**АНОТАЦІЯ**

В дипломній роботі розроблено веб-орієнтовану інформаційну систему рієлторської компанії.

Веб-система надає змогу користувачам здійснити пошук об’єктів для купівлі чи оренди. Користувачі мають змогу зареєструватися в системі і подати заявку на продаж об’єкта або на здачу в оренду. Адміністратор системи має змогу редагувати описи будь-яких об’єктів, управляти довідниками та користувачами.

Для програмної реалізації веб-системи використано середовище розробки Microsoft Visual Studio 2015, мову програмування C#, технологію ASP.NET Web Forms, систему управління базами даних Microsoft SQL Server 2014, технологію доступу до даних .NET Entity Framework.

Обсяг роботи складає 95 сторінок.

В роботі крім основної частини наведено економічну оцінку проектного рішення.

**ANNOTATION**

In the thesis work web-based information system real estate company is developed.

Web system allows users to search for objects for purchase or lease. Users can log in and apply for the sale of an item or to rent. The system administrator can edit the description of any objects, manage vocabulary and users.

For software implementation of web-based system uses Microsoft Visual Studio 2015 development environment, the C# programming language, ASP.NET Web Forms technology, Microsoft SQL Server 2014 database management system, .NET Entity Framework data access technology.

The amount of work is 95 pages.

In addition to the work key part of the economic evaluation of the project design.

ЗМІСТ

[ВСТУП 7](#_Toc443655631)

[РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ЗАСОБІВ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНИХ СИСТЕМ 8](#_Toc443655632)

[1.1. Використання веб-орієнтованих систем 8](#_Toc443655633)

[1.2. Огляд аналогів до розроблюваного програмного забезпечення 9](#_Toc443655634)

[1.3. Сучасні засоби розробки веб-орієнтованих систем 15](#_Toc443655635)

[1.4. Висновки до розділу 17](#_Toc443655636)

[РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ РІЄЛТОРСЬКОЇ КОМПАНІЇ 18](#_Toc443655637)

[2.1. Постановка задачі розробки веб-системи рієлторської компанії 18](#_Toc443655638)

[2.2. Специфікація вимог до програмного забезпечення 19](#_Toc443655639)

[2.3. Обґрунтування вибраних технологій та засобів 25](#_Toc443655640)

[РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІАЦІЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ РІЄЛТОРСЬКОЇ КОМПАНІЇ 28](#_Toc443655641)

[3.1. Архітектура програмного забезпечення 28](#_Toc443655642)

[3.2. Структура програмного рішення 29](#_Toc443655643)

[3.3. Об’єктна модель програмного рішення 29](#_Toc443655644)

[3.4. Проектування графічного інтерфейсу користувача 32](#_Toc443655645)

[3.5. Опис функціональності розробленого програмного забезпечення 33](#_Toc443655646)

[РОЗДІЛ 4. ТЕСТУВАННЯ ТА СУПРОВІД ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ РІЄЛТОРСЬКОЇ КОМПАНІЇ 45](#_Toc443655647)

[4.1. Тестування програмного забезпечення 45](#_Toc443655648)

[4.2. План забезпечення якості веб-орієнтованої системи 47](#_Toc443655649)

[4.3. План впровадження програмного забезпечення 52](#_Toc443655650)

[4.4. План супроводу програмного забезпечення 54](#_Toc443655651)

[РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА 56](#_Toc443655652)

[5.1. Економічна характеристика проектного рішення 56](#_Toc443655653)

[5.2. Розрахунок витрат на розробку та впровадження програмного модулю для забезпечення електронної звітності 57](#_Toc443655654)

[5.3. Визначення комплексного показника якості 63](#_Toc443655655)

[5.4. Визначення експлуатаційних витрат 65](#_Toc443655656)

[5.5. Розрахунок ціни споживання проектного рішення 69](#_Toc443655657)

[5.6. Визначення показників економічної ефективності 70](#_Toc443655658)

[ВИСНОВКИ 73](#_Toc443655659)

[СПИСОК ЛІТЕРАТУРИ 74](#_Toc443655660)

[Додаток A. Модель бази даних 76](#_Toc443655661)

[Додаток Б. Схема бази даних 77](#_Toc443655662)

[Додаток В. Інструкція користувача 78](#_Toc443655663)

[Додаток Д. Лістинг файлу BuildingInfo.aspx 80](#_Toc443655664)

[Додаток Е. Тестові випадки використання програмного забезпечення 94](#_Toc443655665)

# ВСТУП

На сьогодні інформаційні технології використовуються чи не в кожній галузі. Їхній стрімкий розвиток сприяє утворенню все більше компаній, які використовують у своїй діяльності інформаційні системи. Мережа Інтернет використовується чи не в кожному другому програмному забезпечення. Тому на сьогодні є актуальним розробка веб-орієнтованих інформаційних систем.

Існує ряд програмних засобів для швидкої розробки інформаційних систем, їх налаштуванню та впровадження. Однак більшість таких систем має стандарний набір функціоналу та дизайну, і не є орієнтований на вузьке коло задач.

Метою дипломного проектування є розроблення сучасної веб-орієнтованої системи рієлторської компанії. Така система надасть змогу користувачам здійснювати пошук об’єктів для довготривалої і подобової оренди, а також об’єктів купівлі під бізнес та купівлі житла. Користувачі системи також будуть наділені змогою подати заявку на продаж об’єкта або здачу його в оренду. Адміністратор системи зможе управляти користувачами, довідниками, а також редагувати описи приміщень та будівель.

.

# РОЗДІЛ 1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ЗАСОБІВ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНИХ СИСТЕМ

## 1.1. Використання веб-орієнтованих систем

Сьогодні корпоративна інформаційна система стає не тільки системою забезпечення управління, а й інструментом підтримки корпоративних цінностей і нормативно-регламентної бази компанії. Важливою частиною інформаційної системи є корпоративний портал, який служить інструментом трансляції інформації, наприклад, доведення наказів і розпоряджень до виконавців [1].

Перш за все, під веб-системами розуміємо інструмент для досягнення бізнес-целій і реалізації бізнес-стратегії підприємства. Саме цей принцип диктує необхідність розуміння, який саме проект потрібен компанії для презентації себе в Інтернет. Комусь достатньо красивої візитки з динамічної анімацією, а комусь навпаки мінімум стрибаючих елементів, але максимум функціоналу і швидкості знаходження потрібної інформації. Сучасна веб-орінєнтована система має бути інтегрована в діяльність компанії [2].

Веб-орієнтовані системи, зважаючи на ряд їх переваг [3], є популярними чи не в кожній сфері людської діяльності. Сьогодні веб-орієнтовані інформаційні системи широко використовуються:

* бухгалтерський облік – класична область застосування інформаційних систем,
* управління фінансовими потоками;
* управління складом, асортиментом і закупівлями;
* управління виробничим процесом;
* управління маркетингом – збір та аналіз даних про стан ринку в своєму сегменті;
* документообіг;
* оперативне управління підприємством тощо.

## 1.2. Огляд аналогів до розроблюваного програмного забезпечення

На рис. 1.1 подано сторінку компаніії “Авеню”. Ця веб-система надає інформацію про об’єкти нерухомості, їх фотографії (рис. 1.2), контакти власників.

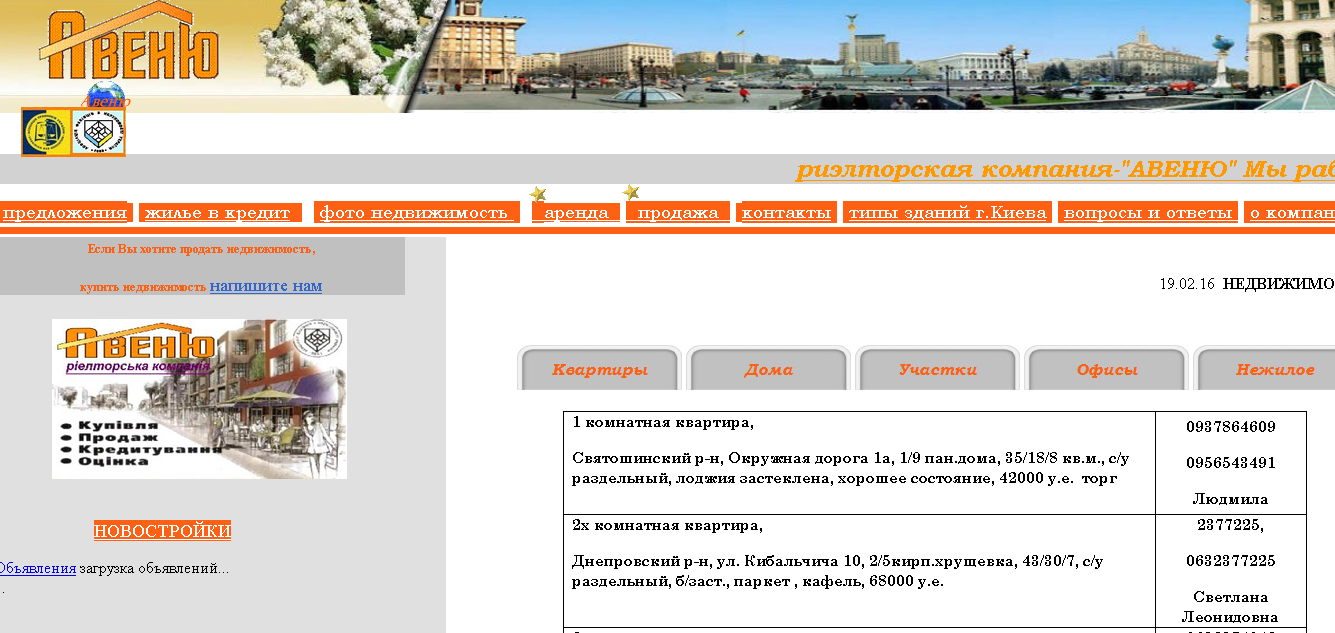


Рис. 1.1. Компанія “Авеню”

В системі розділено функціональність показу зображєнь об’єктів та їх опис, що є дивним рішенням, оскільки користувачу легше орієнтуватися щодо нерухомості, коли подано зображення самого об’єкта.

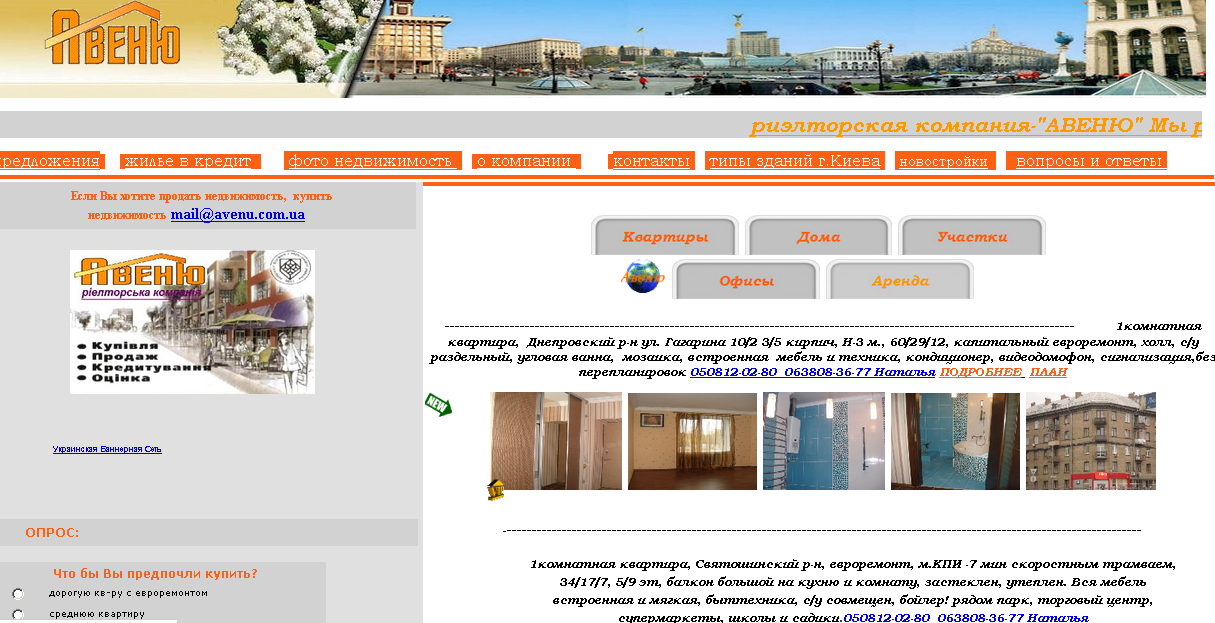


Рис. 1.2. Перелік фотографій об’єктів на сайті “Авеню”

Система не надає зручних засобів для пошуку об’єктів. Користувач немає змоги подати в режимі онлайн заявку на продаж нерухомості або здачі в оренду.

З переваг даної системи слід відзначити опис видів нерухомості і безпосередні контакти власників об’єктів.

На рис. 1.3 подано вигляд сайту компанії “Чайка”. Даний веб-сайт містить недостатньо інформації щодо об’єктів: відсутня інформація про зображення об’єкта, його тип, стан тощо.

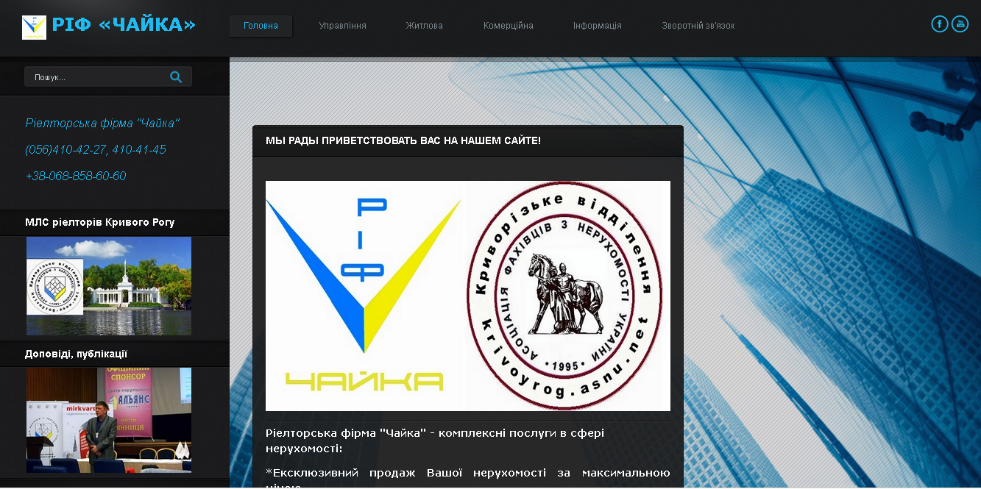


Рис. 1.3. Компанія “Чайка”

Недоліком системи є також відсутня можливість реєстрації користувача в системі а також не дружній дизайн сайту.

На рис. 1.4 подано вигляд сайту компанії “Дуоком”.

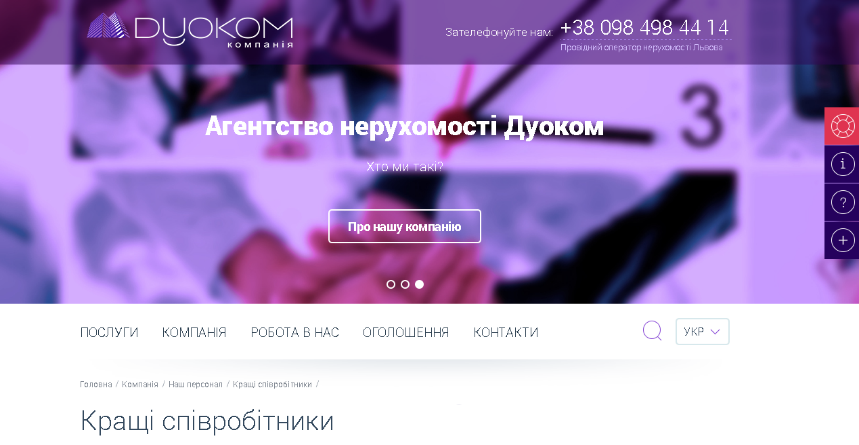


Рис. 1.4. Компанія “Дуоком”

На головній сторінці цієї компанії розміщено інформацію про успішних рієлторів компанії. Сайт містить сучасний дизайн. Користувач має змогу скористатися зручним пошуком об’єктів (рис. 1.5).

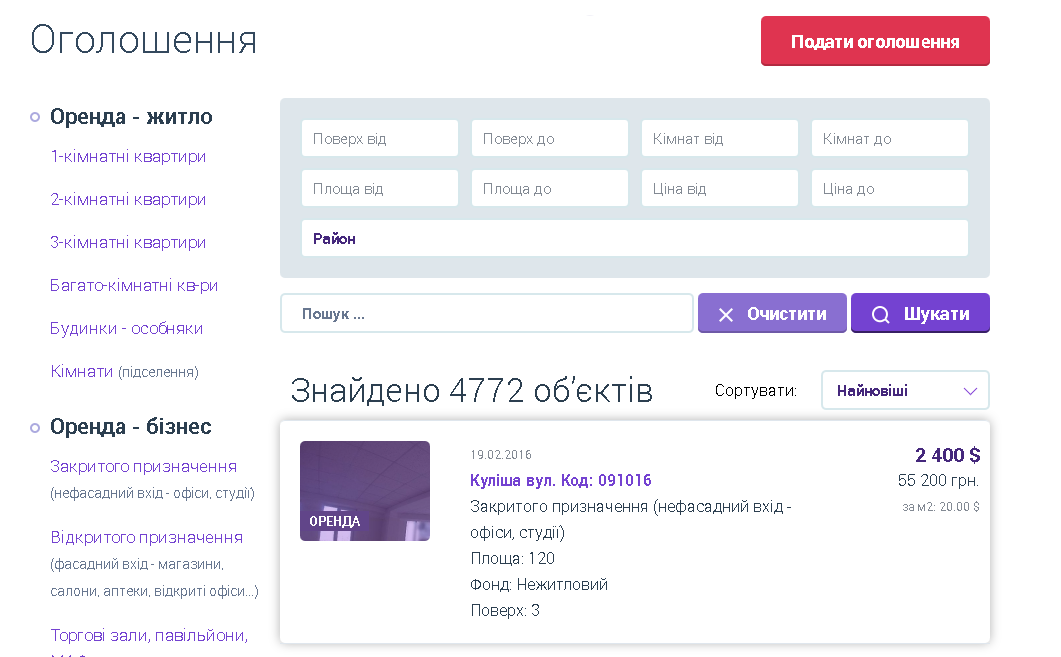


Рис. 1.5. Пошукова форма на сайті компанії “Дуоком”

Пошук об’єктів відбувається за такими полями: поверх, кількість кімнат, площа, ціна, район міста. Однак користувач немає змоги уточнити запит і знайти нерухомість певного виду або вказати площу кухні чи нежитлового приміщення.

На рис. 1.6 подано вигляд сторінки із детальним описом об’єкта нерухомості. Варто відзначити зручний інтерфейс користувача: інформація згрупована і розділена у дві колонки, подано декілька зображень об’єкта.

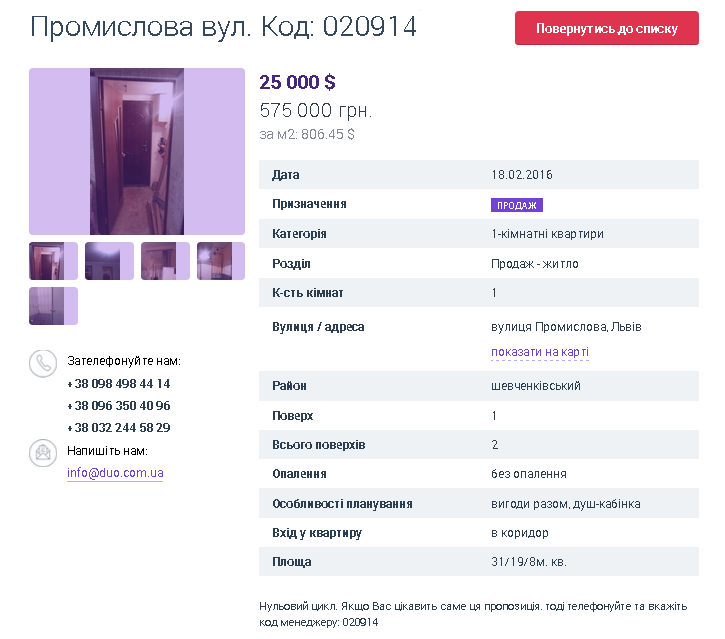


Рис. 1.6. Інформація про об’єкт на сайті компанії “Дуоком”

На рис. 1.7 подано сайт компанії “Траекторія”. Дана веб-система на головній сторінці містить меню для швидкої навігації до потрібного каталогу об’єктів нерухомості.

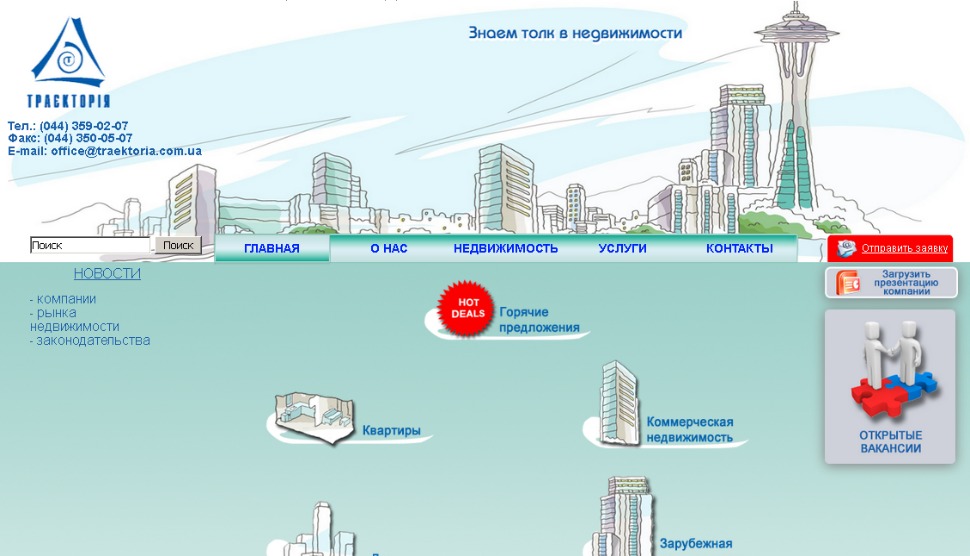


Рис. 1.7. Компанія “Траекторія”

З функціональної точки зору:

* сайт не надає зручного подання результатів пошуку – дані не структуровані, немає інфомації про тип, вид об’єкта тощо (рис. 1.8);
* немає зручних засобів для здійснення пошуку об’єктів;
* немає можливості зареєструватися користувачу в системі.

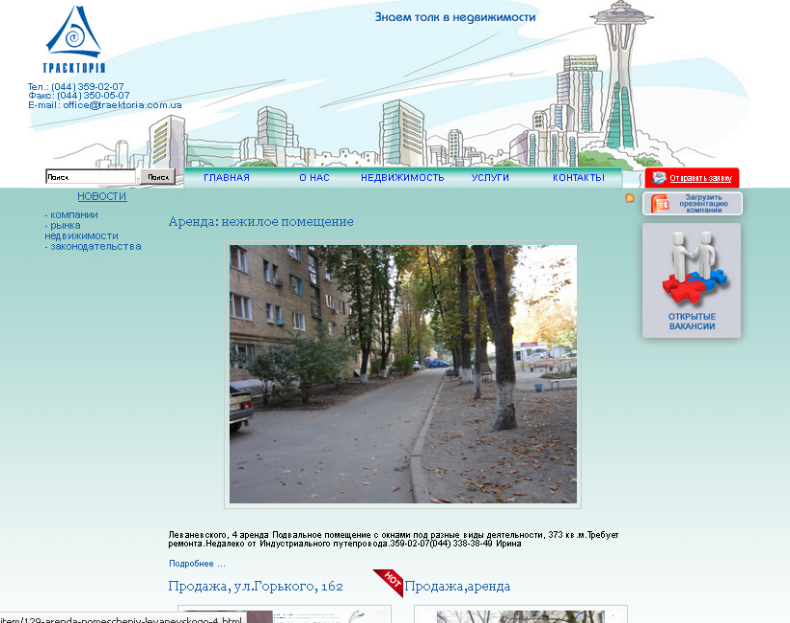


Рис. 1.8. Сторінка із результатами пошуку на сайті компанії “Траекторія”

Загалом, дизайн сайту цієї компанії не можна вважати дружнім, а функціональність яку він надає, це здебільшого статичні сторінки або погано структуровані динамічні дані про об’єкти нерухомості.

На рис. 1.9 подано вигляд сторінки компанії “Золоті ворота”. На головній сторінці розміщено список “гарячих” оголошень. Сайт має просту форму для пошук даних.

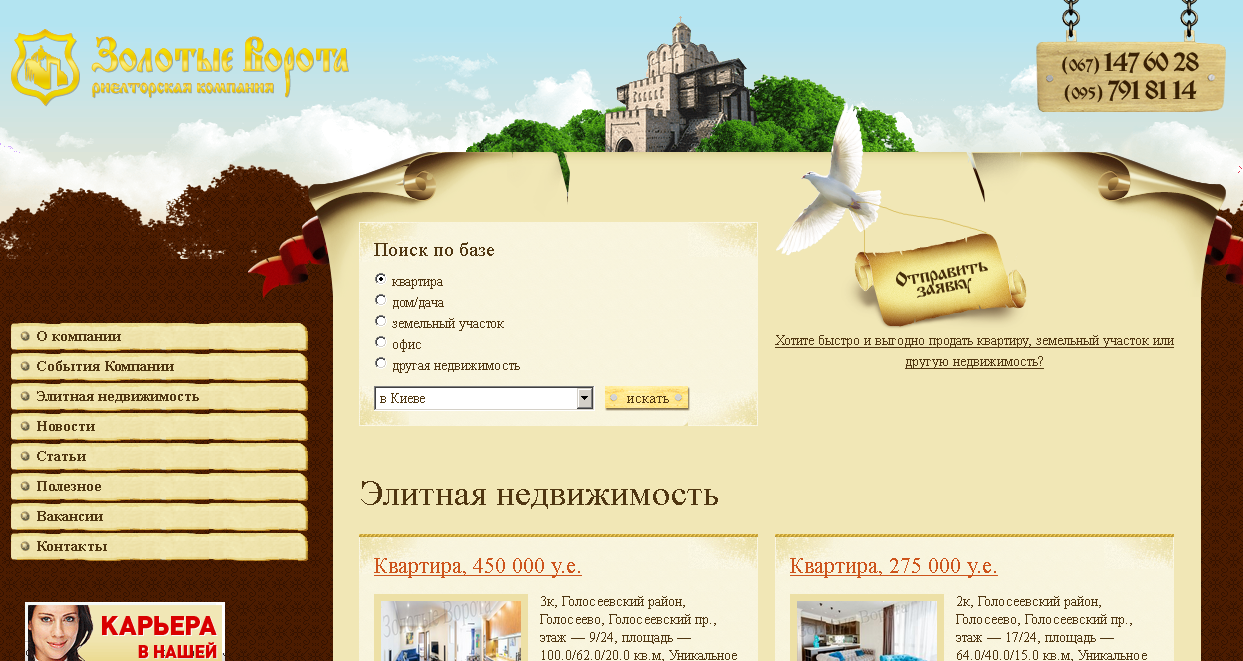


Рис. 1.9. Компанія “Золоті ворота”

На рис. 1.10 наведено пошукову форму даної компанії. Форма включає лише декілька параметрів: район, кількість кімнат, ціна, скільки результатів пошуку відображати на сторінці, і як відсортувати дані. Також користувач має змогу згрупувати дані по району міста.



Рис. 1.10. Пошукова форма на сайті компанії “Золоті ворота”

Натомість відсутня можливість знайти об’єкт нерухомості із вказаною загальною площею, площею кухні, площею кімнат чи площею нежитлових приміщень. Також відсутня можливість подати заявку на здачу в оренду або продаж об’єкта нерухомості онлайн.

На рис. 1.11 подано сторінку із результатами пошуку на сайті компанії “Золоті ворота”.

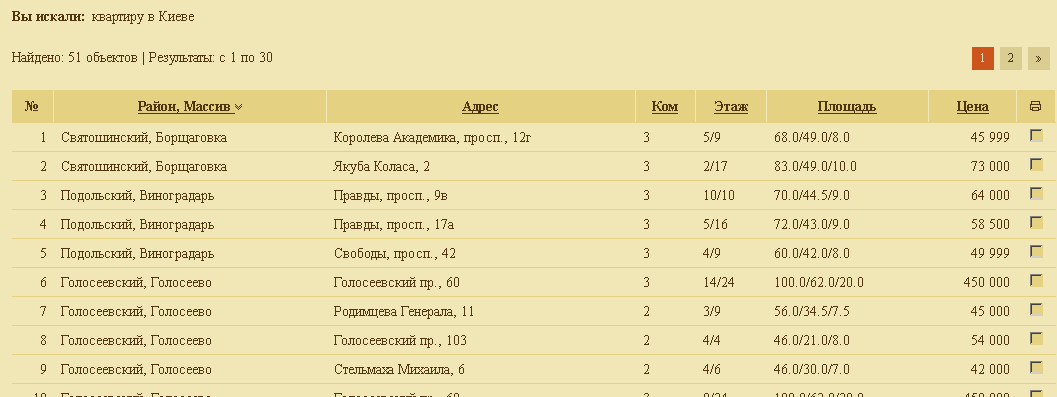


Рис. 1.11. Сторінка із результатами пошуку на сайті компанії “Золоті ворота”

На рис. 1.12 подано вигляд сторінки із детальним описом об’єкта. Слід відзначити, що крім фотографій самого об’єкта також подано план нерухомості.



Рис. 1.12. Детальний опис об’єкта на сайті компанії “Золоті ворота”

Недоліком також є не зручний інтерфейс користувача при подачі даної інформації – інформація сприймається важко, не згрупована, і не виділена на сторінці.

## 1.3. Сучасні засоби розробки веб-орієнтованих систем

На сьогодні існує ряд програмних засобів, які спрямовані на швидку розробку інформаційних систем. Системи управління вмістом (Content based management system, CMS) [9] дають змогу розробити веб-ресурс із стандартним набором функціональності. В подальшому такий веб-ресурс можна вдосконалювати із використанням сторонніх плагінів або програмуючи бажану функціональність.

CMS-системи класифікують на такі [10]:

* **Проста CMS система**. Система утворюється з програмних модулів, для кожної системи вони свої. Модулі одного разу налаштовуються розробником, тим самим жорстко закріплюється структура проекту. Для подальшої зміни структури сайту і його параметрів потрібна участь технічного персоналу. Від користувача системи потрібне знання основ HTML. Сумісність: система сумісна з певними платформами і типами СУБД. Спроба впровадження додаткових модулів, залежно від використовуваних технологій, може привести до повної переробки проекту. Технологічність: динамічне формування сторінок, обмежена пропускна здатність – до 3 000-5 000 відвідувачів на добу. Система інсталюється розробником. Розвиток системи досягається шляхом перекладу сайту на нову, більш пристосовану до поточних завдань, версію системи управління контентом.
* **Шаблонна CMS система**. Єдиний модуль або набір модулів з жорстко закріпленою структурою сайту. Система містить набір сервісних функцій, що дають змогу виконати стандартні дії з сайтом: сформувати стрічку новин, створити / видалити новий розділ, вибрати шаблон для роботи з інформацією, завантажити файл і розставити по тексту картинку, встановити атрибути тексту і т.д. Система сумісна з певними платформами і типами СУБД. Спроба впровадження додаткових модулів також, як і в попередньому випадку, обмежена. Інсталюється розробником. Удосконалення досягається шляхом перекладу сайту на нову, більш складну, версію системи управління контентом.
* **Професійна CMS-система**. Інтуїтивно зрозумілий інтерфейс, візуалізація і розширені можливості редагування. Можливість зміни структури проекту – створення найрізноманітніших сайтів. Сумісність з різними програмно-апаратними платформами Можливість підключення додаткових модулів як від розробника, так і прикладного програмного забезпечення. Можливість швидкого підключення додаткових модулів без порушення структури та ідеології управління веб-ресурсом.
* **Універсальна CMS-система**. Має засоби управління контентом, налаштування функціональності системи, можливість перевизначення атрибутів об'єктів сайту. Система надає засоби для розробки нових сервісів і можливостей. Наявність сертифікованої системи забезпечення безпеки – розмежування прав доступу до системи на внутрішньокорпоративному рівні. Пропускна здатність – не обмежена.

До відомих і найбільш поширених CMS відносяться такі [11]:

* WordPress – безкоштовна CMS, базована на мові PHP.
* Joomla! – безкоштовна CMS, базована на мові PHP.
* Drupal – безкоштовна CMS, базована на мові PHP.
* ExpressionEngine – комерційна CMS, базована на PHP.
* TextPattern – безкоштовна CMS, базована на мові PHP.
* Contao (TYPOlight) – безкоштовна CMS, базована на мові PHP.
* SilverStripe – безкоштовна CMS, базована на мові PHP.
* Umbraco – безкоштовна CMS, базована на технології .NET.
* concrete5 – безкоштовна CMS, базована на мові PHP.
* CushyCMS – безкоштовна CMS.

## 1.4. Висновки до розділу

Таким чином актуальним є розробка веб-орієнтованої інформаційної системи рієлторської компанії. Така система буде мати зручний інтерфейс користувача, надавати йому засоби для ефективного пошуку об’єктів різного типу і форми нерухомості, а також подачу заявок на здачу в оренду чи продаж.

Крім цього веб-система буде використовувати сучасні інформаційні технології, мати адміністративну панель управління користувачами та описом об’єктів.

# РОЗДІЛ 2. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ РІЄЛТОРСЬКОЇ КОМПАНІЇ

## 2.1. Постановка задачі розробки веб-системи рієлторської компанії

Метою дипломного проектування є розроблення веб-орієнтованої інформаційної системи рієлторської компанії.

Веб-система має забезпечити такі можливості:

* авторизація користувача в системі;
* реєстрація користувача в системі;
* пошук будівлі із використанням фільтрів;
* перегляд детального опису нерухомості;
* додання заявки на продаж нерухомості;
* додання заявки на здачу в оренду нерухомості;
* управління користувачами;
* управління довідниками.

Веб-система має мати зручний користувацький інтерфейс та використовувати сучасні технології.

Розроблювана веб-система має забезпечувати такі внутрішні та зовнішні характеристики програмного продукту:

* функціональна придатність;
  + ефективність;
  + надійність;
  + придатність.

Вхідними даними до розробки програмного забезпечення є:

* специфікація вимог замовника;
* документація технології ASP.NET WebForms;
* документація мови програмування C#;
* теорія баз даних.

Вихідними даними є веб-орієнтована інформаційна система рієлторської компанії.

## 2.2. Специфікація вимог до програмного забезпечення

### 2.2.1. Вступ

**2.2.1.1. Призначення, мета**

Метою розроблюваного програмного забезпечення є створення веб-системи для обліку заявок користувачів на здачу приміщення в оренду чи на продаж. Також веб-система має надавати зручні засоби для пошуку потрібної інформації.

**2.2.1.2. Продукти-аналоги**

Аналогами до розроблюваного програмного забезпечення є такі існуючі веб-системи: Park Lane [4], Чайка [5], Компанія Дуоком [6], компанія “Траекторія” [7], компанія “ Золоті ворота” [8] тощо.

### 2.2.2. Загальний опис

**2.2.2.1. Характеристика продукту**

Веб-система має забезпечути реєстрацію та авторизацію користувача в системі; пошук будівлі із використанням фільтрів; перегляд детального опису будівлі; додання заявки на продаж чи здачу в оренду будівлі; управління користувачами; управління довідниками.

**2.2.2.2. Класи користувачів та їх характеристики**

В системі передбачено такі класи користувачів: анонімний користувач, авторизований користувач, адміністратор системи.

1. Анонімний користувач має змогу (рис. 2.1):

* зареєструватися в системі;
* здійснити авторизацію в системі;
* здійснити пошук потрібного об’єкта;
* переглянути детальний опис будівлі;
* переглянути статичні сторінки веб-системи;

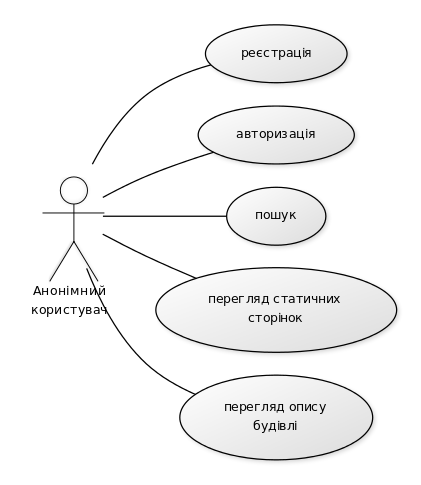


Рис. 2.1. UML діаграма прецедентів для анонімного користувача

2. Авторизований користувач, окрім дій анонімного користувача, має змогу:

* подати заявку на продаж будівлі або приміщення;
* подати заявку на здачу в оренду будівлю або приміщення.

Адміністратор може переглянути подані заявки і змінити їх дані.

На рис. 2.2 наведено діаграму прецедентів для ролі “Авторизований користувач”.

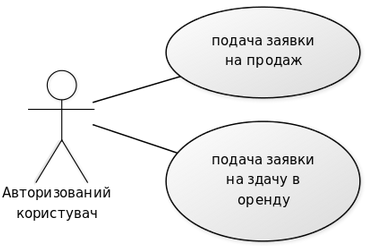


Рис. 2.2. UML діаграма прецедентів для авторизованого користувача

На рис. 2.3 наведено діаграму прецедентів для ролі “Адміністратор”.

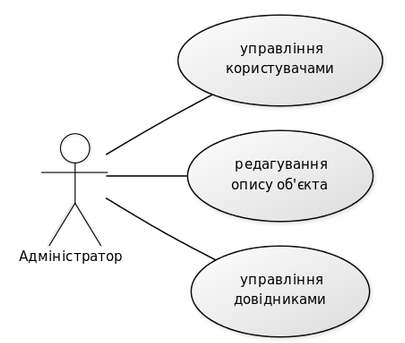


Рис. 2.3. UML діаграма прецедентів для адміністратора системи

3. Адміністратор системи, окрім можливостей анонімного та авторизованого користувачів, має змогу:

* управляти користувачами;
* редагувати опис об’єкта;
* управляти довідниками.

Довідники включають в себе такі сутності: форма об’єкта, тип об’єкта, стан об’єкта, область, місто, район міста.

**2.2.2.3. Середовище функціонування**

### Для взаємодії із веб-системою користувачу потрібно мати в наявності останню версію будь-якого переглядача.

Для серверної частини потрібно:

1. операційна система сімейства Windows;
2. система управління базами даних SQL Server;
3. веб-сервер IIS;
4. фреймворк .NET.

### 2.2.3. Характеристики програмного забезпечення

#### 2.2.3.1. Пошук об’єкта

**2.2.3.1.1. Опис і пріоритет**

Користувач має змогу здійснити пошук об’єкта, використовуючи фільтрування за параметрами об’єкта.

Пріоритет: високий.

**2.2.3.1.2. Послідовності дія/відгук**

Будь-який відвідувач веб-системи має змогу здійснити пошук потрібного об’єкта, із використанням фільтрування за його параметрами. Для цього користувач має відкрити головну сторінку системи, обрати один із пунктів меню, що його цікавить: “Продаж”, “Оренда”, “Купівля”, “Здача в оренду”.

**2.2.3.1.3. Функціональні вимоги**

REQ-1.1: Після вибору конкретного меню користувачу у відповідні поля типу операції над об’єктами підставляються відповідні значення.

REQ-1.2: Результати пошуку відображаються на тій же сторінці.

REQ-1.3: Користувач має змогу здійснювати фільтрування за одним або багатьма характеристиками об’єкта.

REQ-1.4: Після очищення заданих параметрів на формі пошуку, користувачу в якості результатів відображаються актуальні записи.

REQ-1.5: Результати пошуку містять адресу об’єкта, ціну, тип та інші характеристики.

#### 2.2.3.2. Перегляд детального опису об’єкта

**2.2.3.2.1. Опис і пріоритет**

Користувач має змогу переглянути детальний опис об’єкта.

Пріоритет: високий.

**2.2.3.2.2. Послідовності дія/відгук**

Будь-який користувач має змогу переглянути детальний опис об’єкта. Для цього користувачу потрібно відкрити головну сторінку системи і здійснити пошук потрібного об’єкта. Далі потрібно натиснути посилання «детально» або зображення об’єкта. Після цих дій користувачу буде відоображено сторінку із детальним описом вибраного об’єкта.

**2.2.3.2.3. Функціональні вимоги**

REQ-1.1: Анонімний та авторизований користувач має змогу лише переглядати детальний опис будь-якого об’єкта.

REQ-1.2: Авторизований користувач має змогу створити заявку на продаж або здачу в оренду приміщення.

REQ-1.3: Адміністратор має змогу керувати описом будь-якого об’єкта.

REQ-1.4: Якщо сторінку опису товару переглядає не адміністратор, то всі кнопки на сторінці не відображаються.

REQ-1.5: Для анонімного користувача у полі площа кімнат відобрається текстова стрічка із відповідними даними.

REQ-1.6: Для адміністратора у полі площа кімнат відображається меню для створення та внесення змін для даних про площу кімнат.

#### 2.2.3.3. Управління довідниками

**2.2.3.3.1. Опис і пріоритет**

Адміністратор системи має змогу здійснити управління довідниками.

Пріоритет: середній.

**2.2.3.3.2. Послідовності дія/відгук**

Авторизувавшись, адміністратор має натиснути у верхній частині на посилання “Сторінка адміністратора”. Після цього адміністратор має обрати потрібний довідник, натиснувши на відповідне посилання.

**2.2.3.3.3. Функціональні вимоги**

REQ-1.1: Тільки адміністратор може змінювати дані довідників.

REQ-1.2: Якщо сесія закінчилась, і користувач не розпізнається, для доступу до довідників потрібно вимагати авторизацію.

REQ-1.3: До довідників відносяться: інформація про області, міста, райони міста, інформація про користувачів, про тип, форму і статус об’єкта.

#### 2.2.3.4. Формування заявки на продаж або здачу в оренду

**2.2.3.4.1. Опис і пріоритет**

Користувач системи має змогу сформувати заявку на продаж або здачу в оренду приміщення або будівлі.

Пріоритет: високий.

**2.2.3.4.2. Послідовності дія/відгук**

Для подачі заявки на продаж або здачу в оренду користувач має авторизуватись в системі. Якщо користувач немає облікового запису, він має спершу зареєструватись в системі. Після цього, на головній сторінці системи, користувач має обрати один із пунктів додаткового меню.у центральній сцені: “продаж” або “оренда”. Після цих дій користувачу буде відображена сторінка із формою, яку має заповнити користувач. Після цього користувач має підтвердити заявку, натиснувши кнопку «Зберегти».

.**2.2.3.4.3. Функціональні вимоги**

REQ-1.1: Тільки авторизований користувач має змогу створити заявку.

REQ-1.2: Сторінка для внесення заявки містить всі потрібні поля.

### 2.2.4. Вимоги зовнішніх інтерфейсів

#### 2.2.4.1. Користувацькі інтерфейси

Не передбачаються.

#### 2.2.4.2. Програмні інтерфейси

Не передбачаються.

### 2.2.5. Інші нефункційні вимоги

**2.2.5.1. Вимоги продуктивності**

Веб-система має передбачати одночасну взаємодію 100 користувачів.

**2.2.5.2. Вимоги безпеки**

Потрібно зберігати паролі користувачів у захищеному режимі.

Потрібно використовувати ролі для розділення прав доступу до даних.

**2.2.5.3. Атрибути якості програмного продукту**

Доступність – веб-система має бути доступною 24 годин на добу.

Зручність використання – веб-система має володіти зручним та зрозумілим інтерфейсом користувача.

## 2.3. Обґрунтування вибраних технологій та засобів

Для програмної реалізації веб-системи обрано такі технології та інструментальні засоби:

* мова програмування C#;
* середовище розробки Microsoft Visual Studio 2015;
* технологія ASP.NET WebForms;
* система управління базами даних Microsoft SQL Server 2014;
* технологія доступу до даних ADO.NET Entity Framework;
* таблиці каскадних стилів CSS.

Мова програмування C# на сьогодні є однією із найбільш популярних мов для розробки різного роду застосунків [12]. Веб-аплікації, десктнопні проекти, служби, консольні проекти – все це успішно можна реалізувати на цій мові. Мова має схожий до java синтаксис, не використовує вказівників, є більш типізовано бзепечною чим java. Однак основною перевагою мови C# є використання фреймворку .NET Framework, який займається збором сміття і оптимізує використання пам’яті застосуванням [13]. Використання у мові збірок дає змогу забути про контроль версіонування коду.

Середовище розробки Microsoft Visual Studio обрано зважачи на те, що це середовище дає змогу швидко розробляти потужні застосування. Середовище дає змогу використовувати підхід RAD для розробки програмного забезпечення. Крім цього, середовище має потужний інструмент підказок, аналізу коду, формуванню метрик коду тощо [14, 15].

Технологія ASP.NET WebForms є розвитком класичної технології активних серверних сторінок – ASP [16]. Ця технологія увібрала в себе найкращі характеристики ASP та використовує фреймворк .NET Framework. Технологія ASP.NET має альтернативну технологію – ASP.NET MVC, яка працює на основі архітектурного шаблону model-view-controller (модель-представлення-контроллер). На сьогодні ці два підходи майже однаково широко застосовуються [17].

Система управління базами даних Microsoft SQL Server – це система, яка дає змогу безпечно зберігати, вносити, видаляти, оновляти, аналізувати великі масиви даних [18]. На сьогодні цей програмний продукт вдало конкурує із СКБД від компанії Oracle. СКБД Microsoft SQL Server наділена всіма зручними засобами для автоматизації адміністративних рішень, підтримує реплікації даних, дзеркалювання, кластерування, доставку журналів транзакцій тощо. Ця СКБД успішно використовується як у невеликих за розміром компаній, так і у великих за сучасними мірками корпораціях [19, 20].

Технологія .NET Entity Framework – це Object Relation Mapping (ORM), яка дає змогу побудувати логічну модуль до бази даних і працювати в застосуванні не з фізичним доступом до бази даних, а з його віртуальним представленням. Ця ORM дає змогу більше зосередитися на коді, чим на написанні програмного коду для обробки різноманітних полів та їхніх зв’язків. Для ефективності та уніфікацїі запитів до джерел даних, в Entity Framework часто використовують LINQ запити.

Для реалізації графічного інтерфейсу користувача використано мову розмітки html та каскадні таблиці стилів CSS. Використання цих засобів, а також майстер-сторінки дає змогу побудуваи зручне і зрозуміле для користувача застосування.

# РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІАЦІЯ ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ РІЄЛТОРСЬКОЇ КОМПАНІЇ

## 3.1. Архітектура програмного забезпечення

Для розробки веб-системи рієлторської компанії було обрано один із найбільш-популярних шаблонів розробки програмного забезпечення – клієнт-сервер. Даний шаблон передбачає взаємодію трьох компонент: клієнта, сервера застосування та сервера баз даних. У залежності від того, чи робляться додаткові значні обрахунки на стороні клієнта чи ні, розрізняють модель товстого або тонкого клієнта відповідно.

На рис. 3.1 подано схему взаємодії цих компонент. Зауважимо, що у випадку незначного завантаження сервера застосувань та сервера баз даних, ці компоненти можна розмістити на одному сервері.

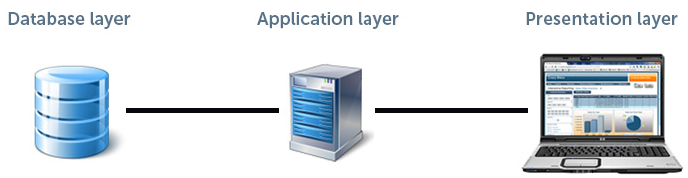


Рис. 3.1. Трирівнева архітектура клієнт-сервер

У зв’язку із тим, що не передбачається додаткова обробка даних на стороні клієнта, для програмної реалізації веб-системи було обрано саме цей підвид шаблону клієнт-сервер.

Таким чином, для взаємодії із застосуванням кінцевому користувачу потрібно буде мати в наявності лише останню версію будь-якого переглядача.

Також, використання трирівневої архітектури надасть змогу в майбутньому легко масштабувати всю систему.

## 3.2. Структура програмного рішення

На рис. 3.2 наведено структуру проектного рішення.

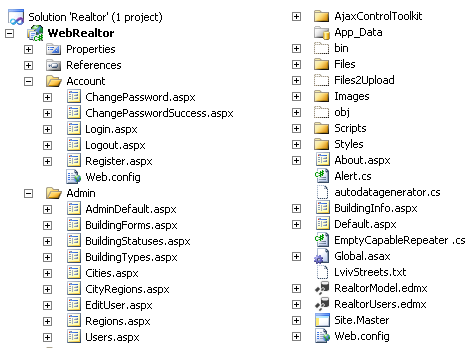


Рис. 3.2. Структура проектного рішення

Проект складається із майстер-сторінки **Site.Master**, сторінок сайту, файлу загальних налаштувань **Web.Config**, каскадних таблиць стилів, javascript та зображень.

В проекті створено два підключення до бази даних із використанням ADO.NET Entity Framework.

## 3.3. Об’єктна модель програмного рішення

На рис. А.1 наведено розроблену модель бази даних, яка містить основні об’єкти. Модель програмного рішення розроблялась у відповідності до принципку Database First, згідно якого спочатку проектується фізична база даних.

Для зберігання інформації про користувачів використано стандартний функціонал ASP.NET – клас **Membership**. Для розділення інформації про будівлі та користувачів використано дві окремі бази даних. База даних **Realtor** містить інформацію про будівлі, база даних **RealtorUsers** – містить стандартний набір таблиць (**aspnetdb**) для роботи із **Membership** та класом **Roles**.

Розроблено такі сутності:

* **Building** – головна таблиця, що містить опис об’єкту.
* **BuildingFloor** – таблиця, що містить поверхи, які відносяться до об’єкта.
* **BuildingForm** – форма об’єкта: житлова, нежитлова тощо.
* **BuildingImage** – зображення об’єкта.
* **BuildingStatus** – стан об’єкта: євроремонт, задовільний, відмінний тощо.
* **BuildingType** – тип об’єкта: виробничий, офіс, торгівельний, незавершений, квартира, будинок тощо.
* **City** – місто.
* **CityRegion** – район міста.
* **Region** – область.
* **RoomSquare** – площа кімнат об’єкта.
* **Type** – тип операції над об’єктом: продаж під бізнес, продаж, оренда довготривала, оренда подобова тощо.

З метою об’єднання декількох видів приміщень в один загальний об’єкт розроблено сутність Building, яка є основним базовим об’єктом для всіх підвидів будівель, приміщень тощо.

На рис. 3.3 подано сутність Building, яка представляє будівлю або приміщення з такими полями:

* BuildingID – унікальний ідентифікатор будівлі, приміщення тощо.
* StreetAddress – адреса, за якою розташований об’єкт.
* BuildingTypeRef – ідентифікатор типу будівлі (виробниче, офіс, торгівельне, незавершене тощо)
* TypeRef – унікальний ідентифікатор типу операції над об’єктом (продаж під бізнес, продаж, оренда довготривала, оренда подобова).



Рис. 3.3. Сутність Building з полями

* TotalRoomNumber – загальна кількість кімнат. Тип може містити значення NULL.
* Price – містить ціну в грн.
* Code – містить код об’єкта.
* Comment – містить коментар до об’єкта.
* PhonePresent – містить інформацію, чи є телефон на об’єкті.
* KitchenSquare – визначає площу кухні. Може бути NULL-значення.
* TotalSquare – визначає загальну площу приміщень.
* NonLivingSquare – визначає загальну площу нежитлових приміщень
* TotalFloor – визначає кількість загалом поверхів у об’єкті.
* DateTimeModified – визначає останню дату внесених змін;
* UserName – визначає обліковий запис користувача, який останнім вніс зміни у відповідну таблицю.
* ContactPhone – представляє контактний телефон.
* BuildingStatusRef – визначає стан об’єкта (євроремонт, старий, новий тощо).
* IsHot – визначає, чи товар є гарячим, себто таким, на який пошируються знижки та акції.
* CityRegionRef – унікальний ідентифікатор, який призначений для заданеня району обраному місту.

На рис. 3.4 наведено UML діаграму класів.



Рис. 3.4. UML діаграма класів

На рис. Б.1 подано сформовану схему бази даних. В додатку В подано інструкцію з використання розробленого програмного забезпечення.

## 3.4. Проектування графічного інтерфейсу користувача

На рис. 3.4 подано вигляд прототипу верхнього банера. Він містить такі об’єкти: логотип компанії, головне меню, компоненту для відображення стану авторизованого користувача тощо.

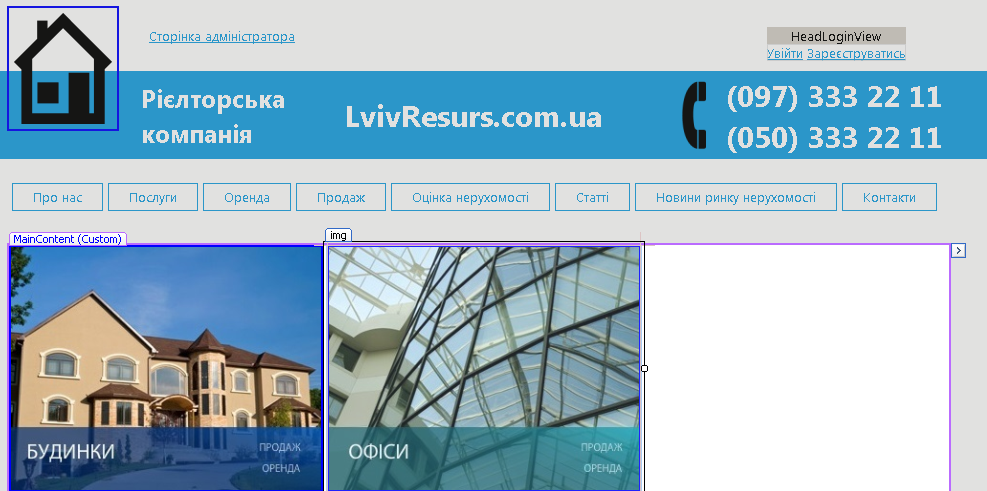


Рис. 3.4. Прототип верхнього банера сторінки

На рис. 3.5 подано вигляд прототипу нижнього банера. Цей банер містить інформацію про гарячі пропозиці, інформацію про останні надходження у базу даних тощо (рис. 3.5).

В центральній сцені розміщені блоки для знайдених зображень. Для проектування графічного інтерфейсу користувача використано CSS, CSS3, HTML тощо.

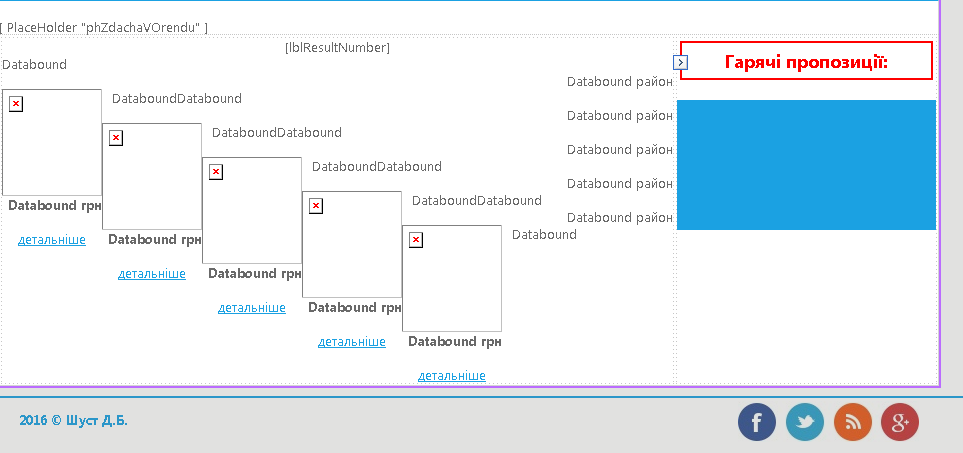


Рис. 3.5. Прототип нижнього банера сторінки

## 3.5. Опис функціональності розробленого програмного забезпечення

На рис. 3.6 подано вигляд головної сторінки для неавторизованого користувача. Такий користувач має змогу переглянути статичні сторінки сайту, здійснити пошук потрібного об’єкту, а також переглянути гарячі пропозиції.



Рис. 3.6. Головна сторінка для неавторизованого користувача

На рис. 3.7 подано вигляд останніх доданих записів та гарячі пропозиції.

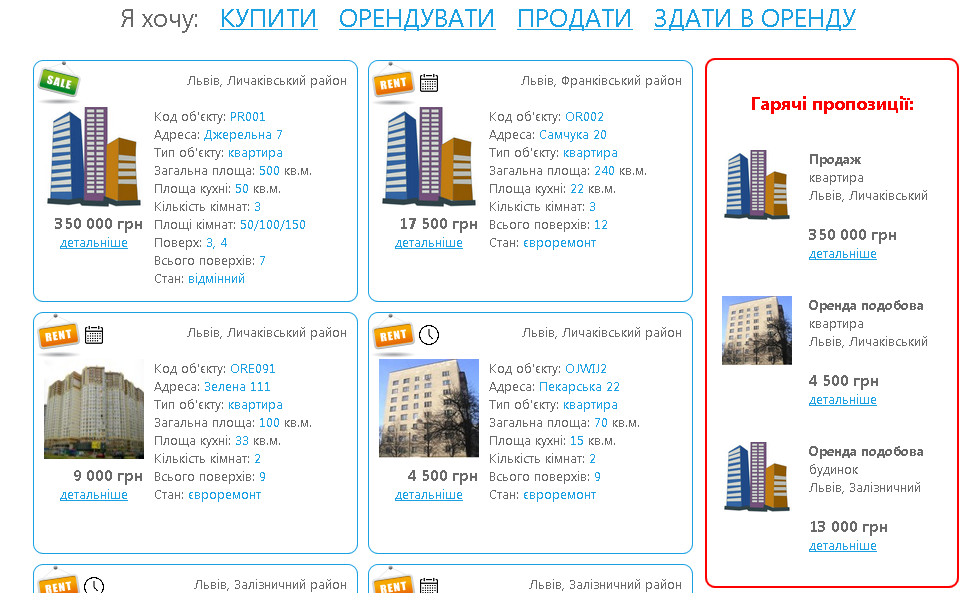


Рис. 3.7. Останні додані записи та гарячі пропозиції

Для кожного запису відображується за наявності його фотографія, або фотографія за замовчуванням, код об’єкту, адреса, район, місто, тип об’єкту, загальна площа, площа кухні, площа кімнат, кількість кімнат, кількість поверхів, стан об’єкта, ціна об’єкта.

Для візуалізації результатів, у залежності від типу операції над об’єктом – продаж, оренда, тощо – у верхньому правому куті відображається відповідна піктограма.

Користувач може переглянути статистику даних (рис. 3.8) – скільки всього є об’єктів у базі, скільки додано за останній місяць та тиждень.

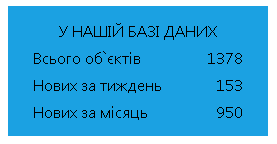


Рис. 3.8. Статистична інформація

Для цього використано такий програмний код:

var iYearNow = DateTime.Now.Year;

var iMonthNow = DateTime.Now.Month;

var startOfWeek = DateTime.Now.AddDays((int) DateTime.Now.DayOfWeek\*-1).AddDays(1);

var endOfWeek = startOfWeek.AddDays(6);

lblTotal.Text = string.Format(msg,

context.Building.Count().ToString(),

context.Building.Count(i => i.DateTimeModified >= startOfWeek &&

i.DateTimeModified <= endOfWeek),

context.Building.Count(i => i.DateTimeModified.Year == iYearNow

&& i.DateTimeModified.Month == iMonthNow));

З метою тестування функціональності веб-системи було розроблено функцію, яка в автоматичному режимі випадковим чином згенерувала дані 1375 об’єктів.

На рис. 3.9 подано вигляд пошукової форми, яка відображається на головній сторінці, якщо користувач натискає на посилання «я хочу купити» або «я хочу орендувати». Після натискання на посилання користувачу відображається форма для пошуку та автоматично згенеровані результати пошуку для запиту купівлі або оренди. У пошуковій формі використано автоматичну підказу для поля адреси. Також для полів місто, тип об’єкта, район використано автоматичне завантаження даних на основі області, форми об’єкта, міста відповідно.

Форма містить для числових полів два поля вводу – для задання мінімально можливого значення і максимально можливого. Поля опрацьовуються окремо, тому є можливість задати або лише нижню межу, або лише верхню, або і верхню і нижню межі одночасно.

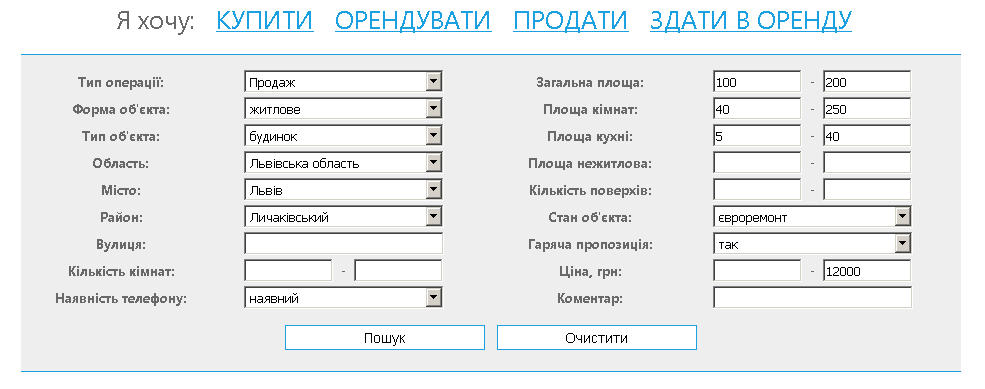


Рис. 3.9. Форма для пошуку

На рис. 3.10 наведено результати пошуку: знайдено 2 записи. Після натискання на кнопку «Пошук» користувачу відобраються релевантні результати, які розбиті на сторінки по 6 об’єктів. Якщо записів більше 6 – відображається компонента посторінкового перегляду. Також з цієї вибірки вибираються перші 5 об’єктів, які входять до гарячих пропозицій.

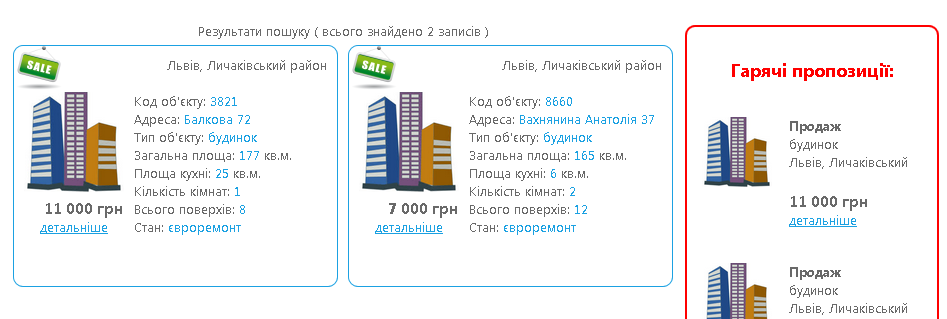


Рис. 3.10. Результати пошуку

Після натискання на кнопку «Очистити» користувачу знову відображаються дані про останні додані записи.

Для відображення окремих блоків об’єктів у результатах пошуку використано компоненту ASP.NET Repeater із такою розміткою:

<asp:Repeater runat="server" ID="repData">

<HeaderTemplate></HeaderTemplate>

<ItemTemplate>

<%# GetDivStyle(Eval("BuildingID")) %>

<span style="float: right;"><%# GetCityAndCityRegionName(Eval("CityRegionRef")) %> район</span><br><br>

<div>

<div style="float: left;">

<a href="BuildingInfo.aspx?id=<%# Eval("BuildingID") %>"><img src="Files/<%# GetImage(Eval("BuildingID")) %>" width="100px" height="100px;"></a><br>

<span style="float: right"></spM><b><big><%# FormatPrice(Eval("Price")) %> грн</big></b></span><br>

<center><a href="BuildingInfo.aspx?id=<%# Eval("BuildingID") %>">детальніше</a></center>

</div>

<div style="float: left; padding-left: 10px; max-width: 190px; word-wrap: break-word;">

<%# GetInfo(Eval("BuildingID")) %>

</div>

</div>

</div>

</ItemTemplate>

<FooterTemplate></FooterTemplate>

</asp:Repeater>

На рис. 3.11 наведено вигляд сторінки для перегляду детального опису об’єкта користувача, який не є адміністратором або модератором, і не є власником запису. Для такого користувача всі поля є заборонені для редагування.

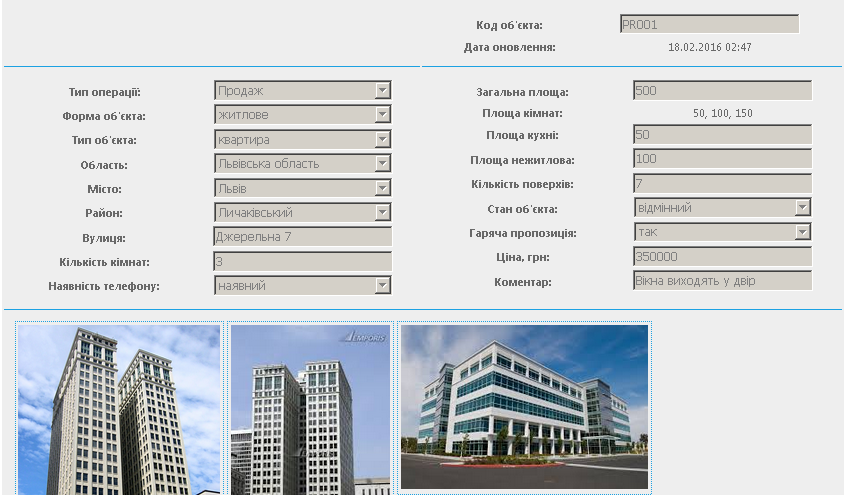


Рис. 3.11. Перегляд детального опису об’єкта

Форма з рис. 3.11 для цього типу користувача дублює пошукову форму з рис. 3.9 за винятком декількох полів: код об’єкта, дата оновлення та зображення.

Для перевірки чи користувач є власником запису використано такий програмний код:

public bool IsOwner

{

get

{

using (var context = new RealtorEntities())

{

if (iBuildingID == -1)

{

return false;

}

var owner = (from i in context.Building

where i.BuildingID == iBuildingID

select i.UserName).FirstOrDefault();

return owner != null &&

(HttpContext.Current.User.Identity.Name.ToLower() == owner.Trim().ToLower());

}

}

}

На рис. 3.12 наведено вигляд сторінки для авторизації в систему.

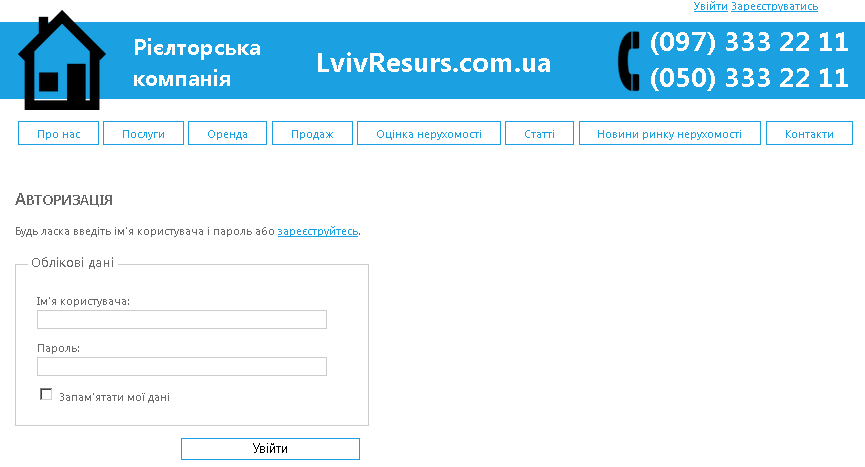


Рис. 3.12. Сторінка для авторизації

Користувач має змогу додати заявку на продаж або здачу в оренду приміщення. Однак, така функціональність передбачена лише для авторизованих користувачів. Тому, якщо користувач не є авторизованим, йому відобразиться сторінка для авторизації. Це здійснено таким програмним кодом:

if (!User.Identity.IsAuthenticated)

{

Response.Redirect("~/Account/Login.aspx");

}

Response.Redirect("~/BuildingInfo.aspx?t=1");

На рис. 3.13 подано вигляд сторінки для реєстрації нового облікового запису в системі.

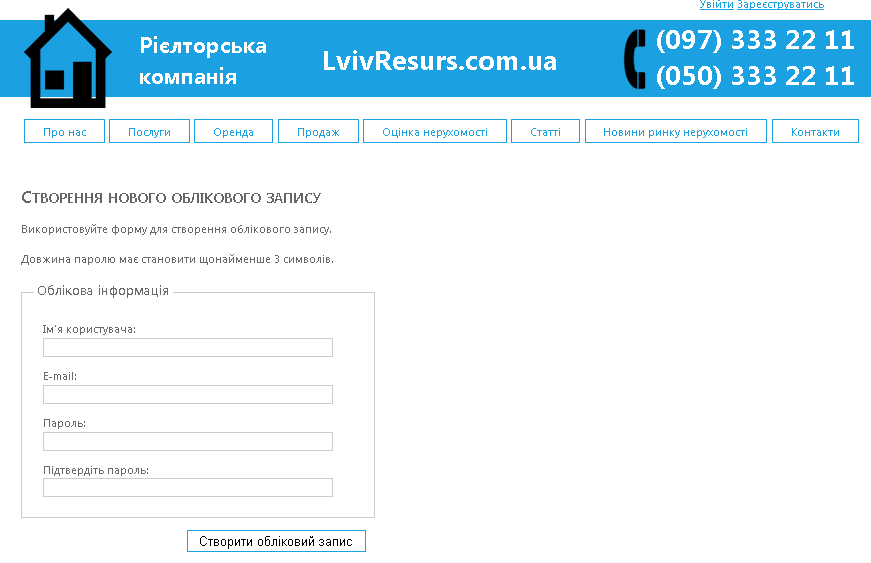


Рис. 3.13. Сторінка для реєстрації нового облікового запису

Якщо користувач авторизований, йому відобразиться сторінка із можливістю вносити дані у поля детального опису об’єкта (рис. 3.14). Після збереження даних про об’єкт, користувачу стає доступним кнопка для додання зображення.

Користувач має змогу додати необмежену кількість зображень до об’єкта. Шлях до цих зображень зберігається в окремій таблиці, яка зв’язана із таблицею Building зовнішним ключем. Самі зображення зберігаються на файловій системі у папці Files. Тому для реалізації можливості завантаження зображень у цю папку, потрібно на сервері встановити відповідні права доступу для цього каталогу.

Користувач також може додати інформацію про площі окремих кімнат (див. рис. 3.14). Для цього використано компоненту GridView.

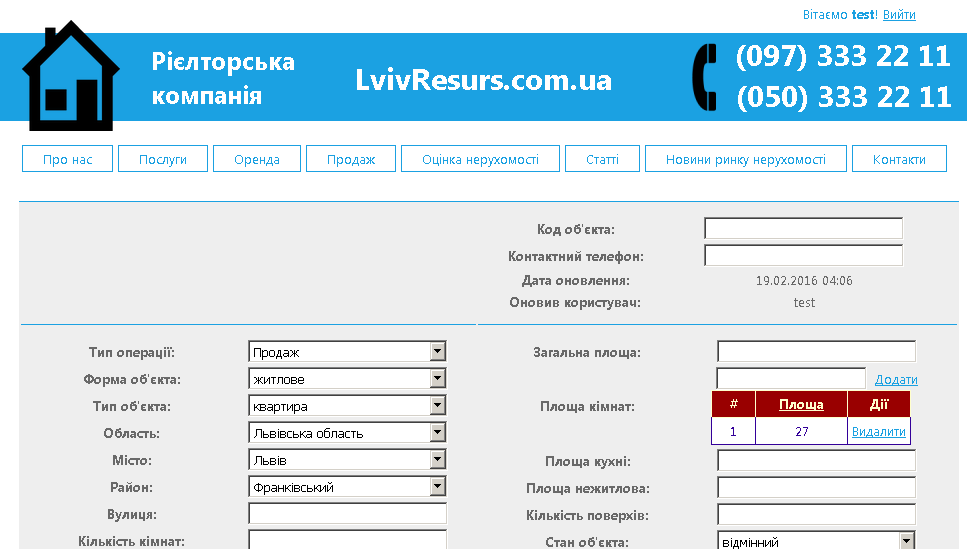


Рис. 3.14. Створення заявки про продаж квартири

На рис. 3.15 подано вигляд меню редагування детальним описом товару для власника запису.

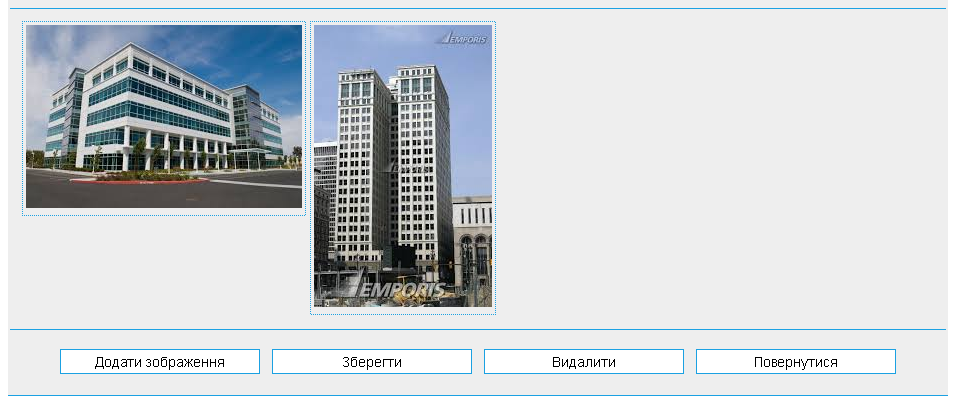


Рис. 3.15. Завантаження фотографій об’єкта

Для завантаження файлу використано компоненту **AsyncFileUpload** із такою розміткою:

<asp:AsyncFileUpload runat="server" ID="asyncFileUpload" Width="690px"

UploaderStyle="Modern" CompleteBackColor="White" UploadingBackColor="#CCFFFF"

ThrobberID="lblProgress" CssClass="FileUploadClass" OnClientUploadError="uploadError"

OnClientUploadComplete="uploadComplete"

OnUploadedComplete="asyncFileUpload\_UploadedComplete"

OnUploadedFileError="asyncFileUpload\_UploadedFileError"></asp:AsyncFileUpload>

і таким програмним кодом:

var fileUploadReceive = asyncFileUpload.PostedFile.FileName;

var uploadedFileSize = asyncFileUpload.PostedFile.ContentLength;

if (uploadedFileSize < 16777216)

{

using (var context = new RealtorEntities())

{

if (iBuildingID == -1) { return; }

context.AddToBuildingImage(new BuildingImage

{

BuildingRef = iBuildingID,

ImagePath = fileUploadReceive

});

context.SaveChanges();

}

asyncFileUpload.SaveAs(MapPath("Files\\" + fileUploadReceive));

}

Обмеження **16777216** встановлено для того, щоб користувач не мав змоги завантажити файл більше 15 Мб.

На рис. 3.16 наведено вигляд головної сторінки адміністратора.



Рис. 3.16. Головна сторінка адміністратора

Адміністратор може управляти формами, типами і станами об’єктів, областями, містами, районами міст, а також користувачами. Крім цього, адміністратор має змогу редагувати будь-який детальний опис об’єкта.

На рис. 3.17 подано вигляд сторінки для управління формами об’єктів. Дя цього використано компоненту GridView та джерело даних EntityDataSource.

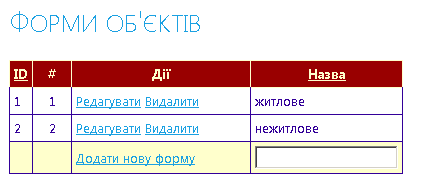


Рис. 3.17. Форми об’єктів

На рис. 3.18 наведено вигляд сторінки для управління об’єктами. Оскільки об’єкт має залежність від форми, то на цій сторінці вибір форми реалізовано за допомогою випадаючого списку.



Рис. 3.18. Типи об’єктів

На рис. 3.19 наведено вигляд сторінки для управління областями.



Рис. 3.19. Області

На рис. 3.20 наведено вигляд сторінки для управління містами. Кожне місто є залежним від області. З метою зручного вибору користувачем певної області для міста додано вибір області через випадаючий список.



Рис. 3.20. Міста

Для цього використано таку розмітку сторінки у компоненті GridView:

<edititemtemplate>

<asp:DropDownList ID="ddlRegion" runat="server" CssClass="field"

AutoPostBack="False" DataSourceID="edsRegions"

selectedvalue="<%# DataBinder.Eval(Container.DataItem,&quot;RegionRef&quot;) %>"

DataTextField="RegionName" DataValueField="RegionID"></asp:DropDownList>

</edititemtemplate>

а також такий програмний код для завантаження даних у рядок створення запису:

protected void gvCities\_OnRowDataBound(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.Footer)

{

var ddlRegion = (DropDownList)e.Row.FindControl("ddlRegionNew");

if (ddlRegion != null)

{

using (var context = new RealtorEntities())

{

ddlRegion.DataSource = from i in context.Region

orderby i.RegionName ascending

select i;

ddlRegion.DataBind();

}

}

}

}

На рис. 3.21 подано вигляд сторінки для управління станом об’єкта.

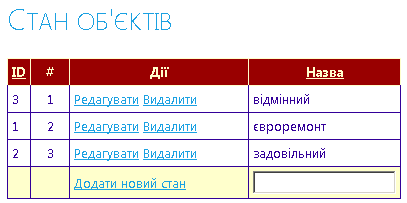


Рис. 3.21. Стан об’єктів

На рис. 3.22 подано вигляд сторінки для управління районами міста.



Рис. 3.22. Райони міста

На рис. 3.23 наведено вигляд сторінки для управління користувачами.

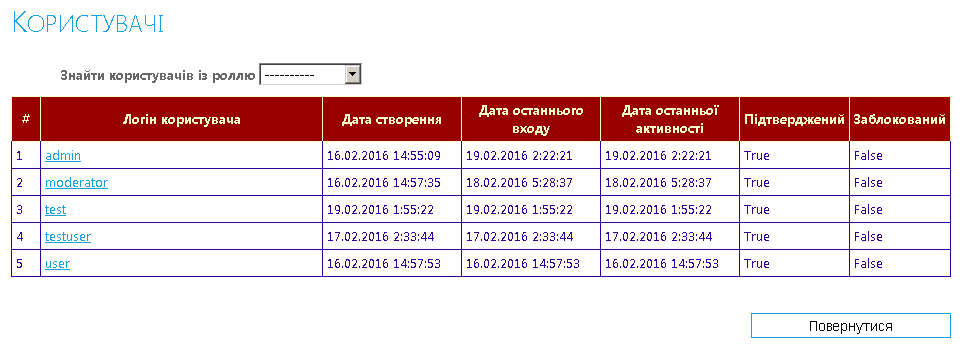


Рис. 3.23. Управління користувачами

Адміністратор може здійснити пошук користувачів за певною роллю, вибравши її з випадаючого списку. Результати будуть автоматично відображені на цій же сторінці.

# РОЗДІЛ 4. ТЕСТУВАННЯ ТА СУПРОВІД ВЕБ-ОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ РІЄЛТОРСЬКОЇ КОМПАНІЇ

## 4.1. Тестування програмного забезпечення

### 4.1.1. Розробка тестів

Розроблено тестові випадки для функціонального тестування та сценарії для тестування продуктивності. В додатку Е подано тестові випадки, які містять детальні кроки та очікувані результати.

### 4.1.2. Функціональне тестування

В табл. 4.1 подано розподіл функціональних тестових випадків і наборів тестових даних для цих випадків за варіантами використання.

Таблиця 4.1.

Розподіл функціональних тестових випадків і наборів тестових даних

| **Варіанти використання** | **Тестові випадки** | **Тестові дані** |
| --- | --- | --- |
| Пошук об’єкта | 5 | 6 |
| Авторизація | 5 | 5 |
| Реєстрація | 5 | 5 |
| Перегляд детального опису об’єкта | 5 | 6 |
| Управління користувачами | 5 | 5 |
| Керування довідниками | 5 | 5 |
| **Разом** | **30** | **32** |

Під час фази розробки тестів було створено перелік функціональних тестових випадків: пошук об’єкта за його характеристиками, авторизація користувача в системі, реєстрація нового облікового запису, перегляд детального опису об’єкта, управління користувачами, керування довідниками.

### 4.1.3. Тестування продуктивності

Для тестування продуктивності веб-системи рієлторської компанії використано програмний засіб WAPT. На рис. 4.1 подано результати тестування середнього відгуку сервера.

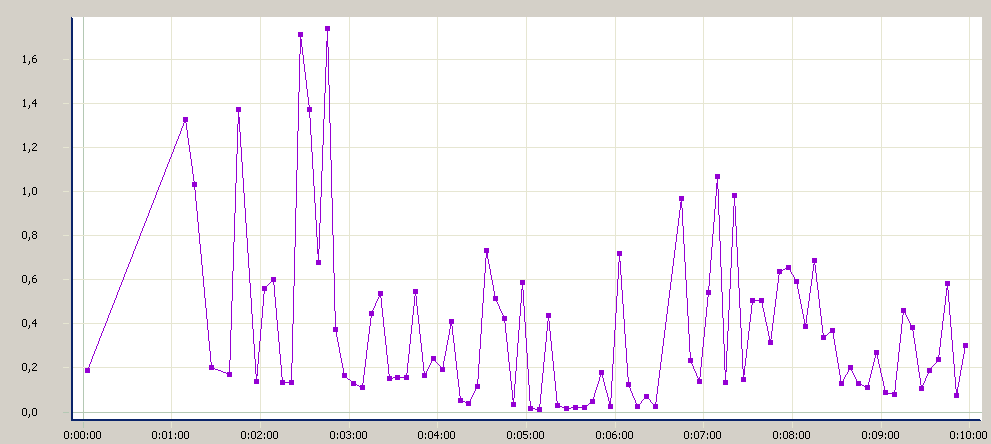


Рис. 4.1. Середній час відгуку сервера

На рис. 4.2 подано результати тестування пропускної здатності веб-системи.

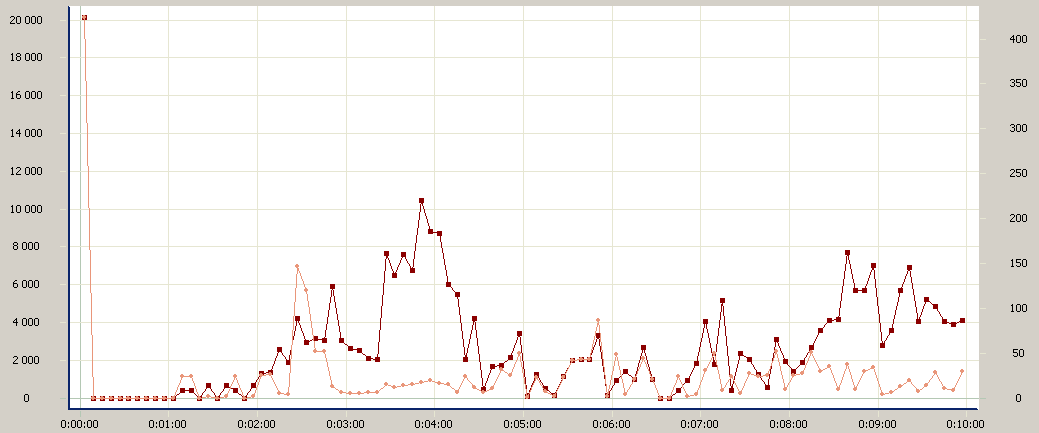


Рис. 4.2. Пропускна здатність сервера

На рис. 4.3 подано результати тестування середнього часу завантаження сторінки.

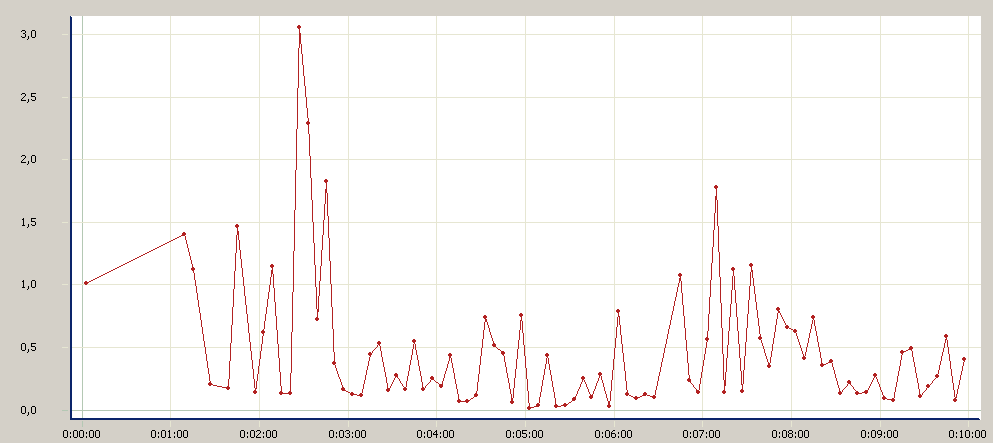


Рис. 4.3. Середній час завантаження сторінки

### 4.1.4. Метрики коду

Для оцінки якості коду побудовано метрики коду у середовищі Visual Studio (рис. 4.4).

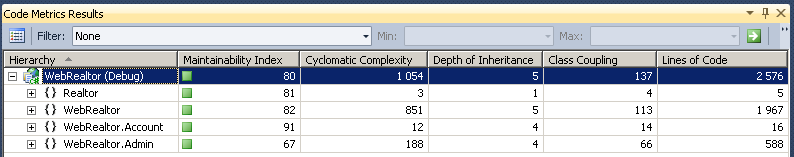


Рис. 4.4. Сформовані метрики коду

### 4.1.5. Результати тестів

За результатами тестування помилок не було знайдено, тому результати тестування вважаються успішними.

## 4.2. План забезпечення якості веб-орієнтованої системи

### 4.2.1. Мета

Веб-орієнтована інформаційна система рієлторської компанії має забезпечити користувача зручним інтерфейсом та засобами пошуку будівель та приміщень за заданими характеристиками.

### 4.2.2. Управління

#### 4.2.2.1. Організація

В табл. 4.2 наведено задіяні на проекті особи та їх ролі.

Таблиця 4.2.

Задіяні на проекті особи та їх ролі

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | | * Назва підрозділу | Склад підрозділу | За що відповідають |
| 1 | | Підрозділ програмної розробки | 1. Керівник  2. Розробник  (дипломник) | Розробка веб-системи |
| 2 | | Підрозділ тестування | 1. Керівник  2. Тестувальник  (дипломник) | Верифікація та валідація веб-системи. |
| 3 | Підрозділ доповнення  функціональності системи | | 1. Керівник  2. Розробник  (дипломник)  3. Тестувальник  (дипломник) | Доповнення функціональності  веб-системи |
| 4 | Відділ ведення документації | | 1. Керівник  2. Розробник  (дипломник) | Документування процесів під час супроводу  веб-системи |

#### 4.2.2.2. Завдання

Основними завданнями є:

* Розробка архітектури веб-системи.
* Розробка моделі сховища даних.
* Вибір інструментів та засобів реалізації.
* Розроблення функціоналу.
* Тестування програмного забезпечення.

#### 4.2.2.3. Ролі та обов’язки

Основні ролі та відповідальності подано в п. 4.2.2.1 цього документу.

#### 4.2.2.4. Оцінка ресурсів, необхідних для забезпечення якості

Оцінка ресурсів, необхідних для забезпечення якості під час тестування подані в табл. 4.3.

Таблиця 4.3.

Оцінка ресурсів, необхідних для забезпечення якості

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Назва ресурсу** | **Вартість** | **Опис** |
| 1 | Microsoft SQL Server Trial | безкоштовно | * Система керування базами даних |
| 2 | Microsoft Visual Studio Trial | безкоштовно | * Середовище для розробки програмного забезпечення |
| 3 | Microsoft IIS 7 | безкоштовно | * Веб-сервер |

### 4.2.3. Документація

#### 4.2.3.1. Мета

Такі документи можуть бути використані в якості вимоги до проектування і виробництва програмного забезпечення:

* + MSFC Software Quality Assurance Requirementsfor MSFC Projects
  + ACIS Program Management Plan: Instrument Software Management Plan

#### 4.2.3.2. Мінімальні вимоги до документації

* + - * 1. **Опис вимог до програмного забезпечення (SRD)**

В табл. 4.4 подано набір компонент для коректної роботи програмного забезпечення.

Таблиця 4.4.

Набір компонент для коректної роботи програмного забезпечення

|  |  |  |  |
| --- | --- | --- | --- |
| № | Файл | Призначення | Належить проекту |
| 1 | Realtor.dll | Бібліотека веб-системи | Веб-орієнтована інформаційна система рієлторської компанії |
| 2 | Realtor | База даних |
| 3 | RealtorUsers | База даних (користувачі) |
| 4 | \*.aspx | Файли логіки веб-системи |
| 5 | \*.css, \*.js, \*.jpg | Файли оформлення веб-системи |

* + - * 1. **Опис дизайну програмного забезпечення (SDD).**

Атрибутами якості програмного забезпечення є:

* Доступність. Робота із програмним забезпеченням можлива з використанням будь-якого сучасного веб-оглядача.
* Зручність використання. Програмне забезпечення має володіти зручним інтерфейсом користувача.
  + - * 1. **План верифікації та валідації**

Такі визначення IEEE ставляться до цього плану SQA:

* + **Перевірка:** процес визначення того, чи продукція стадії життєвого циклу розробки програмного забезпечення дотримується вимог, встановлених на попередньому етапі.
  + **Перевірка:** оцінка в кінці процесу розробки програмного забезпечення для гарантування дотримання вимог.
    - * 1. **Верифікація та валідація**

В рамках перевірки вимог виділено в SQA такі заходи:

* Перевірка наскільки документовані вимоги задовольняють вимогам, описаних в плані проекту.
* Перевірка критичних вимог до виявлення ключових показників або критичних областей програмного забезпечення.
  + - * 1. **Користувацька документація**

Містить чітко сформульовані завдання, які вирішує програмне забезпечення.

* + - * 1. **План конфігураційного управління (SCMP)**

При верифікації, програмне забезпечення тестується формальним, структурованим чином призначеним для забезпечення повного охоплення всіх вимог.

* + - * 1. **Інші документи**

Технічне завдання визначає роботу як переговори з клієнтом.

### 4.2.4. Стандарти, практики, конвенцій, і метрики

* + - 1. **Мета**

Метрики програмного забезпечення дають змогу оцінити характеристики програмного забезпечення числовими показниками.

* + - 1. **Зміст**

Такі стандарти були  використані в якості керівництва для розвитку процесу SQA:

* ANSI / IEEE 730,1: Стандартне програмне забезпечення для планів забезпечення якості.
* ANSI / IEEE 1028: стандарт для програми перевірок і ревізій.
* ANSI / IEEE 1012: стандарт для перевірки програмного забезпечення та перевірки.

### 4.2.5. Тестування

#### 4.2.5.1. Функціональне тестування

### Проведено функціональне тестування для перевірки такої функціональності веб-системи: пошук об’єкта за його характеристиками, авторизація користувача в системі, реєстрація нового облікового запису, перегляд детального опису об’єкта, управління користувачами, керування довідниками.

### 4.2.6. Звітність проблем і коригувальних дій.

При виявленні проблем проводиться процедура аналізу проблем та модифікації програмного забезпечення.

### 4.2.7. Засоби, методи і методик

Необхідними інструментаріями є:

* засіб Microsoft Visual Studio, який дає змогу сформувати метрики програмного забезпечення;
* засіб WAPT, який дає змогу оцінити продуктивність веб-продукту.

### 4.2.8. Контроль середовища

Цей розділ не відповідає даному плану.

### 4.2.9. Контроль розробника

Цей розділ не відповідає даному плану.

### 4.2.10. Збір коментарів, їх підтримка і збереження

Цей розділ не відповідає даному плану.

### 4.2.11. Навчання персоналу

Навчання користувачів програмного забезпечення має проходити до його розгортання та налаштування.

### 4.2.12. Управління ризиками

Уравління ризиками включає кроки, подані в табл. 4.5.

Таблиця 4.5.

Ризики, та заходи щодо їх попередження

|  |  |
| --- | --- |
| **Ризики** | **Заходи щодо попередження** |
| Невиконання термінів | Розставити пріоритети вимог |
| Бюджет | Ефективне планування ресурсів |
| Зміна вимог | Чіткий опис вимог |
| Помилки в проектуванні | Додатковий огляд спроектованої системи. |
| Затримка термінів при розробці модуля. | Виділення додаткових ресурсів на початку розробки. |

## 4.3. План впровадження програмного забезпечення

### 4.3.1. Вступ

Веб-система рієлторської компанії надає змогу користувачам здійснити пошук за критеріями об’єкта. Користувачі мають змогу купити, продати, взяти в оренду приміщення чи квартиру.

### 4.3.2. Ролі і відповідальності

В табл. 4.6 подано ролі та їх відповідальності.

Таблиця 4.6.

Ролі та їх відповідальності

|  |  |  |
| --- | --- | --- |
| **Роль** | **Виконувані функції** | **Необхідні знання та навики** |
| Мережевий адміністратор | Контроль над доступом системи до мережі Інтернет | Спеціаліст середнього рівня кваліфікації. |
| Адміністратор баз даних | Контроль за цілісністю бази даних | Спеціаліст середнього рівня кваліфікації. |
| Користувач | Використання системи | Базовий рівень знань ПК. |

### 4.3.3. Часовий план

Для реалізації проекту розгортання та налаштування веб-системи потрібно в середньому 8 год (табл. 4.7).

Таблиця 4.7.

Часовий план розгортання та налаштування

|  |  |  |
| --- | --- | --- |
| № | Найменування задачі | Час впровадження |
| 1 | Встановлення та налаштування  Microsoft SQL Server 2014 | 2 год. |
| 2 | Встановлення та налаштування IIS | 2 год. |
| 3 | Розгортання веб-системи | 2год |
| 4 | Ввід в експлуатацію | 2 год |
| **Всього:** | | **8 год** |

### 4.3.4. Навчання персоналу

Навчання має відбуватися до встановлення та налаштування програмного забезпечення.

### 4.3.5. Робоче середовище

До складу технічних засобів повинен входити комп'ютер, що виконує роль веб-сервера і сервера баз даних: процесор: 2 Гц; пам’ять: 2048 Мб; жорсткий диск: 40 Гб; операційна система: сімейства Windows.

### 4.3.6. Процедура інсталяції

Для встановлення програмного забезпечення потрібно:

1. розгорнути базу даних Realtor з резервної копії;
2. створити веб-сайт у IIS і скопіювати файли системи;
3. змінити стрічку з’єднання із базою даних у файлі налаштувань web.config.

### 4.3.7. Запуск продукту в експлуатацію

Після встановлення, налаштування і вдалого проходження всіх тестів веб-система може бути запущена для користування.

## 

## 4.4. План супроводу програмного забезпечення

### 4.4.1. Вступ

Веб-орієнтована інформаційна система рієлторської компанії надає зручні засоби користувачам для ефективного пошуку товару та перегляду його характеристик.

### 4.4.2. Огляд системи

**Характеристики продукту**

Основними функціональними можливостями розробленого програмного забезпечення є: пошук об’єкта за його характеристиками, авторизація користувача в системі, реєстрація нового облікового запису, перегляд детального опису об’єкта, управління користувачами, керування довідниками.

### 4.4.3. Середовище супроводу

Комп’ютерна мережа.

### 4.4.4. Персонал супроводу

#### 4.4.4.1. Функції персоналу супроводу

* підтримка стабільності доступу до мережі Інтернет;
* підтримка стабільності та цілісності роботи БД;
* вдосконалення наявних функцій програмного забезпечення.

#### 4.4.4.2. Розподіл обов’язків персоналу супроводу

Розподіл обов’язків персоналу супроводу наведено у табл. 4.8.

Таблиця 4.8

Розподіл обов’язків персоналу супроводу

|  |  |  |
| --- | --- | --- |
| **Команда** | **Категорія супроводу** | **Детальна інформація** |
| Команда 1 | Коригувальний супровід | Вдосконалення веб-системи, виконуване після передачі в експлуатацію для усунення збоїв. |
| Команда 2 | Вдосконалюючий супровід | Модифікація веб-системи на етапі експлуатації для підвищення продуктивності і зручності супроводу. |

### 4.4.5. Процедури супроводу

Процес супроводу починається одразу після впровадження програмного забезпечення в експлуатацію.

# РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

## 

## 5.1. Економічна характеристика проектного рішення

Сьогодні програмне забезпечення є найрозповсюдженішим та одним із найважливіших аспектів інформаційного світу, тому дуже часто виникає необхідність в оновленні принципу роботи та захисту програмних продуктів, за допомогою яких суспільство обмінюється інформацією. Дуже важливим є створення веб-орієнтованої інформаційної системи рієлторської компанії.На початкових стадіях, а саме стадії виробництва прийняте рішення щодо ефективності програмного забезпечення обумовлюється:

* витратами на розробників продукту;
* витратами на купівлю периферійних пристроїв та іншого інструменту;
* витратами на оновлення програмного забезпечення.

Щодо стадії експлуатації, то ефективність програмного продукту забезпечується:

* виконання вимог користувача;
* ціна подальшої підтримки програмного забезпечення;
* зручністю та легкістю використання та оновлення продукту;
* якістю і кількістю обслуговуючого персоналу.

Метою даного дипломного проекту є розробка веб-орієнтованої інформаційної системи рієлторської компанії, яка забезпечить зручний перегляд об’єктів нерухомості за визначеними критеріями пошуку.

З точки зору економіки, то головна ідея впровадження проектного рішення полягає в зменшенні та оптимізації витратних коштів на створення і подальшу підтримку програмного продукту.

## 5.2. Розрахунок витрат на розробку та впровадження програмного модулю для забезпечення електронної звітності

1) Витрати нарозроблення і впровадження програмного засобу (*К*) визначаються як:

*Кзаг* = *К1* + *К2*,(5.1)

де*К1* – витрати на розроблення програмного засобу, грн.;

*К2 –* витрати на налагодження і дослідну експлуатацію програмного засобу на ЕОМ, грн.

Витрати на розроблення програмного засобу передбачають:

1. витрати на оплату праці розробників ();
2. відрахування у спеціальні державні фонди ()
3. вартість додаткових виробів, що закуповуються ();
4. транспортно-заготівельні витрати ();
5. витрати на придбання спецобладнання ();
6. накладні витрати ();
7. інші витрати ().

Для проведення розрахунків витрат на оплату праці необхідно визначити категорії працівників, котрі беруть участь у проектуванні, їх чисельність, середньоденну заробітну плату спеціаліста відповідної категорії та трудомісткість робіт у людино-днях (людино-годинах).

Середньоденна заробітна плата *і*-го розробника () обчислюється за формулою:

 (5.2)

де** – основна місячна заробітна плата розробника *і-ої* спеціальності, грн.;

** – місячний фонд робочого часу, днів (24 дні).

У проекті бере участь 4 особи: керівник проекту, програміст, тестер та співробітник-консультант. Їх заробітна плата становить:

* Керівник проекту

**грн.;

* Програміст

** грн.;

* Тестер

**грн.;

* Співробітник-консультант

**грн.

Трудомісткість робіт над проектом становить: 15 людино-днів для керівника проекту, 45 людино-днів для програміста, 8людино-днів для тестера та 10 людино-днів для співробітника-консультанта.

Розрахунок витрат на оплату праці усіх розробників проекту обчислюємо за формулою:

 (5.3)

де – чисельність розробників проекту і-ої спеціальності, чол.;

 – час, витрачений на розроблення проекту працівником *і*-ої спеціальності, дні;

* –*денна заробітна плата розробника *і*-ої спеціальності, грн.

**грн.

Розрахунок витрат на оплату праці розробників зводиться у табл. 5.1.

Таблиця 5.1

Розрахунок витрат на оплату праці

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Спеціальність розробника | Кількість розробників, чол. | Час роботи, дні | Денна заробітна плата розробника, грн. | Витрати на оплату праці, грн. |
| 1 | Керівник проекту | 1 | 15 | 166,67 | 2500,05 |
| 2 | Програміст | 1 | 45 | 187,50 | 8437,50 |
| 3 | Співробітник-консультант | 1 | 15 | 145,83 | 2187,45 |
| 4 | Тестер | 1 | 8 | 104,17 | 833,36 |
| *Разом* | | | | | 13958,36 |

2) Витрати на оплату праці працівникам тягнуть за собою додаткові зобов'язання підприємства перед державними фондами обов'язкового страхування*.* Підприємство зобов'язане здійснювати відрахування до фондів: єдиного соціального внеску(34,7%).

 (5.4)

де *ВФ* – нарахування єдиного соціального внеску;

*ВОП* – витрати на оплату праці.

грн.

3) Витрати на додаткові вироби, що закуповуються () (папір, дискети тощо) визначаються за їхніми фактичними цінами з врахуванням найменування, номенклатури танеобхідної їх кількості в проекті.

Вихідні дані та результати розрахунків оформляються у табл.5.2. Транспортно-заготівельні витрати () становлять 14% від суми витрат на додаткові вироби, що закуповуються.

Таблиця 5.2

Розрахунок витрат на куповані вироби

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва купованих виробів | Марка, тип | Кількість на розробку, шт. | Ціна за одиницю, грн. | Сума витрат, грн. | Сума витрат з урахуванням транспортно- заготівельних витрат, грн. |
| 1 | Олівець | CENTRUM HB Gotic,з гумкою 83535 | 3 | 3,50 | 10,50 | 11,97 |
| 2 | Ручка | Flair X5, чорний | 1 | 3,00 | 3,00 | 3,42 |
| 3 | Маркер | Optima,чорний | 1 | 32,50 | 32,50 | 37,05 |
| 4 | Клей-олівець | VGR, 36г | 5 | 3,50 | 17,50 | 19,95 |
| *Разом:* | | | | | 63,50 | 72,39 |

4) Витрати на придбання спецобладнання () для проведення експериментальних робіт розраховуються в тому випадку, коли для розроблення та впровадження проектного рішення необхідне придбання додаткових технічних засобів.

Вартість спецобладнання длявиконання конкретного проекту визначається на основі специфікації в їх потребі та фактичних цін з урахуванням транспортно-заготівельних витрат.

Транспортно-заготівельнівитрати () становлять 11 % від суми витрат на придбання спецобладнання.

Витрати на придбання спецобладнання наведені у табл. 5.3:

Таблиця 5.3

Витрати на придбання спецобладнання

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва обладнання | Марка, тип | Кількість на розробку, шт. | Ціна за одиницю, грн. | Сума витрат з урахуванням транспортно-  заготівельних витрат, грн. |
| 1. | Монітор | Dell P2214H | 1 | 4089,00 | 4538,79 |
| 2. | Системний блок | Lenovo Idea Centre H535 | 1 | 8295,00 | 9207,45 |
| 3. | Клавіатура | Acme Ultrathin Bluetooth BK01 | 1 | 945,00 | 1048,95 |
| 4. | Мишка | Genius NetScroll 120USB | 1 | 329,00 | 365,19 |
| Разом: | | | | 13658,00 | 15160,38 |

5) Накладні витрати *()* проектних організацій передбачають витрати на управління, загальногосподарські, невиробничі витрати. Вони становлять 22% витрат на оплату праці.

**грн.

6) Інші витрати (*)* – це витрати, які не враховані в попередніх статтях витрат. Їх розраховують за встановленим відсотком (8%) до витрат на оплату праці.

 грн.

7) Витрати на розробку проектного рішення обчислюємо за формулою:

 (5.5)

грн.

Витрати на налагодження і дослідну експлуатацію системи визначаємо згідно з формулою:

 (5.6)

де  – вартість однієї години роботи ПК, грн.\год;

 – кількість годин роботи ПК на налагодження програми, год.

Вартість однієї години роботи ПК – 1,824 грн., тобто спожита електроенергія: 0,2 КВт/год. × 1,824 грн. (тариф) = 0,2365 грн.

На налагодження системи було витрачено 24 днів (24 × 8 = 192 годин часу), тому:

 грн.

Отже, витрати на розробку і впровадження програмного засобу становлять:

 грн.

Результати розрахунків кошторису на розробку проектного рішення наведено у табл. 5.4.

Таблиця 5.4

Кошторис витрат на розробку проектного рішення

|  |  |
| --- | --- |
| Найменування елементів витрат | Сума витрат, грн. |
| Витрати на розробку проектного рішення, у т.ч.: |  |
| витрати на оплату праці | 13958,36 |
| відрахування єдиного соціального внеску | 4843,55 |
| витрати на додаткові вироби, що закуповуються | 72,39 |
| витрати на придбання спецобладнання | 15160,38 |
| накладні витрати | 3070,83 |
| інші витрати | 1116,67 |
| витрати на відлагодження і дослідну експлуатацію системи | 45,40 |
| *Разом* | 38267,58 |

## 5.3. Визначення комплексного показника якості

Комплексний показник якості () визначається шляхом порівняння показників якості проектованої системи і вибраного аналогу.

За аналог обираємо продукт "РК Маклер", що відповідає проектному рішенню (проектованій системі) по функціональному призначенню та по сфері застосування , і є широко представлений на обраному ринку.

Для визначення  використовується система показників технічного рівня і якості, яка містить в собі наступні групи:

1. Показники призначення:
   1. Актуальність даного програмного забезпечення;
   2. Уніфікація;
   3. Автоматизоване оновлення програмного забезпечення;
2. Показники надійності:
   1. Ймовірність помилки;
   2. Ймовірність безвідмовної роботи.
3. Показники безпеки:
   1. Можливість не санкціонованого доступу до даних.
4. Патентно-правові показники:
   1. Патентно-правовий статус.
5. Ергономічні показники:
   1. Зручність в експлуатації.

Комплексний показник якості проектованої системи визначаємо методом арифметичного середньозваженого з формули:

 (5.7)

де – кількість одиничних показників (параметрів), прийнятих для оцінювання якості проектованої системи;

 – коефіцієнт вагомості кожного з параметрів щодо їхнього впливу на технічний рівень та якості проектованої системи, причому:



(5.8)

 – часткові показники якості, визначені порівнянням числових значень одиничних показників проектованої системи і аналога за формулами:

 або  (5.9)

де , – кількісні значення і-го одиничного показника якості відповідно проектованої системи і аналога.

З попередніх двох формул вибирається та, в якій збільшення відповідає покращенню показника якості проектованої системи. Результати розрахунку наведені у табл. 5.5.

Таблиця 5.5

Визначення комплексного показника якості проектованої системи (аналога)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показники | Числове значення показників, бали | | Відносний показник якості, | Коефіцієнт вагомості |  |
| Аналог | Проект, прогр. продукт |  |
| Актуальність даного ПЗ | 10 | 9 | 0,9 | 0,125 | 0,1125 |
| Уніфікація | 7 | 9 | 1,3 | 0,09 | 0,117 |
| Зручність в експлуатації | 5 | 9 | 1,8 | 0,15 | 0,27 |
| Автоматизоване оновлення ПЗ | 2 | 10 | 5 | 0,19 | 0,95 |
| Патентно-правовий статус | 10 | 5 | 0,5 | 0,085 | 0,04 |
| Ймовірність безвідмовної роботи. | 6 | 9 | 1,5 | 0,16 | 0,15 |
| Можливість не санкціонованого доступу до даних | 5 | 9 | 1,8 | 0,1 | 0,18 |
| Кількість збоїв в програмі | 4 | 10 | 2,5 | 0,1 | 0,25 |
| *Разом* | | | | 1,00 | 2,07 |

## 5.4. Визначення експлуатаційних витрат

При порівнянні програмних засобів в експлуатаційні витрати включають вартість підготовки даних () і вартість годин роботи ПК (). Одноразові експлуатаційні витративизначаються за формулою:

 (5.10)

де – одноразові експлуатаційні витрати на проектне рішення (аналог), грн.;

 –вартість підготовки даних для експлуатації проектного рішення (аналога), грн.:

 – вартість машино-годин роботи ПК для проектного рішення (аналога), грн.

Річні експлуатаційні витрати визначаються за формулою:

 (5.11)

де  – експлуатаційні річні витрати проектного рішення, грн.;

 – періодичність експлуатації проектного рішення (аналога), разів/рік.

Вартість підготовки даних для експлуатації проектного рішення (аналога) () визначаються за формулою:

 (5.12)

де –номери категорій персоналу, які беруть участь у підготовці даних;

 *–*чисельність співробітників -ї категорії, чол.;

 – трудомісткість роботи співробітників -ї категорії, год.;

 –середньогодинна ставка робітника -ї категорії з врахуванням відрахувань до державних цільових фондів, грн./год.

Середньогодинна ставка оператора визначається за формулою:

 (5.13)

де – основна місячна зарплата працівника -ї категорії, грн.;

 – коефіцієнт, який враховує збори до спеціальних державних фондів (*b*=0,347);

 – місячний фонд робочого часу. год.

Кількість робочих годин у місяці год.

Отже, для проектного рішення середньогодинна ставка становить:

грн.

грн.

грн.

грн.

Вартість підготовки даних для роботи на ПК дорівнює:

 грн.

Витрати на експлуатацію ЕОМ визначаються за формулою:

 (5.14)

де – витрати машинного часу для одноразової реалізації проектного рішення (аналога), год.;

– вартість однієї машинної години роботи ЕОМ.

Вартість машино-годин роботи ПК для проектного рішення становить 27,85 грн. Тоді одноразові експлуатаційні витрати на проектне рішення становлять:

 грн.

Вихідні дані та результати розрахунків витрат на підготовку даних для експлуатації проектного рішення зводяться у табл. 5.6.

Таблиця 5.6

Розрахунок витрат на підготовку даних для роботи на ЕОМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категорія персоналу | Чисельність співробітників -ої категорії, чол. | Час роботи співробітників *і*-ої категорії, год. | Середньогодинна ЗП співробітника *і*-ої категорії, грн. | Витрати на підготовку даних, грн. |
| Проектне рішення | | | | |
| Керівник проекту | 1 | 15 | 28,06 | 420,90 |
| Програміст | 1 | 24 | 38,59 | 926,16 |
| Тестер | 1 | 15 | 24,55 | 368,25 |
| Консультант | 1 | 8 | 17,53 | 140,24 |
| *Разом* | | | | 1855,55 |
| Аналог | | | | |
| Менеджер IT проекту | 1 | 24 | 17,54 | 420,94 |
| Архітектор ПО | 1 | 24 | 45,60 | 1094,44 |
| Адміністратор баз даних | 1 | 24 | 22,45 | 538,80 |
| *Разом* | | | | 2054,18 |

Над проектом-аналогом працює 1 Менеджер IT проекту,1 Архітектор ПО 1 Адміністратор баз даних. Їх місячні заробітні ставки відповідно 2500 грн., 6500 грн. та 3200грн. Тоді середньогодинна ставка для кожного з них становить:

грн.

грн.

грн.

Вартість підготовки даних для роботи на ПК дорівнює:

 грн.

Таким чином, одноразові експлуатаційні витрати по проектному рішенню складають:

 грн.

Періодичність використання програмного засобу проектного рішення становить 4рази/рік, натомість аналогу 8 раз/рік.

Річні експлуатаційні витрати складають:

 грн.

 грн.

## 5.5. Розрахунок ціни споживання проектного рішення

Ціна споживання () - це витрати на придбанняі експлуатацію проектного рішення за весь строк його служби:

 (5.15)

де  – ціна придбання проектного рішення, грн.;

 – теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

 (5.16)

де– норматив рентабельності (30%);

 – витрати на прив’язку та освоєння проектного рішення на конкретному об’єкті (500,00грн.);

 – витрати на доукомплектування технічних засобів на об’єкті (0 грн.);

– ставка податку на додану вартість (20 %).

Ціна придбання проектного рішення дорівнюватиме:

 грн.

Теперішня вартість витрат на експлуатацію проектного рішення розраховується за формулою:

 (5.17)

де  – річні експлуатаційні витрати в -ому році, грн.;

*Т*- строк служби проектного рішення (2 роки):

R - річна ставка проценту банків (21%).

Оскільки впродовж всього строку експлуатації , то:

 (5.18)

де *РV* – ставка дисконту на період *Т*, яка визначається залежно від процентної ставки (*R=0,21*) і періоду експлуатації (*Т*).

Оскільки термін експлуатації проектного рішення становить 2 роки, тоді *РV* = 1,46. Отже, теперішня вартість витрат на експлуатацію проектного рішення становить:

 грн.

Таким чином ціна споживання проектного рішення становить:

грн.

Аналогічно визначається ціна споживання для аналогу. Визначимо теперішню вартість витрат на експлуатацію аналогу. Термін експлуатації аналогу становить 2 років, тоді *РV* = 1,46. Оскільки , то:

 грн.

Отже, ціна споживання проекту-аналогу становить:

 грн.

## 5.6. Визначення показників економічної ефективності

1) Показник конкурентоспроможності:

 (5.19)



2) Економічний ефект в сфері експлуатації (грн.):

 (5.20)

 грн.

3) Економічний ефект в сфері проектування (грн.):

 (5.21)

 грн.

Оскількита, то розраховується:

4) Додатковий економічний ефект в сфері експлуатації (грн.):

 (5.22)

грн.

5) Додатковий економічний ефект в сфері проектування (грн.):

 (5.23)

 грн.

6) Термін окупності витрат на проектування рішення (років):

 (5.24)

 роки.

Результуючі показники економічної ефективності зводяться у табл. 5.7.

Таблиця 5.7

Показники економічної ефективності проектного рішення

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування показників | Одиниці вимірювання | Значення показників | |
| Аналог | Проектне рішення |
| 1. Капітальні вкладення | грн. |  | 38267,58 |
| 2. Ціна придбання | грн. | 60399,25 | 30084,89 |
| 3. Річні експлуатаційні виграти | грн. | 16796,64 | 4848,35 |
| 4. Ціна споживання | грн. | 64717,30 | 37163,48 |
| 5. Економічний ефект в сфері експлуатації | грн. |  | 11807,85 |
| 6. Додатковий економічний ефект в сфері експлуатації | грн. |  | 15432,00 |
| 7. Економічний ефект в сфері проектування | грн. |  | 10314,36 |
| 8. Додатковий економічний ефект в сфері проектування | грн. |  | 17619,00 |
| 9. Термін окупності витрат на проектування рішення | роки |  | 4,2 |
| 10. Коефіцієнт конкурентоспроможності |  |  | 1,19 |

**5.7. Висновок до розділу**

Метою дипломного проекту є розробка «**Веб-орієнтованої інформаційної системи рієлторської компанії**».Система покликана автоматизувати операції щодо пошуку в оренду та купівлю нерухомого майна працівниками рієлторської компанії. «**Веб-орієнтована інформаційна система рієлторської компанії**» буде мати наступні переваги над її аналогами. А саме:

* легка та зрозуміла у користуванні, що дає економію у підготовці до роботи з програмою;
* у порівнянні з аналогами невелика вартість продукту;
* можливість виникнення збоїв у програмі зведена до мінімуму.

В результаті виконання роботи,було досягнуто наступних показників:

* економічний ефект в сфері експлуатації становить 11807,85 грн;
* додатковий економічний ефект в сфері експлуатації становить 15432,00 грн;
* економічний ефект в сфері проектування становить 10314,36 грн;
* додатковий економічний ефект в сфері проектування становить 17619,00 грн.

Таким чином, виходячи з усіх розрахунків, термін окупності програмного модуля складає 4,2 роки.

# ВИСНОВКИ

В роботі розроблено веб-орієнтовану інформаційну систему рієлторської компанії.

Веб-система покликана оптимізувати процеси підбору приміщення, квартири, будівлі працівниками рієлторської компанії.

Користувач системи має змогу ефективно здійснювати пошук, додавати заявку на продаж або здачу в оренду приміщення тощо.

Адміністратор системи має змогу управляти користувачами, управляти довідниками, переглядати детальні описи об’єктів.

В роботі сформовано специфікацію вимог до програмного забезпечення, обгрунтовано вибір засобів та технологій для реалізації веб-системи, описано її функціональність та тестування, сформовано плани забезпечення якості та впровадження програмного забезпечення

Економічна оцінка проектного рішення показала актуальність та доцільність розробленого програмного забезпечення у порівнянні з існуючими інформаційними веб-системами.

# СПИСОК ЛІТЕРАТУРИ

1. Способи використання веб-технологій сучасними підприємствами [Електронний ресурс]. – Режим доступу: http://uastudent.com/sposoby-vykorystannja-veb-tehnologij-suchasnymy-pidpryjemstvamy/ (2016).
2. Семь принципов создания современных веб-приложений [Електронный ресурс]. – Режим доступа: https://habrahabr.ru/ post/242429/ (2016).
3. Web-приложения – поступь будущего [Електронный ресурс]. – Режим доступа: http://www.webmascon.com/topics/technologies/12a.asp (2016).
4. Компанія "Park Lane" [Електронний ресурс]. – Режим доступу: http://parklane.ua/uk/ (2016).
5. Компанія "Чайка" [Електронний ресурс]. – Режим доступу: http://rifchaika.at.ua/ (2016).
6. Компанія "Дуоком" [Електронний ресурс]. – Режим доступу: http://www.duo.com.ua/kompaniya/nash-personal/kraschi-spivrobitnyky (2016).
7. Компанія "Траекторія" [Електронний ресурс]. – Режим доступу: http://www.traektoria.com.ua/ (2016).
8. Компанія "Золоті ворота" [Електронний ресурс]. – Режим доступу: http://www.goldengate.kiev.ua/ (2016).
9. Countent management system [Electronic resource]. – Web access: https://en.wikipedia.org/wiki/Content\_management\_system (2016).
10. Общая классификация CMS [Елетронный ресурс]. – Режим доступа: http://vismech.ru/aboutcms/obshaya-klassifikaciya-cms/ (2016).
11. Top 10 content management systems [Electronic resource]. – Web access:http://www.webdesignerdepot.com/2011/10/top-10-content-management-systems/ (2016).
12. Troelsen A. C# 6.0 and the .NET 4.6 Framework. 7th ed. / A. Troelsen, P. Japikse. – Apress, 2015. – 1625 p.
13. Whitaker R.B. The C# Player's Guide / R.B. Whitaker. – Starbound Software, 2015. – 368 p.
14. Johnson B. Professional Visual Studio 2015 / Bruce Johnson. – Wrox, 2015. – 1320 p.
15. Edwards L. Visual Studio 2015 For Beginners / Lewis Edwards. – CreateSpace Independent Publishing Platform, 2015. – 122 p.
16. The Asp.Net site [Electronic resource]. – Web access: http://www.asp.net/web-forms/books (2016).
17. Penberthy W. Beginning ASP.NET for Visual Studio 2015 / William Penberthy. – Wrox, 2016. – 792 p.
18. Jones D. Learn SQL Server Administration in a Month of Lunches: Covers Microsoft SQL Server 2005-2014 / D. Jones. – Manning Publications, 2014. – 256 p.
19. Rankins R. Microsoft SQL Server 2014 Unleashe / R. Rankins, P. Bertucci, C. Gallelli, A. Silverstein. – Sams Publishing, 2015. – 1992 p.
20. Mistry R. Introducing Microsoft SQL Server 2014 / Ross Mistry, Stacia Misner. – Microsoft Press, 2014. – 144 p.

# Додаток A. Модель бази даних

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Рис. А.1. Модель бази даних | | | | | |
| Виконав | Шуст Д.Б. |  |  | Модель бази даних | Сторінка |
| Перевірив | Тушницький Р.Б. |  |  | 1 з 1 |
| ІПДО кафедра ПЗ | | | |

# Додаток Б. Схема бази даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Рис. Б.1. Схема бази даних | | | | |
| Виконав | Шуст Д.Б. |  | Схема бази даних | Сторінка |
| Перевірив | Тушницький Р.Б. |  | 1 з 1 |
| ІПДО кафедра ПЗ | | |

# Додаток В. Інструкція користувача

**В. 1. Компоненти програмного забезпечення**

Перелік компонент розробленої веб-системи, пояснення їх призначення та інформація про приналежність до проекту наведено у табл. В.1.

Таблиця В.1.  
 Набір компонент для коректної роботи програмного забезпечення

|  |  |  |  |
| --- | --- | --- | --- |
| № | Файл | Призначення | Належить проекту |
| 1 | Realtor.dll | Бібліотека веб-системи | Веб-орієнтована інформаційна система рієлторської компанії |
| 2 | Realtor | База даних об’єктів та їхніх характеристик |
| 3 | RealtorUsers | База даних користувачів |
| 4 | \*.aspx | Файли розмітки веб-системи |
| 5 | \*.css, \*.js, \*.jpg | Файли оформлення веб-системи |

Для запуску веб-системи потрібно мати в наявності веб-сервер IIS із підтримкою .NET Framework та СКБД MS SQL Server 2014.

**В.2. Встановлення програмного забезпечення**

Для встановлення програмного забезпечення потрібно:

1. Розгорнути базу даних Realtor з резервної копії.
2. Створити веб-сайт у IIS і перемістити файли веб-системи у відповідну директорію.
3. Змінити параметри з’єднання із базою даних у файлі налаштувань web.config.

**В.3. Налаштування програмного забезпечення**

Веб-система не потребує додаткових налаштувань.

**В.4. Базові функції програмного забезпечення**

1. Пошук об’єктів певного типу. Користувач має змогу здійснити пошук об’єктів обраного типу. Для цього йому потрібно відкрити систему. Після цього користувачу буде відображено головну сторінку веб-системи. Далі користувач у блоці «я хочу» має обрати тип операції над об’єктом – продаж, купівля, оренда, здача в оренду. Після цього користувачу буде відображено форму для задання параметрів пошуку. Змінивши параметри пошуку і натиснувши на кнопку «Пошук» користувач отримає перелік шуканих об’єктів.
2. Перегляд сторінки із детальним описом об’єкта. Користувач має змогу переглянути сторінку із описом будь-якого об’єкта. Якщо користувач є неавторизованим, то він зможе лише переглянути дані. Якщо користувач є авторизованим і є власником запису, то він зможе його відредагувати. Якщо користувач є авторизований як адміністратор, то він має змогу редагувати дані будь-якого об’єкта.
3. Подання заявки на здачу житла в оренду. Користувач має змогу подати заявку на здачу житла в довгострокову або подобову оренду. Для цього йому потрібно авторизуватись в системі і в блоці «я хочу» обрати пункт меню «здати в оренду». Після цього користувачу буде відображена форма для внесення даних у базу даних. У відповідному полі тип операції буде вказано «Оренда довгострокова» та «Оренда подобова».
4. Управління користувачами та довідниками. Адміністратор системи має змогу управляти користувачами та довідниками. Для цього йому потрібно авторизуватись в систему. Натиснути на посилання «Сторінка адміністратора». Обрати потрібне меню для продовження своїх дій.

# Додаток Д. Лістинг файлу BuildingInfo.aspx

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Script.Services;

using System.Web.Services;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using AjaxControlToolkit;

using Realtor;

namespace WebRealtor

{

public partial class BuildingInfo : Page

{

private int iBuildingID = -1;

private int iStartupType = -1;

protected void Page\_Load(object sender, EventArgs e)

{

var id = Request.QueryString["id"];

if (!string.IsNullOrEmpty(id))

{

try

{

iBuildingID = int.Parse(id);

}

catch (FormatException)

{

Response.Redirect("~/Default.aspx");

}

}

var t = Request.QueryString["t"];

if (!string.IsNullOrEmpty(t))

{

try

{

iStartupType = int.Parse(t);

}

catch (FormatException)

{

iStartupType = -1;

}

}

if (!Page.IsPostBack)

{

using (var context = new RealtorEntities())

{

#region Load DDL

if (iStartupType != -1)

{

var actions = new List<int>();

if (iStartupType == 1)

{

actions.Add(1);

actions.Add(8);

}

if (iStartupType == 2)

{

actions.Add(2);

actions.Add(3);

}

ddlType.DataSource = context.Type.Where(i => actions.Contains(i.TypeID)).OrderBy(i => i.TypeName).Select(i => i);

ddlType.DataBind();

}

else

{

ddlType.DataSource = context.Type.OrderBy(i => i.TypeName).Select(i => i);

ddlType.DataBind();

}

ddlBuildingForm.DataSource = context.BuildingForm.OrderBy(i => i.BuildingFormName).Select(i => i);

ddlBuildingForm.DataBind();

ddlBuildingForm.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

ddlBuildingType.DataSource = context.BuildingType.OrderBy(i => i.BuildingTypeName).Select(i => i);

ddlBuildingType.DataBind();

ddlBuildingType.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

ddlRegion.DataSource = context.Region.OrderBy(i => i.RegionName).Select(i => i);

ddlRegion.DataBind();

ddlRegion.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

ddlCity.DataSource = context.City.OrderBy(i => i.CityName).Select(i => i);

ddlCity.DataBind();

ddlCity.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

ddlCityRegion.Items.Insert(0, new ListItem {Text = "-------", Value = "-1"});

ddlBuildingStatus.DataSource = context.BuildingStatus.OrderBy(i => i.BuildingStatusName).Select(i => i);

ddlBuildingStatus.DataBind();

#endregion

#region Load Data

if (iBuildingID != -1)

{

if (!HttpContext.Current.User.IsInRole("administrator") && !HttpContext.Current.User.IsInRole("moderator"))

{

pnlData.Enabled = false;

trContactPhone.Visible = false;

trUserName.Visible = false;

trButtons.Visible = false;

btnSave.Visible = false;

btnDelete.Visible = false;

btnAddImage.Visible = false;

}

var buildings = from i in context.Building

where i.BuildingID == iBuildingID

select i;

if (buildings.Any())

{

var building = buildings.First();

#region Fill Data

tbStreet.Text = building.StreetAddress != null ? building.StreetAddress.Trim() : string.Empty;

ddlBuildingType.SelectedValue = building.BuildingTypeRef.ToString();

var iBuildingType = building.BuildingTypeRef;

var buildingForm = context.BuildingType.Where(i => i.BuildingTypeID == iBuildingType).Select(i => i.BuildingFormRef).FirstOrDefault();

if (buildingForm != null && ddlBuildingForm.Items.FindByValue(buildingForm.ToString()) != null)

{

ddlBuildingForm.SelectedValue = buildingForm.ToString();

}

ddlType.SelectedValue = building.TypeRef.ToString();

tbRoom.Text = building.TotalRoomNumber.HasValue ? building.TotalRoomNumber.Value.ToString() : string.Empty;

tbPrice.Text = building.Price.HasValue ? building.Price.Value.ToString() : string.Empty;

tbCode.Text = building.Code != null ? building.Code.Trim() : string.Empty;

tbComment.Text = building.Comment != null ? building.Comment.Trim() : string.Empty;

ddlPhonePresent.SelectedValue = building.PhonePresent.HasValue ? (building.PhonePresent.Value ? "1" : "0") : "-1";

tbKitchenSquare.Text = building.KitchenSquare.HasValue ? building.KitchenSquare.Value.ToString() : string.Empty;

tbTotalSquare.Text = building.TotalSquare.HasValue ? building.TotalSquare.Value.ToString() : string.Empty;

tbNonLivingSquare.Text = building.NonLivingSquare.HasValue ? building.NonLivingSquare.Value.ToString() : string.Empty;

tbTotalFloors.Text = building.TotalFloor.HasValue ? building.TotalFloor.Value.ToString() : string.Empty;

lblDateTime.Text = building.DateTimeModified.ToString("dd.MM.yyyy hh:mm");

lblUserName.Text = building.UserName.ToString();

tbContactPhone.Text = building.ContactPhone != null ? building.ContactPhone.Trim() : string.Empty;

ddlBuildingStatus.SelectedValue = building.BuildingStatusRef.ToString();

ddlIsHot.SelectedValue = building.IsHot ? "1" : "0";

#endregion

#region City and Region Info

var iCityRegionID = building.CityRegionRef;

var cityInfo = from i in context.CityRegion

join j in context.City on i.CityRef equals j.CityID

where i.CityRegionID == iCityRegionID

select new

{

j.RegionRef,

j.CityID

};

if (cityInfo.Any())

{

var cityData = cityInfo.First();

var iRegionID = cityData.RegionRef;

var iCityID = cityData.CityID;

ddlRegion.SelectedValue = iRegionID.ToString();

ddlCity.DataSource =

context.City.Where(i => i.RegionRef == iRegionID)

.OrderBy(i => i.CityName)

.Select(i => i);

ddlCity.DataBind();

ddlCity.Items.Insert(0, new ListItem {Text = "-------", Value = "-1"});

if (ddlCity.Items.FindByValue(iCityID.ToString()) != null)

{

ddlCity.SelectedValue = iCityID.ToString();

}

ddlCityRegion.DataSource =

context.CityRegion.Where(i => i.CityRef == iCityID)

.OrderBy(i => i.CityRegionName)

.Select(i => i);

ddlCityRegion.DataBind();

ddlCityRegion.Items.Insert(0, new ListItem {Text = "-------", Value = "-1"});

if (ddlCityRegion.Items.FindByValue(building.CityRegionRef.ToString()) != null)

{

ddlCityRegion.SelectedValue = building.CityRegionRef.ToString();

}

}

#endregion

DoBindRoomSquares();

DoLoadImages();

}

else

{

Response.Redirect("~/Default.aspx");

}

}

else

{

trDateTime.Visible = false;

trUserName.Visible = false;

trImages.Visible = false;

lnkbtnAddRoomSquare.Enabled = false;

btnAddImage.Visible = false;

if (!HttpContext.Current.User.IsInRole("administrator") && !HttpContext.Current.User.IsInRole("moderator"))

{

Response.Redirect("~/Default.aspx");

}

}

#endregion

}

}

}

[WebMethod, ScriptMethod]

public static List<string> GetStreets(string prefixText, int count, string contextKey)

{

using (var context = new RealtorEntities())

{

prefixText = prefixText.Trim();

var res = (from i in context.Building

where i.StreetAddress.Contains(prefixText)

orderby i.StreetAddress ascending

select i.StreetAddress).Distinct().ToList();

return res.Any() ? res : new List<string> { "Нічого не знайдено" };

}

}

protected void ddlRegion\_OnSelectedIndexChanged(object sender, EventArgs e)

{

if (ddlRegion.SelectedValue != string.Empty && ddlRegion.SelectedIndex != 0)

{

using (var context = new RealtorEntities())

{

try

{

var iRegionID = int.Parse(ddlRegion.SelectedValue);

ddlCity.DataSource = context.City.Where(i => i.RegionRef == iRegionID).OrderBy(i => i.CityName).Select(i => i);

ddlCity.DataBind();

ddlCity.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

var iCityID = ddlCity.SelectedValue != string.Empty && ddlCity.SelectedIndex != 0 ? int.Parse(ddlCity.SelectedValue) : -1;

ddlCityRegion.DataSource = context.CityRegion.Where(i => i.CityRef == iCityID).OrderBy(i => i.CityRegionName).Select(i => i);

ddlCityRegion.DataBind();

ddlCityRegion.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

}

catch (FormatException)

{

return;

}

}

}

else

{

ddlCity.Items.Clear();

ddlCity.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

}

}

protected void btnSave\_Click(object sender, EventArgs e)

{

if (HttpContext.Current.User.IsInRole("administrator") || HttpContext.Current.User.IsInRole("moderator"))

{

try

{

using (var context = new RealtorEntities())

{

if (iBuildingID == -1)

{

if (ddlCityRegion.SelectedValue == "-1")

{

Alert.Show("Виберіть район в місті");

return;

}

#region Add New

var build = new Building();

build.StreetAddress = tbStreet.Text.Trim();

build.BuildingTypeRef = int.Parse(ddlBuildingType.SelectedValue);

build.TypeRef = int.Parse(ddlType.SelectedValue);

if (tbRoom.Text.Trim() != string.Empty)

{

build.TotalRoomNumber = int.Parse(tbRoom.Text.Trim());

}

if (tbPrice.Text.Trim() != string.Empty)

{

build.Price = float.Parse(tbPrice.Text.Trim());

}

build.Code = tbCode.Text.Trim();

build.Comment = tbComment.Text.Trim();

if (ddlPhonePresent.SelectedValue != "-1")

{

build.PhonePresent = ddlPhonePresent.SelectedValue == "1";

}

if (tbKitchenSquare.Text.Trim() != string.Empty)

{

build.KitchenSquare = float.Parse(tbKitchenSquare.Text.Trim());

}

if (tbTotalSquare.Text.Trim() != string.Empty)

{

build.TotalSquare = float.Parse(tbTotalSquare.Text.Trim());

}

if (tbNonLivingSquare.Text.Trim() != string.Empty)

{

build.NonLivingSquare = float.Parse(tbNonLivingSquare.Text.Trim());

}

if (tbTotalFloors.Text.Trim() != string.Empty)

{

build.TotalFloor = int.Parse(tbTotalFloors.Text.Trim());

}

build.DateTimeModified = DateTime.Now;

build.UserName = HttpContext.Current.User.Identity.Name.ToLower();

build.ContactPhone = tbContactPhone.Text.Trim();

build.BuildingStatusRef = int.Parse(ddlBuildingStatus.SelectedValue);

build.IsHot = ddlIsHot.SelectedValue == "1";

build.CityRegionRef = int.Parse(ddlCityRegion.SelectedValue);

context.AddToBuilding(build);

context.SaveChanges();

Alert.Show("Дані збережено");

Response.Redirect("~/BuildingInfo.aspx?id=" + build.BuildingID);

#endregion

}

else

{

#region Update

var builds = from i in context.Building

where i.BuildingID == iBuildingID

select i;

if (builds.Any())

{

var build = builds.First();

build.StreetAddress = tbStreet.Text.Trim();

build.BuildingTypeRef = int.Parse(ddlBuildingType.SelectedValue);

build.TypeRef = int.Parse(ddlType.SelectedValue);

if (tbRoom.Text.Trim() != string.Empty)

{

build.TotalRoomNumber = int.Parse(tbRoom.Text.Trim());

}

else

{

build.TotalRoomNumber = null;

}

if (tbPrice.Text.Trim() != string.Empty)

{

build.Price = float.Parse(tbPrice.Text.Trim());

}

else

{

build.Price = null;

}

build.Code = tbCode.Text.Trim();

build.Comment = tbComment.Text.Trim();

if (ddlPhonePresent.SelectedValue != "-1")

{

build.PhonePresent = ddlPhonePresent.SelectedValue == "1";

}

else

{

build.PhonePresent = null;

}

if (tbKitchenSquare.Text.Trim() != string.Empty)

{

build.KitchenSquare = float.Parse(tbKitchenSquare.Text.Trim());

}

else

{

build.KitchenSquare = null;

}

if (tbTotalSquare.Text.Trim() != string.Empty)

{

build.TotalSquare = float.Parse(tbTotalSquare.Text.Trim());

}

else

{

build.TotalSquare = null;

}

if (tbNonLivingSquare.Text.Trim() != string.Empty)

{

build.NonLivingSquare = float.Parse(tbNonLivingSquare.Text.Trim());

}

else

{

build.NonLivingSquare = null;

}

if (tbTotalFloors.Text.Trim() != string.Empty)

{

build.TotalFloor = int.Parse(tbTotalFloors.Text.Trim());

}

else

{

build.TotalFloor = null;

}

build.DateTimeModified = DateTime.Now;

build.UserName = HttpContext.Current.User.Identity.Name.ToLower();

build.ContactPhone = tbContactPhone.Text.Trim();

build.BuildingStatusRef = int.Parse(ddlBuildingStatus.SelectedValue);

build.IsHot = ddlIsHot.SelectedValue == "1";

build.CityRegionRef = int.Parse(ddlCityRegion.SelectedValue);

context.SaveChanges();

Alert.Show("Дані збережено");

}

#endregion

}

}

}

catch (Exception ex)

{

Alert.Show("Виникла помилка. Дані не збережено");

}

}

else

{

Response.Redirect("~/Default.aspx");

}

}

protected void btnDelete\_Click(object sender, EventArgs e)

{

if (iBuildingID != -1 && (HttpContext.Current.User.IsInRole("administrator") || HttpContext.Current.User.IsInRole("moderator")))

{

using (var context = new RealtorEntities())

{

var build = from i in context.Building

where i.BuildingID == iBuildingID

select i;

var rooms = from i in context.RoomSquare

where i.BuildingRef == iBuildingID

select i;

foreach (var i in rooms)

{

context.DeleteObject(i);

context.SaveChanges();

}

var floors = from i in context.BuildingFloor

where i.BuildingRef == iBuildingID

select i;

foreach (var i in floors)

{

context.DeleteObject(i);

context.SaveChanges();

}

var images = from i in context.BuildingImage

where i.BuildingRef == iBuildingID

select i;

foreach (var i in images)

{

context.DeleteObject(i);

context.SaveChanges();

}

context.DeleteObject(build);

context.SaveChanges();

}

Response.Redirect("~/Default.aspx");

}

}

protected void ddlBuildingForm\_OnSelectedIndexChanged(object sender, EventArgs e)

{

if (ddlBuildingForm.SelectedValue != string.Empty && ddlBuildingForm.SelectedIndex != 0)

{

try

{

var iFormID = int.Parse(ddlBuildingForm.SelectedValue);

using (var context = new RealtorEntities())

{

ddlBuildingType.DataSource = context.BuildingType.Where(i => i.BuildingFormRef == iFormID).OrderBy(i => i.BuildingTypeName).Select(i => i);

ddlBuildingType.DataBind();

ddlBuildingType.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

}

}

catch (FormatException)

{

return;

}

}

}

protected void lnkbtnAddRoomSquare\_OnClick(object sender, EventArgs e)

{

if (iBuildingID == -1)

{

Alert.Show("Спершу збережіть інформацію про об'єкт");

}

else

{

if (tbRoomSquare.Text.Trim() != string.Empty)

{

try

{

var squareValue = float.Parse(tbRoomSquare.Text.Trim());

using (var context = new RealtorEntities())

{

context.AddToRoomSquare(new RoomSquare {BuildingRef = iBuildingID, RoomSquareValue = squareValue});

context.SaveChanges();

tbRoomSquare.Text = string.Empty;

}

DoBindRoomSquares();

}

catch (FormatException)

{

Alert.Show("Коректно введіть дані");

}

}

else

{

Alert.Show("Коректно введіть дані");

}

}

}

protected void btnDeleteRoomSquare\_Click(object sender, EventArgs e)

{

var button = sender as LinkButton;

try

{

#region Try to delete

using (var context = new RealtorEntities())

{

var sid = int.Parse(((TextBox)button.NamingContainer.FindControl("tbRoomSquareIDHid2")).Text.Trim());

var roomSquare = from i in context.RoomSquare

where i.RoomSquareID == sid

select i;

if (roomSquare.Any())

{

context.DeleteObject(roomSquare.First());

context.SaveChanges();

}

}

#endregion

}

catch (Exception ex)

{

Alert.Show(ex.ToString());

}

DoBindRoomSquares();

}

private void DoBindRoomSquares()

{

#region Room Info

using (var context = new RealtorEntities())

{

if (HttpContext.Current.User.IsInRole("administrator") || HttpContext.Current.User.IsInRole("moderator"))

{

phRoomSquares.Visible = true;

lblRoomSquares.Visible = false;

var rooms = from i in context.RoomSquare

where i.BuildingRef == iBuildingID

orderby i.RoomSquareValue ascending

select i;

gvRoomSquares.DataSource = rooms;

gvRoomSquares.DataBind();

gvRoomSquares.Visible = rooms.Any();

}

else

{

lblRoomSquares.Text = string.Join(", ", (from i in context.RoomSquare

where i.BuildingRef == iBuildingID

orderby i.RoomSquareValue ascending

select i.RoomSquareValue).ToList());

phRoomSquares.Visible = false;

lblRoomSquares.Visible = true;

}

}

#endregion

}

protected void ddlCity\_OnSelectedIndexChanged(object sender, EventArgs e)

{

if (ddlCity.SelectedValue != string.Empty && ddlCity.SelectedIndex != 0)

{

using (var context = new RealtorEntities())

{

try

{

var iCityID = int.Parse(ddlCity.SelectedValue);

ddlCityRegion.DataSource = context.CityRegion.Where(i => i.CityRef == iCityID).OrderBy(i => i.CityRegionName).Select(i => i);

ddlCityRegion.DataBind();

ddlCityRegion.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

}

catch (FormatException)

{

return;

}

}

}

else

{

ddlCityRegion.Items.Clear();

ddlCityRegion.Items.Insert(0, new ListItem { Text = "-------", Value = "-1" });

}

}

protected void btnCancel\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Default.aspx");

}

protected void btnAddImage\_OnClick(object sender, EventArgs e)

{

mpePhoto2.Show();

}

#region File Upload

protected void asyncFileUpload\_UploadedComplete(object sender, AsyncFileUploadEventArgs e)

{

var fileUploadReceive = asyncFileUpload.PostedFile.FileName;

var uploadedFileSize = asyncFileUpload.PostedFile.ContentLength;

if (uploadedFileSize < 16777216)

{

using (var context = new RealtorEntities())

{

if (iBuildingID == -1)

{

return;

}

context.AddToBuildingImage(new BuildingImage

{

BuildingRef = iBuildingID,

ImagePath = fileUploadReceive

});

context.SaveChanges();

}

asyncFileUpload.SaveAs(MapPath("Files\\" + fileUploadReceive));

}

}

protected void asyncFileUpload\_UploadedFileError(object sender, AsyncFileUploadEventArgs e)

{

}

protected void lnkbtnFileModalPopup\_Click(object sender, EventArgs e)

{

if (iBuildingID != -1)

{

mpePhoto2.Show();

}

}

protected void lnkbtnCloseFileModalPopup\_Click(object sender, EventArgs e)

{

mpePhoto2.Hide();

DoLoadImages();

}

#endregion

private void DoLoadImages()

{

if (iBuildingID != -1)

{

using (var context = new RealtorEntities())

{

var images = from i in context.BuildingImage

where i.BuildingRef == iBuildingID

select i;

repImages.DataSource = images;

repImages.DataBind();

trImages.Visible = images.Any();

}

repImages.Visible = true;

}

else

{

repImages.Visible = false;

trImages.Visible = false;

}

}

protected void lnkbtnDeleteImage\_OnClick(object sender, EventArgs e)

{

var button = sender as LinkButton;

try

{

#region Try to delete

using (var context = new RealtorEntities())

{

var sid = int.Parse(((TextBox)button.NamingContainer.FindControl("tbBuildingImageIDHid")).Text.Trim());

var images = from i in context.BuildingImage

where i.BuildingImageID == sid

select i;

if (images.Any())

{

context.DeleteObject(images.First());

context.SaveChanges();

}

}

#endregion

}

catch (Exception ex)

{

Alert.Show(ex.ToString());

}

DoLoadImages();

}

}

}

# Додаток Е. Тестові випадки використання програмного забезпечення

Таблиця Е.1

ID № 1:: TestCase Пошук об’єктів довготривалої оренди

|  |  |
| --- | --- |
| Користувач має змогу здійснити пошук об’єктів довготривалої оренди | |
| Steps (Кроки/дії) | Expected Results (Очікувані результати) |
| 1. Відкрити головну сторінку системи. 2. Натиснути в блоці «Я хочу» на посилання «Орендувати». 3. Вказати необхідні параметри пошуку і натиснути кнопку «Шукати». 4. Натиснути кнопку «Очистити». | 1. Користувачу відображено результати пошуку останніх внесених об’єктів, а також в блоці «я хочу» наявний пункт меню «Орендувати». 2. Користувачу відображена форма для задання параметрів пошуку. У полі тип операції є значення “Оренда довготривала” та “Оренда подобова”. 3. Користувачу відображено релевантні до заданих параметрів результати пошуку. 4. Користувачу відображено останні додані об’єкти. |

Таблиця Е.2

ID № 2:: TestCase Перегляд неавторизованим користувачем детального опису об’єкта

|  |  |
| --- | --- |
| Неавторизовані користувачі мають змогу переглянути сторінку із детальним описом вибраного об’єкта | |
| Steps (Кроки/дії) | Expected Results (Очікувані результати) |
| 1. Обрати товар викорис-товуючи головну сторінку для пошуку. 2. Перейти на сторінку опису об’єкта. | 1. Користувачу доступна функціональність вибору об’єкта шляхом відображення посилання «детально», а також наявністю посилання на об’єкт, яке розміщено на фотографії об’єкту. 2. Користувачу доступний перегляд детального опису об’єкта, але не доступна зміна жодного параметру. |

Таблиця Е.3

ID № 3:: TestCase Управління користувачами

|  |  |
| --- | --- |
| Адміністратор системи має змогу управляти користувачами | |
| Steps (Кроки/дії) | Expected Results (Очікувані результати) |
| 1. Авторизуватись в систему. 2. Натиснути на посилання «Сторінка адміністратора». 3. Натиснути на посилання «Користувачі». 4. Натиснути на посилання облікового запису будь-якого користувача. | 1. Користувачу відображено головну сторінку системи, а також наявне меню адміністратора. 2. Коритувачу відображено сторінку із переліком можливих дій. Наявне посилання «Користувачі». 3. Адміністратору відображено сторінку із переліком зареєстрованих користувачів. 4. Адміністратору відображено сторінку для управління обліковим записом обраного користувача. |