МІНІСТЕРСТОВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп‘ютерної та програмної інженерії
Кафедра комп’ютерних інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.С. Савченко
«____» _____2021 р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СПУПЕНЯ «БАКАЛАВР»

Тема: «Веб-додаток на базі каталогу місць проведення дозвілля»

Виконавець: студентка УС-411 Безверха Катерина Сергіївна
(студент, група, прізвище, ім'я, по батькові)

Керівник: к. т. н., доцент Холявкіна Тетяна Володимирівна
(науковій ступень, вчене звання, прізвище, ім'я, по батькові)

Нормоконтролер: ст. викл. Шевченко О.П. _____
(П.І.Б.) (підпис)

КИЇВ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп‘ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Освітній ступінь: Бакалавр

Галузь знань, спеціальність, спеціалізація: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ
Завідувач кафедри

А.С. Савченко
“_____”_____ 2021 р.

ЗАВДАННЯ

на виконання дипломного проекту студента

Безверха Катерина Сергіївна

(прізвище, ім’я, по батькові)

1.Тема проекту: «Веб-додаток на базі каталогу місць проведення дозвілля» затверджена наказом ректора від 22.04.2021р. №636/ст.

2. Термін виконання роботи: з 10.05.2021р. по 20.06.2021р.

3. Вихідні дані до роботи: хмарне середовище Microsoft Azure; середовища розробки MS SQL, Visual studio; full-stack фреймворк – .NET Framework; мова програмування – C#; технології front-end частини – HTML, CSS, Razor, JavaScript, jQuery, Bootstrap, AJAX; схема додатку MVC; трирівнева архітектурна модель додатку.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка завдання, реалізація завдання та особливості архітектури, аналіз виконаного завдання та його характеристики, висновки.

5. Перелік обов’язкового графічного матеріалу: зображення діаграм бази даних, скріншоти вікон серверу, схеми «MVC» та архітектури веб-додатку, скріншоти вікон середовища розробки, зображення скріншотів частин готового проекту.

6. Календарний план-графік

№ п/п	**Завдання**	**Термін виконання**	**Підпис керівника**
1.	Аналіз літератури та джерел за темою дипломного проекту.	10.05.2021 14.05.2021	
2.	Розроблення та затвердження плану дипломного проекту.	15.05.2021 17.05.2021	
3.	Проведення консультації з науковим керівником щодо створення першого розділу.	18.05. 2021 19.05. 2021	
4.	Розробка розділу 1	20.05. 2021 25.05.2021	
5.	Розробка розділу 2	26.05. 2021 31.05. 2021	
6.	Розробка розділу 3	01.06. 2021 07.06. 2021	
7.	Висновки та оформлення пояснювальної записки дипломного проекту.	08.06. 2021 11.06. 2021	
8.	Підписання необхідних документів у встановленому порядку.	12.06.2020 13.06.2021	
9.	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	14.06.2021 15.06.2021	

7. Дата видачі завдання: 10.05.2021р.

Керівник дипломного проекту _____ Холявкіна Т.В.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Безверха К.С.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Веб-додаток на базі каталогу місць проведення дозвілля» викладена на 66с., містить 36 рис., 20 літературних джерел, 2 додатки.

Мета роботи – розробити веб-додаток «Каталог місць проведення дозвілля», для зберігання в одному місці всіх можливих розваг та пропозицій проведення дозвілля, зі зручною та зрозумілою будь-якому користувачу системою фільтрування та привабливим виглядом сторінок.

Методи дослідження: аналіз теоретичного матеріалу веб-сайтів та веб-додатків схожого направлень; аналіз та вибір методів і технологій реалізації веб-додатків і баз даних.

Об'єкт дослідження – веб-додаток.

Предмет дослідження – хмарний веб-додаток на базі каталогу місць проведення дозвілля.

Новизна роботи -

Результати дипломного проекту рекомендується використовувати користувачам з метою пошуку інформації щодо пропозицій розваг та відпочинку, з можливістю сортувати інформацію за категоріями та характеристиками, а також власникам бізнесу з відповідною тематикою для рекламування своїх послуг.

ВЕБ-ДОДАТОК, ХМАРНІ ТЕХНОЛОГІЇ, СЕРВЕРИ, БАЗИ ДАНИХ, SQL, C#, VISUAL STUDIO, AZURE, MVC, ТРИРІВНЕВА АРХІТЕКТУРА, RAZOR, HTML, CSS, JAVASCRIPT, BOOTSTRAP, AJAX, REGEX.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

VCS – Система контролю версій

HTML – Мова розмітки гіпертексту

CSS – Каскадна таблиця стилів

MVC – Модель-представлення-контролер

DAL – Шар доступу до даних

BAL – Шар бізнес-логіки

XML – Розширювана мова розмітки

AJAX – Асинхронний Javascript та XML

SQL – Декларативна мова програмування

ВСТУП

Сьогодні звичайна доросла людина проводить більшу частину свого життя за роботою, а коли постає питання відпочинку, обирає вже звичний для себе варіант. Як правило, це зустрічі в кафе з друзями або виїзд на природу. Проте мало хто замислюється, скільки насправді існує цікавих та незвичних варіантів проведення дозвілля: від стрибків с парашутом до кінних прогулянок за містом. Зібрання багатьох подібних пропозицій в одному місці допоможе людям набагато якісніше організовувати власний відпочинок або проводити вільний час з користю, наприклад відкривши для себе нове хобі.

Метою роботи є проектування єдиної бази усіх розважальних пропозицій, щоб користувачі мали змогу в одному місці знайти для себе саме той варіант проведення дозвілля, який би вони хотіли.

Особливістю даного проекту та основною відмінністю від сайтів зі схожою тематикою буде наявність не лише загальних порад виду «щось може вас зацікавити», а конкретних пропозицій від власників розважального бізнесу, зі вказанням цін, контактної інформації та розподілом за категоріями. Також для зручності пошуку буде реалізовано систему характеристик позицій, які дозволять користувачам одразу відфільтровувати непотрібні варіанти або навпаки, знаходили лише ті, що підходять для вказаного віку, кількості людей, сезону, тощо.

Для того, щоб охопити якнайбільше потенційних користувачів, позиції будуть збиратись та сортуватись за містами, в яких (або поблизу яких) вони розташовуються. Слід також звернути увагу на той факт, що питання проведення дозвілля постає не тільки у місцевих мешканців, а й у туристів та приїжджих. Для їх комфорту буде реалізовано можливість обирати мову, якою їм було б зручніше за все отримувати інформацію. Тому деякі дані в базі будуть зберігатись у трьох екземплярах, перекладених кожною мовою відповідно (українська, російська та англійська).

Створення вищеописаної інформаційної системи буде корисним не лише для користувачів, а і для власників розважального бізнесу. Адже єдина база розваг, окрім

того, що стане черговою рекламою пропозицій, дозволить більш цілеспрямовано знаходити потенційних клієнтів.

Дипломний проект складається з трьох розділів.

В першому розділі роботи розглянуто предметну область «каталог місць проведення дозвілля», її основні поняття та особливості, на основі яких спроектовано базу даних. Розглянуто можливість використання хмарних сервісів Azure для функціонування бази даних і веб-додатку. Проаналізовано структуру веб-додатку і його взаємодію з базою даних, а також описано основні технології реалізації проекту.

В другому розділі розглянуто технології реалізації рішення проекту та безпосередньо самого веб-додатку у середовищі Visual Studio. Розглянуто детально архітектуру функціонування кожного з рівнів, та основну логіку системи. Наведено приклади використання технологій на сторінках користувача та адміністратора.

В третьому розділі проаналізовано розроблений проект та розписано весь функціонал системи. Розглянуто різницю між адміністративним додатком та користувацьким. Пояснення супроводжуються скріншотами.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ БАЗИ ДАНИХ

1.1. База даних в хмарному середовищі

Для початку роботи над додатком необхідно детально проаналізувати предметну область та спроектувати базу даних.

1.1.1. Предметна область

Предметною областю проекту є каталог місць проведення дозвілля. Тобто під це визначення підпадають всі можливі розважальні пропозиції, починаючи від звичайних прогулянок та екскурсій, закінчуючи стрибками з парашутом та квест-кімнатами. Отже, спроектована система має бути максимально універсальною та гнучкою. Всі дані для веб-додатку будуть зберігатися у хмарній базі даних. Важливим також є якісне проектування таблиць та зв’язків, тому що у випадку помилки додаток може працювати неправильно, або не відповідати всім поставленим вимогам. Основна інформація, яку має надавати система – це назва розваги, зображення та опис. Саме ці дані будуть надаватися користувачам веб-додатку. Так як проект має бути універсальним, то всі позиції розваг у такому вигляді потребують певних характеристик для сортування та зручної навігації. Тому необхідним буде створення загальних категорій, за якими будуть відсортовані пропозиції, а також пошукової системи, яка буде знаходити співпадіння символів і допомагати користувачам орієнтуватися на сторінці.

1.1.2. Проектування бази даних

Метою створення бази даних є складання каталогу місць проведення дозвілля, доступ до якого буде здійснюватись через веб-додаток. Відповідний сайт має надавати користувачам інформацію щодо категорій розважальних заходів, їх місцезнаходження, ціну та часи роботи. Крім того, на сайті має розміщуватись декілька зображень до кожної позиції, короткий опис послуги та посилання на контакти власників і офіційні сторінки в мережах. Також для зручності користувачів слід реалізувати перемикач мов, а до кожної позиції додати відповідні властивості для більш легкого сортування.

В основу бази даних буде покладено реляційну модель, тобто табличне представлення даних. Кожна таблиця міститиме власні рядки для збереження даних, а взаємозв'язки між записами будуть здійснюватися через так звані ключі, прив'язані до рядків (Primary Key та Foreign Key).

Для того, щоб спроектована база даних містила всю необхідну інформацію, умовно виділимо основні групи, що повинні бути представлені у вигляді таблиць бази. Для наглядності побудуємо до кожної групи діаграми за допомогою інструментарію середовища SQL Server Management Studio, в яких буде відображено таблиці з назвами, назви рядків і типи даних, що вони містить, а також зв'язки між самими таблицями.

Перша група таблиць буде стосуватися об'єктів. До її складу увійдуть такі таблиці: Subjects, SubjectsImages, SubjectsCategoryMap, SubjectsKindsMap, SubjectsText (Рис. 1.1.). Розглянемо їх детальніше:

1. Об'єкти (Subjects) – безпосередньо позиції запропонованих розваг;
2. Зображення Об'єктів (SubjectImages) – фотографії у звичайному та зменшеному розмірах для перегляду на сторінці;
3. Опис Об'єктів (SubjectText) – опис позицій розваг на декількох доступних мовах;

4. Карта Категорій Об’єктів (SubjectsCategoriesMap) – список категорій кожного об’єкта;
5. Карта Типів Категорій Об’єктів (SubjectsFeatureKindsMap) – список типів категорій об’єктів.

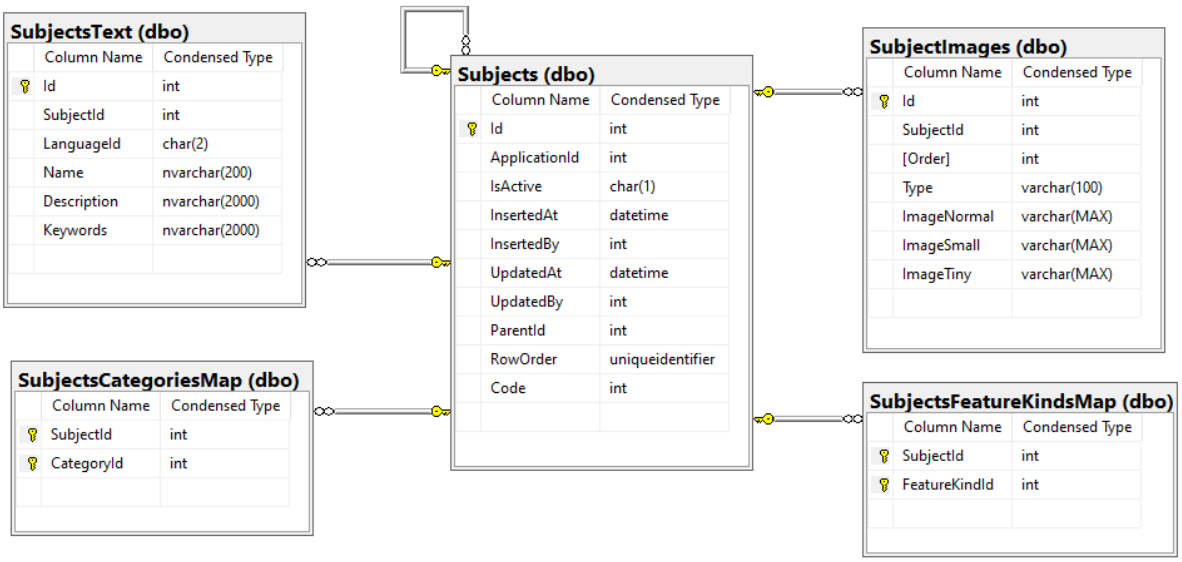

Рис. 1.1. Зв'язки між таблицями групи об'єктів

Друга група таблиць буде стосуватися категорій об’єктів. До її складу увійдуть таблиці: Categories, CategoriesFeatureMap, CategoriesHierarchyMap, CategoriesText (Рис. 1.2.). Розглянемо їх детальніше:

1. Категорії (Categories) – номери ідентифікаторів категорій для зручної роботи з базою;
2. Опис категорій (CategoriesText) – назви груп позицій за видом діяльності (наприклад, ресторан, польоти, прогулянки, тощо);
3. Карта Властивостей Категорій (CategoriesFeatureMap) – список властивостей категорій;
4. Карта ієрархії категорій (CategoriesHierarchyMap) – категорії, що посилаються самі на себе (категорії підкатегорій).

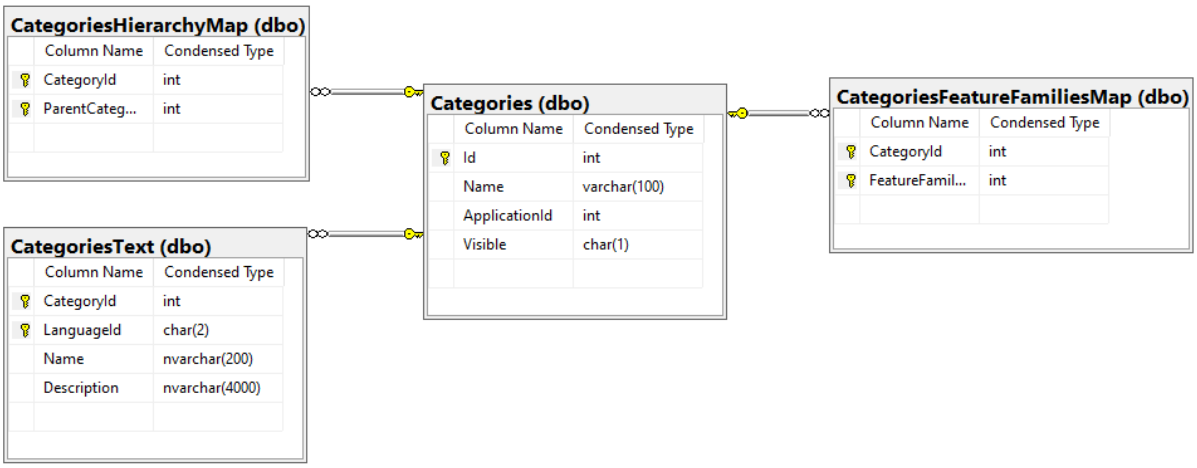

Рис. 1.2. Зв’язки між таблицями групи категорій

Третя група таблиць буде стосуватися характеристик категорій. До її складу увійдуть таблиці: FeatureFamilies, FeatureFamiliesText, FeatureKinds, FeatureKindsText, FeatureModes, Features (Рис. 1.3.). Розглянемо їх детальніше:

1. Характеристики (Features) – номери ідентифікаторів властивостей для зручної роботи за базою;
2. Опис властивостей (FeatureKinds) – опис властивостей характеристики (наприклад, видима/невидима, порядок відображення, тощо);
3. Опис характеристик (FeaturesKindsText) – назви характеристик позицій для зручного фільтрування (наприклад ціна, email і т.д.);
4. Сімейство характеристик (FeatureFamilies) – безпосередньо самі характеристики (наприклад, ціна, сезон, майсер-клас, тощо);
5. Назви сімейств характеристик (FeatureFamiliesText) – назви самих характеристик;
6. (FeatureModes) – типи характеристик (числове значення, текст, тощо).

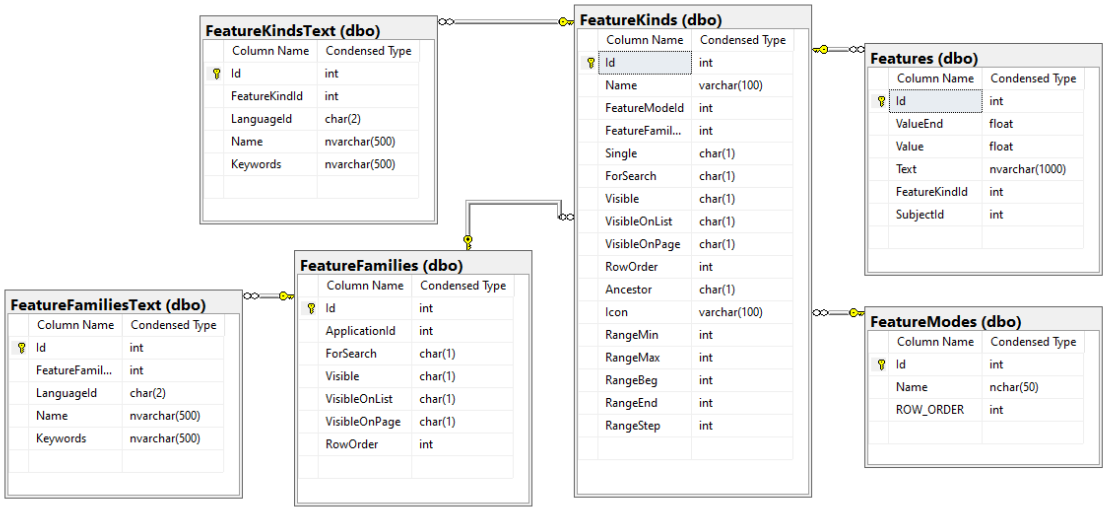

Рис. 1.3. Зв’язки між таблицями групи властивостей

До останньої групи увійдуть таблиці, які не відносяться до жодної групи з попередніх. Віднесемо сюди такі таблиці: Languages, Roles, Access, Modules, RolesAccessModulesMap, Users, UsersRolesMap, Towns, TownsText (Рис. 1.4.). Розглянемо їх детальніше:

1. Мови (Languages) – перелік доступних мов з іконками прапорців;
2. Користувачі (Users) – клієнти веб-додатку, що бажають зареєструватися на сайті;
3. Ролі (Roles) – спеціальна таблиця для розподілу доступу та можливостей клієнтів (адміністратор або користувач);
4. Доступ (Access) – таблиця що містить типи доступу (читання, зміна і т.д.);
5. Модулі (Module) – містить види функціоналу для яких буде надано доступ (наприклад редагування об’єктів або їх характеристик);
6. Карта доступу до модулів за ролями (RolesAccessModulesMap) – ідентифікатори функціоналів і типів доступу для ролей;
7. Карта ролей користувачів (UsersRolesMap) – ролі користувачів для розподілу прав (адміністратор, користувач, оператор);
8. Міста (Towns) – ідентифікатори міст на різних мовах;
9. Назви Міст (TownsText) – назви міст на різних мовах.

Рис. 1.4. Зв’язки між додатковими таблицями та іншими групами таблиць

1.1.3. Технології реалізації бази даних

Робота з базою даних буде проводитись за допомогою структурованої мови запитів SQL в програмному середовищі Visual Studio. В роботі будуть використані представлення, збережені процедури і користувацькі функції.

Збереженими процедурами називають спеціальні набори SQL-інструкцій, що компілюються одноразово та зберігаються на сервері. Завдяки використанню таких процедур можливо підвищити швидкодійність, тому що замість збереження запиту, який часто використовується, клієнти мають змогу посилатися на відповідну збережену процедуру, яка одразу ж буде оброблена сервером. На відміну від користувацьких функцій збережена процедура викликається за допомогою функції CALL або EXECUTE. У порівнянні зі збереженими процедурами, користувацькі функції мають трохи менше можливостей, але натомість вони дозволяють повертати скалярне значення або цілу таблицю. Використання представлень дозволить значно спростити декілька таблиць, поєднавши їх в одну віртуальну. Адже представлення – це логічна таблиця, яка не є самостійною частиною набору даних, а являє собою результат виконання запиту (SELECT). Такий спосіб визволення даних з таблиць у момент звернення до представлення дає більш гнучку можливість

налаштувань прав доступу до даних, а також дозволяє розділяти логіку збереження даних та програмного забезпечення.

Розглядаючи питання розташування бази даних, одним з найоптимальніших варіантів буде використання послуг хмарних сервісів. Хмарні бази даних створюються і зберігаються в хмарних сховищах, але у порівнянні з функціоналом традиційних баз даних, вони мають декілька істотних переваг. По-перше, такі бази будуть мати більшу гнучкість налаштувань завдяки власним ресурсам хмари. По-друге, при роботі з хмарними сервісами користувачі мають змогу керувати віддаленою базою будь-де і будь-коли, без потреби купувати додаткове обладнання. По-третє, автоматизоване хмарне програмне забезпечення у багатьох випадках може замінити кваліфікованих адміністраторів баз даних, а використання найновіших технологій захисту гарантовано вбереже цілісність даних навіть в екстрених випадках.

Тож для зберігання створеної бази даних використаємо можливості хмарного сервісу Microsoft Azure. Основними підготовчими діями для внесення бази у хмару є створення серверу бази даних Azure SQL Server, а також групи ресурсів для більш зручного керування всіма ресурсами пов’язаними з базою і майбутнім веб-додатком.

Для створення групи ресурсів слід зайти у розділ Resource groups та вибрати опцію «Додати». Далі – ввести назву майбутньої групи ресурсів, вказати регіон місцезнаходження та підписку Azure, до якої буде прив’язана ця група (Рис.1.5.).

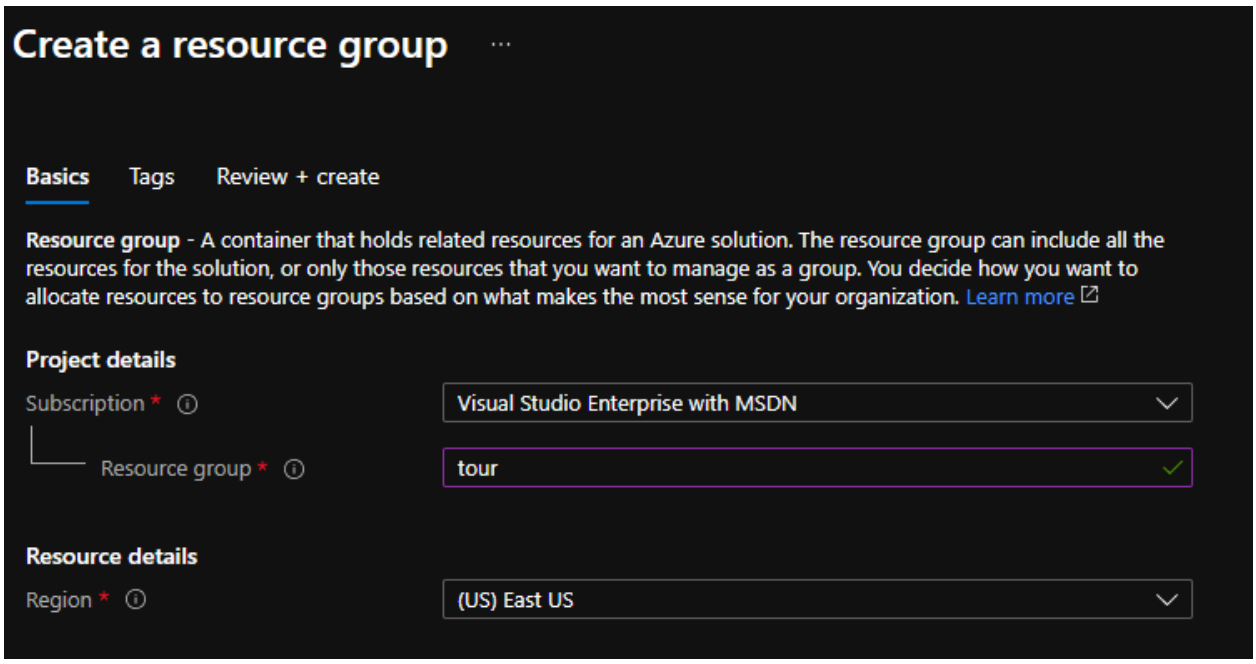

Рис. 1.5. Вікно створення групи ресурсів

В ході створення серверу обираємо глобально унікальне ім'я – («ek1zrnzvc5») та ім'я адміністратора – («sadmin»). Після чого вводимо пароль для входу і обираємо місцерозташування – (East US) (Рис. 1.6.).

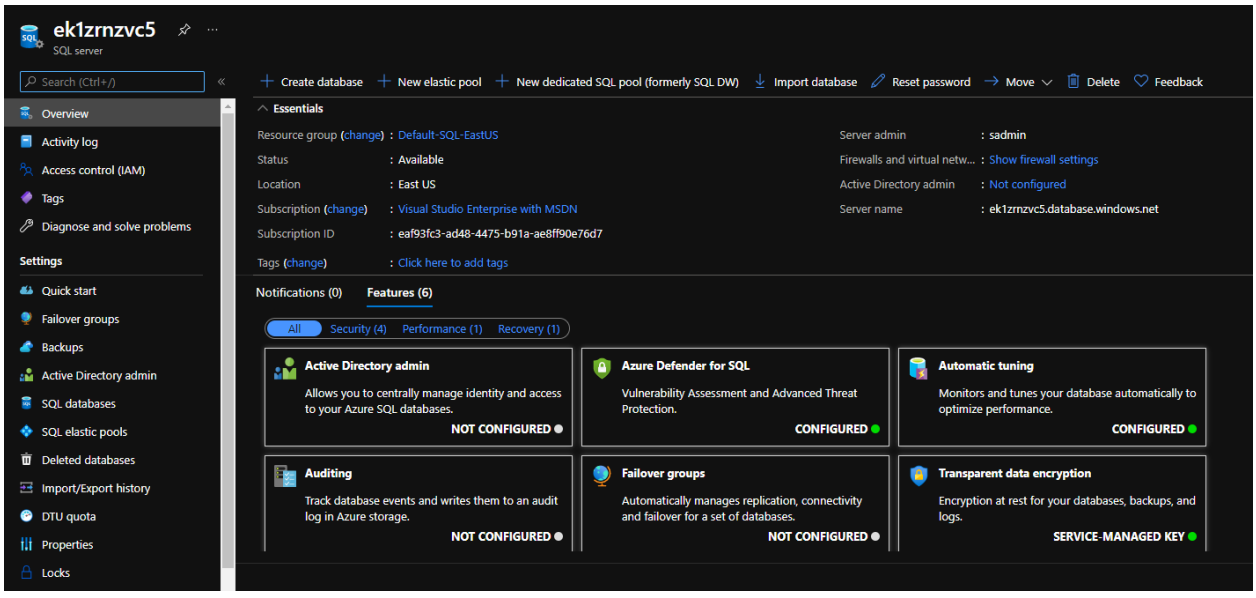

Рис. 1.6. Створений діючий SQL-сервер

В ході створення бази даних SQL обираємо тип ресурсу «окрема база даних» і вводимо назву «tour». Наступним кроком призначаємо у відповідних вікнах створені до цього групу ресурсів та сервер.

Для подальшої роботи з базою буде використовуватись програмне середовище Visual Studio з інтегрованими сервісами Azure. Для цього створюємо новий проект типу «Database Project», а для підключення до бази використаємо текстову строку «Connection String», яка містить іменовані дані для доступу (Рис. 1.7.). В ході цих дій було отримано можливість створювати таблиці та вносити до них зміни через середовище Visual Studio. Так, за допомогою запитів мови SQL було відтворено всі спроектовані раніше таблиці та зв’язки між ними. [1]

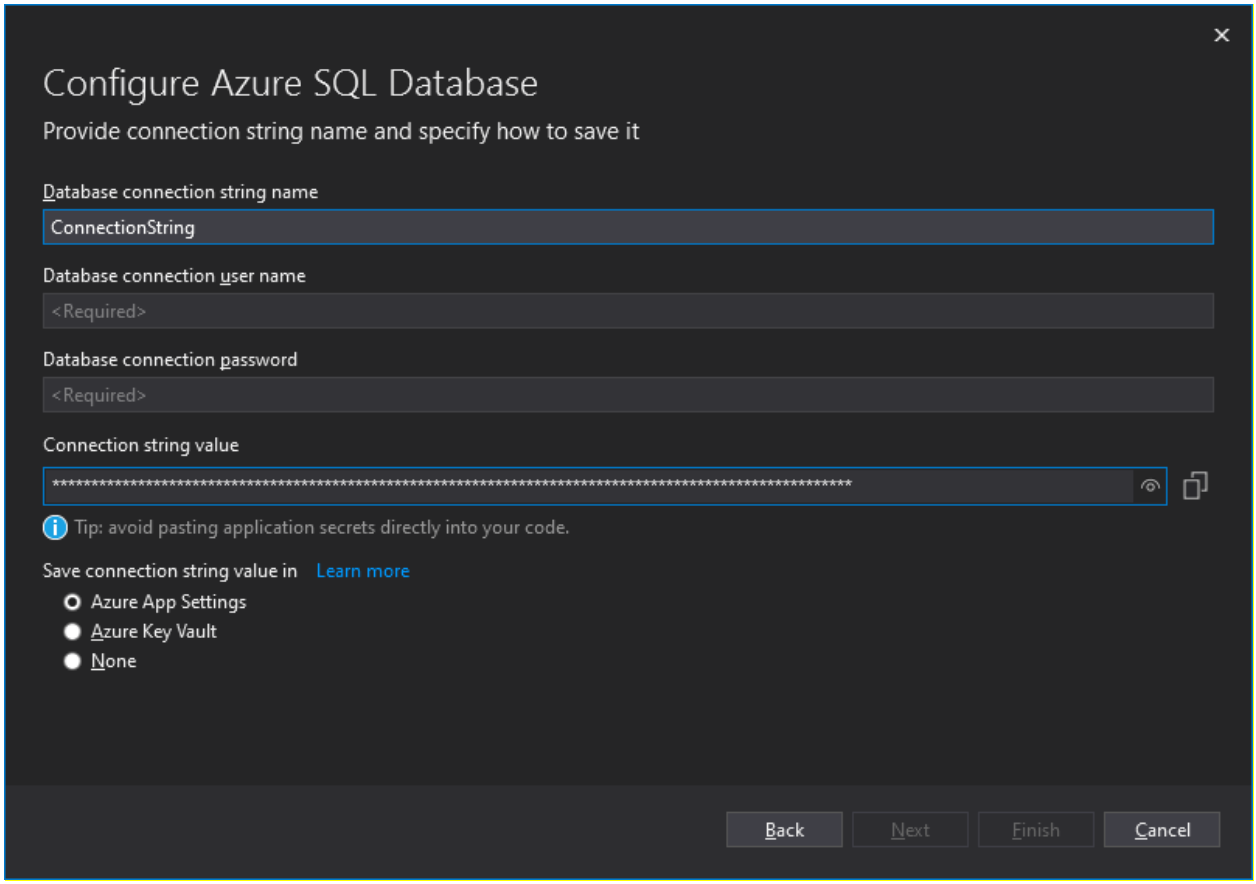

Рис. 1.7. Підключення до бази даних за допомогою Connection String через середовище Visual Studio

1.2. Інтеграція веб-додатку з базою даних

Веб-додаток має безперешкодно взаємодіяти з базою даних, отримувати інформацію від сервера та за потребою вносити зміни у сховище. Розглянемо детальніше процес інтеграції спроектованої БД з веб-додатком та структуру останнього, яка буде найоптимальнішою в даному випадку.

1.2.1. Структура і технології реалізації веб-додатку

Веб-додаток являє собою веб-сайт, в якому кінцевий зміст сторінок визначатиметься за запитом користувача. Тобто, на відміну від звичайного веб-сайту, який містить лише статичні, підготовані завчасно HTML-файли, сторінки веб-додатку є динамічними і генеруються в залежності від введених умов.

Додаток буде реалізовано мовою C# на програмній платформі .NET 4.8. В якості програмного середовища, як і для роботи з базами даних, буде використано Visual Studio. В основу роботи веб-додатку буде покладено так званий принцип MVC (Model (Модель) – View (Представлення) – Controller (Контролер)) (Рис. 1.8.). Цей принцип дозволить розділити логіку додатку, його зовнішній вигляд та взаємодію с користувачем, завдяки чому написання коду буде більш безпечним в плані виникнення помилок, а сам написаний код – більш зрозумілим та структурованим. Як і видно із назви, компонент View (він же Представлення) буде описувати лише зовнішній вигляд додатку; Model (Модель) – містити звернення до бази даних, а також здійснювати обробку даних; Controller (Контролер) – виступатиме в ролі елементу-сполучення між Моделлю і Представленням. Тобто Контролер буде отримувати дані від користувача, передавати їх у Модель, після чого отримувати оброблений результат й передавати його до Представлення. [2]

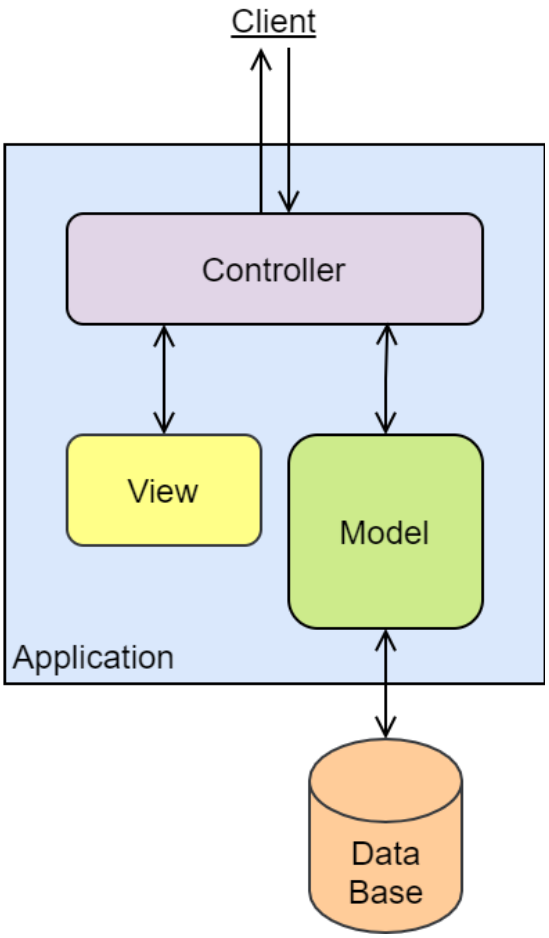

Рис. 1.8. Графічне зображення функціонування моделі MVC

Завдяки використанню принципу MVC стає можливим використання спеціального движку представлення – Razor. Цей движок дозволяє використовувати сторінки написані за допомогою HTML та C# в одному. Тобто робить можливим поєднання двох синтаксисів. [Приклад використання робочої моделі MVC – Додаток А]

Крім того сам веб-додаток використовуватиме трирівневу клієнт-серверну архітектуру і складатиметься з 3-х шарів: шар доступу до даних (Data Access Layer), шар логіки (Execution Layer) та шар представлення (Presentation Layer) (Рис. 1.9.). Такий архітектурний розподіл дозволить розбити додаток на підзадачі різних рівнів абстракцій. Наприклад, шар доступу до даних – надає доступ до баз даних та повертає або створює записи в сховищі. Шар логіки –

правила за якими визначаються залежності та поведінка об'єктів в системі. Шар представлення – являє собою користувацький інтерфейс. [3]

Зовнішній вигляд сторінок веб-додатку буде описано за допомогою мови розмітки HTML та каскадних таблиць стилів CSS. Для опису логіки функціонування використаємо мову програмування JavaScript і JQuery.

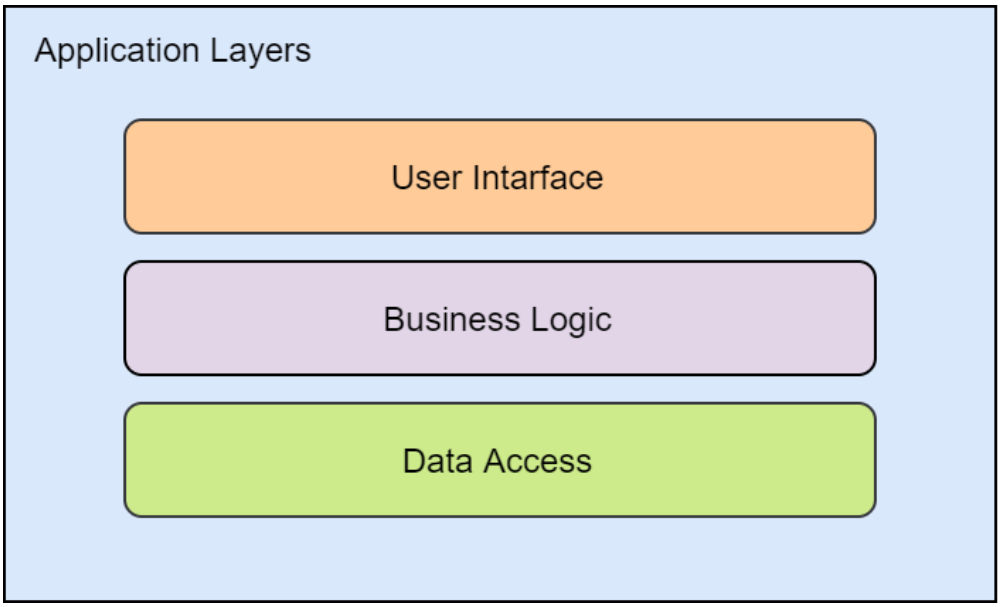

Рис. 1.9. Графічне зображення трирівневої архітектури додатку

Готовий веб-додаток буде містити дві сторінки для перегляду: користувацьку та адміністративну. На відміну від користувацької сторінки, адміністративна буде мати інструменти керування елементами (наприклад, можливість додавати, змінювати або видаляти дані з бази). Доступ до цієї сторінки буде надаватися лише користувачам з присвоєними правами адміністратора або оператора. Для всіх інших користувачів інтернету буде відкриватись основна користувацька сторінка, яка дозволить лише для переглядати інформацію, без можливості внесення змін.

1.2.2. Обмін даними між веб-додатком і базою даних

Обмін даними буде відбуватися в межах шару Data Access через MS SQL Client за допомогою SQL команд. В методі формується нова SQL команда або викликається збережена процедура і відправляється на сервер. [Приклад збереженої процедури – Додаток Б] Після обробки сервер повертає дані або результат виконання. Якщо потрібно передати великий або складний об'єкт, його буде серіалізовано, тобто перетворено у XML-документ, який передається за призначенням. В такому випадку сервер обробляє XML-файл. Файл з розширенням .xml – це текстовий файл, що розширює мову розмітки (eXtensible Markup Language) і описує документ.

При XML-серіалізації в потік XML серіалізуються лише відкриті поля і значення властивостей об'єкта, але не враховується інформація про тип. Центральним класом такої серіалізації є клас XmlSerializer. Він створює файли C# і компілює їх у файли DLL. Найбільш важливими методами цього класу є Serialize і Deserialize, які відповідають за прямий та зворотній процеси серіалізації. Проте сервер, який буде використовуватися при розробці веб-додатку може обробляти серіалізовані XML-файли та не потребує десеріалізації. [4]

Для оптимізації роботи системи буде використано багатопотоковість, в тому числі при запитах до серверу баз даних. Тобто деякі важливі процеси будуть виконуватися паралельно (наприклад, отримання списку категорій та їх характеристик із серверу), а тривалість метода, в якому вони будуть викликані визначатиметься найдовшою тривалістю виконання потоку.

Щоб пришвидшити завантаження деяких даних буде запроваджено систему кешування, тобто збереження деяких відносно статичних даних у кеші. Кеш – це спеціальна пам'ять з більшою швидкістю доступу, яка складається з набору записів. Кожен запис асоційований з елементом даних, що являє собою копію цього ж елемента в основній пам'яті. Крім того, кожен запис має ідентифікатор (він же «тег»), який визначає відповідність між

елементами даних у кеші та їх копіями в основній пам’яті. Дані в кеші зазвичай зберігаються на пристрої зі швидким доступом, наприклад на оперативній пам’яті. Зазвичай обсяг пам’яті кешу досить невеликий, але це компенсується високою швидкістю доступу.

Висновок до розділу

Створення бази даних на сьогодні є дуже важливою задачею для будь-якого спеціаліста в області інформаційних технологій. А використання хмарних сховищ з метою зберігання бази робить цю задачу ще більш актуальною для сучасного світу. Адже управління віддаленими серверами зараз є дуже простим та доступним майже для будь-якого користувача. Крім того перевага над використанням фізичних серверів є доволі значною: всі проблеми зі збереження цілісності даних та функціонуванні структур беруть на себе самі сервіси. Віддалені бази даних також дають можливість керувати ними з будь-якої точки та декільком користувачам одночасно.

На сьогоднішній день веб-додатки мають значні переваги над звичайними веб-сайтами. Основними перевагами є простота встановлення та користування, а також висока надійність. Завдяки використанню багаторівневої архітектури (в даному випадку трирівневої) веб-додаток стає більш продуктивним та масштабованим. В ході розробки додатку така архітектура дозволить програмістам розроблювати кожен рівень (шар) одночасно, не залежачи один від одного. Крім того, у випадку виникнення неполадок стає можливим працювати лише із шаром, в якому виникла проблема, не торкаючись компонентів інших рівнів.

РОЗДІЛ 2
СТУКТУРА І ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ

2.1. Веб-додаток у середовищі Visual Studio

Інтегроване середовище для розробки Microsoft Visual Studio має вбудовані інструменти для роботи з багатьма видами проектів, у тому числі й з веб-додатками. Крім того, веб-додаток повинен взаємодіяти з веб-сервером за допомогою браузера. Так як сервер з базою даних вже було створено у попередньому розділі за допомогою хмарного сховища Microsoft Azure, залишається створити сам додаток і налаштувати його взаємодію з сервером.

2.1.1. Рішення проекту веб-додатку

Для створення веб-додатку у середовищі Visual Studio вже є всі необхідні готові файли шаблонів, що дозволить значно скоротити час розробки. Отже при створенні нового проекту слід обрати мову C# у списку запропонованих мов програмування та шаблон веб-додатку ASP.NET Core Web App. За замовчуванням такий тип проекту використовує технологію Razor для обробки запитів. [5] Після ряду наступних стандартних налаштувань проекту (таких як ім'я, розташування на диску, версії фреймворку, тощо) ріщення проекту матиме такі базові компоненти: Connected Services – підключені сервіси Azure, Dependencies – всі залежності (додані пакети та бібліотеки), Properties – вузол, який містить деякі налаштування проекта (наприклад, налаштування адреси запуску, тощо.), файл appsettings.json – конфігураційний файл проекту та два стартових класи: Program.cs та

Startup.Cs. Перший відповідає за налаштування та запуск веб-хосту, а другий містить логіку обробки вхідних запитів. [6] Для подальшої роботи над проектом необхідно створити декілька компонентів різних типів. Після всіх маніпуляцій рішення проекту буде мати наступний вигляд (Рис. 2.1.):

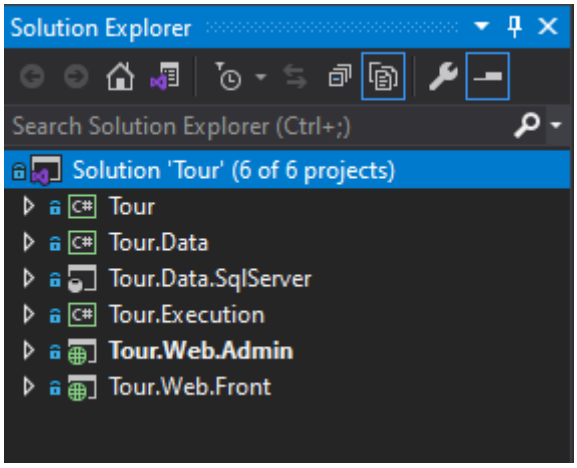

Рис. 2.1. Зовнішній вигляд рішення проекту веб-додатку

Як видно з Рис. 2.1., готове рішення проекту для веб-додатку має назву «Tour» та містить компоненти:

Tour – основа системи, де зберігаються загальні класи для функціонування всього проекту, а також конфігураційні файли та кеш. Компонент Tour містить такі класи: Cache (для кешування чего?), Conversion (для перетворення зображень в bitmap з метою збереження в базі), Configuration (додає об’єктам файли конфігурацій), DataProcessor (для перетворення відповіді SQL сервера в об’єкт у відповідності з класом), Definition (містить додаткові класи для опису модулів системи, до яких буде надано доступ певного виду), Helper (містить методи, що допомогають форматувати введені строки), Utilities (містить методи для функціонування логіки системи).

Компонент Tour.Data являє собою шар доступу до даних (DAL). Тут містяться методи для роботи з базою та класи, куди заносяться відповіді від сервера баз даних. Для більш зручної розробки класи, що використовуються

для повернення відповіді від сервера були внесені до папки Domain, а методи для звернень до БД – до папки Service.

Tour.Data.SqlServer – модель даних, що представляє собою саму БД. Компонент призначений для роботи з базою: зміни, зроблені в цій моделі відображаються в самій базі на сервері.

Компонент Tour.Execution являє собою шар бізнес-логіки (BAL). Тут зберігається бізнес-логіка. Тут зберігається вся логіка роботи додатку, окрім звернень до серверу баз даних (наприклад, регулює фільтрування позицій за категоріями).

Tour.Web.Admin – безпосередньо сам веб-додаток, що призначений для використання операторами та адміністратором, з можливостями внесення змін до бази (наприклад, додавання та видалення позицій, редагування описів, тощо).

Tour.Web.Front – веб-додаток, призначений для перегляду користувачами.

2.1.2. Система контролю версій Git

Щоб процес розробки був більш безпечний в плані виникнення помилок для відстеження змін буде використано систему контролю версій Git.

Система контролю версій (VCS) являє собою програмне забезпечення, яке дозволяє відстежувати зміни в документах, при необхідності проводити їх відкат, визначати, хто і коли вніс виправлення і т.п. Існують локальні, централізовані та децентралізовані системи. При використанні локальних систем VCS попередні версії зберігаються локально на комп’ютері. У випадку з централізованими системами використовується єдиний сервер, на якому зберігаються всі файли. Доступ до нього можуть отримати одразу декілька клієнтів. Такий підхід доволі зручний при розробці програми одразу декількома програмістами, або у разі роботи з декількох комп’ютерів. Розподілені системи контролю версій не просто завантажують стани файлів у

певний період часу, а повністю копіюють весь репозиторій (повний бекап усіх даних). Тобто у випадку поломки одного з серверів обміну даними, будь-який клієнтський репозиторій може бути скопійовано на інший сервер для подовження роботи. [7]

Середовище Visual Studio 2019 має власний вбудований клієнт Git. На Рис. 2.2. зображено вікно історії змін у проекті, автор, дата та ідентифікаційний код.

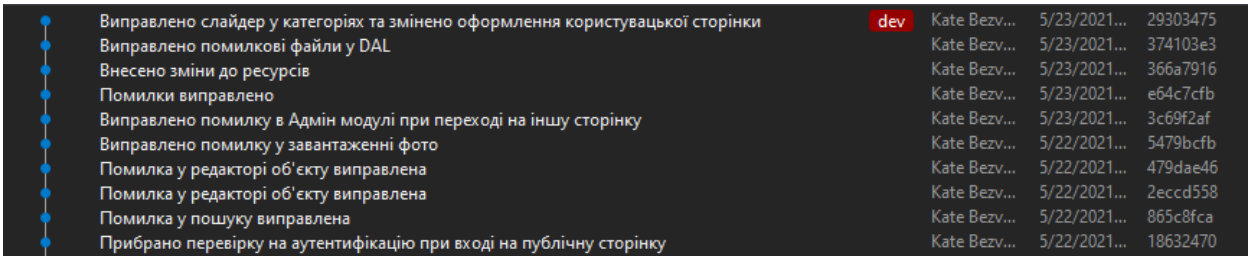

Рис. 2.2. Вікно історії системи контролю версій

2.2. Архітектурні рівні додатку

Як вже було зазначено, веб-додаток складатиметься з трьох основних архітектурних рівнів: DAL, BAL і User Interface. Розглянемо їх детальніше.

2.2.1. Рівень Data Access

Архітектурно рівень доступу до даних (DAL) знаходиться в самому низу та не може посилатися ні на які інші рівні. В рішенні проекту цей рівень представлено компонентом під назвою Tour.Data. До його функціоналу входить отримання даних від SQL-сервера та комунікація з БД. Дані, які DAL отримує від бази розміщуються у відповідні класи (Entities). Тобто Entities – це і є представлення даних у вигляді класів, завдяки яким додаток може їх розуміти та використовувати. Комунікація рівня доступу до даних з базою відбувається за допомогою SQL-команд (в тому числі й збережених процедур),

що містяться в методах на цьому рівні. [8] Рівень бізнес-логіки (BAL) за потребою звертається до методів DAL для отримання даних.

Для визначення моделей, об’єкти яких будуть отримуватися, до папки Domain додано відповідні класи, а саме: Application, Categories, Country, Features, IdName, Image, Language, Role, Subjects, Text та User.

Як приклад, представлений вміст класу Subjects. Для того, щоб успішно спілкуватися з тими частинами коду, які потребують властивості, а не поля використовуються ключові слова get і set. За допомогою методів GetValue відповідні дані підтягуються з бази та зберігаються у вигляді властивостей однойменних змінних:

```
public class aSubjectText : IRowLight
 {
   public int SubjectId { get; set; }
   public string LanguageId { get; set; }
   public string LanguageName { get; set; }
   public string Name { get; set; }
   public string Description { get; set; }
   public string Keywords { get; set; }
   public virtual void Init(SqlDataReader readerSQL, List<string> ColumnList)
   {
     SubjectId = Command.GetValue<int>(readerSQL, "SubjectId");
     LanguageId = Command.GetValue<string>(readerSQL, "LanguageId");
     LanguageName = Command.GetValue<string>(readerSQL, "LanguageName");
     Name = Command.GetValue<string>(readerSQL, "Name");
     Description = Command.GetValue<string>(readerSQL, "Description");
     Keywords = Command.GetValue<string>(readerSQL, "Keywords");
   } }
```

2.2.2. Рівень Business Logic

Рівень представлення не може напряму отримувати дані з БД. Тому в ролі посередника існує рівень бізнес логіки, який інкапсулює всі необхідні обчислення. Рівень бізнес-логіки або BAL представлено в проекті компонентом Tour.Execution та частково у самих додатках адміністратора та користувача. Класи, що керують логікою всієї системи, представляють собою елемент контролер та знаходиться у папках Controllers компонентів веб-додатків Tour.Web.Admin та Tour.Web.Front. Саме контролер обробляє дії користувачів під час роботи веб-додатку, шляхом звернення до методів рівня DAL. Також контролер може звертатися до інших частин BAL. Таким чином, інформація отримана у вигляді відповідей від DAL обробляється для формування наступного рівня архітектури – шару представлення. [9]

Як приклад, даний перевіряє, який з видів доступу присвоєно користувачу:

```
public class UserExecution
 {
  public bool UserHasAccess(ApplicationType applicationId, int userId, Module module, Access access)
  {
   var result = new Data.Service.UserService().UserHasAccess(userId, module, access);
   return result;
  }
 }
```

Таким чином, спочатку іде перевірка прав доступу користувача, після чого створюються об’єкти класу Tour.Data (рівень DAL). Потім за допомогою створених об’єктів формується запит до бази даних. Отримані дані

поміщуються в модель та передаються до шару представлення. Отже BAL виконує функції посередника між шаром доступу до БД та шаром представлення, отримуючи об’єкти з одного рівня та передаючи їх до іншого. Tour.Web.Front – веб-додаток, сторінки якого призначені для перегляду користувачами.

2.2.3. Рівень User Interface

Рівень представлення, або Presentation Layer є найвищим з рівнів архітектури веб-додатку. Він відповідає за взаємодію додатку з користувачем. Саме шар представлення отримує модель даних, оброблену контролером, на базі якої будується вигляд веб-сторінки. Дані обробляються за допомогою мови Razor – суміші HTML та C#. [10] В системі рівень представлення – це компоненти веб-додатків Tour.Web.Admin та Tour.Web.Front. Наприклад, виведення даних про категорії позицій на сторінку буде виглядати наступним чином:

```
<div class="row">
    @foreach (var item in Model.CategoryList)
    {
      @Html.Partial("CategoryItem", item);
    }
  </div>
```

2.3. Основні технології

Оглянемо основні технології завдяки яким стала можлива реалізація тематичного вед-додатку, мету використання кожної технології та конкретні приклади.

2.3.1. Мова програмування C#

C# - сучасна об’єктно-орієнтована мова програмування, що дозволяє створювати безліч додатків різних типів. Однією з переваг даної мови є наявність великої кількості вбудованих бібліотек та шаблонів, використання яких дозволить значно скоротити час розробки. Інструментарій мови C# надає можливості для рішення великого кола задач. Аналогічно іншим об’єктно-орієнтованим мовам програмування, синтаксис C# має дві великі категорії типів: базові типи, які вже є наявними, та класи, які програміст може створювати самостійно. [11] Вся основна логіка проекту побудована саме цією мовою. Наприклад, даний цикл foreach перебирає наявні, «увімкнені» типи характеристик:

```
foreach (var featureKindBase in featureKindListBase)
    {
     var _featureKind = featureKindList.Where(x => x.Id == featureKindBase.Id).FirstOrDefault();
     if (_featureKind != null)
     {
      featureKindBase.Checked = true;
      featureKindBase.RangeBeg = _featureKind.RangeBeg;
      featureKindBase.RangeEnd = _featureKind.RangeEnd;
     }
    }
```

2.3.1. HyperText Markup Language

Оглянемо основні технології завдяки яким стала можлива реалізація Завдяки використанню мови гіпертекстової розмітки браузер «розуміє», як треба правильно відображати завантажену сторінку. В ході виконання проекту

мова HTML використовується для розмітки сторінок веб-додатку. Сама по собі мова складається з тегів, тобто певних команд, які перетворюються у візуальні об’єкти в браузері користувача. Отже HTML задає лише каркас веб-сторінки, в якому можуть бути прописані посилання, таблиці, зображення, блоки, абзаци, форми, заголовки і таке інше. Проте для створення повноцінного сайту, крім мови розмітки слід використовувати ще як мінімум так звану таблицю стилів CSS. [12] При створенні HTML-коду для веб-додатку слід враховувати, що він має дві основні сторінки для відображення (клієнтську та адміністративну), а отже елементи та розмітка для них будуть декілька відрізнятись. Розглянемо як приклад використання мови HTML для створення таблиці на сторінці адміністратора (Рис. 2.3.):

```
<table class="table table-striped">
  <thead>
   <tr>
    <th>Id</th>
    <th>Name</th>
    <th>Name List</th>
    <th>Shared</th>
    <th>For Search</th>
    <th>Visible</th>
    <th>Visible On List</th>
    <th>Visible On Page</th>
    <th></th>
   </tr>
  </thead>
  <tbody>
```

Id	Name	Name List	Shared	For Search	Visible	Visible On List	Visible On Page	
13	Location	Location, Расположение, Розташування,		✔	✔	✔	✔	
1018	Capacity	Capacity, Вместимость, Місткість,		✔	✔	✔	✔	
1017	Type of aircraft	Type of aircraft, Тип воздушного судна, Тип повітряного судна,		✔	✔	✔	✔	
1019	System	System, Система, Система,	✔		✔	✔	✔	
1020	Season	Season, Сезон, Сезон,		✔	✔	✔	✔	
1021	Excursion	Excursion, Экскурсия, Екскурсія,		✔	✔	✔	✔	
1022	Walking on a horse	Walking on a horse, Прогулки на лошади, Прогулянки на коні,	✔	✔	✔	✔	✔	

Рис. 2.3.Таблиця створена за допомогою мови розмітки

2.3.2. Cascading Style Sheets

Каскадна таблиця стилів, або скорочено CSS підключається до коду HTML-сторінки. Якщо HTML описує основні елементи у вікні браузера, то таблиця стилів описує їх зовнішній вигляд, тобто стильові атрибути. CSS має схожий на мову розмітки синтаксис, завдяки якому можна описати відступи, вирівнювання, позиціонування, прозорість, межі, кольори та багато інших властивостей. Завдяки використанню цієї технології зовнішній вигляд сторінок веб-додатку можна зробити набагато привабливішими та більш струсткрованими. Стилі можна розміщувати прямо на самій HTML сторінці всередині тегу <head>. Для цього треба лице вписати ще один додатковий тег <style>, який буде вказувати на застосування CSS. Також можливе застосування стилів через аналогічний атрибут доданий до деяких окремих тегів. Ще один спосіб використання цієї технології – це прив’язати до HTML сторінки окремий файл з розширенням .css. Для цього слід створити файл з таким розширенням та прив’язати його до основної сторінки за допомогою тегу <link>. Застосування останнього способу більш зручне, адже завдяки винесенню форматування стилів в окремий файл працювати з ними стає набагато легше. [13] Як прикляд розглянемо застосування стилів CSS до елементів користувацької сторінки, а саме до організації позицій на сторінці та їх оформлення (Рис. 2.4.):

.subject_list_container .item { height: 250px; width: 100%; padding: 15px; margin-bottom: 0px; vertical-align: top; border: solid 1px silver; position: relative; background-color: floralwhite; opacity: 2; }

.subject_list_container .item .title { display: block; width: 100%; color: #00759A; text-decoration: none; font-size: 14px; line-height: 14px; margin-bottom: 5px; }

.subject_list_container .item .title div { height: 120px; margin-bottom: 5px; background-repeat: no-repeat; }

.subject_list_container .item .owner { display: inline-block; font-size: 12px; font-family: Calibri; color: brown; line-height: 12px; }

.subject_list_container .item .control { position: absolute; top: 0px; right: 0px; border: solid 1px silver; border-radius: 3px; background-color: antiquewhite; padding: 0px 3px 0px 3px; }

.subject_list_container .item .season img { height: 20px; width: 20px; }

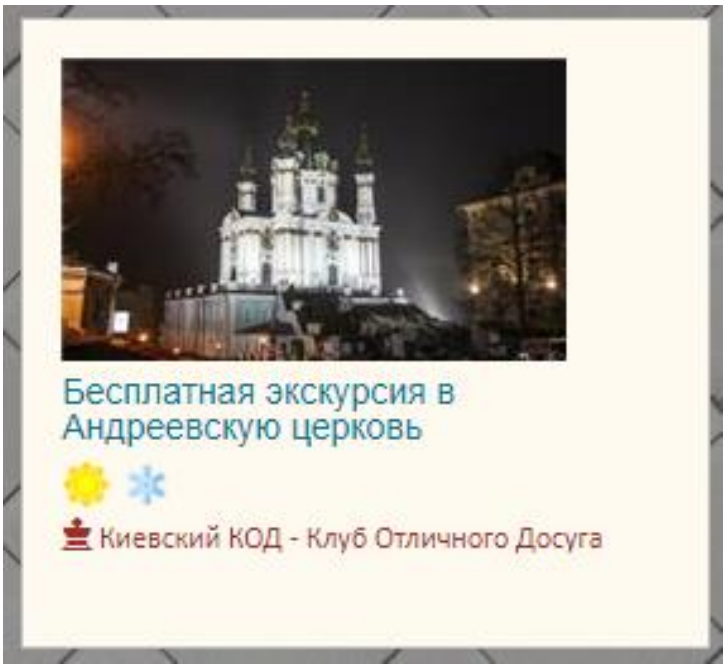

Рис. 2.4. Стильове оформлення одного з елементів позицій

2.3.3. Javascript

Javascript (або JS) – це об’єктно-орієнтована мова програмування, що використовується у написанні сценарієв для веб-сторінок. У браузері для JavaScript доступним є все, що пов'язано з маніпулюванням веб-сторінками, взаємодією з користувачем і веб-сервером. Наприклад, в браузері JavaScript може модифікувати HTML-код та стилі, реагувати на дії користувача, додавати новий HTML-код на сторінку, змінювати існуючий вміст, модифікувати стилі, реагувати на дії користувача, клацання миші, відправляти мережеві запити на віддалені сервера, завантажувати файли, тощо. [14]

Для полегшення написання коду до Javascript підключена спеціальна додаткова бібліотека під назвою JQuery. Вона дозволяє створювати складні події, використовуючи при цьому набагато менше строк коду. Наприклад, за допомогою цього фреймворку можна робити анімації з елементами сторінки, або маніпулювати стилями CSS. Базовий синтаксис виглядає наступним чином: «$(селектор).дія()». Знак «$» для вказання початку роботи з JQuery, (селектор) – для запиту або пошуку елементу HTML, дія() – безпосередньо дія виконувана над обраним елементом. [15] В проекті JQuery використовується в тому числі й у функції видалення фільтрів для пошуку:

```
function DeleteAllFilters() {
$('.search_container input.search_text').val('');
$('.feature_family_container input:checked').each(function () { $(this).prop('checked', false); });
}
```

2.3.4. Bootstrap

Bootstrap являє собою набір шаблонів оформлення HTML, CSS та JavaScript для створення сайтів та веб-додатків. Такі набори включають в себе готові веб-форми, кнопки, мітки, блоки навігації та інші компоненти веб-інтерфейсу. Використання цього фреймворку дозволяє значно пришвидшувати розробку фронтенду сайтів та інтерфейсів адміністративних сторінок. До складу Bootstrap входять: інструменти для створення макету (система сіток, контейнері, адаптивних утиліт та ін.), готові компоненти (кнопки, форми, слайдери, випадаючі списки, тощо), класи для стилізації базового контенту (тексту, зображень, кода, таблиць і т.д.), утилітних класів для рішення традиційних задач (наприклад, вирівнювання тексту, відображення скритих елементів, задання кольору, відступів, фону, тощо.). [16] Однією з головних особливостей Bootstrap є можливість групування строк та колонок сайту за допомогою сітки. За замовчуванням у сітці цього фреймворку 12 колонок. Колонка має клас col-x-x, де перший x – позначення пристрою, а другий – кількість колонок від 1 до 12. [17] В ході роботи над проектом Bootstrap було використано для створення випадаючих меню, кнопок, а також для групування об’єктів на сторінці (Рис. 2.5.):

```
<div class="row">
   <div class="col-sm-5" style="padding-left: 40px;">
    @for (int i = 0; i < Model.TextList.Count(); i++) ...
   </div>
   <div class="col-sm-3 subject_categories">
    <fieldset class="fieldset" style="min-height: 500px;">...</fieldset>
   </div>
   <div class="col-sm-4" id="feature_list_container" style="min-height: 400px; padding-right: 40px;">
    @Html.Action("FeatureListFromDataBase", Model)  </div>  </div>
```

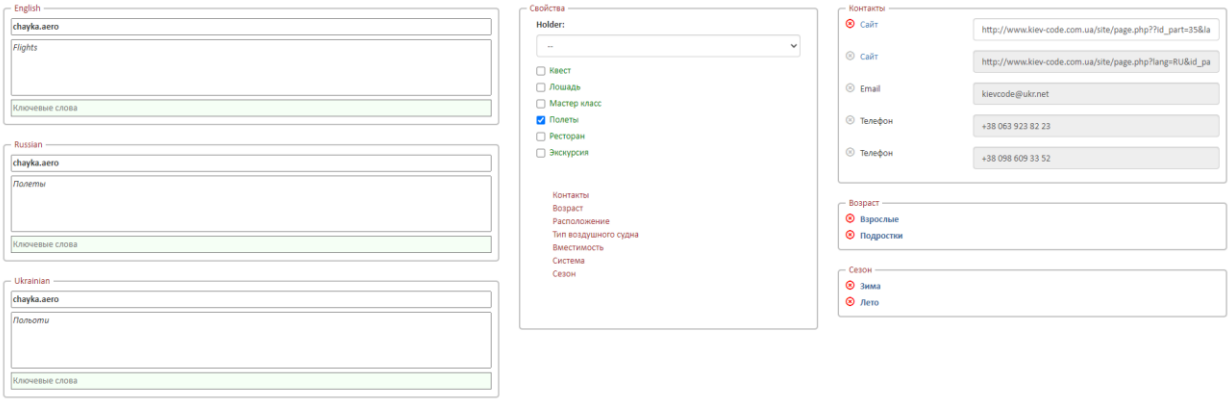

Рис. 2.5. Згруповані об’єкти на сторінці

2.3.5. Razor

Razor – це простий програмний синтаксис для вбудовування серверного коду у веб-сторінку. Синтаксис Razor заснований на платформі ASP.NET, спеціально розробленої для створення веб-додатків. Веб-сторінки Razor можна описати як HTML-сторінки з двома типами вмісту: HTML-вміст і код Razor. Коли сервер читає сторінку, він спочатку запускає код Razor, а потім відправляє HTML-сторінку в браузер. Код, що виконується на сервері, може виконувати завдання, які неможливо виконати в браузері, наприклад, доступ до бази даних сервера. Серверний код може створювати динамічний HTML-контент на льоту, перш ніж він буде відправлений у браузер. Якщо дивитися в браузері, HTML, створений серверним кодом, нічим не відрізняється від статичного HTML-вмісту. Razor підтримує C# і використовує символ @ для переходу від HTML до C#. Коли за символом @ слідує зарезервоване ключове слово, він переходить до розмітки, специфічної для Razor. В іншому випадку він перейде до простого C#. [18] Як приклад використання в проекті, Razor був задіяний для створення таблиці з категоріями на адміністративній сторінці (Рис. 2.6.):

```
<tbody>
    @foreach (var item in Model.CategoryHierarchyList)
    {
      <tr class='@(item.Visible ? "for_search" : "")' data-id="@item.Id">
        <td class="number"><span>@item.Id</span></td>
        <td><span>@item.Name</span></td>
        <td class="colored"><span>@item.NameList</span></td>
        <td class="colored"><span>@item.ParentNameList</span></td>
        <td class="control">
          <span class="glyphicon glyphicon-edit" onclick="OpenEditor(@item.Id)"
title="Edit"></span>
          <span class="glyphicon glyphicon-cog"
onclick="OpenFeatureFamilyEditor(@item.Id)" title="Features"></span>
          <span class="glyphicon glyphicon-trash"
onclick="DeleteCategory(@item.Id)" title="Delete"></span>
        </td>
      </tr>
    }
  </tbody>
```

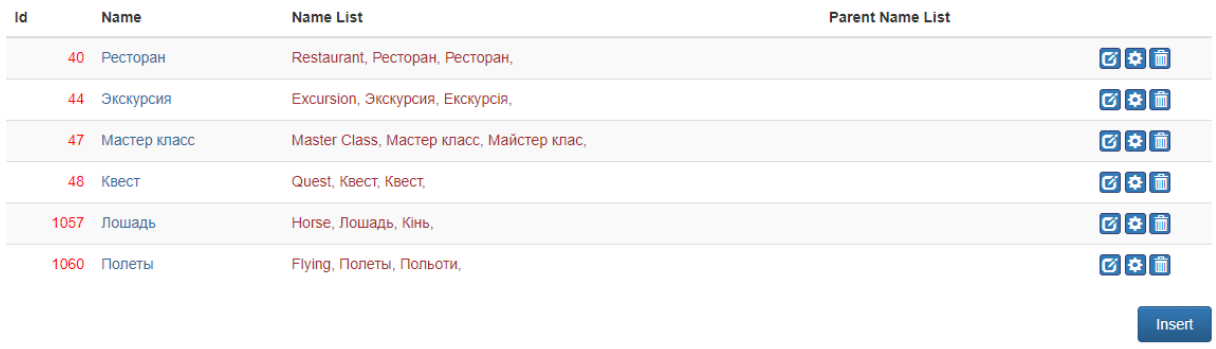

Рис. 2.6. Таблиця побудована за допомогою Razor

2.3.6. AJAX

AJAX (абревіатура від Asynchronous Javascript and XML) – це технологія взаємодії з сервером без перезавантаження сторінки. Ajax працює асинхронно по відношенню до іншої частини коду, і, оскільки, нема потреби кожного разу оновлювати сторінку повністю, швидкість роботи з сайтом значно зростає. В роботі технології можна виділити 4 основні етапи:

1. Користувач викликає AJAX. Зазвичай це реалізується за допомогою якої-небудь кнопки;
2. Система відправляє на сервер запит і всі можливі дані. Наприклад, може знадобитися завантаження конкретних даних з бази;
3. Сервер отримує відповідь від бази даних й відправляє інформацію у браузер;
4. Javascript отримує відповідь, обробляє його і виводить користувачу.

Для обміну даними на сторінці створюється об'єкт XMLHttpRequest, він буде виконувати функцію посередника між браузером і сервером. Запити можуть відправлятися в одному з двох типів - GET і POST. У першому випадку звернення проводиться до документа на сервері, в ролі аргументу йому передається URL сайту. Для запобігання переривання запиту можна скористатися функцією JavaScript Escape. Для великих обсягів даних застосовується функція POST. Серверна частина обробляє дані, що надходять і на їх підставі створює нову інформацію, яка буде відправлена клієнтові. Як відповідь сервер використовує простий текст, XML і JSON. У першому випадку результат можна відразу ж відобразити на сторінці. При отриманні XML-документа його зазвичай конвертують в HTML і виводять на екран. Якщо відповідь отримана в форматі JSON, клієнту слід виконати отриманий код. Після цього буде сформовано об'єкт JavaScript. [19] В проекті за допомогою функції $.ajax() у додатку адміністратора викликається модальне вікно для редагування інформації про категорію (Рис. 2.7.):

```
$.ajax({
        type: "post",
        url: '/Category/OpenEditor',
        data: { id: id },
        dataType: 'html',
        success: function (response) {
            $('.editor').html(response);
        },
        error: function (response) {
            $('.editor').html(response.responseText);
        }
    });
```

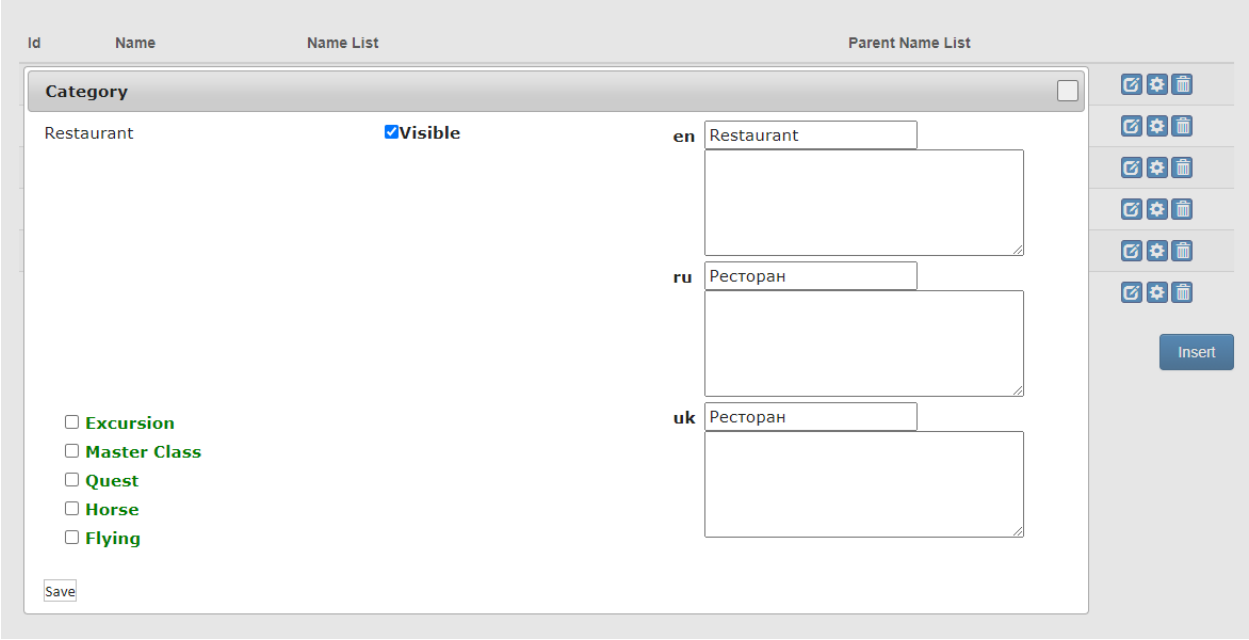

Рис. 2.7. Модальне вікно викликане за допомогою технології AJAX

2.3.7. Regular Expressions

Регулярні вирази – це формальна мова пошуку та здійснення маніпуляцій з підрядком в тексті, заснована використанні метасимволів. Для пошуку використовується так звана маска, тобто рядок із символів, що задає правило пошуку. Для маніпуляції з текстом додатково задається рядок заміни, яка також може містити в собі спеціальні символи. Багато сучасних мов програмування містить вбудовану підтримку регулярних виразів. Центральним класом при роботі з регулярними виразами у C# є клас Regex. Завдяки спеціальному синтаксису регулярних виразів можливо провести такі операції як перевірку на відповідність на початку або в кінці рядка, повтор попереднього або наступного символу певну кількість разів, відповідність алфавітному або цифровому символу, тощо. [20] Клас Regex було використано у проекті для заміни невірних символів з метою автоматичного форматування рядка, наприклад, від час внесення оператором опису позицій:

result = Regex.Replace(result, @"[ ]+", " "); - один чи більше пробілів змінюється на один;

result = Regex.Replace(result, @"[ ]+,", ","); - один чи більше пробілів з комою змінюється на кому;

result = Regex.Replace(result, @"\([ ]+", "("); - якщо після скобки, що відкривається, один чи більше пробілів, то замінюється на скобку;

result = Regex.Replace(result, @"[ ]+\)", ")"); - якщо після скобки, що закривається, один чи більше пробілів, то замінюється на скобку.

Висновок до розділу

В розділі було розглянуто структуру веб-додатку у середовищі Visual Studio та його взаємодію з базою даних, описано функціональну модель кожного з рівнів архітектури та технології з конкретними прикладами, за допомогою яких було реалізовано проект. Середовище розробки Visual Studio було обрано через наявність вбудованих інструментів роботи з хмарою та базою даних, завдяки чому процес створення додатків став набагато простішим. Крім того, в цьому середовищі є наявною система контрол версій, яка допомагає зберігати окремі версії проекту на будь-якому етапі розробки, у зв'язку з чим пошук та виправлення помилок стали набагато простішими. Також завдяки структурі рішення проекту дуже зручно працювати з окремими рівнями додатку, не зачіпаючи при цьому всі інші.

Рівень DAL функціонує за допомогою мови програмування C# та структурованої мови запитів SQL. Таким чином, на цьому рівні відбувається вся комунікація з базою даних. Рівень BAL функціонує за допомогою тієї ж мови C# та вбудованих додаткових фреймворків (Regex). Рівень представлення використовує технології гіпертекстової мови розмітки та її шаблонів, каскадних таблиць стилів, мови логіки скриптів Javascript та її розширень. Як і для фкнкціонування контроллера, логіка рівн презентації регуляється за допомогою синтаксису Razor. Всі основні дії, пов'язані з базою даних виконуються через виклик збережених процедур – завчасно підготованих конструкцій SQL-коду, які викликаються у випадку необхідності.

РОЗДІЛ 3
АНАЛІЗ ПРОВЕДЕНОГО ПРОЕКТУВАННЯ ТА ПІДВЕДЕННЯ ПІДСУМКІВ

3.1. Огляд функціоналу користувацького додатку

Розглянемо докладніше результат розробки веб додатку, а саме його користувацької та адміністративної сторінок. Зовнішній вигляд домашньої користувацької сторінки зображено на Рис. 3.1. На головній сторінці міститься зображення-логотип з назвою веб-додатку («Tour») та категорії для вибору користувачів. Шапка у вигляді синьої лінії містить назву веб-додатку зліва та перемикач мов і кнопку для входу до акаунту справа. Фонове зображення також однакове для всіх сторінок та змінюється відносно пори року, установленої на комп'ютері користувача. На Рис. 3.1. фонове зображення, яке відповідає порі «літо».

Рис. 3.1 Зовнішній вигляд домашньої користувацької сторінки

3.1.1. Перегляд списку позицій

Після переходу користувача до певної категорії на новій сторінці завантажується список позицій справа та їх характеристик для сортування зліва (Рис.3.2.). Внизу списку з позиціями містяться дві кнопки, які завантажують наступний набір позицій, або попередній, якщо користувач переглядає не перші 25 елементів списку. Назва обраної користувачем категорії значиться у самому верху списку. До шапки веб-сторінки додається рядок пошуку з кнопкою та число знайдених пропозицій.

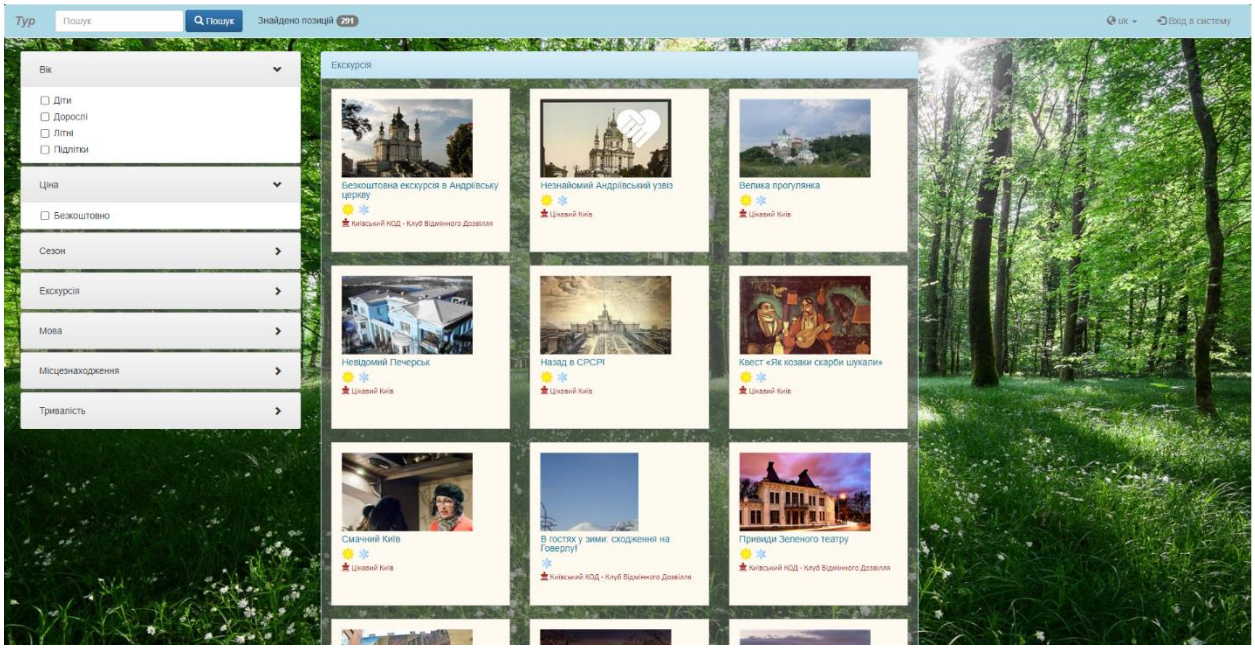

Рис. 3.2. Зовнішній вигляд сторінки з пропозиціями категорії «Екскурсія»

3.1.2. Перегляд характеристик позицій

Позиції різних категорій містять різні характеристики. Наприклад, на Рис. 3.3. зображено характеристики категорії «Майстер-клас». Властивості кожної характеристики відображені списком з можливістю згортування та розгортування. Навпроти кожної властивості стоїть поле-чекбокс для можливості сортування позицій.

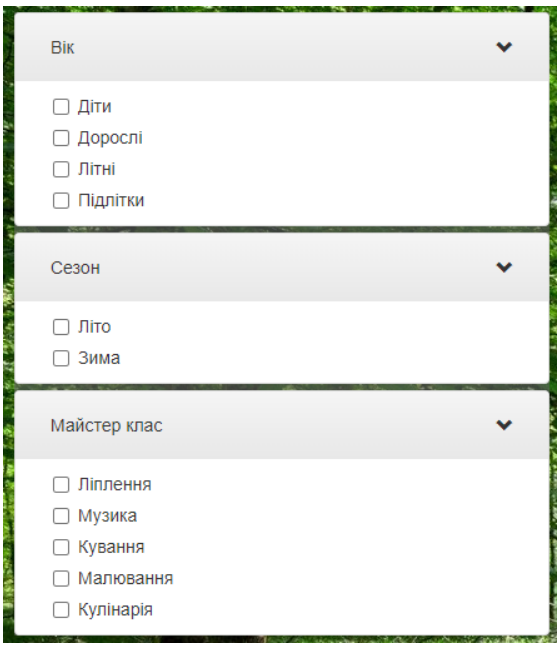

Рис. 3.3. Характеристики позицій в категорії «Майстер-клас»

3.1.3. Фільтрація позицій за характеристиками

Користувач може відфільтровувати позиції використовуючи властивості характеристик. Для цього йому потрібно поставити галочки навпроти потрібних властивостей. Сортування виконується автоматично. Після виконання процедури всі позиції, що не містять потрібних властивостей будуть прибрані з вікна перегляду. Обрані налаштування з’являються у вигляді додаткових фільтрів зверху з хрестиками для прибирання. Кількість відображуваних на сторінці позицій також зміниться. Для відміни фільтрів користувачу слід прибрати галочки навпроти властивостей, або натиснути «Очистити фільтр» для прибирання одразу всіх. На Рис. 3.4. зображено вікно перегляду позицій, що відповідають сортуванню за фільтрами.

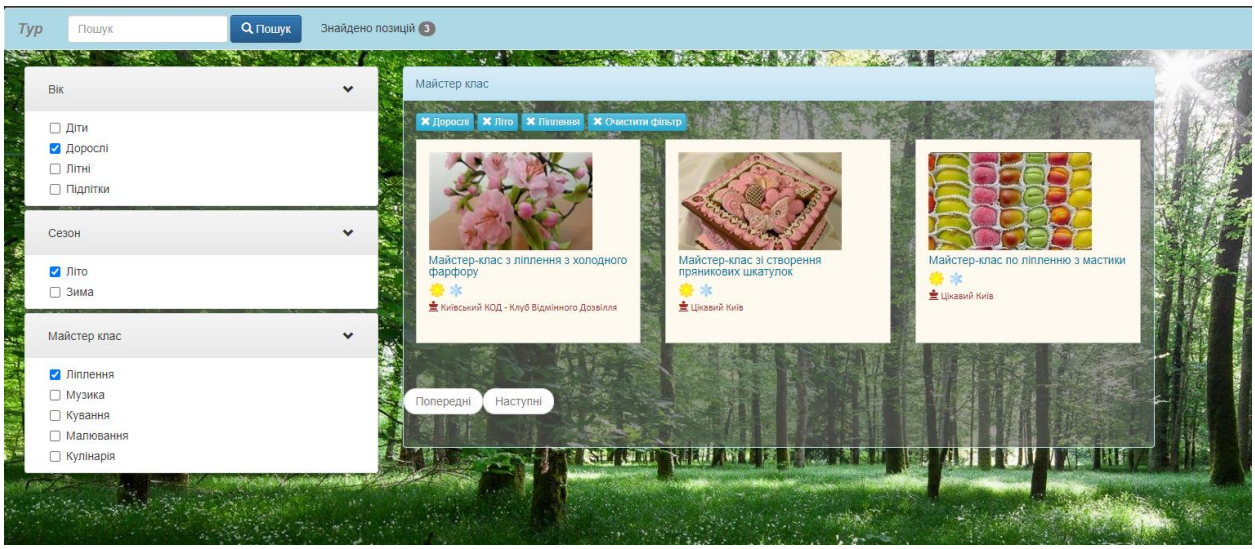

Рис. 3.4. Вікно перегляду позицій із застосованими фільтрами

3.1.4. Пошук позицій за назвами та ключовими словами

Окрім сортування за характеристиками користувачі можуть скористатися пошуком у верхній частині веб-сторінки. Для цього потрібно ввести слово або частину слова тією мовою, якою на даний момент перекладено сторінку та натиснути кнопку «Пошук». Система перевіряє співпадіння символів у пошуковому рядку та назві позиції і видає результат. Окрім пошуку за назвою можна ввести ключове слово, що описує бажання користувача. Якщо таке ключове слово присутнє для конкретної позиції, вона теж підлягає відображенню як знайдений результат. На Рис. 3.5. зображено результат подібного пошуку, в результаті якого було знайдено позиції за назвою та за ключовими словами.

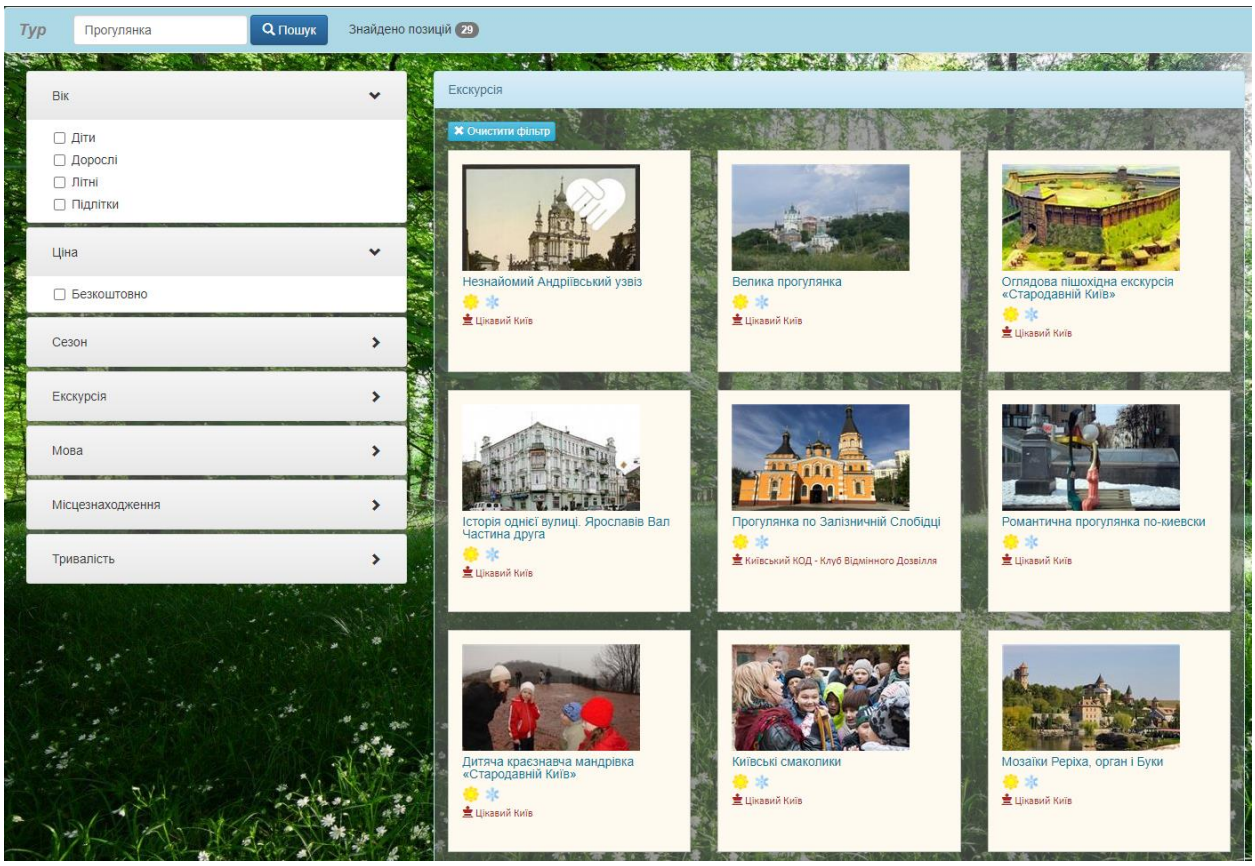

Рис. 3.5. Результат роботи пошукової системи

3.1.5. Перегляд сторінки позиції

Кожна позиція у меню представлена у вигляді квадратної «карточки» із зображенням, назвою, іконкою сезону та посиланням на сайт (Рис. 3.6.). Для отримання більш розгорнутої інформації користувачу треба натиснути на назву, яка його зацікавила, після чого відкриється сторінка зі збільшеними фотографіями та детальним описом (Рис. 3.7.). У вікні справа буде зазначено всі властивості характеристик для обраної позиції. При наявності більш ніж одної фотографії можна буде переглядати їх, натискаючи на стрілки справа та зліва. Внизу буде відображено опис до кожної позиції мовою, яка встановлена на сайті.

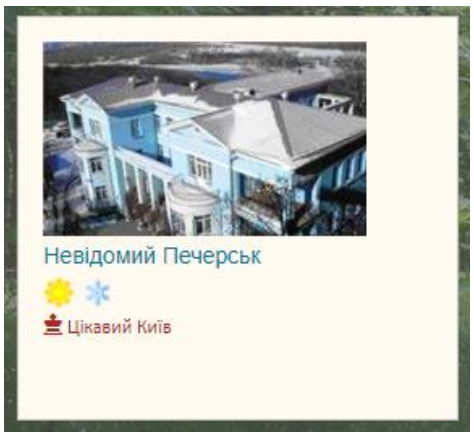

Рис. 3.6. Зображення позиції на сторінці перегляду

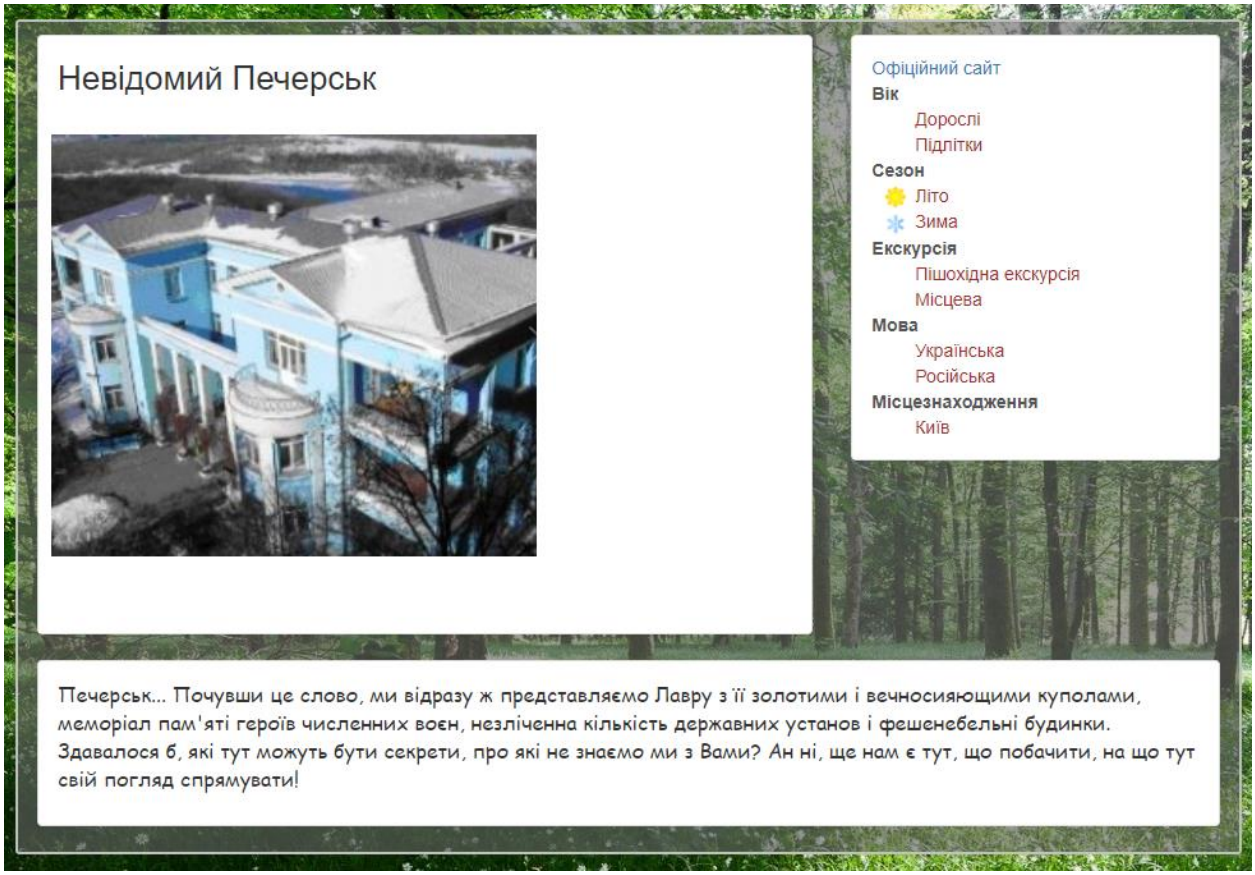

Рис. 3.7. Сторінка окремої позиції

3.1.6. Перемикання мов

Для зручності користувачів у веб-додатку реалізована можливість перемикання мов. Це вибір із трьох наявних мов: української, російської та англійської. Кнопка для перемикання знаходиться у шапці веб-сторінки справа (Рис. 3.8.).

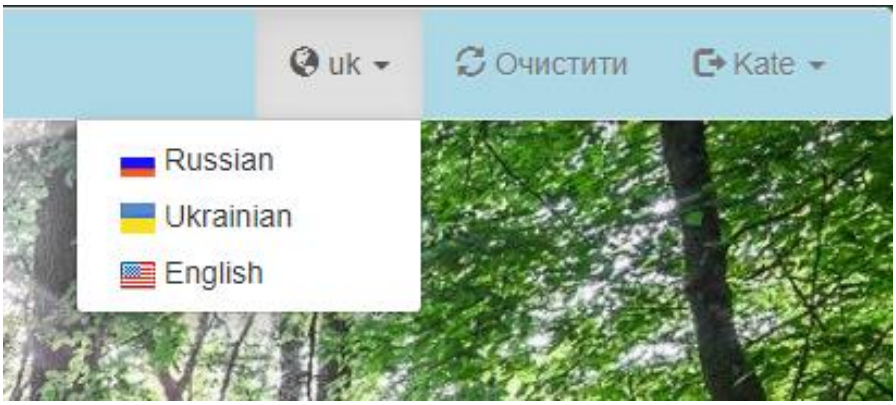

Рис. 3.8. Кнопка перемикання мов

За замовчуванням вся інформація на сайті викладена українською.

3.2. Огляд функціоналу адміністративного додатку

Для того, щоб редагувати інформацію можна було зручніше та швидше, було створено адміністративну сторінку. Ця сторінка прив’язана до тієї ж бази даних та використовує аналогічні методи логіки функціонування, але не призначена для показу звичайним користувачам. Розглянемо детальніше її готовий функціонал.

3.2.1. Вхід та головна сторінка

При спробі зайти на адміністративну сторінку одразу виникає помилка у зв'язку з відсутністю прав доступу (Рис. 3.9.). Зайти можуть лише зареєстровані користувачі, які є в базі додатку та яким при цьому надано відповідні права.

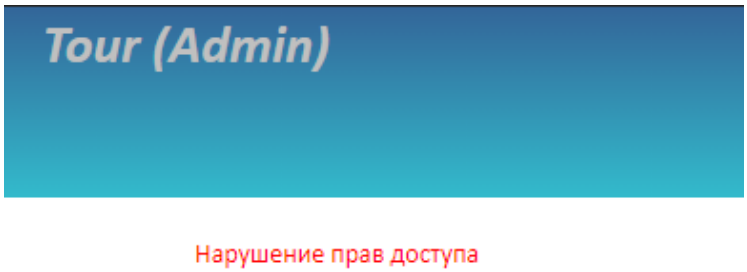

Рис. 3.9. Помилка доступу для незареєстрованих осіб

Для того, щоб адміністратору отримати доступ до сторінки, йому слід ввести свої логін та пароль, які збережені в базі (Рис. 3.10.).

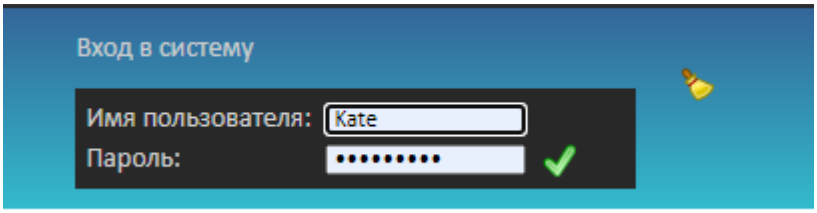

Рис. 3.10. Аутентифікація користувачів адміністративної сторінки

Після введення правильних логіну та паролю система перевірить, який тип прав закріплено за цим користувачем (лише перегляд, чи перегляд та внесення змін). Основною метою створення адміністративного додатку є редагування даних для користувацького веб-додатку. Всі наявні там дані можна розділити на три великі групи: «Категорії», «Властивості» та «Об'єкти». При отриманні доступу до сторінки адміністратора оператор, чи

інша особа потрапить на основну сторінку з однойменними розділами (Рис. 3.11.). Кожен розділ відкриє список наявних даних та інструменти для їх зміни.

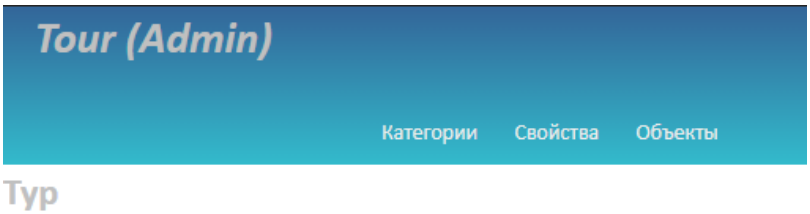

Рис. 3.11. Вигляд основного меню адміністративної сторінки

3.2.2. Редагування позицій

Основною інформацією веб-додатку є дані про розваги (вони ж Позиції або Об’єкти). При переході адміністратором на сторінку «Об’єкти» відкриється список основних позицій (Рис. 3.12.).

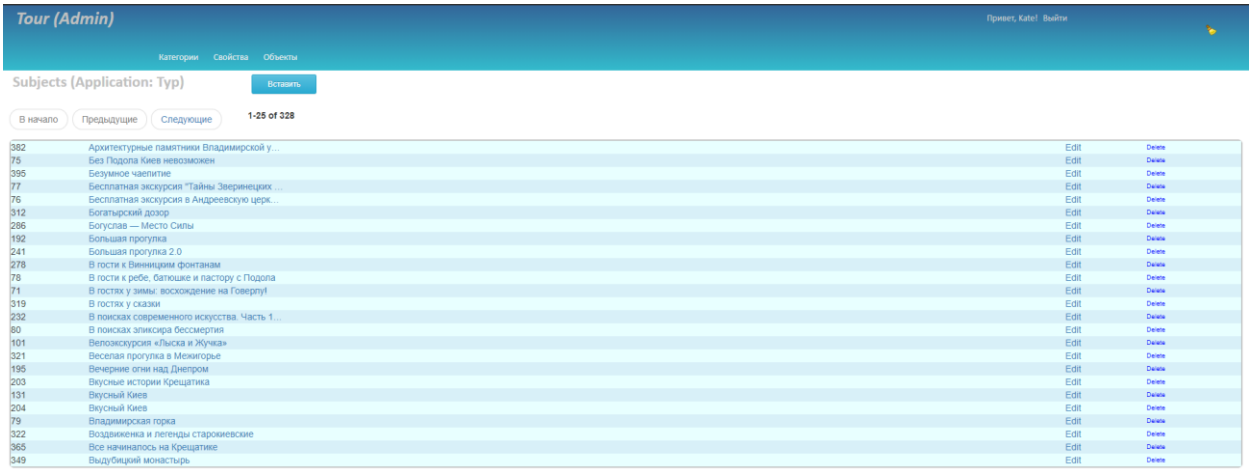

Рис. 3.12. Сторінка зі списком всіх наявних позицій

Для зручності назви позицій відсортовані за алфавітом в кількості по 25 штук на сторінці. Для переходу до наступних використовуються кнопки «Попередні», «Наступні» та «У початок». Зліва від кожної позиції вказано її ідентифікатор в базі, а справа – кнопки «Змінити» та «Видалити». Для того,

щоб додати новий об’єкт слід натиснути зверху кнопку «Вставити», після чого відкриється редактор створення нової позиції (Рис. 3.13.).

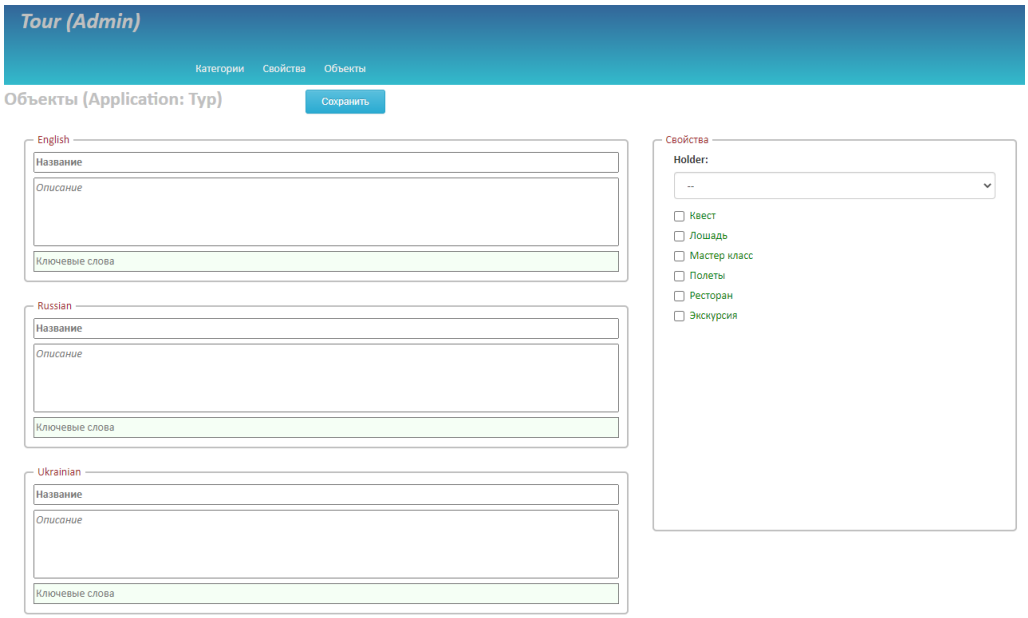

Рис. 3.13. Редактор нової позиції

Щоб створити позицію спочатку слід ввести для ней назву, опис та ключові слова трьома мовами, а також обрати категорію та батьківський об’єкт (опціонально). Після введення всіх даних та натискання кнопки «Зберегти» нова позиція з’явиться у списку. Для подальшого редагування треба вибрати потрібну позицію зі списку всіх наявних на натиснути навпроти неї кнопку «Змінити», після чого відкриється редактор характеристик та зображень (Рис. 3.14.).

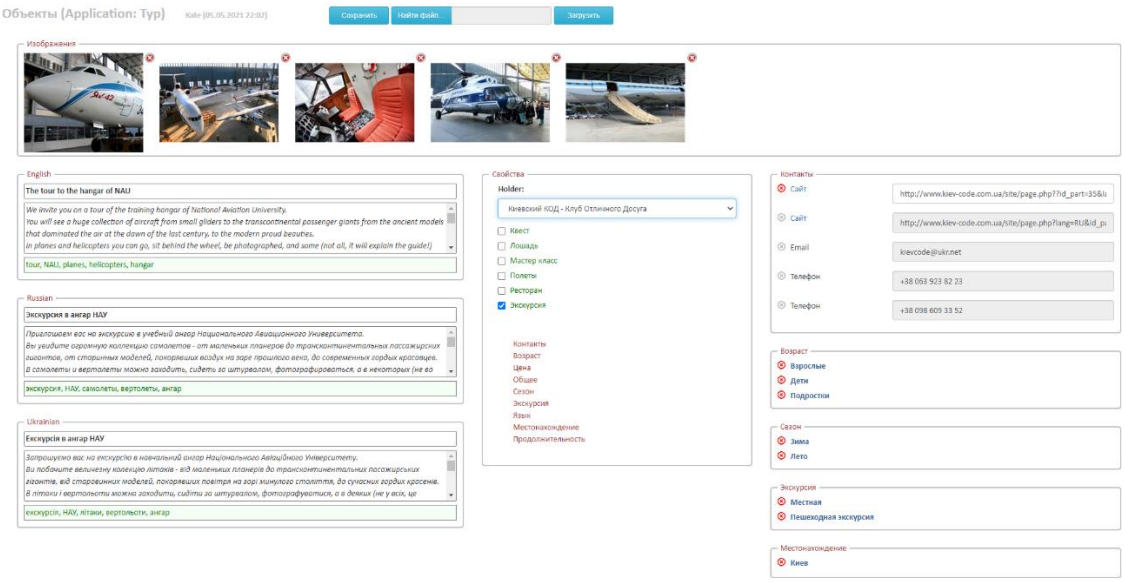

Рис. 3.14. Редактор характеристик позиції

3.2.3. Редагування категорій

Якщо при створенні нової позиції у списку була відсутня потрібна категорія, то її можна додати у розділі «Категорії». При переході до цієї сторінки відкривається список всіх наявних категорій та інструменти для їх редагування (Рис. 3.15.).

Id	Name	Name List	Parent Name List	
40	Ресторан	Restaurant, Ресторан, Ресторан,		
44	Экскурсия	Excursion, Экскурсия, Екскурсія,		
47	Мастер класс	Master Class, Мастер класс, Майстер клас,		
48	Квест	Quest, Квест, Квест,		
1057	Лошадь	Horse, Лошадь, Кінь,		
1060	Полеты	Flying, Полеты, Польоти,		

Insert

Рис. 3.15 Сторінка зі списком всіх наявних категорій

Окрім назв категорій трьома мовами список містить ідентифікатори в базі даних, ім’я батьківської категорії, якщо воно є та три кнопки для керування: для редагування інформації про категорію (Рис. 3.16.), для редагування характеристик для кожної категорії (Рис. 3.17.) та видалення.

Натискання кнопки «Вставити» відкриє порожнє вікно, аналогічно до вікна редагування інформації.

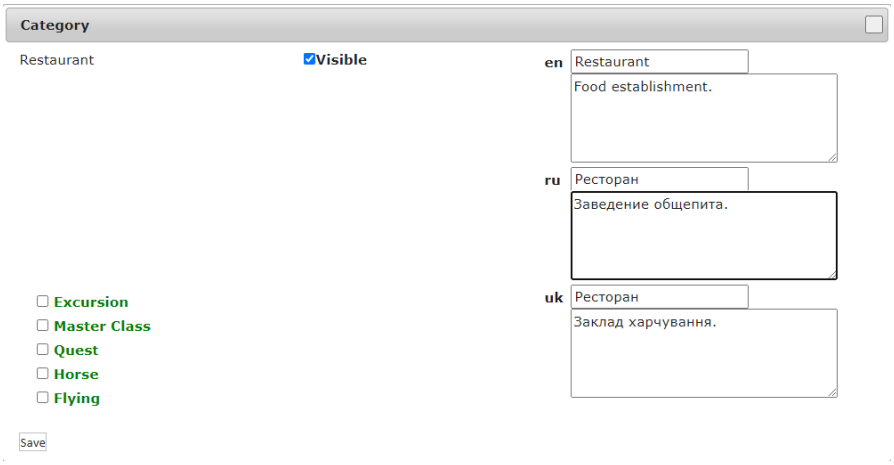

Рис. 3.16. Вікно редагування інформації про категорію

Рис. 3.17. Вікно редагування характеристик для категорії

3.2.4. Редагування характеристик

Останніми даними для редагування залишаються характеристики та їх властивості. На сторінці властивостей знаходяться всі властивості, представлені у вигляді таблиці та відсортовані за характеристиками (Рис.3.18.).

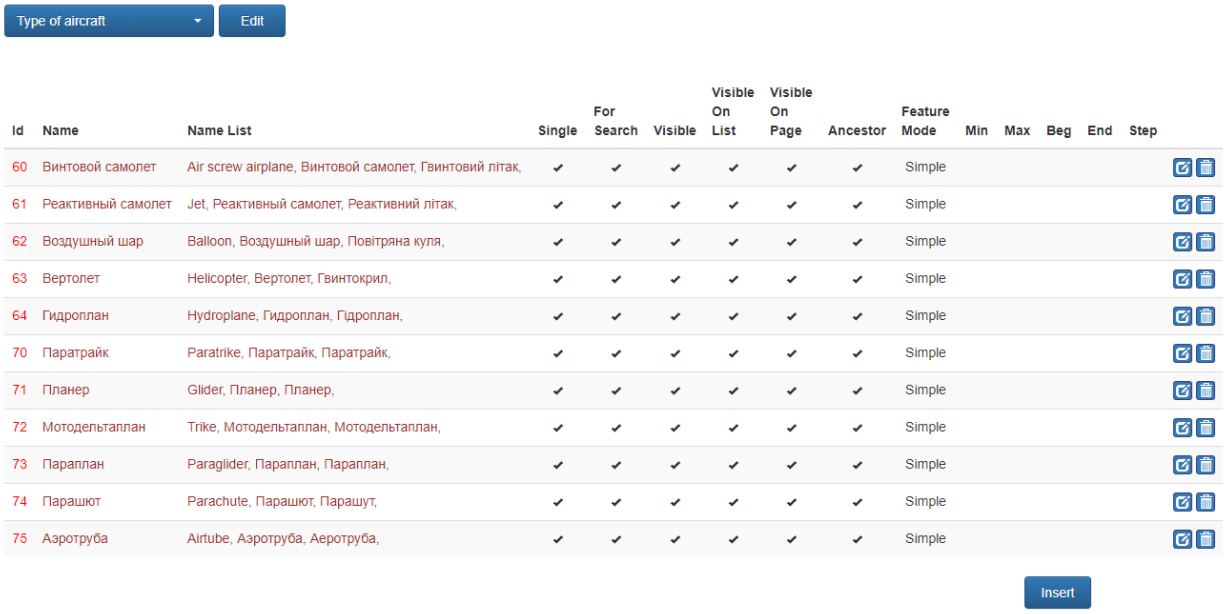

Рис. 3.18. Сторінка зі списком всіх наявних властивостей і характеристик

Для того, щоб обрати властивості якої характеристики відобразити, слід вибрати потрібну характеристику з випадаючого меню зверху (Рис. 3.19.). Всі наявні властивості будуть відображені у вигляді списку. Окрім назв, ідентифікаторів у базі та кнопок керування вони містять додаткові поля, що вказують на видимість на сторінці, видимість для пошуку користувача, батьківську властивість, а також опціональний діапазон значень. Для редагування інформації про властивість слід натиснути кнопку справа, після чого відкриється спеціальне модальне вікно (Рис. 3.20.). При натисканні на кнопку «Вставити» відкриється аналогічне, але порожнє вікно.

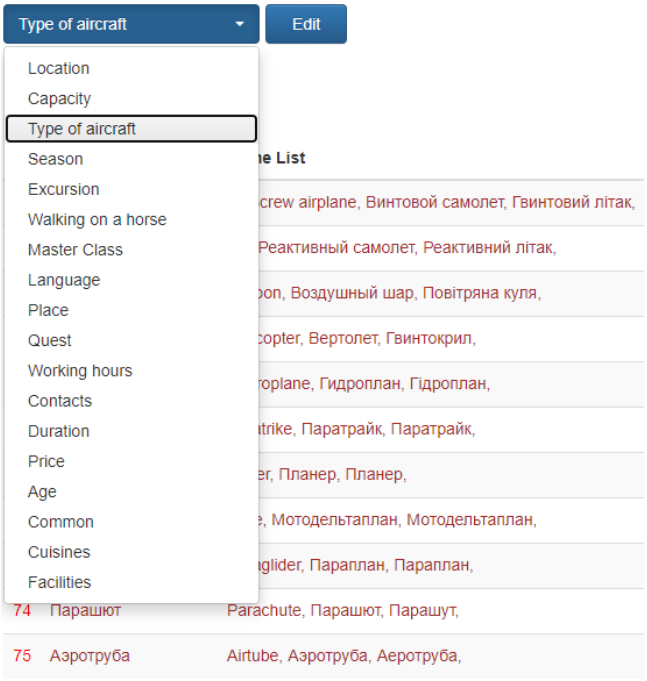

Рис. 3.10. Список всіх характеристик

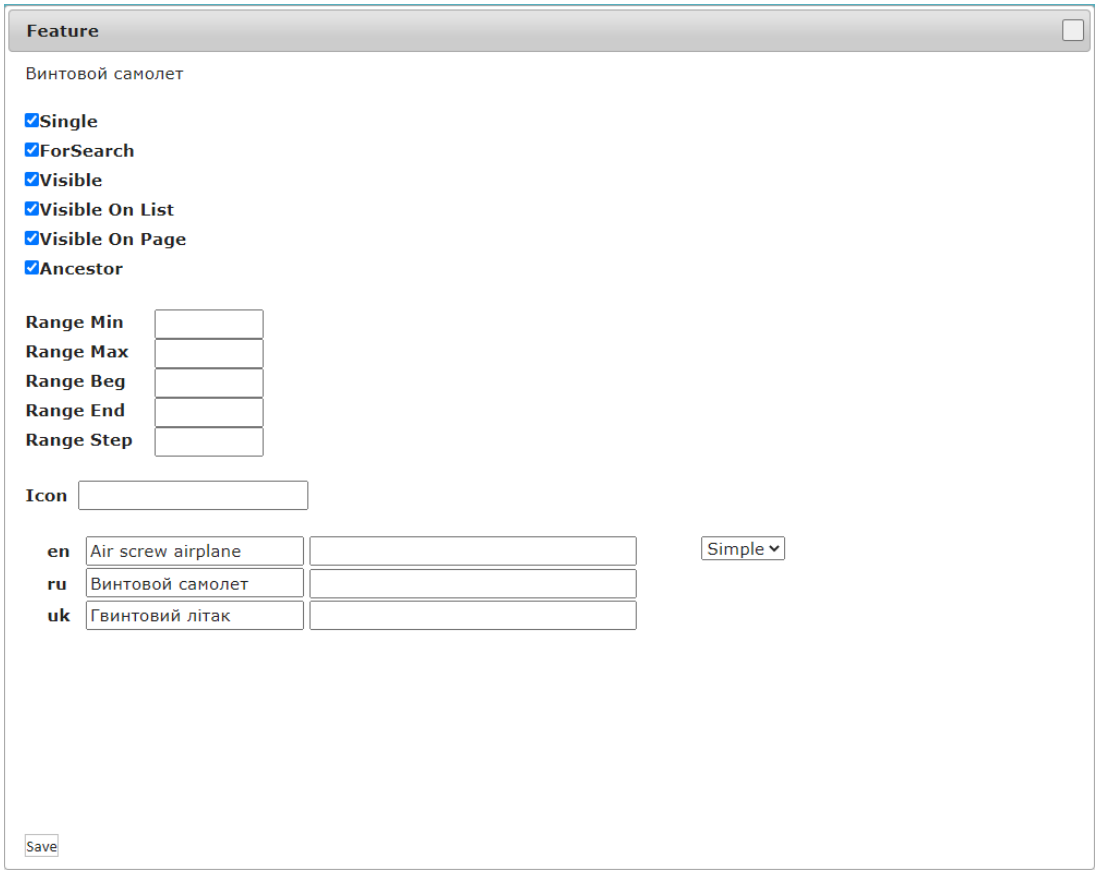

Рис. 3.20. Вікно редагування інформації про властивості характеристик

Висновок до розділу

В розділі описується кінцевий результат розробки дипломної роботи з теми «Хмарний веб-додаток на базі каталогу місць проведення дозвілля», проводиться огляд всього функціоналу та роз’яснюється принцип користування. Окремо розглядається функціонал користувацького веб-додатку та адміністративного.

На відміну від додатку користувача, який призначений для перегляду позицій, вибору категорій та характеристик з метою отримання інформації з бази, додаток адміністратора слугує для того, щоб редагувати інформацію в базі. Через цю відмінність можливості та зовнішній вигляд цих двох додатків відрізняються. Якщо сторінки користувача мають бути максимально привабливими та інтуїтивно зрозумілими для користування, то сторінки адміністраторів не потребують зовнішньої привабливості, а їх інтерфейс має біти максимально зручним та повнофункціональним. Для досягнення зовнішньої привабливості в першому випадку було застосовано багато стильових особливостей (наприклад зміна фонового рисунку в залежності від пори року на комп’ютері користувача), а для більш повного розуміння інформації – перекладач для трьох мов. Крім того, за допомогою легкої навігації, пошуку та сортуванню користувач має змогу отримати максимально наближений до своїх вимог результат.

Додаток адміністратора навпаки, не має зайвого стильового оформлення з метою меншого відволікання майбутніх працівників від роботи з базою, але натомість надає поний контроль над даними, включаючи їх зміну, редагування, видалення та додавання. Завдяки системі перевірки прав доступу зайти на адміністративну сторінку можуть лише перевірені користувачі, а отже дані надійно захищені.

ВИСНОВКИ

В ході виконання дипломної роботи було розроблено веб-додаток на тему «Каталог місць проведення дозвілля» та базу даних, що містить інформацію для нього. Для проектування та зберігання бази даних було обрано хмарне середовище Azure, яке забезпечує надійні сховища та сервери. Використане середовище розробки Visual Studio має вбудовані інструменти роботи з хмарою Azure, що забезпечило якісне підключення та взаємодію БД із веб-додатком, а також зробило можливим реалізацію двох частин додатку з однією й тією ж базою даних. Як результат роботи – робочий веб-додаток з легким, інтуїтивно зрозумілим інтерфейсом для користувачів та надійна та зручна адміністративна сторінка для операторів. Сторінка адміністратора забезпечує повний контроль над усією доступною для показу інформацією шляхом внесення змін у БД. Це означає, що будь-яка людина може дуже швидко навчитися оперувати базою даних.

Створений веб-додаток є корисним та актуальним не лише для звичайних користувачів, а й для власників розважального бізнесу. Показ пропозицій з контактними даними компаній є гарною рекламою, а можливість налаштовувати характеристики об’єктам дозволяє дуже точно знаходити те, що задовольнить потреби всіх споживачів.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Quickstart: Create an Azure Database for MySQL server by using the Azure portal [Electronic resource]. – Access mode: https://docs.microsoft.com/ (lastaccess: 19.05.2021). – Title from the screen.

2. Что такое MVC: рассказываем простыми словами [Електронний ресурс]. – Режим доступу: https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami(датазвернення:19.05.2021). – Назва з екрану.

3. Трехуровневая архитектура приложения [Електронний ресурс]. – Режим доступу: https://it-matika.pro/blog/novye-it-terminy/trehurovnevaya-arhitektura-prilozheniya(датазвернення:19.05.2021). – Назва з екрану.

4. XML serialization [Electronic resource]. – Access mode: https://docs.microsoft.com/en-us/dotnet/standard/serialization/introducing-xml-serialization(lastaccess: 21.05.2021). – Title from the screen.

5. Quickstart: Use Visual Studio to create your first ASP.NET Core web app [Electronic resource]. – Access mode: https://docs.microsoft.com/en-us/visualstudio/ide/quickstart-aspnet-core?view=vs-2019(lastaccess: 21.05.2021). – Title from the screen.

6. Начало работы с ASP.NET Core [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/aspnet5/1.2.php(датазвернення:21.05.2021). – Назва з екрану.

7. Git для начинающих. Часть 1. Что такое системы контроля версий? [Електронний ресурс]. – Режим доступу: https://devpractice.ru/git-for-beginners-part-1-what-is-vcs/(датазвернення:22.05.2021). – Назва з екрану.

8. Data Access Layer [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/mvc5/23.7.php(датазвернення:22.05.2021). – Назва з екрану.

9. Business Logic Layer [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/mvc5/23.8.php(датазвернення:22.05.2021). – Назва з екрану.

10. Presentation Layer [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/mvc5/23.9.php(датазвернення:22.05.2021). – Назва з екрану.

11. Язык программирования C#: краткий обзор [Електронний ресурс]. – Режим доступу: https://techrocks.ru/2019/02/16/c-sharp-programming-language-overview/(датазвернення:24.05.2021). – Назва з екрану.

12. Что такое HTML и почему его должен знать каждый веб-разработчик [Електронний ресурс]. – Режим доступу: https://skillbox.ru/media/code/chto_takoe_html/ (датазвернення:24.05.2021). – Назва з екрану.

13. Что такое CSS: объясняем простыми словами [Електронний ресурс]. – Режим доступу: https://gb.ru/posts/chto-takoe-css-obyasnyaem-prostymi-slovami(датазвернення:24.05.2021). – Назва з екрану.

14. Введение в JavaScript [Електронний ресурс]. – Режим доступу: https://learn.javascript.ru/intro(датазвернення:24.05.2021). – Назва з екрану.

15. jQuery Syntax [Електронний ресурс]. – Режим доступу: https://html5css.ru/jquery/jquery_syntax.php (датазвернення:24.05.2021). – Назва з екрану.

16. Что такое Bootstrap и зачем он нужен? [Електронний ресурс]. – Режим доступу: https://itchief.ru/bootstrap/introduction (датазвернення:25.05.2021). – Назва з екрану.

17. Что такое сетка бутстрап? [Електронний ресурс]. – Режим доступу: https://webformyself.com/chto-takoe-setka-butstrap/(датазвернення:25.05.2021). – Назва з екрану.

18. ASP.NET Razor - C# and VB Code Syntax [Electronic resource]. – Access mode: https://www.w3schools.com/asp/razor_syntax.asp(lastaccess: 25.05.2021). – Title from the screen.

19. Технология AJAX [Електронний ресурс]. – Режим доступу: https://wiki.rookee.ru/ajax/(датазвернення:26.05.2021). – Назва з екрану.

20. Регулярные выражения [Електронний ресурс]. – Режим доступу: https://metanit.com/sharp/tutorial/7.4.php(датазвернення:27.05.2021). – Назва з екрану.

ДОДАТКИ

Додаток А

```
public ActionResult Index(int? featureFamilyId)
                    // перевірка доступу
                    {       if (!UserHasAccess(Module.FeatureKind, Access.Read,
log: true))
                            return View(ErrorPages.AccessErrorView);
                    // Створення екземплярів шару даних
                    var dProject = new Data.Service.ProjectService();
                    var dFeature = new Data.Service.FeatureService();
                    var dFeatureFamily = new
Data.Service.FeatureFamilyService();
                    // Створення моделі даних для сторінки
                    var model = new FeatureKindListModel();
                    // Заповнення моделі даними
                    model.FeatureFamilyList =
dFeatureFamily.GetFeatureFamilyList(forSearchOnly: false);
                    if (model.FeatureFamilyList.Count == 0)
                    {
                            ViewBag.Message = "No one Feature Family item
defined";
                            return View("Error");                   }
```

```
                // Отримання id типу характеристики
                featureFamilyId = featureFamilyId ?? model.FeatureFamilyList.FirstOrDefault().Id;
                // Отримання типу характеристики
                model.FeatureFamily = model.FeatureFamilyList.Where(x => x.Id == featureFamilyId).FirstOrDefault();
                // Отримання списку видів характеристик по знайденому типу
                model.FeatureKindList = new Data.Service.FeatureService().GetFeatureKindDetailedList(featureFamilyId: model.FeatureFamily.Id, languageId: CurrentLanguageId);
                return View(model);
            }
```

```
-- Оновлення категорії

-- Параметри

CREATE PROCEDURE [dbo].[spCategoryManagementUpdateCategory]
	@ApplicationId		INT,
	@Id			INT = NULL,
	@Visible		CHAR(1) = NULL,
	@TextList		XML = NULL,
	@ParentCategoryList	XML = NULL
AS
BEGIN
	BEGIN TRY
	-- Початок транзакції для оновлення даних в категорії
		BEGIN TRANSACTION;
			IF @Id IS NULL
			BEGIN
				INSERT INTO Categories (ApplicationId, Name)
VALUES (@ApplicationId, 'Automatic');
				SET @Id = @@IDENTITY;
			END
			ELSE
BEGIN

UPDATE Categories SET

		ApplicationId = @ApplicationId,
		Visible = @Visible
```

```
WHERE Id = @Id;
                    END;
                IF @TextList IS NOT NULL
                    BEGIN
                    DELETE FROM CategoriesText WHERE CategoryId = @Id;
INSERT INTO CategoriesText (CategoryId, LanguageId, [Name], [Description])
                    SELECT DISTINCT
                    @Id,
                    Tbl.Col.value('@language', 'CHAR(2)'),
                    Tbl.Col.value('@val_name', 'NVARCHAR(100)'),
                    Tbl.Col.value('@val_description', 'NVARCHAR(4000)')
                    FROM @TextList.nodes('/root/rec') AS Tbl(Col);
END;
                IF @ParentCategoryList IS NOT NULL
BEGIN
        DELETE FROM CategoriesHierarchyMap WHERE CategoryId = @Id;
        INSERT INTO CategoriesHierarchyMap (CategoryId, ParentCategoryId)
                    SELECT DISTINCT
                    @Id,
                    Tbl.COL.value('@ParentCategoryId', 'INT')
                    FROM @ParentCategoryList.nodes('/root/rec') AS Tbl(COL);
                    END;
                COMMIT TRANSACTION;    END TRY
```

```
-- Дії у випадку помилки
        BEGIN CATCH
                IF @@TRANCOUNT > 0
                        ROLLBACK TRANSACTION;
                DECLARE @ErrorNumber INT = ERROR_NUMBER();
DECLARE @ErrorLine INT = ERROR_LINE();
                DECLARE @ErrorMessage NVARCHAR(4000) =
ERROR_MESSAGE();
                                DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
                DECLARE @ErrorState INT = ERROR_STATE();
                PRINT 'Actual error number: ' + CAST(@ErrorNumber AS
VARCHAR(10));
                PRINT 'Actual line number: ' + CAST(@ErrorLine AS
VARCHAR(10));
                RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);
        END CATCH
END;
```