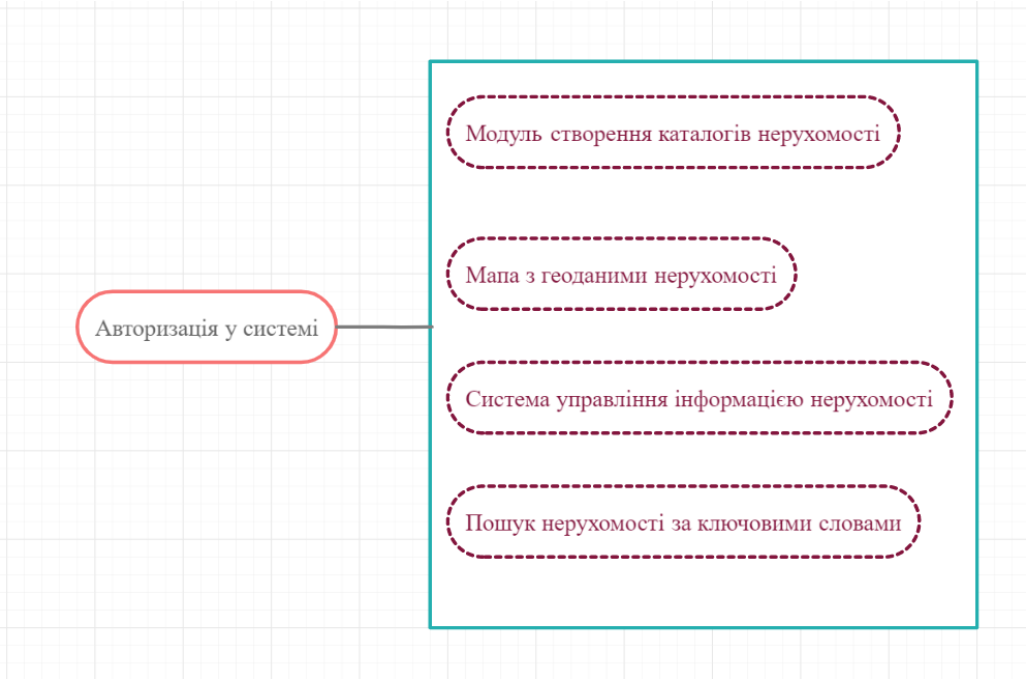

Рисунок 1.3 – Функціонал системи каталогізації нерухомості

На сьогоднішній день будь-який бізнес нерухомості має досить велику базу нерухомості. Використання цієї бази у системі каталогізації нерухомості значно полегшить бізнес процеси та надасть усім користувачам швидкий та легкий доступ до всієї потрібної інформації.

Виходячи з цього, можна припустити, що завданням системи є надання доступу до інформації нерухомості та функціоналу системи і забезпечення захисту від розповсюдження конфіденціальної інформації.

Для великих компаній, які займаються нерухомістю завданням модернізації системи управління нерухомості є забезпечення доступності до системи та розподіл функціоналу між працівниками з різним рівнем доступу.

Використання Інтернету або локальних мереж при взаємодії з працівниками, звичайно ж, очевидно. Але не варто забувати про безпеку переданих даних. При цьому питання безпеки носять чільний характер, ніж доступність джерела інформації. Так само потрібно пам'ятати про те, що деяка інформація має конфіденціальний характер. Інакше система не матиме сенсу і не зможе конкурувати з традиційними методами.

Процес створення та заповнення інформації системи управління каталогу нерухомості вимагає знань мов програмування. Ці знання можуть бути недоступні для користувачів. Тому розробник ядра системи повинен застосовувати більш складні технічні засоби програмування, створюючи додатковий функціонал, який допоможе перенести існуючу базу даних нерухомості у систему з правильною структурою даних. Створення такого механізму переносу даних, надасть власнику системи можливість долучити до систему більшу кількість користувачів, які мають власну базу нерухомості.

З ростом попиту до використання електронних каталогів та удосконалення сучасних технологій зв'язку можливі ситуації, коли вимоги до функціонального стану системи підвищаться. При цьому потрібно буде додавати нові функції і можливості. Це так само слід врахувати при побудові.

Найголовнішою вимогою до системи управління каталогом нерухомості є те, що процес управління нерухомості повинен якомога менше відрізнятися від традиційного процесу.

В результаті проведеного аналізу виявлено основні вимоги до системи управління каталогом нерухомості:

- ідеологія і методи управлення нерухомості не повинні відрізнятися від традиційних;
- система повинна надавати доступ до інформації нерухомості;
- інтерфейс системи повинен бути доступний будь-якому користувачеві;
- система повинна бути стійка при роботі і безпечна;
- робота в системі повинна бути зручна для використання.

2 МОДЕЛЮВАННЯ СИСТЕМИ ТА ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Загальні вимоги

Для веб-сервісу системи управління та каталогізації нерухомості буде використовуватися сучасна система, з метою підвищення якості доступу до інформації, а так само для організації та каталогізації нерухомості, редагування інформації.

Реалізація повинна являти собою програмний продукт, що дозволяє створити середовище для доступу к каталогам нерухомості. Взаємодія з системою повинно здійснюватися через глобальну мережу з використанням протоколів прикладного рівня.

Система повинна відповідати наступним вимогам [4]:

- доступність: здатність надавати доступ до інформації з точки віддаленого доступу;
- адаптованість: здатність адаптувати інформацію нерухомості відповідно до індивідуальних потреб користувачів;
- ефективність: здатність збільшувати ефективність і продуктивність, скорочуючи час і витрати на отримання інформації;
- довговічність: здатність відповідати новим технологіям без додаткової і дорогої доопрацювання;
- можливість багаторазового використання: здатність використовувати систему в різних контекстах.

Проведений аналіз дав змогу сформулювати наступні вимоги до створюваної системи управління каталогом нерухомості:

- користувачі системи повинні бути розділені за зобов'язаннями і мати певні права доступу до елементів системи. Рекомендується створити різні види користувачів за напрямком їх діяльності, наприклад: адміністраторів

системи, спеціаліст з оцінки нерухомості, спеціаліст з продажу нерухомості і так далі;

- адміністратори повинні мати можливість редагувати всю введену в систему інформацію, мати доступ до всіх модулів, а так само додавати користувачів в систему;

- для інших користувачів повинна бути можливість обмежити доступ до будь-якого модуля;

- в системі необхідно організувати механізми редагування матеріалів зв’язаних з нерухомістю, способи їх опису, пошуку та каталогізації. Всі введені дані повинні бути чітко структуровані і представлятися у вигляді певного інформаційного об’єкту;

- програмну реалізацію потрібно створити на мовах HTML, C#, SQL, TYPESCRIPT та застосувати фреймворк Angular;

- модулі програм повинні бути зрозумілими - досить короткими для простоти, але занадто маленькі модулі породжують занадто багато гілок в структурі дерева, і ієрархія стає сама по собі важкою для розуміння. Обов'язкове використання класів, функцій і процедур, замість повторення коду;

Бази даних необхідно реалізовувати з використанням серверу Microsoft SQL.

Права користувачів повинні бути організовані в контексті бази даних, а так само файлової системи. Для кожного курсу рекомендується організувати власну директорію для зберігання файлів, завантажених розробником [14].

2.2 Архітектура веб-застосунків

Згідно з вимогами нових технологій інформаційна система повинна відповідати останнім технічним досягненням і для системи повинна бути розроблена певна архітектура [5]. Нині використовується триланкова (Three - Tier) архітектура веб-застосунків (рис. 2.1).

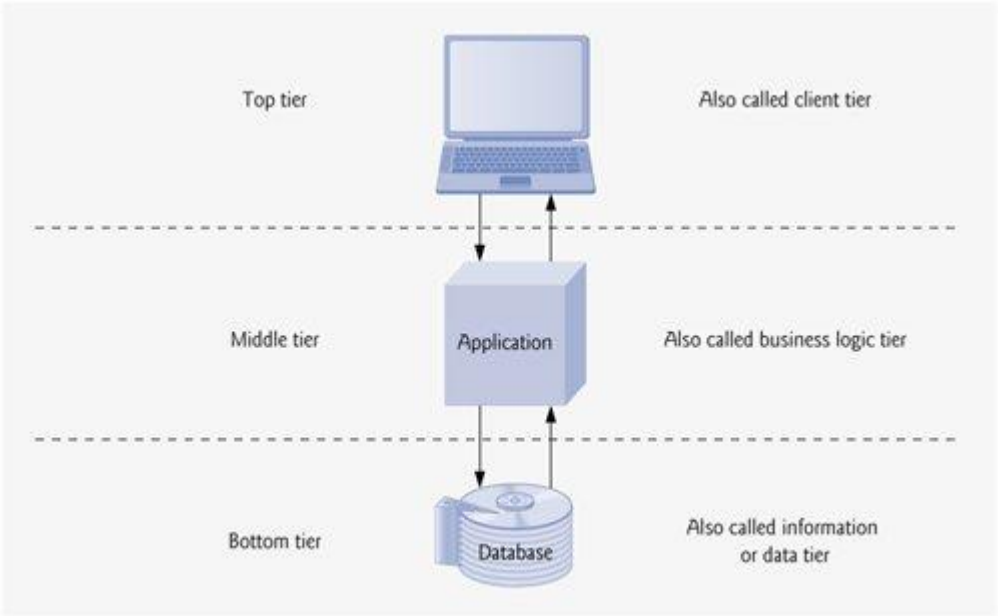

Рисунок 2.1 – Триланкова архітектура веб-застосунків

В сучасній триланковій архітектурі веб-сервер – сервер застосунків – сервер БД використовують різні комбінації платформ, розглянемо декілька основних:

а) web -сервер Apache, сервер застосунків PHP, сервер БД MySQL;

б) веб-сервер Microsoft Information Services, сервер застосунків .Net Core Framework, сервер БД Microsoft SQL.

Таким чином, вибір програмного забезпечення для вирішення поставленого завдання зводиться до вибору однієї з двох вказаних платформ.

Істотним недоліком такої архітектури є те, що клієнту доводиться чекати завантаження кожної наступної сторінки.

Для усунення цього недоліка можна використовувати принципи SPA (Single Page Application). Особливість архітектури SPA полягає в тому, що всі елементи, необхідні для роботи веб-застосунку знаходяться на одній сторінці. Також даний вид додатків завантажує додаткові модулі після запиту від користувача

При завантаженні нових модулів в SPA контент на них оновлюється тільки частково, так як немає необхідності повторно завантажувати незмінні

елементи [6]. Це збільшує швидкість відповіді і скорочує передається обсяг даних між браузером і сервером. Даний вид софта за способом взаємодії з користувачем найбільше схожий на роботу десктопних додатків, але на сервері.

Переваги Single Page Applications:

- доступність. Можна отримати моментальний доступ до функціонала з будь-якого типу пристрою без проблем з сумісністю, об'ємом пам'яті, потужностями або часом на установку;
- універсальність. Використовувати застосунок можна практично з будь-якого пристрою, якщо на ньому є доступ до інтернету;
- швидкість. Одна сторінка зі всім необхідним не тільки економить час на повторну завантаження даних, а й підвищує продуктивність роботи.

Недоліки Single Page Applications:

- труднощі з SEO. Особливості SPA ускладнюють або унеможливлюють процес індексації пошуковими системами всіх модулів програми. Це може викликати труднощі з оптимізацією;
- навантаження на браузер. Через те, що клієнтські фреймворки важкі, вони досить довго завантажуються.

2.3 Вибір сучасних засобів для створення інтерфейсу

HTML (від англ. HyperText Markup Language) - мова гіпертекстової розмітки. Мова служить для створення веб-сторінок. Він інтерпретується браузером і відображається у вигляді документа у зрозумілій для людини формі.

HTML - це невід'ємна складова і основа практично будь-який веб-сторінки. В першу чергу виступає як засіб логічної розмітки сторінки. Заповнення вмісту всередині сторінки реалізується за допомогою так званих дескрипторів. Дескриптори також часто називають «тегами». Теги - це спеціальні маркери, які певним чином інтерпретуються браузером. Суть тегів

в тому, що вміст сторінки, укладену в різні теги, по-різному обробляється браузером. Наприклад, ми можемо зробити висновок контент сторінки в тег списку, таблиці або параграфа, і дане вміст буде вважатися браузером так, який тег ми вибрали. В результаті розмітка за допомогою тегів надає вмісту якийсь сенс. Мова HTML має досить велику історію розвитку, і за цей час зазнав значних змін. Велика частина змін пов'язана з додаванням в мову нових тегів і припиненням використання застарілих.

HTML - це всього лише розмітка абсолютно нечитабельним документа. Однак потрібно відзначити те, що засобами мови ми все ж можемо управляти, як зовнішнім виглядом, так і логічною структурою. Але такий підхід вважається неправильним і застарілим, тому що настроїти вигляд існує окрема мова.

CSS (від англ. Cascading Style Sheets) - каскадні таблиці стилів. Це мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки. Мета створення CSS це розмежувати структурну частину від зовнішнього вигляду. Це служить для того, щоб гнучко управляти зовнішнім виглядом документа і мінімізувати обсяг повторюваного коду.

За допомогою мови CSS, ми можемо надавати стилі таким типам документів як: HTML, XHTML, SVG, XUL. Ми будемо працювати переважно з веб-сторінками, тому використовуємо мову розмітки HTML. Для того, щоб почати використовувати CSS, потрібно цей документ якось пов'язати зі стилями, тобто повідомити HTML-документу, що він буде оформлений за допомогою CSS. Підключення може відбуватися різними способами, в залежності від поставленого завдання. Таблиці стилів можуть розташовуватися як всередині самого HTML документа, так і зовні окремим файлом з розширенням css. Важливо розуміти, що CSS-файл - це текстовий файл з набором інструкцій, що описують зовнішній вигляд документа. Якщо таблиця стилів знаходиться в окремому файлі, то вона підключається до документа за допомогою спеціального тега link, який повинен розташовуватися в цьому документі всередині тега head. Найчастіше ми саме

так підключаємо стилі. При цьому в атрибуті href вказується шлях до підключається файлу стилів. Кожне правило складається з селектора і блоку оголошень. Задавати стилі можна як конкретним селекторам, так і створювати класи та ідентифікатори в файлі HTML і через зовнішній файл CSS стилізувати документ.

TypeScript - це мова на основі JavaScript. TypeScript додає до JavaScript необов'язкові типи, які підтримують інструменти для масштабних програм JavaScript для будь-якого браузера, для будь-якого хосту та будь-якої ОС. TypeScript компілюється у зручний для читання JavaScript на основі стандартів. Головним завданням є розширення можливостей веб-документа через маніпулювання DOM-елементами на веб-сторінці. DOM - це об'єктна модель документа (від англ. Document Object Model). Це деревоподібний вигляд документа, тобто вкладені теги всередині іншого стають дочірніми вузлами. За допомогою JavaScript ми можемо отримувати вузли в DOM, змінювати, видаляти, додавати і навіть змінювати зв'язки між ними. Рівно як і з CSS ми підключаємо JavaScript через головний документ, через тег script. Можна підключати зовнішній файл в тезі script через атрибут src, або писати код всередині самого тега.

Для реалізації веб-застосунку було вирішено використовувати зовнішній файл скриптів. JavaScript застосовується для створення різних анімацій, меню, що випадає або слайдера, суть яких полягає в тому, щоб в певному обмеженому просторі відбувалося чергування різних об'єктів. Часто JavaScript використовується і для первинної перевірки даних, які користувач вводить в форми. Існує велика кількість бібліотек та фреймворків. Щоб спростити код ми будемо використовувати фреймворк Angular 10. Одне з головних переваг бібліотек та фреймворків це крос-браузерні інтерфейс до методів DOM. До того ж фреймворки позбавляють веб-розробників від необхідності вивчати в деталях сам JavaScript та TypeScript, надаючи ряд зручних у використанні інструментів, що дозволяють з легкістю управляти об'єктною моделлю документа [7-8].

2.4 Вибір веб-сервера

Apache та Nginx - два найбільш рейтингових сервери згідно даних Інтернет станом на грудень 2020 року.

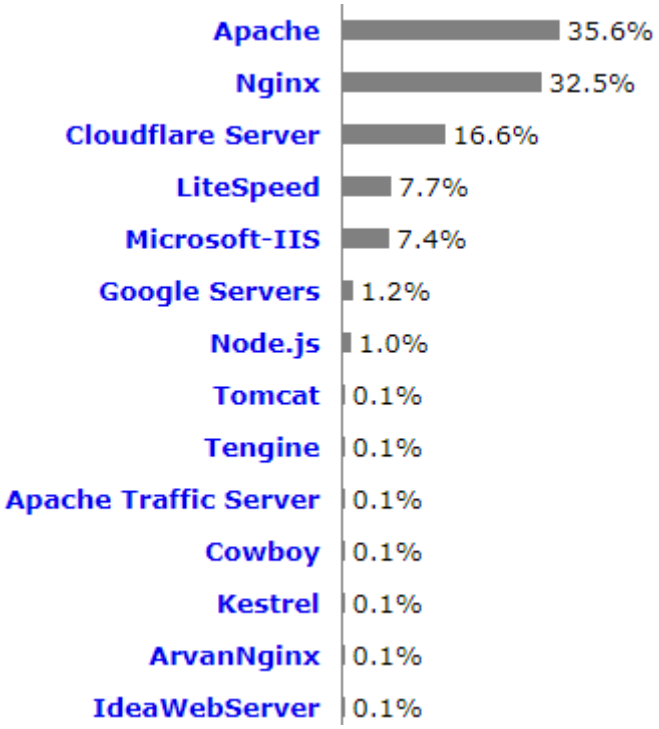

Рисунок 2.2 – Рейтинг серверів

Судячи з наведених даних, 35,6% світових серверів припадає на частку Apache, 32,5% – на долю nginx, 7,4% – IIS, та інші [9].

При аналізі платформи IIS – C#(.NET) - Microsoft SQL було виявлено такі переваги:

- відкритий початковий код;
- велика поширеність, що спрощує пошук хостингу з прийнятними умовами;
- низька вартість володіння (як розробки застосунків, так і експлуатації систем);
- висока продуктивність і доступність;

- простота адміністрування і управління – шляхом правки параметрів в текстовому конфігураційному файлі;
- добре документування, що істотно спрощує розробку веб-застосунків;
- добра підтримка об'єктно-орієнтованого підходу.

Зважаючи на виявлені переваги для реалізації системи було вибрано веб-сервер IIS.

2.5 Вибір сервера застосунків

При використанні веб-сервера IIS можливе використання середовищ на мові C#.

C# надійна та швидка мова програмування. Один і той самий C#-код за допомогою платформи .Net Core може працювати в незмінному вигляді на різних веб-серверах і операційних системах. Мова постійно поліпшується та динамічно розвивається.

Фреймворк .Net Core має відкритий початковий код, таким чином э можливість проаналізувати недоліки та мінімізувати їх при розробці веб застосунка.

Мова C# працює в середовищах Unix, Windows, MacOS, і розроблена з урахуванням інтеграції з різними веб-серверами, у даному випадку буде використовуватися IIS, отже цільовою операційною системою буде Windows. Веб-сервер IIS також є вільно поширюваною в Інтернет технологією. Мова C# та платформа .Net Core працює і з іншими веб-серверами. Сценарії можуть переноситися між серверними платформами без яких-небудь істотних змін. Крім того, застосунки написані на C# підтримують контейнерізацію docker, це дозволяє отримати ряд додаткових вигод при взаємодії з серверами на різних платформах.

C# надає велику кількість вже розроблених бібліотек, що значно спрощує процес розробки. C# створений на основі мові C++ і має добре

детермінований та інтуїтивно зрозумілий синтаксис. Досвідчені програмісти можуть легко додавати нові функціональні можливості. І розширений набір функцій, наявний в бібліотеці доповнень, є свідченням того, що саме так найчастіше і відбуваються.

Застосунок C#, при наявності досвіду, можна розробити за відносно малий проміжок часу, та реалізувати його надійним та легким для розширення функціоналу.

В C# включена підтримка багатьох баз даних, що робить написання веб-застосунків з використанням БД до неможливості простим.

Ось неповний перелік підтримуємих БД:

SQL Lite, PostgreSQL, MySQL, MariaDB, Firebird, Azure Cosmos DB, Db2, Informix та інші.

Приведений список далеко не повний. Мова надає розробнику усі необхідні інструменти для реалізації взаємодії з базою даних. Вона має достатню кількість функцій для реалізації поставлених завдань. Тому, ця мова має усі потрібні можливості для реалізації складних систем.

Для розробки мовою C# використовується середовище розробки - Visual Studio.

2.6 Шаблон MVC

Модель-вигляд-контролер (англ. Model-view-controller, (MVC)) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення [10].

Цей шаблон поділяє систему на три частини: модель даних (Model), вигляд даних (View) та контролер (Controller) (рис. 2.3). Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

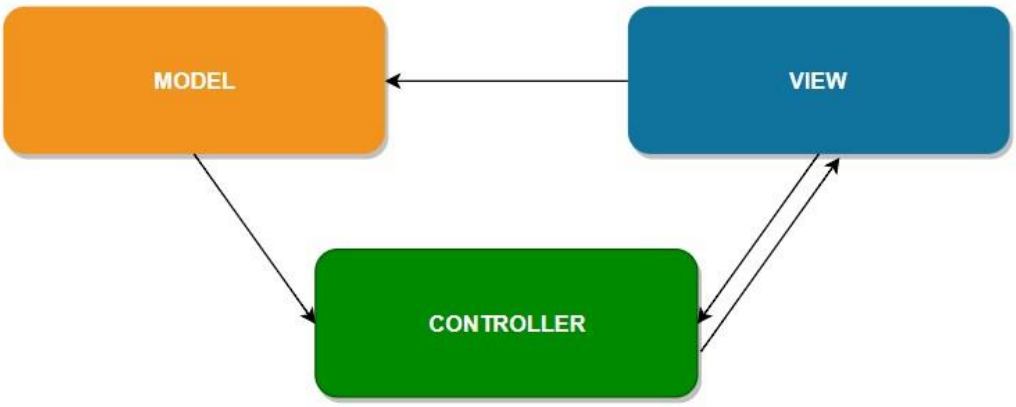

Рисунок 2.3 – Діаграма взаємодії між компонентами шаблону

Мета шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Архітектурний шаблон MVC поділяє програму на три частини. У тріаді до обов'язків компоненту Модель входить зберігання даних і забезпечення інтерфейсу до них. Вигляд відповідальний за представлення цих даних користувачеві. Контролер керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов'язаних

областей, наприклад, різні таблиці і поля форм, в яких відображається інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача (зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле).

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам Моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

2.7 Фреймворк для Javascript

Фреймворк (англ. Framework, каркас, платформа, структура, інфраструктура) – інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою [7].

Програмний фреймворк (англ. software framework) – це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Відповідно – програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту. Побудова кінцевого продукту відбувається, зазвичай, на базі єдиного API.

Одна з головних переваг, при використанні каркасних застосунків, полягає в тому, що такі програми мають стандартну структуру. Каркаси застосунків стали популярними з появою елементів інтерфейсу, які мали тенденцію до реалізації стандартної структури для застосунків. З їх

використанням стало набагато простіше створювати засоби для автоматичного створення графічних інтерфейсів, оскільки структура внутрішньої реалізації коду програми стала відома заздалегідь. Для забезпечення каркасу, зазвичай, використовують підходи об'єктно-орієнтованого програмування, наприклад, частини програми можуть успадковуватися від базових класів фреймворка.

Angular – JavaScript-фреймворк з відкритим вихідним кодом призначений для розробки односторінкових додатків.

В ході проведення аналізу виявлено такі основні особливості Angular, які можна використати для розробки веб-застосунку:

- висока продуктивність;
- єдина структура проекту;
- реактивне програмування з RxJS;
- шаблони, засновані на розширенні HTML;
- кешування сторінок та окремих фрагментів;
- перехоплення та обробка помилок;
- введення та валідація веб-форм;
- кросбраузерна підтримка HTTP, WebSockets, Service Workers;
- наявніть dependency injection та чітка структура залежностей;
- генерація базової структури проекту;
- підтримка тем оформлення для їх легкої зміни;
- можливість підключення сторонніх бібліотек.

Розробка веб-застосунків мовою Typescript може здійснюватися як в будь-якому простому текстовому редакторові, наприклад, блокноті Windows, так і в потужних WYSIWYG-редакторах або інтегрованих середовищах розробки. До найбільш відомих редакторів відносяться:

а) Visual Studio Code;

б) Jetbrains Webstorm;

в) Angular IDE;

г) Net Beans IDE.

У роботі використовувався редактор - Visual Studio Code.

2.8 Вибір системи управління базами даних

Критерії оцінювання СУБД розділені на чотири групи незалежно від того, як оцінки використовуватимуться : для порівняння СУБД або для оцінки параметрів системи. Ці групи такі:

- економічні (ціна придбання, повна вартість володіння або TCO – Total Cost Ownership, коефіцієнт повернення інвестицій або ROI, життєздатність постачальника, риски, вигоди, що не оцінюються в грошових одиницях, та ін.);
- технічні (функції, включаючи застосунки, рівень інформаційних технологій, прийнятих в реалізації, відкритість, надійність, ефективність, перспективи розвитку, рівень технічної підтримки, та ін.).

СУБД повинна відповідати таким основним вимогам:

- несуперечливість даних;
- актуальність даних, що зберігаються;
- багатоаспектне використання даних;
- можливість модифікації системи;
- надійність;
- швидкість доступу.

Для реалізації поставлених задач оптимальним варіантом буде реляційна СУБД.

Так як для створення інформаційної системи управління каталогом нерухомості в роботі обрано веб-сервер IIS та мову C# доцільним буде використання СУБД Microsoft SQL, яка адаптована під взаємодію з сервером застосунків C#.

Програмне забезпечення Microsoft SQL є багатопотоковим, розрахованим на багато користувачів і надійним сервером баз даних SQL. Сервер Microsoft SQL призначений як для обслуговування критично

важливих, сильно завантажених виробничих систем, так і для вбудовування в програмне забезпечення масового застосування. Microsoft SQL – торговельна марка, належна Microsoft [11].

Microsoft SQL постійно розвивається та покращується, що робить цю технологію перспективною та позбавляє необхідності в пошуку більш сучасної та ефективнішою технології у майбутньому. В зв'язці з платформою .Net Core та пакетом бібліотек Dapper надає розробнику та користувачу надійну та швидку систему для зберігання даних. Dapper – пакет бібліотек для .Net, які спрощують процес розробки доступу та взаємодії з базою даних.

2.9 Вибір платформи та фреймворку для розробки мовою C#

Для розробки веб-застосунку було вибрано мову програмування C#, для виконання поставлених завдань недостатньо вибрати лише мову програмування. Для розробки повноцінного веб-застосунку є потреба в виборі платформи та фреймворку для розробки проекту. Сучаснішою платформою для розробки мовою C# являється .Net Core, а фреймворком – ASP.Net Core [12].

.Net Core - це модульна платформа для розробки програмного забезпечення з відкритим вихідним кодом. Сумісна з такими операційними системами як Windows, Linux і macOS. Була випущена компанією Microsoft.

ASP.Net Core – це вільно-розповсюджуваний крос-платформний фреймворк для створення веб-додатків з відкритим вихідним кодом. Дана платформа розробляється компанією Майкрософт спільно з спільнотою і має велику продуктивність в порівнянні з ASP.NET

ASP.NET Core може працювати поверх крос-платформного середовища .NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS, Linux. За допомогою ASP.NET Core ми можемо створювати крос-платформні додатки. Тобто ми можемо запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання

веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget.

Однією з визначальних особливостей .NET Core є гнучке розгортання: ви можете встановити платформу або як частина свого застосування, або окремо.

.Net Core має вбудоване введення залежностей (Dependency Injection), спрощений високопродуктивний модульний конвеєр HTTP-запитів, інструментарій, що спрощує процес сучасної веб-розробки.

В рамках крос-сумісності платформа для розробки додатків включає в себе модульну інфраструктуру. Вона видається через NuGet, і ви можете отримати доступ до пакетним функцій, а не до однієї великої збірці. Як розробник ви можете створювати легкі додатки, що містять тільки необхідні пакети NuGet, що зробить вашу програму безпечніше і продуктивніше.

Доступ до командного рядка (CLI) на будь-якій підтримуваній платформі. Інтерфейс CLI потрібен для виконання певних команд для запуску та розгортання проекту. Подібно платформі .NET Core, CLI є кросплатформним, розробник має можливість використовувати його на більшості сучасних операційних систем.

В результаті аналізу існуючих технологій для веб-застосунка виявлено набір технологій, які є оптимальними для реалізації поставленої задачі:

- веб-сервер IIS;
- сервер застосунків C# на платформі .Net Core;
- сервер БД MS SQL;
- Microsoft Sql Management Studio для адміністрування бази даних.

3 ПОБУДОВА МОДЕЛІ ВЕБ-СЕРВІСУ

3.1 Опис інформаційної системи

Для того щоб подальший виклад розробки було більш зрозумілим і чітким, потрібно визначити якийсь розподіл процесів і об'єктів, що використовуються в процесі моделювання.

Для системи керування каталогом нерухомості застосовується програмно-апаратний комплекс, який дозволяє спростити процеси управління певним набором інформації про нерухомість.

При моделюванні системи було виявлено такі об’єкти та процеси:

- адміністратор системи – роль у системі, яка надає можливість розподілу дозволів та певних ролей для користувачів системи.

- користувач - особа, яка має певний набір дозволів виданий адміністратором, для доступу к функціям перегляду, редагування та видалення інформації про нерухомість .

- роль – набір дозволів, які видаються користувачу згідно з його професійною діяльністю або іншими потребами.

- дозвіл – певне правило, яке дозволяє користувачу використовувати певні функції системи.

- об’єкт нерухомості - сукупність інформації об'єднана в певний інформаційний об’єкт.

- каталог нерухомості - база нерухомості, яка може бути згрупована та посортована за певною ознакою. До каталогу має доступ будь-який зареєстрований користувач, але для повного доступу до інформації повинен мати певні дозволи від адміністратору.

- мапа нерухомості - набір матеріалів у вигляді тексту, цифрових зображень на мапі. База даних нерухомості містить координати завдяки яким можна побачити їх положення на мапі системи.

- пошук нерухомості та інформації про нерухомість – дає можливість користувачу знайти певну інформації у базі даних за різноманітними ознаками.

- зміна інформації залежно від року – надає можливість користувачу отримати інформації про нерухомість залежно від вибраного року, історичну (інформації за попередні роки) та поточну.

3.2 Процес керування та перегляду інформації нерухомості

Процес доступу до інформації про нерухомість організовано на підставі потреб бізнесу або державних структур, які будуть використовувати дану систему. Функціонал може відрізнятися залежно від призначення системи. Незалежно від призначення використовуються базові функції, які притаманні створюваній системі. До таких функцій системи управління каталогом нерухомості належать наступні (рис.3.1):

а) розподіл доступу до функцій системи;

б) пошук об’єктів нерухомості;

в) мапа нерухомості;

г) редагування інформації нерухомості.

Рисунок 3.1 – Функції системи управління каталогом нерухомості

Основними видами інформаційних об’єктів зв’язаних з нерухомістю є:

- оцінка вартості нерухомості;
- інформація про продаж нерухомості;
- певні пільги на вартість нерухомості;
- судові справи(скарги) зв’язані з вартістю нерухомості;
- інформація про район до якого прилягає нерухомість.

Розглянемо оцінку вартості нерухомості, як основний вид діяльності для системи. При будь-яких продажах, ремонтних роботах та інших видах діяльності потребується оцінка вартості нерухомості, та інформація про цю оцінку може бути збережена та відредагована у системі.

Крім того оцінка нерухомості може змінюватися декілька разів на рік впродовж років існування нерухомості, тому такі процеси потребують зберігання інформації за весь період існування нерухомості або період обслуговування цієї нерухомості структурою що використовує об’єкт.

Інформація про продаж нерухомості – для збереження актуальної інформації про поточних власників нерухомості або попередніх власників, система повинна надавати користувачу можливість управління такою інформацією.

Пільги на вартість нерухомості – ця інформація буде корисною при розрахунках певних податків зв’язаних з нерухомістю, а можливість зберігання інформації за декілька років значно спростить цей процес.

Судові справи(скарги) – зберігання інформації про скарги та можливість перерахунку певних пільг та оцінки нерухомості в результаті схвалення або відхилення скарги це значна функція для управління інформацією про нерухомість.

Район нерухомості – це інформація яка допоможе розподіленню сфер діяльності наприклад пожежних частин, лікарень та т.д. Також ця інформація може використовуватись для підрахунку певних витрат зв’язаних з районом.

Такий набір інформаційних об’єктів допоможе будь-якій організації спростити свої процеси розрахунку певних вартостей та податків на

нерухомість, або ж повністю автоматизує ці процеси .

Для користувача важливо, щоб система надавала користувачу можливість змінити часовий проміжок у системі для відображення інформації за певний період часу. Адміністратор системи матиме можливість налаштувати такі проміжки часу за потребами організації, які використовують систему.

Також для того, щоб система була гнучкою та користувачі системи мали можливість налаштувати її під свої потреби наступні засоби:

- можливість сховати та показати певні модулі системи;
- можливість сховати та показати певні інформаційні поля об’єкта;
- можливість змінити назву певних інформаційних полів об’єкта;
- можливість створити нове інформаційне поле об’єкта;
- можливість змінити порядок інформаційних полів об’єкта;
- можливість змінювати межі інформаційного року системи.

Таким чином користувачі системи зможуть налаштувати її під себе та розширити з мінімальними затратами часу.

При моделюванні системи потрібно передбачити, що різні користувачі системи можуть користуватися декількома базами нерухомості.

Один користувач може мати можливість використовувати різні бази нерухомості, для цього у системі повинна бути можливість зміни бази даних. Адміністратор системи має можливість надати доступ користувачу до будь-якої бази даних.

Проаналізувавши можливості та вимоги системи, можна виділити такі варіанти користування системою:

- адміністратор системи створює нових користувачів, надає їм певні дозволи на використання функціоналу системи, займається налаштуванням системи під певну особливість роботи своєї організації, надає доступ до різних баз даних за необхідністю;

- користувач залежно від ролі та дозволів наданих адміністратором, має можливість знайти у пошуку потрібний йому об’єкт нерухомості,

переглянути його географічну інформацію на мапі та отримати доступ до процесів та інформаційних об’єктів зв’язаних з нерухомістю.

Модуль управління користувачами передбачає можливість створення нового профіля користувача. Так як така система повинна бути закритою та захищеною, тому система не передбачає реєстрування самим користувачем і функція створення нових користувачів доступна тільки адміністратором. Після створення профілю користувач має підтвердити свою електронну пошту та отримати доступ до системи.

Адміністратор після створення профілю переходить до процесу надання ролі та дозволів для нового користувача.

Роль це набір дозволів, які відкривають користувачу певний функціонал системи, але після видачі ролі адміністратор може відкоригувати дозволи індивідуально для даного користувача.

Система надає користувачу набір дозволів, які поділяються на декілька типів – доступ на перегляд, доступ на редагування, доступ на видалення. Без дозволу на перегляд користувач не матиме можливість редагувати та видаляти, навіть якщо матиме такий дозвіл.

Пошук дозволяє знайти об’єкт нерухомості по будь-якому критерію пошуку. Інформація яка була введена користувачем у полі пошуку буде порівняна з кожним полем інформаційного об’єкту нерухомості.

Після того, як користувач знайшов потрібний йому об’єкт нерухомості або набір об’єктів, він має можливість переглянути коротку інформацію про нерухомість та переглянути її локацію на мапі. Якщо цієї інформації недостатньо, користувач має можливість відкрити повну інформацію про нерухомість.

Система надає користувачу доступ до додаткової інформації про нерухомість, якщо користувачу потрібно знайти пільги, судові справи і т.д. Система надає можливість ввести додаткову інформацію про нерухомість до системи.

3.3 Класифікації моделей інформаційної системи з точки зору організаційно-технологічного підходу

Функціональність інформаційних систем залежить від організаційно-управлінської структури організації, технології документообігу та всього, що необхідно для підтримки ефективного бізнесу. За сучасного рівня розвитку програмного забезпечення, існує безліч різноманітних програмних засобів обробки інформації, написаних різними мовами програмування за вище переліченими методами. Різноманіття програмних пакетів (ПП) пов’язано із специфікою кожної галузі, в якій проводиться обробка.

Водночас характерною особливістю web-орієнтованих інформаційних систем, зокрема систем управління контентом web-сайтів, є наявність великої кількості документальної інформації, причому самі ці документи часто генеруються динамічно, на основі тих чи інших процедур. Тому web-орієнтована інформаційна система, яка ґрунтується на знаннях, повинна бути орієнтована на роботу як з онтологіями, так і з документами і множинами документів.

Переважно, під час розроблення web-орієнтованих інформаційних систем, які мають використовуватися в мережі Internet чи в локальній мережі окремої установи, постає задача створення структурно складних систем. Такі системи складаються з компонентів, кожен з яких виконує свою роль у життєвому циклі системи, і задачі, що виконуються ними, є самостійними і можуть бути окремими повнофункціональними системами.

Тому в основу моделі інформаційної бази web-орієнтованої системи необхідно покласти дві вузлові компоненти: онтологія предметної області та множина документів, а також зв’язки між цими компонентами. Іншими словами, необхідно побудувати граф, який складається не з логічно розрізнених документів, що характерно для більшості сучасних web-орієнтованих систем, а з описів реальних класів предметної області, їх екземплярів та зв’язків між ними, а також пов’язаних з ними документів.

Тобто, можна говорити про проблему занурення множини документів, які можуть бути статичними або генеруватися динамічно, в загальну семантику предметної області. Детальніше формалізувати зв’язки між семантикою предметної області та множиною документів можна за формальною моделю онтології.

Слід також зазначити, що базу знань, яку покладено в основу інтелектуальної інформаційної системи, можна розглядати як наповнення власне онтології як концептуальної схеми. Формально розглядаємо модель інформаційного наповнення web-орієнтованої системи як трійку:

$$M =< W^*, D, L >, \tag{3.1}$$

де M – онтологія предметної області, W* – розширена онтологія, наповнення онтології W конкретними екземплярами класів (фактично – база знань), D – множина документів; L – множина зв’язків між W* та D. На найзагальнішому рівні, в термінології UML власне онтологія пов’язана з діаграмою класів, а наповнення бази знань – з діаграмою екземплярів. Але об’єктно-орієнтована реалізація онтологічного підходу, незважаючи на спорідненість цих напрямків, має свою специфіку.

Зокрема, в зазначається, що розроблення онтологій значно відрізняється від проектування класів і зв’язків в об’єктно-орієнтованому програмуванні.

Об’єктно-орієнтоване програмування зосереджується переважно на методах класів: програміст приймає проектні рішення за операторними методами класу, тоді як розробник онтології приймає рішення, ґрунтуючись на структурних властивостях класу. У результаті структура класу і зв’язки між класами в онтології значно відрізняються від структури подібної предметної області в об’єктно-орієнтованій програмі. У показано, що значеннями функцій інтерпретації онтології можуть бути елементи множини документів D.

При цьому мова повинна йти не тільки власне про онтологію, але і про всю базу знань, побудовану на її основі. Якщо онтологія розглядається як трійка, де Q – множина класів, які відповідають поняттям предметної області; I – множина зв'язків між ними, а F – множина функцій інтерпретації, то розширена онтологія описується як трійка , де Q* – множина класів разом з їх екземплярами, I* – множина зв'язків між цими елементами, а F* – множина функцій інтерпретації, визначених у найпростішому випадку на елементах з Q*, I*. Тоді елементи D можуть бути значеннями функцій з F*. Іншими словами, будемо вважати документ d релевантним відносно W*, якщо існують хоча б один вузол i/та функція інтерпретації f такі, що d=f(w).

Важливими елементами web-орієнтованої інформаційної системи повинні стати класи, які реалізують функції інтерпретації та можуть використовуватися для динамічного формування документів, а також класи, які забезпечують відображення отриманих документів. Наведене співвідношення може стати основою для динамічного формування переліку споріднених документів. Для документа d=f(w) спорідненими документами можуть вважатися зокрема такі:

- всі документи s такі, що s=h(w), де h пов'язане з F* (тобто документи, пов'язані з тим самим вузлом іншими функціями інтерпретації); можливість сховати та показати певні інформаційні поля об'єкта;
- всі документи s такі, що s=g(u), де функції інтерпретації g пов'язані з f, вузли и пов'язані з w.

Принципово важливим є те, що зв'язки і можливі переходи між документами повинні насамперед визначатися зв'язками між класами онтології та їх екземплярами. Тому ключовим компонентом стає система навігації, яка на основі онтології забезпечує динамічне формування навігаційних графів, які визначають можливі переходи web-сайтом.

Якщо ввести міри, які характеризують семантичну близькість понять онтології, а також ступені важливості функцій інтерпретації, це дасть змогу проранжувати документи за мірою їх спорідненості до даного. Цікаво також

дослідити, як введені так міри спорідненості співвідносяться з традиційними підходами до визначення мір схожості документів на основі аналізу власне текстів.

У цьому контексті стає очевидним, що проблема динамічного формування документів, споріднених до даного, якнайтісніше пов’язана з проблемою підвищення повноти та релевантності інформаційного пошуку. Дійсно, незалежно від того, чи користувач вийшов на певний вузол онтології в процесі навігації по сайту, чи він ввів відповідне ключове слово в полі введення – йдеться про формування переліку документів, пов’язаних з цим вузлом, і ранжування цих документів за мірою релевантності.

Провівши формальний опис інформаційних систем, визначимо типові операції над нею. Розділимо такі операції на дві основні групи.

До першої групи належать операції, що модифікують наявну систему шляхом зміни її внутрішньої структури – у таких операціях бере участь одна інформаційна система.

Друга група операцій формується на основі операцій, традиційних для теорій формальних систем та абстрактних автоматів. У таких операціях беруть участь дві або більше інформаційні системи; результатом операції є нова система, що їх об’єднує.

Розглядаючи операції 2-ї групи, вважатимемо, що вихідний потік може розщеплюватися або дублюватися та одна з його копій – передаватися в іншу систему. У результаті застосування таких операцій до двох або більше інформаційних систем утворюється нова інформаційна система, що містить як компоненти вихідні системи. Запити до цих систем уже надходитимуть від вищого сервісу, а не від користувачів системи. Відповіді компонент спрямовуються як вхідний потік у вищу систему.

Типовими представниками першої групи є такі операції:

- операція визначення параметра. Визначимо перетворення інформаційної системи IS на IS’ визначаючи параметр як фіксацію значення одного з параметрів Pj(i) запиту Qi. У разі такої операції невизначеність

поведінки системи залишається такою самою.

- операція усунення параметра. Визначимо перетворення інформаційної системи IS у IS’ шляхом усунення параметра як усунення одного з параметрів Pj(i) запиту Qi з розгляду. У разі застосування цієї операції зростає невизначеність поведінки системи.

- операцію усунення параметра доцільно застосовувати до системи з метою побудови її загальної моделі, придатної для презентації;

- операція визначення параметра з запиту. Визначимо операцію визначення параметра з запиту як злиття групи можливих запитів в один та визначення нового параметра в цьому запиті. Область визначення нового параметра повинна бути скінченною множиною, кожен з елементів якої однозначно відповідає одному з усунутих збігаються. Таку операцію застосовують лише у випадку, коли кортежі параметрів запитів збігаються. Тому перед застосуванням цієї операції доцільно узгодити кортежі запитів за допомогою операції усунення та визначення параметрів. Очевидно, що за такої операції зростає невизначеність поведінки системи;

- операція визначення запиту з параметра. Ця операція визначення запиту з параметра є оберненою до попередньої (один запит розбивається на групу запитів за допомогою усунення параметра з переліченою областю визначення). Така операція породжує групу запитів, у яких домени параметрів запитів збігаються. При такій операції зростає визначеність поведінки системи. Тому ця операція є не зовсім коректною у тому сенсі, що в системі недостатньо інформації для її здійснення. Тому необхідно проводити додаткові дослідження для уточнення поведінки системи;

Аналогічно формулюються операції для станів та відповідей інформаційної системи. До другої групи належать такі операції:

Послідовне з’єднання двох ІС. З’єднання відбувається послідовно (відповідно до технології клієнт-сервер). У такому випадку вихідний символ IS1 формує запит IS2. Так отримуємо інформаційну систему IS, яка є послідовним об’єднанням двох IS1 та IS2. Її вхідний алфавіт збігається з

вхідним алфавітом IS1, а множина вихідних символів з вихідним алфавітом – IS2. Невизначеність поведінки кожної з систем IS1 та IS2 залишається незмінною, а поведінку нової системи прогнозують так: вхідні сигнали для IS2 є вихідними для IS1, тому імовірність вхідного символу IS2 отримуватиметься так:

$$\frac{1}{\mathrm{N}}\sum_{j=1}^{N} P r_j(A_j^{(1)}) \xrightarrow[N\to\infty]{} \Pr(Q_j^{(2)}), \tag{3.2}$$

де $\mathrm{P}\mathrm{r}_{\mathrm{j}}(A_j^{(1)})$ – імовірність вихідного символу Ai для IS1 на j-й ітерації.

Імовірність кожного стан у та переходу в межах підсистеми IS1 залишається незмінною, а для підсистеми IS2 коригується згідно із змінами ймовірностей вхідних символів. Така операція характерна для серверів переадресації, довідкових систем, електронних бібліотек тощо [13].

2. Паралельне з'єднання двох ІС. З'єднання відбувається паралельно (два сервера об'єднуються в один). У такому випадку отримуємо інформаційну систему IS, яка є паралельним об'єднанням двох IS1 та IS2. Її вхідний алфавіт утворюється як об'єднання вхідних алфавітів IS1 та IS2, а вихідний алфавіт є об'єднанням вхідних алфавітів IS1 та IS2. Невизначеність поведінки кожної із систем IS1 та IS2 залишається незмінною, а поведінку нової системи прогнозують так: вхідні сигнали IS є вхідними для IS1 або IS2, тому імовірність вхідного символу IS отримують за виразом:

$$\Pr(\mathrm{Q}_{\mathrm{j}}) = \Pr(Q_j^{(k)})\Pr(I S_k), \tag{3.3}$$

де $\Pr(Q_j^{(k)})$, k = 1,2 – імовірність вхідного символу Qi для ISk; $\Pr(I S_k)$, k = 1,2 – імовірність, що вхідний символ належить до алфавіту підсистеми ISk

3. Замикання ІС. Вихідний потік спрямовується як вхідний у ту саму систему (сервер замикається сам на себе). Поведінка нової системи буде загалом схожа на поведінку вихідної.

Одним з наслідків побудови моделі web-системи є можливість формального опису її поведінки. Такий опис має імовірнісний характер та відображає асимптотичну поведінку системи.

Досліджуючи поведінку системи, розглядають абстрактний часовий параметр. Розглянемо проблему визначення часового параметра. Формальний підхід, описаний в, ґрунтується на виділенні певної впорядкованої множини, що розглядається як множина моментів часу.

Елементи множин вхідних сигналів та реакцій системи є результатами застосування певних функцій до моментів часу. При моделюванні систем архітектури клієнт–сервер, враховуючи те, що будь-яка дія системи визначається запитом клієнта, за формальний параметр часу доцільно приймати номер за порядком запиту клієнта. Множина номерів запитів – множина натуральних чисел.

3.4 Декомпозиція моделі системи

При проектуванні складної системи її розбивають на частини, кожна з яких потім розглядається окремо. Можливі два різні способи такого розбиття на підсистеми: структурне (або функціональне) розбиття і об'єктна (компонентна) декомпозиція.

При функціональної декомпозиції програмної системи її структура може бути описана блок-схемами, вузли яких представляють собою «обробні центри» (функції), а зв'язки між вузлами описують рух даних. У системі керування каталогом нерухомості можна виділити такі підсистеми: підсистема пошуку нерухомості і підсистема накопичення та редагування інформаційних об’єктів зв’язаних з нерухомістю. В технології програмування процеси, прохідні в таких системах, називають back-end і front-end,

відповідно. Front-end (фронт-енд) і back-end (бек-енд) - це узагальнені терміни, які відображають початковий і кінцевий стани процесу. Front-end відповідає за отримання введення (вхідної інформації) в будь-яких формах від користувача і обробку отриманої інформації в ту форму, яку back-end здатний використовувати. Front-end - це інтерфейс між користувачем і back-end'ом. Співвідношення функцій системи з процесами front-end і back-end наведені в табл. 3.2.

Таблиця 3.2 – Співвідношення функцій системи

Процеси	Функції
Front-end	1. Вивід інтерфейсу користувача 2. Вивід інформації про нерухомість 3. Вивід інформації користувачів 4. Заповнення форм для створення та редагування інформації 5. Відправка запитів до back-end
Back-end	1. Взаємодія з базами даних 2. Обробка інформації для оновлення даних та посилання даних на front-end

4 РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ КОРИСТУВАЧІВ СИСТЕМОЮ УПРАВЛІННЯ КАТАЛОГОМ НЕРУХОМОСТІ

4.1 Структурна схема веб-застосунку інформаційної системи управління каталогом нерухомості

Структурна схема клієнту веб-застосунку інформаційної системи управління каталогом нерухомості наведена на рисунку 4.1.

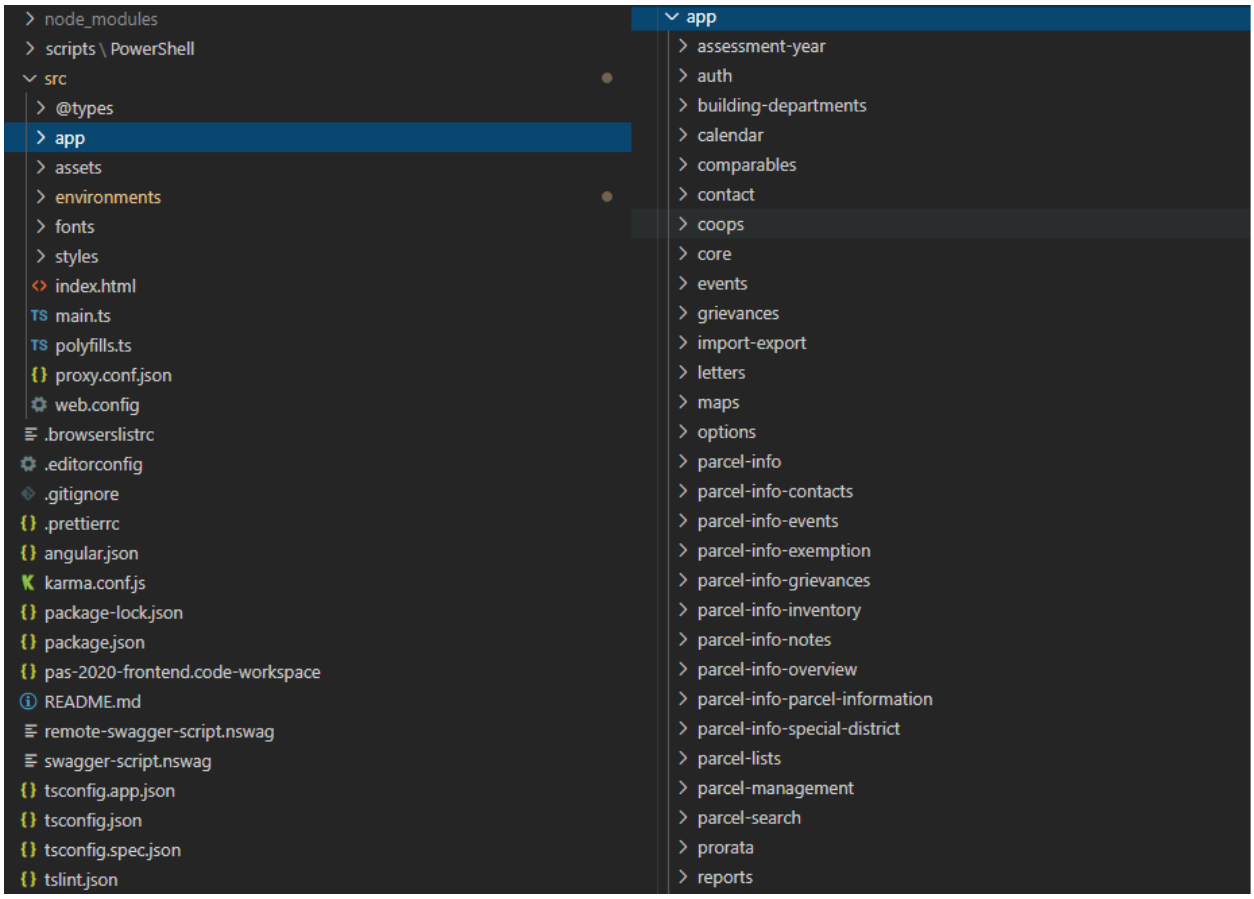

Рисунок 4.1 – Структурна схема клієнту веб-застосунку інформаційної системи управління каталогом нерухомості

У клієнті використовується стандартна архітектура технології Angular. Додаток представляє собою набір модулів – головний модуль додатку, який

з’єднує в один додаток усі модулі, функціональні модулі які відповідають за певнший функціонал додатку, та один спільний модуль який містить у собі функціонал який може бути перевикористан у декількох модулях.

Кожен модуль ділиться на такі складові частини:

- компоненти – відповідають за візуальну частину веб сторінки та містять у собі певну логіку для функціонування цієї сторінки

- сервіси – використовуються компонентами для обробки певних данних для подальшого їх використання

- дірективи – використовуються для різноманітних маніпуляцій с DOM

- фільтри – використовуються для зміни вигляду інформації перед виводом на html сторінку

Для створення клієнту додатка використовується засіб Redux, а саме його реалізація у Angular – NGRX. За допомогою цієї технології створюється певне дерево стану усього додатку. Коли клієнт отримує дані від сервісу або змінює їх сам, він записує у сховище стану певне значення, яке потім може бути використане у будь-якій клієнтській часті додатку.

Маніпуляції зі станом цих даних проводяться централізовано за домопогою можливостей ngrx. Таким чином у нас є одне місце за домопогою якого ми можемо оновлювати стан даних оминаючи оновлювання даних з різних місць.

Частини цієї технології можно поділити на такі сутності:

- дія – певний тип маніпуляція с даними сховища, містять у собі саме тип дії та певну інформацію, яка потрібна для виконання цієї дії;

- редуктор – це прості функції які приймають дію та попередній стан сховища, маючи ці дані отримують наступний стан сховища та записують його;

- эффекти – також мають справу з діями, але у відмінності від редукторів не працює зі станом сховища, а в більшості випадків з API;

- селектори – використовуються для отримання інформації зі сховища

компонентами.

Основна перевага цієї технології це оптимізація клієнту. Зміна стану сховища відбувається не так часто, та інформація на сторінках оновлюється тільки якщо дані в сховищі дійсно змінилися.

Структурна схема серверу веб-застосунку інформаційної системи управління каталогом нерухомості наведена у рисунку 4.2.

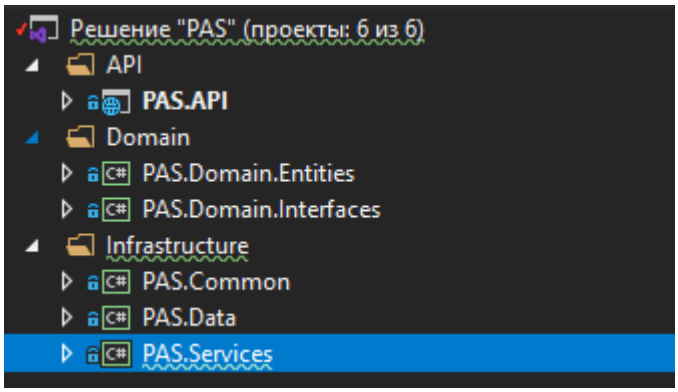

Рисунок 4.2 – Структурна схема серверу веб-застосунку інформаційної системи управління каталогом нерухомості

На серверній частині додатку використовується триярусна архітектура, яка поділяється на такі частини:

- рівень бази данних, головна задача якого, операції запису та считування інформації за бази данних. А також формування архітектури бази данних;
- рівень бізнес логіки, задача якого обробка інформації з клієнту для подальшого запису у базу данних, та обробка інформації з бази данних для подальшої передачі до клієнту в правильному вигляді;
- рівень програмного інтерфейсу, задача якого це комунікація та обмін інформацією з клієнтом.

Така архітектура допомогає розділити проект на рівні віподвідальності та пришвидшую розширення та розробку проекту.

4.2 Розробка адаптивної та кросбраузерної веб-сторінки

Адаптивний веб-дизайн (в англійській мові «responsive web design») - це дизайн веб-сторінок, що забезпечує відмінне сприйняття на різних пристроях, підключених до Інтернету.

Це означає, що один і той же сайт можна переглядати на самих різних пристроях, незалежно від дозволу і формату екрану, - смартфонах, планшетах, ноутбуках і т.д. При цьому перегляд буде однаково зручний для всіх форматів - користувачам мобільних пристроїв, наприклад, не потрібно буде розширювати окремі області сайту, щоб не промахнутися мимо потрібної посилання.

Адаптивний дизайн покликаний зробити веб-сторінки і відображення їх вмісту відповідними тому пристрою, з якого вони є видимими.

Можливості адаптивного веб-дизайну:

- велика розмаїтість пристроїв, з яких можна виходити в Інтернет. В даний час існує безліч пристроїв, якими люди користуються, в тому числі, і для того, щоб виходити в Інтернет. Всі ці пристрої розрізняються розміром екрану, дозволом і, відповідно, тим, як може відображатися на них веб-сайт. Тому важливо, щоб ваш сайт добре виглядав і правильно відображався у будь-якого з користувачів, незалежно від того, який пристрій він використовує;
- популярність мобільних пристроїв з виходом в Інтернет і збільшення мобільного Інтернет-трафіку. З ростом популярності мобільних пристроїв кількості користувачів, які заходять з них на сайти, помітно збільшилася, тому просто ігнорувати їх вже не можна - це не одна-дві людини на півроку, це значна частина вашої аудиторії, і їм повинно бути зручно користуватися вашим сайтом (інакше вони не будуть цього робити);
- термінова інформація. Якщо ваш ресурс містить новинну / термінову інформацію, і висока ймовірність, що користувачеві може знадобиться прочитати цю інформацію саме з телефону (бо інших пристроїв у нього під

рукою немає) в даний момент часу, потрібно подбати про те, щоб у нього була можливість це зробити.

Для того, щоб сайт однаково виглядав в Internet Explorer, в FireFox, в Opera та в Google Chrome (самих різних версій), веб-дизайнер повинен подбати про кросбраузерність веб-застосунку з самого початку роботи над проектом.

При розробці веб-застосунку необхідно визначитися, з якими саме браузерами повинен працювати сайт. Для цього потрібно вивчити статистику відвідувань сайтів (наприклад, зі схожою тематикою), в якій є можливість подивитися, якими браузерами користувалися його відвідувачі. Сервіс Google Analytics, наприклад, надає таку можливість. Таким чином, визначивши найбільш популярні з точки зору кількості відвідувань браузери, можна адаптувати веб-застосунок до них.

Різні браузери дотримуються загальні правила і стандарти, але мають різні алгоритми обробки Html-кодів і каскадних таблиць CSS. І тому не завжди один і той же елемент виглядає однаково в різних браузерах.

Рішень такої проблеми може бути декілька. Найбільш поширений спосіб, який застосовується багатьма розробниками веб-застосункців - це написання наборів спеціальних селекторів або правил, що підтримуються тільки певними браузерами. Тобто, якщо необхідно коректно відображати сайт, скажімо, в трьох браузерах, то потрібно написати спеціальну логіку для кожного з цих браузерів окремо. Такий варіант не являється оптимальним при розробці на велику кількість браузерів. В такому випадку потрібно використовувати елементи, які однаково працють на різних браузерах

4.3 Розробка інтерфейсу системи

Для забезпечення зручної роботи користувачів, інтерфейс повинен включати наступні елементи:

- блок ідентифікації сайту - частина графічного інтерфейсу, що

дозволяє відвідувачеві однозначно визначити, що являє ресурс, яким він користується. Являє собою логотип (знак) сайту, його назва та, можливо, короткий опис.

- гіперпосилання - частина документа, що посилається на інший елемент (текст, заголовок, примітка, зображення) в самому документі, на інший об'єкт (файл, директорія, додаток), розташований на локальному комп'ютері або в комп'ютерній мережі, або на елементи цього об'єкта.

- навігація - елемент інтерфейсу, що дозволяє виконувати перехід користувача до певних структурних частин (сторінок) сайту. Складається з головного і додаткового меню і являє собою набір гіперпосилань.

- контент - власне сам зміст сайту є сукупністю текстової та графічної інформації.

Оперуючи цими поняттями, можна описати зовнішній вигляд системи, що відповідає вимогам зручності використання. Власне проектування графічного інтерфейсу для сайту являє собою організацію всіх необхідних елементів на сторінці, а так само розробку їх зовнішнього вигляду.

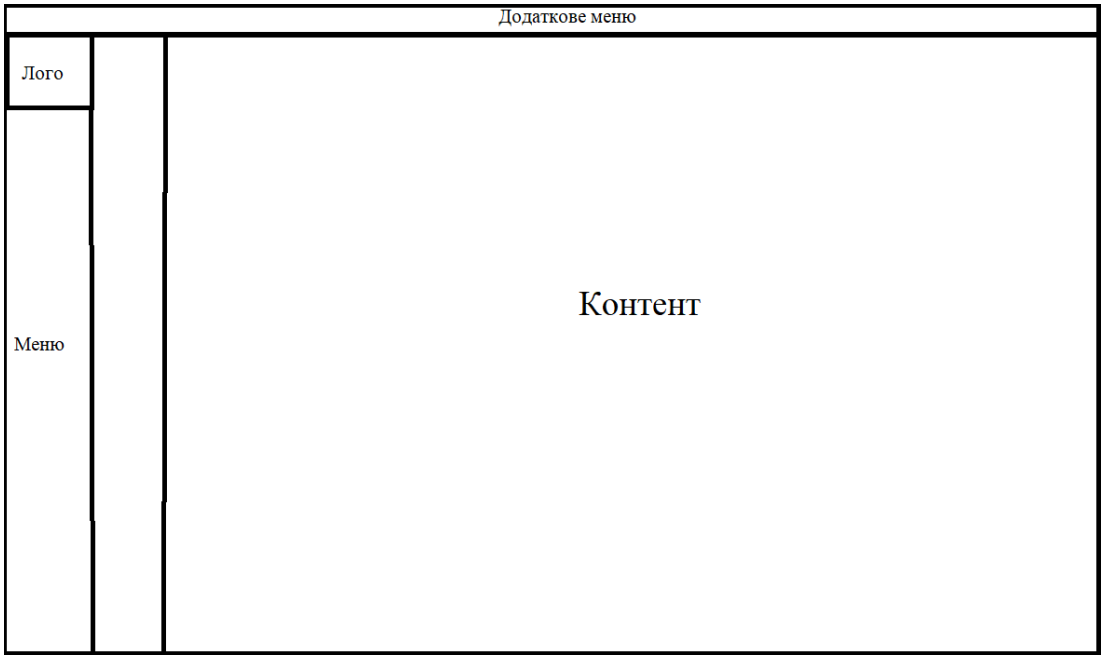

Рисунок 4.3 – Структура інтерфейсу сайту

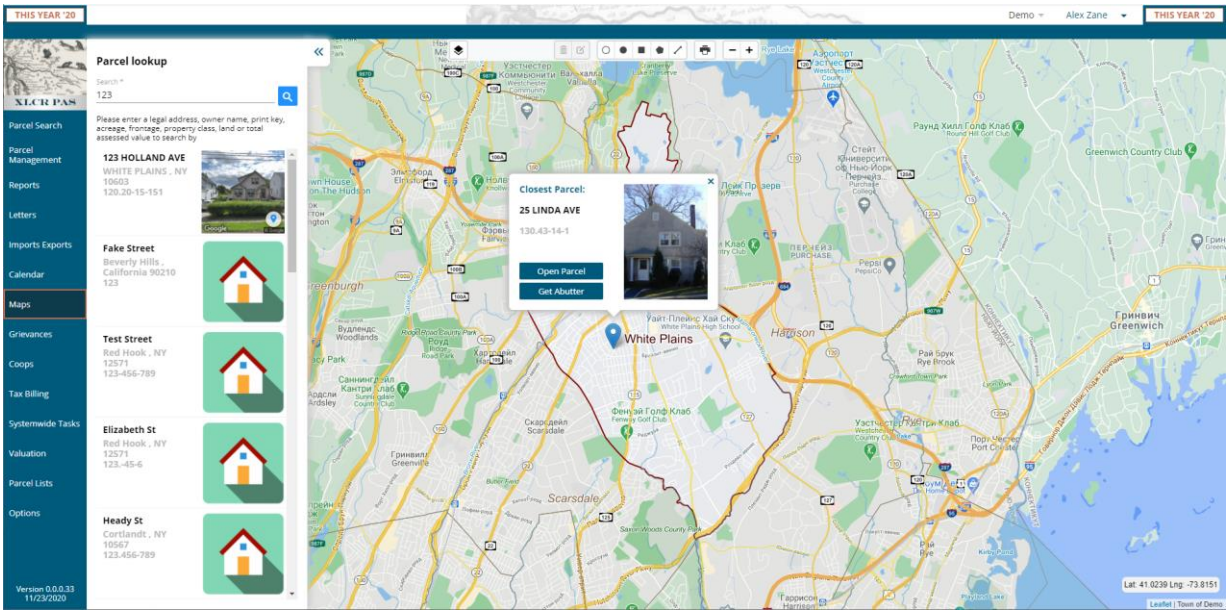

Рисунок 4.4 – Зовнішній вигляд графічного інтерфейсу системи

Блок ідентифікації сайту являє собою логотип сайту. Основне меню доступно всім користувачам, але пункти меню можуть буди недоступні в залежності від дозволів користувача та налаштувань системи. У цьому меню можна знайти посилання на мапу, пошук нерухомості, налаштування системи і т.д. Другий рівень основного меню з’являється при переходи за певним посиланням основного, якщо цей функціонал має декілька функціональних можливостей, які потребують додаткових посилань. За допомогою додаткового меню користувач має можливість зміни інформаційного року для інформації про нерухомість, зміни бази даних та вийти з профілю. Готовий графічний інтерфейс зображений на (рис. 4.4).

4.4 Розробка бази даних системи

Для зберігання інформації в системі управління каталогом нерухомості передбачається використання баз даних MSSQL. Потрібно розробити структуру бази даних, щоб почати процес програмування.

Таблиця 4.1 – База даних

Назва таблиці	Поле	Опис
Parcel	parcelId	Ідентифікатор
	Latitide Longtitude	Поля координат, які використовуються для відображення на мапі
	LegalCity LegalState LegalAdress LegalAdressNumber	Поля, які використовуються для відображення адреси нерухомості
User	UserId	Ідентифікатор користувача
	Email	Електронна адреса
	PaswordHash	Хєш паролю для аутентифікації користувача
SpecialDistrict	specialDistrictId	Ідентифікатор району
	specialDistrictCode	Унікальний код району
ParcelSpecialDistrict	parcelId	Ідентифікатор нерухомості
	specialDistrictCode	Унікальний код району
	calculationAmount	Сума податку нерухомості для району
	percentage	Процент податку нерухомості для району

Продовження таблиці 4.1

ParcelSale	ParcelId	Ідентифікатор нерухомості, яка була продана
	SaleId	Ідентифікатор продажу нерухомості
	SortOrder	Порядок сортування для виводу
TaxAttribute	taxAttributeId	Унікальний ідентифікатор атрибуту податку
	taxAttributeName	Назва атрибуту податку
SpecialDistrictTaxAttribute	taxAttributeId	Унікальний ідентифікатор атрибуту податку
	specialDistrictid	Унікальний ідентифікатор району
	attributePercentage	Процент податку атрибута для району
Role	RoleId	Ідентифікатор ролі
	Name	Назва ролі
UserRole	UserId	Ідентифікатор користувача до якого прив’язано роль
	RoleId	Ідентифікатор ролі

Продовження таблиці 4.1

Permission	PermissionId	Ідентифікатор дозволу
	Name	Назва дозволу
RolePermission	RoleId	Ідентифікатор ролі, до якої прив'язуються дозволи
	PermissionId	Ідентифікатор дозволу
Exemption	ExemptionId	Ідентифікатор пільги
	Percent	Процент знижки
	Amount	Сума знижки
	ExemptionCode	Унікальний код пільги
Sale	SaleId	Ідентифікатор продажу нерухомості
	SalePrice	Сумма отримана при продажі нерухомості
	DateSale	Дата продажу
	ContactId	Ідентифікатор контакту нового користувача

Продовження таблиці 4.1

TenantEnvironmentSettings	TenantId	Ідентифікатор бази даних для якої зберігаються налаштування системи
	Json	Налаштування системи та динамічні поля, які зберігаються в базі даних у форматі Json

Також кожна таблиця містить допоміжні для системи поля, userIdCreated та userIdModified. Ці поля використовуються для слідкуванням за змінами у системі, коли користувач створює або змінює будь-яку сутність в базі даних, ідентифікатор користувача записується в відповіде поле. Також у кожній таблиці знаходяться dateCreated та dateModifed. Для запису дат змін даних у таблицях.

4.5 Функціонал користувача

Після створення профілю у системі та надання певних дозволів, користувач отримує доступ до функціоналу системи. У кожного типу користувача є доступ до особистого кабінету, де він може отримати інформацію про свого облікового запису. Приклад сторінки користувача наведено на (рис. 4.4).

Головною функцією системи є пошук, перегляд загальної інформації та геоінформації для нерухомості. Тому після створення профілю користувач має можливість перейти на вкладку за посиланням «Maps» (рис. 4.5).

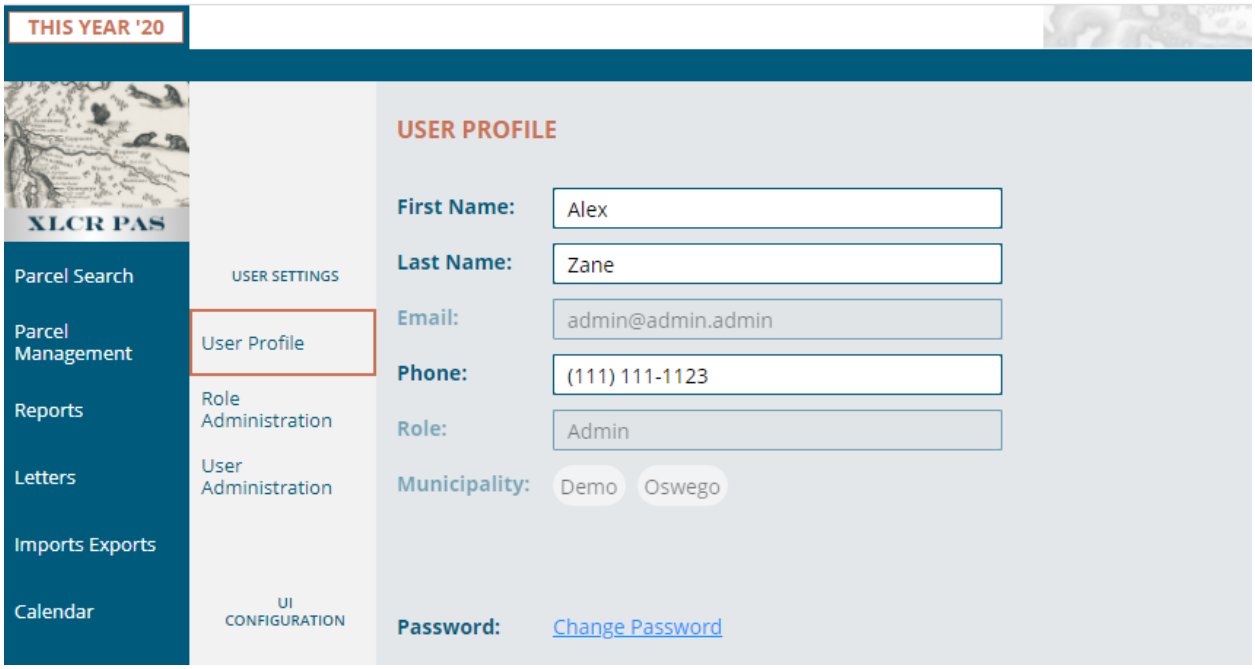

Рисунок 4.5 – Вид сторінки користувача

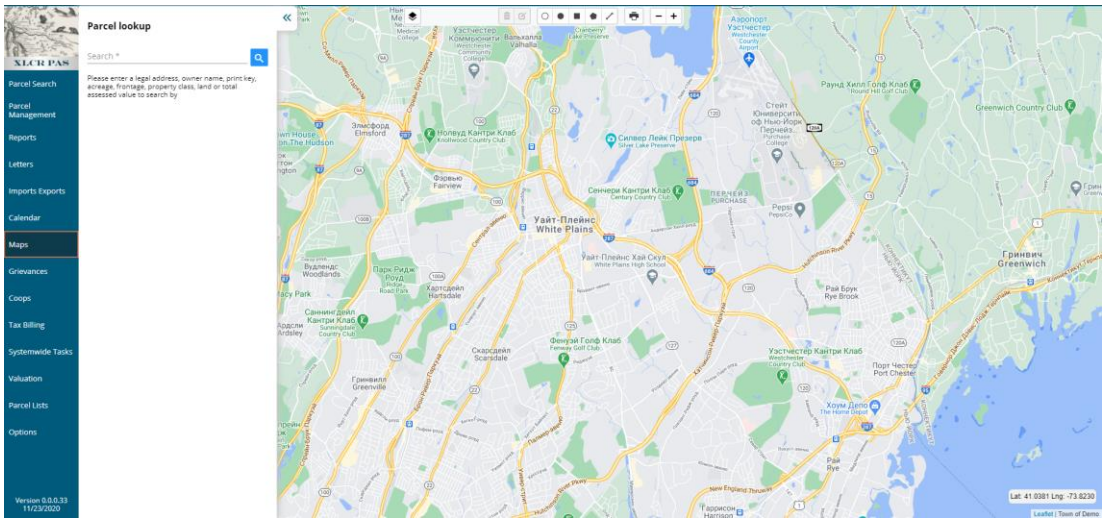

Рисунок 4.6 – Вид сторінки мапи

На цій сторінці крім мапи наявний функціонал пошуку та перегляду інформації про нерухомість. Для цього достатньо ввести певну інформацію про нерухомість, яка описана під полем пошуку та отримати список нерухомості, яка підпадає під критерії пошуку (рис. 4.7).

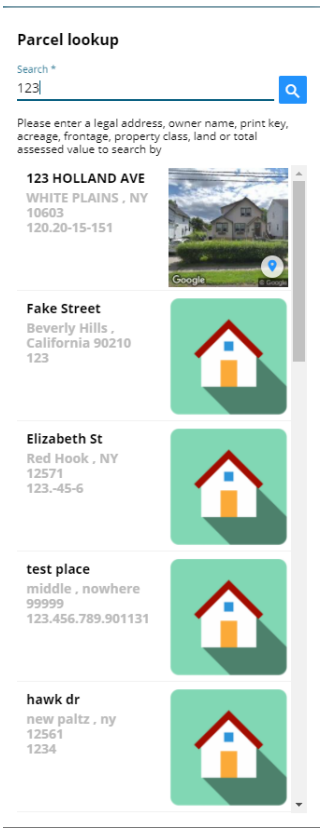

Рисунок 4.7 – Панель пошуку нерухомості

Також при натисканні на певну адресу на мапі можна побачити фото, яке зв’язано з цією нерухомістю, якщо вона міститься у базі даних. Та отримати інформацію зв’язану з цією нерухомістю (рис. 4.8).

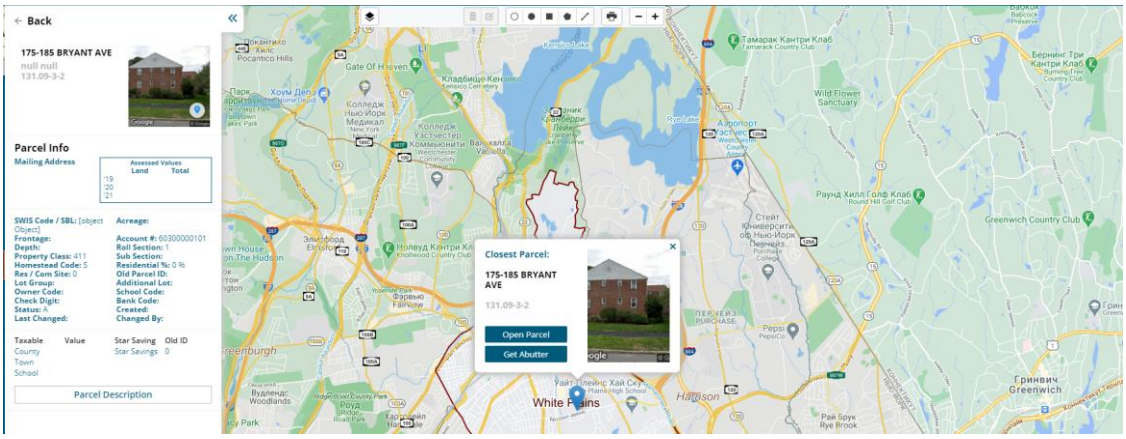

Рисунок 4.8 – Панель інформації про нерухомість

Для пошуку більш детальної інформації про нерухомість користувач може скористуватися пошуком за певними інформаційними полями сутностей, які зв’язані з нерухомістю та саме нерухомістю. Користувач має можливість знайти нерухомість за власником, адресою, спеціальним кодом і т.д. (рис. 4.9).

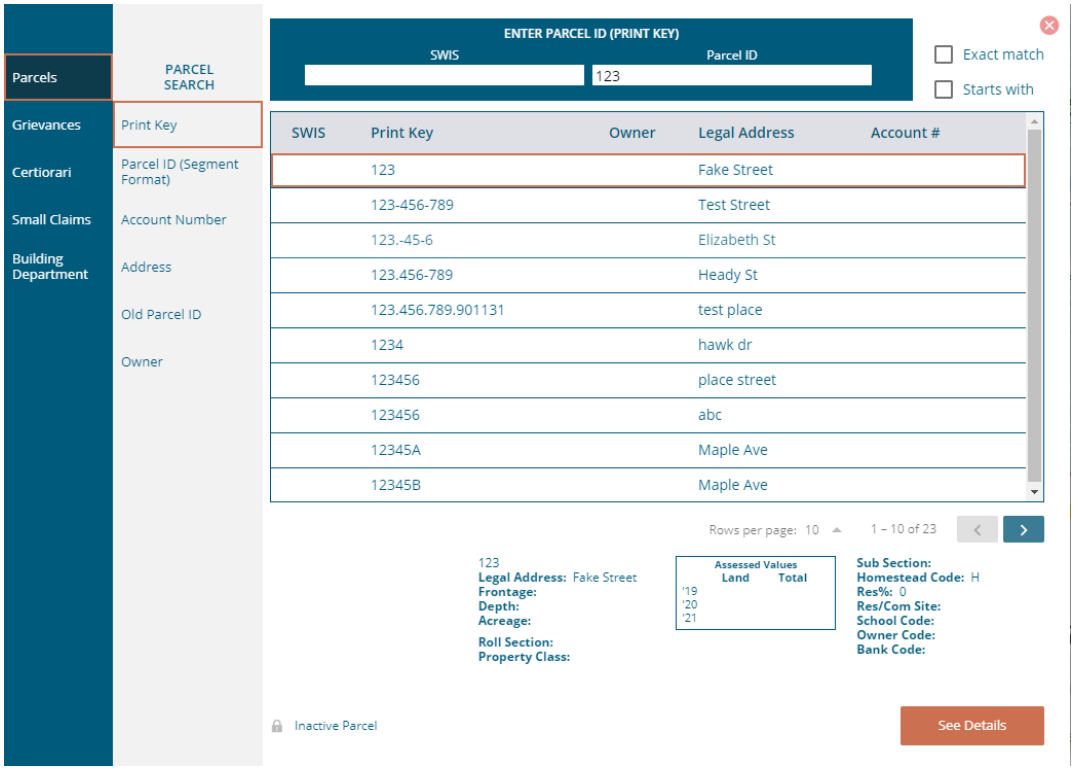

Рисунок 4.9 – Панель пошуку нерухомості

Користувач має можливість натиснути на кнопку «See Details» та перейти на сторінку інформації про нерухомість. Де він має змогу переглянути інформацію про саме нерухомість, продажі нерухомості, пільги та скарги зв’язані з нею. А також при наявності певних дозволів для кожної зі зв’язаних сутностей, користувач має можливість редагувати інформацію (рис. 4.10).

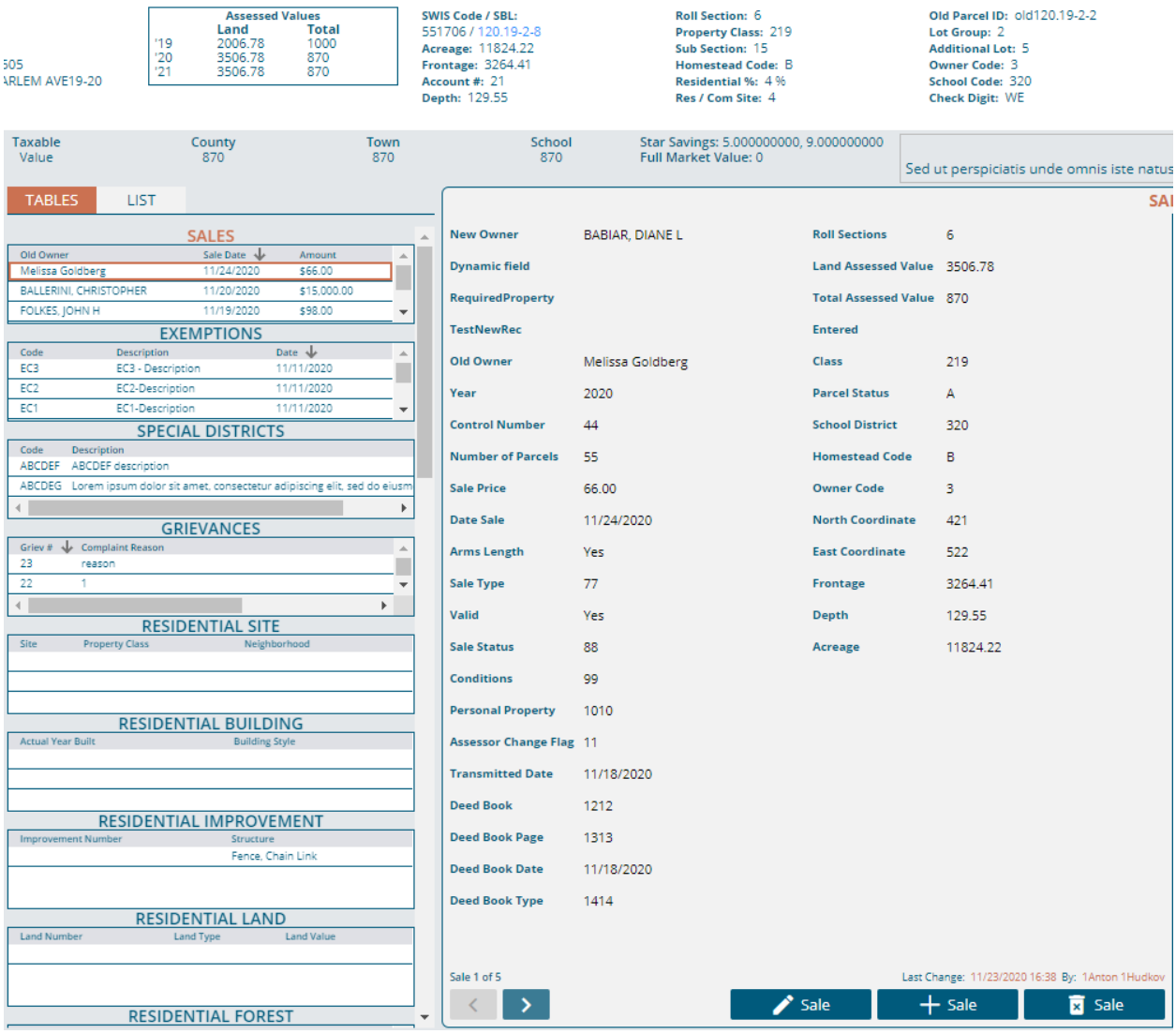

Рисунок 4.10 – Вид сторінки інформації про нерухомість

4.6 Швидкодія системи

Головною особливістю розробленої системи полягає в швидкодії пошуку нерухомості відносно подібним систем. Проаналізувавши швидкість пошуку різних систем маємо такі дані:

- час завантаження сторінки пошуку становить 946 мс, а запит пошуку займає 74 мс для розробленої системи;

- час завантаження сторінки пошуку становить 1290 мс для сайту американської компанії пошуку нерухомості Zillow, так як сайт не підтримує технологію SPA, то час на запит пошуку входить до часу завантаження

сторінки;

- для сайту компанії Trulia час завантаження сторінки пошуку становить 1340 мс, а час виконання запиту пошуку - 1200 мс;

- для сайту компанії Redfin час завантаження сторінки пошуку становить 3740 мс, а час виконання запиту пошуку - 725 мс.

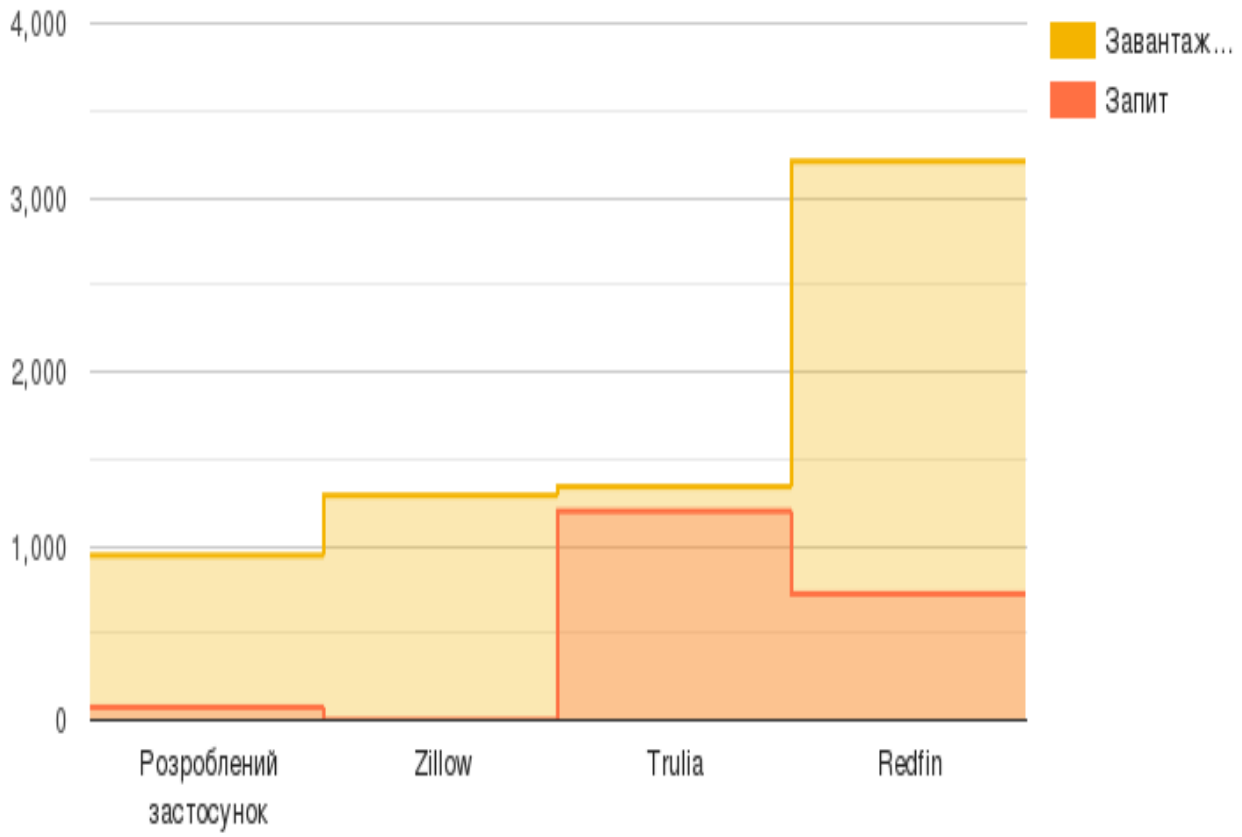

Рисунок 4.11 – Діаграма швидкості пошуку нерухомості у різних системах

Також для аналізу продуктивності розробленого застосунку відносно інших популярних систем було використано технологію компанії Google – Lighthouse. Результати аналізу наведено у таблиці 4.2 та у вигляді діаграми (рис. 4.12).

Таблиця 4.2 – Порівняння продуктивності сервісів

Критерії	Розроблений застосунок	Redfin	Zillow	Trulia
Продуктивність застосунку	87% що, згідно алгоритмів Lighthouse, означає що час від початку завантаження до можливості взаємодії складає менше 4.3 секунд	12% - що, згідно алгоритмів Lighthouse, означає що час від початку завантаження до можливості взаємодії більше 5.8 секунд	18% - що, згідно алгоритмів Lighthouse, означає що час від початку завантаження до можливості взаємодії складає більше 5.8 секунд	34% - що, згідно алгоритмів Lighthouse, означає що час від початку завантаження до можливості взаємодії в діапазоні 4.4–5.8 секунд
Доступність	89%	72%	86%	77%
Використання кращих методів розробки	93%	86%	79%	79%

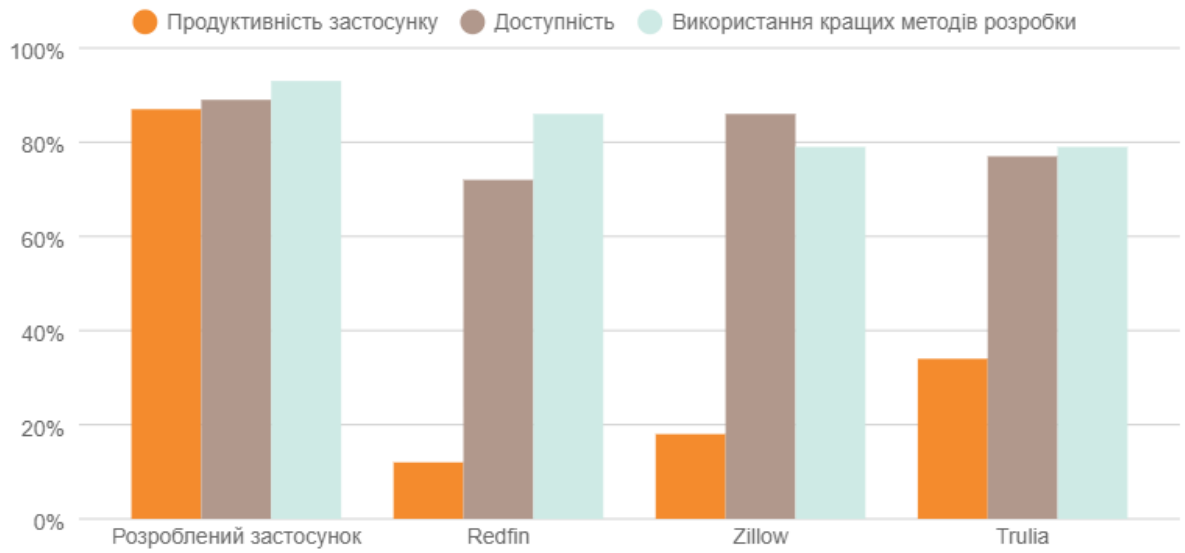

Рисунок 4.12 – Діаграма результатів оцінювання за допомогою Lighthouse

Проаналізувавши отримані оцінки Lighthouse, можна зробити висновок, що сучасні технології, підходи до розробки та архітектурні рішення, які були використані у процесі моделювання та розробки системи задовольняють потребам користувачів та мають переваги від існуючих систем.

4.7 Розроблені інструменти розширення об’єктів системи

Реляційні бази даних не надають розробнику динамічної зміни структури бази в процесі роботи додатку, але для кінцевого користувача може бути важливо доповнювати, або змінювати інформаційні поля об’єкту (в нашому випадку нерухомості). Для вирішення цієї проблеми було реалізовано певний метод розширення системи. Для кожної з баз даних, які використовуються у системі, розробник системи пише певний файл налаштування у форматі JSON. Цей файл містить список інформаційних полів, та їх налаштування, наведено у прикладі 4.1

```
"exemptions":[
        {
            "key":"exemptionStatus",
            "column":"ExemptionStatus",
            "label":"Exemption Status1",
            "enabled":true,
            "order":0,
            "dataType":4,
            "isRequired":false,
            "group":"OverallInformation"
        }…]
```

Приклад 4.1– Файл конфігурації системи

Для кожної сутності написані налаштування, які містять ключову назву поля у базі даних, порядок відображення на інформаційній сторінці системи, статус поля – активне чи не активне, тип даних для відображення, флаг обов'язковості. Цього файлу достатньо для налаштування вже існуючих інформаціоних характеристик та атрибутів системи. Для легкого розширення інформаційних об'єктів системи зі сторони користувача системи, базу даних доповнено такими таблицями бази даних (рис 4.13).

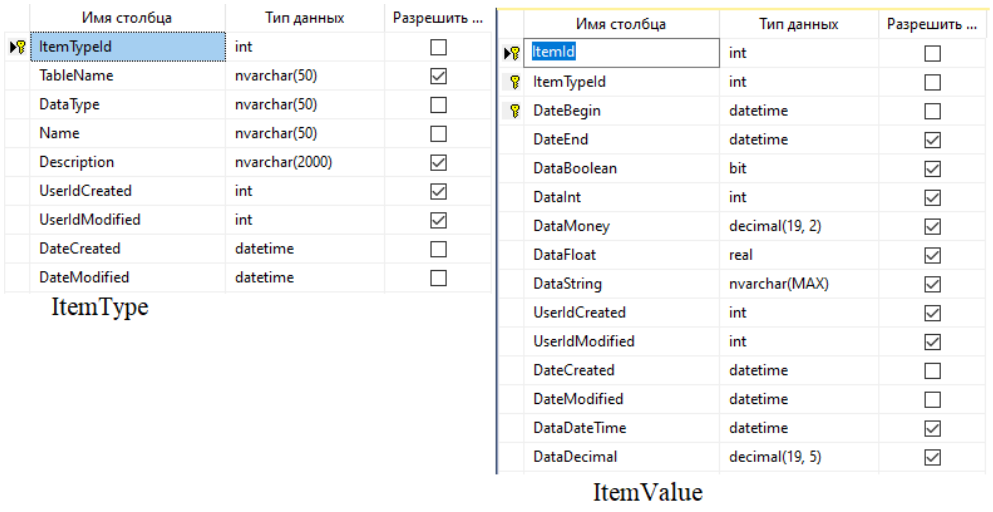

Рисунок 4.13 – Структура таблиць для реалізації динамічних полів

Наведені таблиці дозволяють для кожного інформаційного об'экту системи створити запис у таблиці ItemType, указавши назву таблиці сутності, назву поля та тип данних для використовування у системі.

Таблиця itemValue потрібна для того, щоб зберігати значення динамічних полів для певного єкземпляра сутності. Для цього указуємо унікальний ідентифікатор певного інформаційного об’єкта, унікальний ідентифікатор динамічного поля, та саме значення цього поля в один із стовпців таблиці, який відповідає певному типу даних.

Таким чином реалізован механізм за домогою, якого можна керувати наповненням інформаційних об’єктів системи управління каталогом нерухомості зі сторони користувача системою, а не розробника. При аналізі існуючих систем аналогічного підходу до реалізації подібного функціоналу не виявлено.

ВИСНОВОК

Для досягнення поставленої мети кваліфікаційної роботи було проаналізовано систему управління каталогом нерухомості, змодельовано таку систему, обрано програмне забезпечення, розроблено веб-застосунок системи, побудовано модель веб-сервісу, спроектовано базу даних. Результатом цих заходів став створений веб-сервіс системи управління каталогом нерухомості.

Проведений аналіз можливостей застосування сучасних технологій для управління каталогом нерухомості показав, що глобальна мережа Інтернет є найбільш придатним комунікаційним середовищем для організації системи управління каталогом нерухомості. Створений веб-застосунок у повній мірі використовує основні можливості новітніх технологій для вирішення прикладної задачі. Для більш швидкого розширення інформаційних об’єктів системи та реалізації усіх можливостей веб-застосунку було покращено методи розробки архітектури бази даних.

З технічної точки зору впровадження розробленої системи не вимагає великих економічних витрат. Об'єктно-орієнтований підхід, використаний в процесі розробки, дозволяє здійснювати подальшу модернізацію системи.

Функціональні можливості спроектованого та розробленого веб-застосунку можуть бути доопрацьовані та розширені у відповідності до вимог тієї сфери, де буде застосовано розроблену систему.

Практичне значення одержаних результатів полягає в запропонованих технологічних рішеннях удосконалення процесів побудови інформаційних систем, які можут бути використані розробниками програмного забезпечення при розробці веб-застосунків та при моделюванні систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методи та засоби мультимедійних інформаційних систем: навч. посіб. / Т. М. Басюк, П. І. Жежнич; Нац. ун-т «Львів. політехніка». – Львів: Вид-во Львів. політехніки, 2015. – 426 с. – Бібліогр.: с. 413 –416.

2. Касапов В.О., Носик А.М. Аналіз систем классифікації та пошуку інформаційних об’єктів [Текст] : Проблеми інформатизації: тези доповідей восьмої міжнародної науково-технічної конференції 26-27 листопада 2020 року. / Черкаси: ЧДТУ; Баку: ВА ЗС АР; Бельська-Бяла: УТГН; Харків: НТУ «ХПІ», 2020. – С. 51.

3. Объектно-ориентированный анализ и проектирование [Текст] / М. Бретт, П. Гери, У. Девид. – М. : Издательский дом «Питер», 2013. – 171 с.

4. Касапов В.О., Носик А.М. Методи побудови інформаційних систем [Текст] : Інформаційні проблеми теорії акустичних, радіоелектроиних і телекомунікаційних систем: тези доповідей дев’ятої міжнародної науково-технічної конференції 11-13 листопада 2020 року. / Харків: НТУ «ХПІ», 2020. – С. 61-62.

5. Архітектура програмного забезпечення [Електронний ресурс] : Режим доступу: https://soandso.biz/blog/software-engineering/arhitektura-programnogo-zabezpechennya.html

6. Разработка одностраничных веб-приложений [Текст] / М. Миковски, Д. Пауелл. – М. : ДМК Пресс, 2014. – 43 с.

7. Angular и TypeScript. Сайтостроенние для профессионалов [Текст] / Я.Файн – М. : Издательский дом «Питер», 2017. – 464 с.

8. Веб-приложения на JavaScript [Текст] / А. Маккоу. – М. : Издательский дом «Питер», 2012. – 56 с.

9. Usage statistics of web servers [Електронний ресурс] – Режим доступу : www / URL: https://w3techs.com/technologies/overview/web_server.

10. Шаблоны проектирования веб-приложений [Текст] / П. Вора. – М. : Эксмо, 2011. – 656 с

11. Тарасов, С. СУБД для программиста. База данных изнутри. – М.: ЗАО «Издательство Соломон», 2015. – 322 с.

12. C# и .NetCore. Кросс-платформенная разработка для профессионалов [Текст] / М. Прайс. – М. : Издательский дом «Питер», 2016. – 640 с.

13. Бойко Н.І. Моделювання Web-орієнтованих систем та напрямки розвитку WEB-ресурсів [Електронний ресурс] – Режим доступу до ресурсу: http://ena.lp.edu.ua:8080/bitstream/ntb/19246/1/3-Boyko-16-25.pdf