МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

«Веб-сервіс для перегляду культурних заходів міста»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Прикладне програмування»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи ПП-41

_____Лабчук В.А._____
(прізвище та ініціали)

Керівник _____Зосімов В.В._____
(прізвище та ініціали)

_____ д.ет.н., доц._____
(науковий ступінь, звання)

Унікальність тексту 95%

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *прикладних інформаційних систем*
Протокол № від р.

зав. кафедри Плескач В.Л.

Київ – 2023

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	14.10.2022	виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	24.10.2022	заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	31.10.2022	виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	01.11.2022	виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	08.11.2022	виконано
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	21.12.2022	виконано
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	31.01.2023	виконано
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	30.03.2023	виконано
9.	Подання роботи у першому варіанті	28.04.2023	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	03.05.2023	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**23.05.2023**	
12.	Врахування зауважень керівника і подання роботив остаточному варіанті (з відповідним висновком про допуск) на кафедру	26.05.2023	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	12.06.2023	
14.	Захист кваліфікаційної роботи бакалавра	26.06.2023	

Здобувач вищої освіти _____
(підпис)

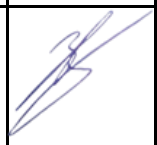
Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Календарний план дипломної роботи	1
Відомість дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	6
2	9
3	23
Висновки	2
Перелік використаних джерел	2
Додатки	10

<table>
<tr><td></td><td></td><td></td><td></td><td colspan="3" rowspan="2">ДП ХХХХ 00.000.00</td></tr>
<tr><td></td><td>ПІБ</td><td>Підп.</td><td>Дата</td></tr>
<tr><td>Розробн.</td><td>Лабчук В.А.</td><td></td><td></td><td rowspan="4">Відомість

дипломної

роботи</td><td>Лист</td><td>Листів</td></tr>
<tr><td>Керівн.</td><td>Зосімов В.В.</td><td>  </td><td></td><td></td><td></td></tr>
<tr><td>Н/контр.</td><td>Макаренко С.А.</td><td>  </td><td>26.05.2023</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Зав.каф.</td><td>Плескач В.Л.</td><td>  </td><td></td></tr>
</table>

АНОТАЦІЯ

Дипломна робота: 54 с., 15 рис., 5 табл., 16 джерел, 4 дод.

Кваліфікаційна робота присвячена проектуванню та розробленню веб-сервісу перегляду культурних заходів міста.

Метою дипломної роботи є ефективний спосіб донесення всіх культурних заходів міста до містян та гостей міста з метою сприяння і розвитку культури.

Для досягнення поставленої мети треба вирішити такі **завдання**:

- дослідити загально-теоретичні засади розробки і впровадження веб-сервісів, ознайомитися із практичною користю створення веб-сервісу для даної тематики;
- здійснити аналіз програмно-технологічних рішень інформаційних систем перегляду культурних заходів міста;
- спроектувати, реалізувати та впровадити веб-сервіс перегляду культурних заходів міста з урахуванням інженерії вимог.

Об’єкт дослідження.

Процеси перегляду культурних заходів міста.

Предмет дослідження.

Програмно-технічні, організаційні засади, принципи, підходи щодо побудови програмної системи перегляду культурних заходів.

Методи дослідження.

Теорія управління для дослідження теоретичних аспектів ведення систем перегляду, емпіричний аналіз і синтез систем, що застосовувався при вивченні прикладів сучасних методів побудови систем перегляду, UML-моделювання, аналогія залучені в процесі проектування, розробки та побудови власної системи перегляду культзаходів, метод порівняння, що застосовано для аналізу наявних ресурсів та програмних систем перегляду.

Ключові слова: веб-сервіс, перегляд, культурні заходи, технологія ASP.NET Core.

ABSTRACT

Thesis: 54 pages, 15 figures, 5 tables, 16 sources, 4 appendices.

This thesis is devoted to the design and development of a web-service software system for viewing of the cultural events in the cities.

The purpose of the thesis is an effective way to inform all citizens and guest of the city about all cultural events in the city for the development of the culture.

To achieve this goal you need to solve the following **tasks**:

- To study the general theoretical foundations of the development and implementation of web-services, to get acquainted with the practical benefit of the creation of this web-service for this topic;
- To carry out the analysis of software and technological decisions of construction of viewing city's cultural events information systems;
- To design, implement, implement a web-service of viewing cultural events in the city, taking into account the engineering requirements.

Object of study.

Viewing processes of cultural events in the cities.

Subject of study.

Software and technical, organizational principles, principles, approaches to building a software system for viewing of cultural events in the city.

Research methods.

Management theory for research of theoretical aspects of viewing systems, empirical analysis and synthesis of systems used in studying examples of modern methods of building viewing systems, UML-modeling, analogy involved in the design, development and construction of own cultural activities viewing system, comparison method which is used to analyze the available resources and software systems of viewing cultural events.

Keywords: web-service, viewing, cultural events, ASP.NET Core technology.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ

ІКТ – Інформаційно-комунікаційні технології;

ORM – object relational mapper (об’єктно-реляційний мапер);

IDE – integrated development environment (інтегроване середовище розробки);

ВС – веб-сервіс;

MVC – model view controller (модель, представлення, контролер);

англ. – англійська;

ШІ – Штучний інтелект;

ІТ – Інформаційні технології;

БД – База даних;

ОС – Операційна система;

ПЗ – Програмне забезпечення;

LTS – Long-time support (довга підтримка, версії ПЗ з подовженим терміном підтримки);

EF – entity framework (технологія платформи .NET для взаємодії з базами даних).

ВСТУП

Культура є невід'ємною частиною нашого життя. Це стало ще краще зрозуміло зараз в умовах війни, адже боротьба іде не тільки за територію, але й за культурні цінності. Крім того відвідування культурних подій ще й може слугувати гарним, а головне корисним відпочинком. Для ефективного функціонування культури і для донесення до громадян всіх запланованих культурних заходів створюється цей сервіс.

Актуальність цієї теми зумовлено тим, що ІКТ та цифрові технології стрімко розвиваються, а країна потребує перемоги не лише на військовому фронті, але й в культурному полі. Проведення таких культурних заходів потрібне і для інших країн, а особливо для України, з метою підвищення рівня культури населення, а дана програмна система покликана фактично інформувати жителів про всі ці культурні події.

Метою кваліфікаційної роботи бакалавра є ефективне сповіщення містян про заплановані культурні заходи міста з метою підвищення рівня культури серед населення.

Завдання дослідження:

- дослідити загально-теоретичні засади розробки і впровадження веб-сервісів, ознайомитися із практичною користю створення веб-сервісу для даної тематики;
- здійснити аналіз програмно-технологічних рішень інформаційних систем перегляду культурних заходів міста;
- спроектувати, реалізувати та впровадити веб-сервіс перегляду культурних заходів міста з урахуванням інженерії вимог;

Об'єктом дослідження кваліфікаційної роботи бакалавра є процеси перегляду культурних заходів міста.

Предметом дослідження кваліфікаційної роботи бакалавра є програмно-технічні, організаційні засади, принципи, підходи щодо побудови веб-сервісу перегляду культурних заходів міста.

Методи дослідження: теорія управління для дослідження теоретичних аспектів ведення систем перегляду, емпіричний аналіз і синтез систем, що застосовувався при вивченні прикладів сучасних методів побудови систем е-комерції, порівняння, UML-моделювання, та аналогія залучені в процесі проектування, розробки та побудови власної системи перегляду, метод порівняння, що застосовано для аналізу наявних ресурсів та програмних систем перегляду культурних заходів.

Практичне значення одержаних результатів полягає у тому, що розроблена прикладна система може стати корисним, зручним та сучасним рішенням для містян, а також гостей міста, які в умовах ведення війни потребують доступу до культури а також змістовного проведення часу та відпочинку. Розроблена програмна система може бути застосована для підняття культурного рівня жителів, корисного проведення часу, а також полегшення емоційного стану містян, що на сьогодні є достатньо важливим.

Структура роботи:

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, розподілених на підрозділи та висновку.

РОЗДІЛ 1 СУЧАСНІ ПІДХОДИ ДО РОЗРОБЛЕННЯ І ВПРОВАДЖЕННЯ ВЕБ-СЕРВІСІВ

1.1. Загальне поняття веб-сервісів

Веб-сервіс – програмна система зі стандартизованими інтерфейсами, ідентифікована веб-адресою url. Такий сервіс може взаємодіяти з іншими сервісами та сторонніми додатками за допомогою повідомлень, заснована на певних протоколах. Ця річ зазвичай використовує розширювану мову розмітки XML (eXtensible Markup Language) для обміну інформацією.

Клієнт має змогу отримати xml-відповідь від сервера, попередньо надіславши йому повідомлення.

На рисунку 1.1 зображено вигляд веб-сервісів.

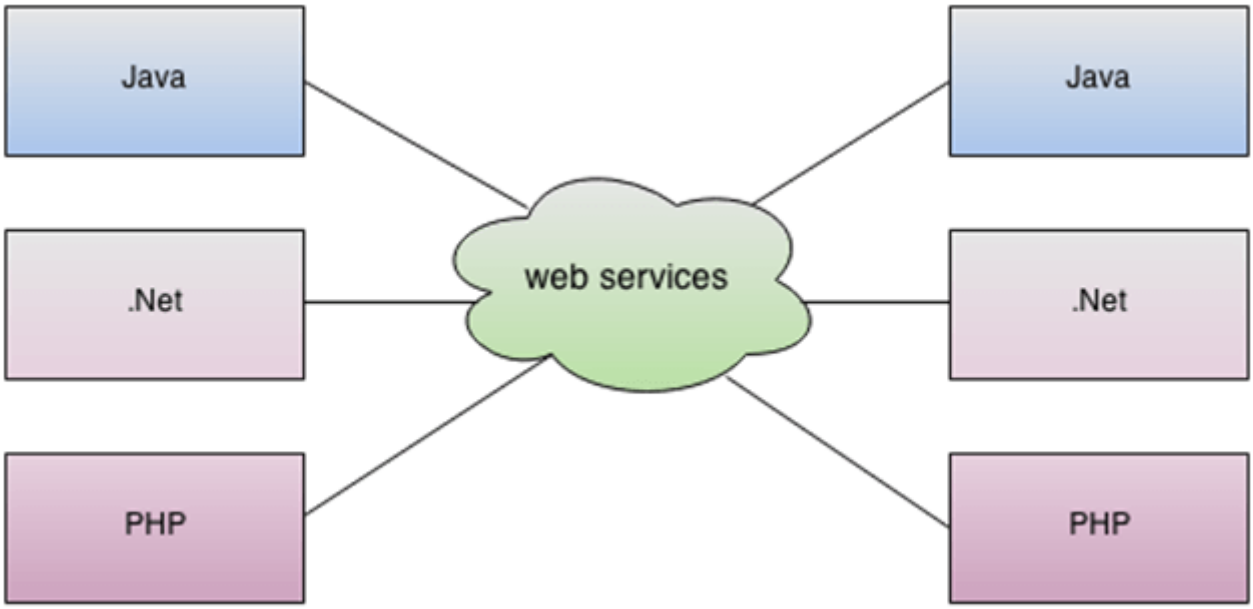


Рисунок 1.1 – вигляд веб-сервісу

Веб-сервіси мають ряд властивостей, зокрема:

- Слабка зв’язаність – клієнт не є жорстко прив’язаним то веб-сервісу напряму, можливі зміни в майбутньому. Це дозволяє інтегруватися між різними системами, до прикладу використовувати веб-сервіс з різних ОС клієнта та різних ПК;

- Крупнозернистість – передбачає використання слабкоспецифічних процесів на відмінну від мов програмування, що дозволяє отримати правильну кількість бізнес-логіки;
- Робота в синхронному та асинхронному режимах – це дозволяє забезпечувати слабку зв’язаність. Клієнт може працювати асинхронно, отримуючи результат будь-коли в подальшому, тоді як клієнти, що не підтримують такого режиму роботи, можуть працювати синхронно, отримуючи відповідь після завершення роботи сервісу;
- Базованість на XML – цей формат використовується на рівнях опису даних та їх транспортування. При цьому виключається прив’язка до мереж, операційних систем або платформ;
- Підтримка виклику віддалених процедур (RPC – remote procedure calls) – це відбувається через надання власних еквівалентних процедур або ж через переклад вхідних викликів у виклики Java або .NET-компонентів. Для цього використовуються протоколи, засновані на XML;
- Підтримка обміну документами – можна працювати не лише з сирими даними, але й з комплексними документами.

1.2. Протоколи та типи веб-сервісів

Для веб-сервісів існує багато комунікаційних протоколів на базі XML-формату, зокрема:

1. BEEP (blocks extensible exchange protocol)
2. WSFL (web services flow language)
3. SOAP (simple object access protocol)
4. UDDI (universal description, discovery and integration)

Веб-сервіси поділяються на 2 великих типи:

- RESTful (REpresentational state transfer)
  Переваги:

А) швидкість – немає надто чітких стандартів. До того ж можливо використовувати не лише формат XML, але й звичайний текст, JSON та HTML;

Б) можливість використання SOAP в якості імплементації;

В) дозвіл використання різних форматів даних - не лише XML, але й звичайний текст, JSON та HTML;

Г) незалежність від платформи та мови програмування.

- SOAP – рекомендація спільноти w3c для взаємодії 2 програм. Є платформонезалежним та мовонезалежним (тобто підходить для різних мов програмування) xml-протоколом.

  Переваги:

  А) ws-безпека – власна безпека для SOAP;

  Б) незалежність від платформи та мови програмування – можна виконувати на будь-якій платформі та писати будь-якою мовою програмування.

  Недоліки:

  А) швидкість, потреба в ресурсах та пропускній здатності – тут використовується лише XML-формат, а його потрібно парсити для читання. При цьому треба чітко слідувати стандартам при роботі з SOAP, а їх немало;

  Б) залежність від WSDL (мови опису веб-сервісів) – інших механізмів для SOAP не передбачено.

Для порівняння SOAP та REST слугує таблиця 1.1:

Таблиця 1.1 – порівняння rest та soap

Відмінність	REST	SOAP
Призначення	Архітектурний стиль	Протокол
Розшифрування	REpresentational state transfer	Simple object access protocol

Продовження таблиці 1.1

Можливість використання суперника	Так, бо це концепт і він може використовувати різні протоколи: http, soap тощо	Ні, бо це протокол
Бажаність	Більш бажаний	Менш бажаний
Формат даних	XML, JSON, HTML, звичайний текст	Лише XML
Безпека	Наслідування безпеки з транспортного рівня	Своя
Необхідність в пропускній здатності та ресурсах	Менша	Більша
Стандартизація	Допускається гнучкість	Необхідно чітко слідувати
Механізм для викриття бізнес-логіки	URI	Сервісні інтерфейси

1.3. Практична користь створення веб-сервісу системи перегляду заходів

На сьогодні до культурних заходів можна віднести достатньо багато категорій, зокрема:

1. Концерти;
2. Стендапи;
3. Театри;
4. Кіно;
5. Заходи для дітей;
6. Виставки;
7. Екскурсії;

8. Майстер-класи.

Виходячи з цього різноманіття, важливість культурних заходів важко переоцінити. Особливо актуальними такі події є сьогодні. Можна зазначити декілька причин цього:

a) Спосіб провести час та відволіктись від буденності;
b) Підвищення рівня культури суспільства, що особливо актуально під час війни на знищення;
c) Спосіб відпочинку з користю.

Зрозуміло, що для ефективного донесення інформації про майбутні культзаходи необхідні інформаційно-технологічні засоби. Актуальність таких засобів збільшується, враховуючи наявність внутрішньо переміщених осіб в країні та розвиток інформаційних технологій в цілому. Це дозволяє не лише ефективніше інформувати про культурні події, але й здійснювати їх аналіз та обробку. До прикладу, в одному місті можуть бути різні культурні заклади: філармонії, театри, музеї, кінотеатри. Веб-сайти у таких закладів можуть бути або відсутні, або окремі, тобто ті, які дають доступ до подій лише в даному конкретному закладі. При цьому неможливо подивитися всі культурні заходи міста в єдиному місці. Це недостатньо зручно, особливо якщо людина слабко ознайомлена з містом, як наприклад внутрішньо переміщена особа.

Веб-сервіси для взаємодії з культурними заходами можуть містити як звичайний перелік таких заходів, так і можливість придбання квитка на такий захід. В подальших розділах будуть розглянуті деякі приклади досить популярних вже існуючих подібних сервісів, серед яких такі українські сервіси як karabas.com, moemisto.ua, concert.ua, intenet-bilet.ua. Деякі з наведених ресурсів представляють більш вичерпний список заходів, і взагалі міст, деякі ж обмежуються досить невеликою кількістю об’єктів. Деякі дані звичайно ж будуть дублюватися, однак наша ціль – об’єднати дані (звичайно ж при об’єднанні буде уникнено дублювання).

РОЗДІЛ 2 АНАЛІЗ ПРОГРАМНИХ РІШЕНЬ ПЕРЕГЛЯДУ КУЛЬТУРНИХ ЗАХОДІВ МІСТ

2.1. Програмна система karabas com

Програмна система karabas com представляє собою веб-сайт, який дозволяє переглядати та купувати квитки на різноманітні культурні події у різних містах.

На стартовій сторінці відображена стрічка подій, можливість вибору типу події, список закладів та можливість вибору міста, це відображено на рисунку 2.1.

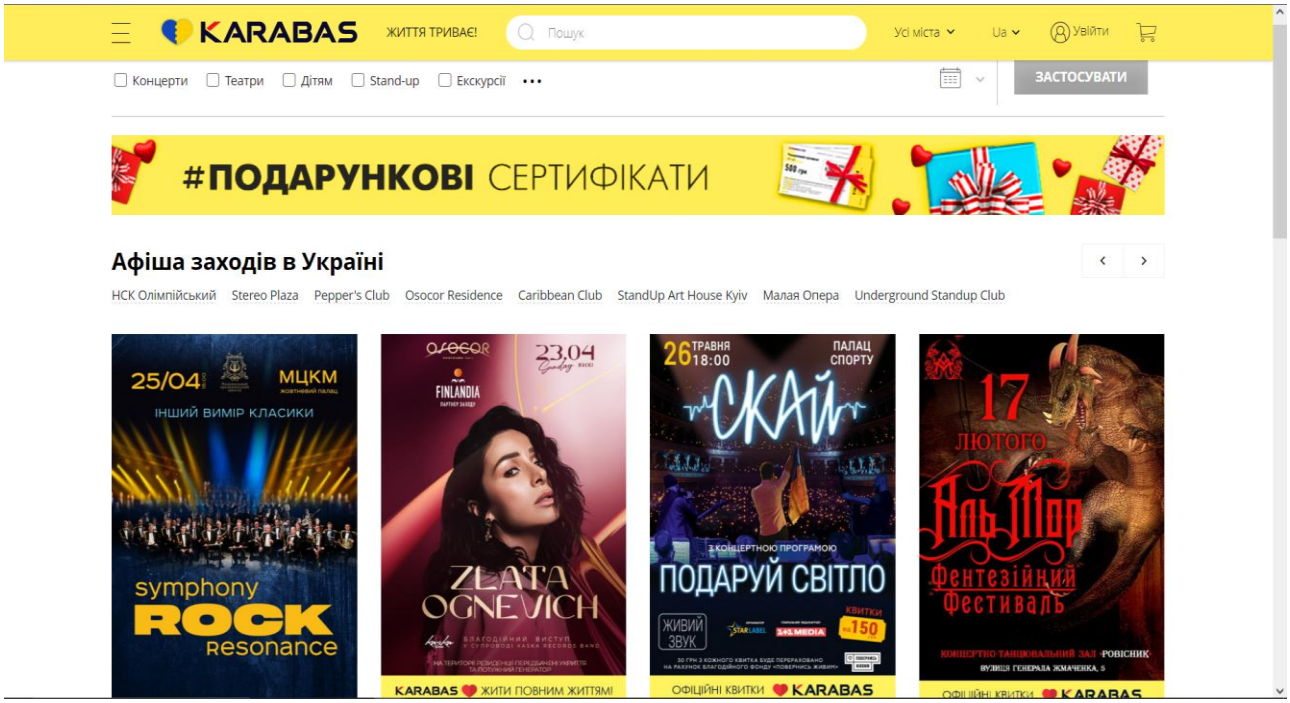


Рисунок 2.1 – стартова сторінка веб-сервісу karabas com

Рисунок 2.2 представляє перелік міст в системі. В системі представлена дійсно велика кількість міст, зокрема і досить невеликі міста (зокрема Мукачево, Стрий, Вараш, Сміла, Шепетівка тощо), що можна побачити далеко не завжди в подібних сервісах, адже деякі з них не працюють навіть в значній частині немалих міст.

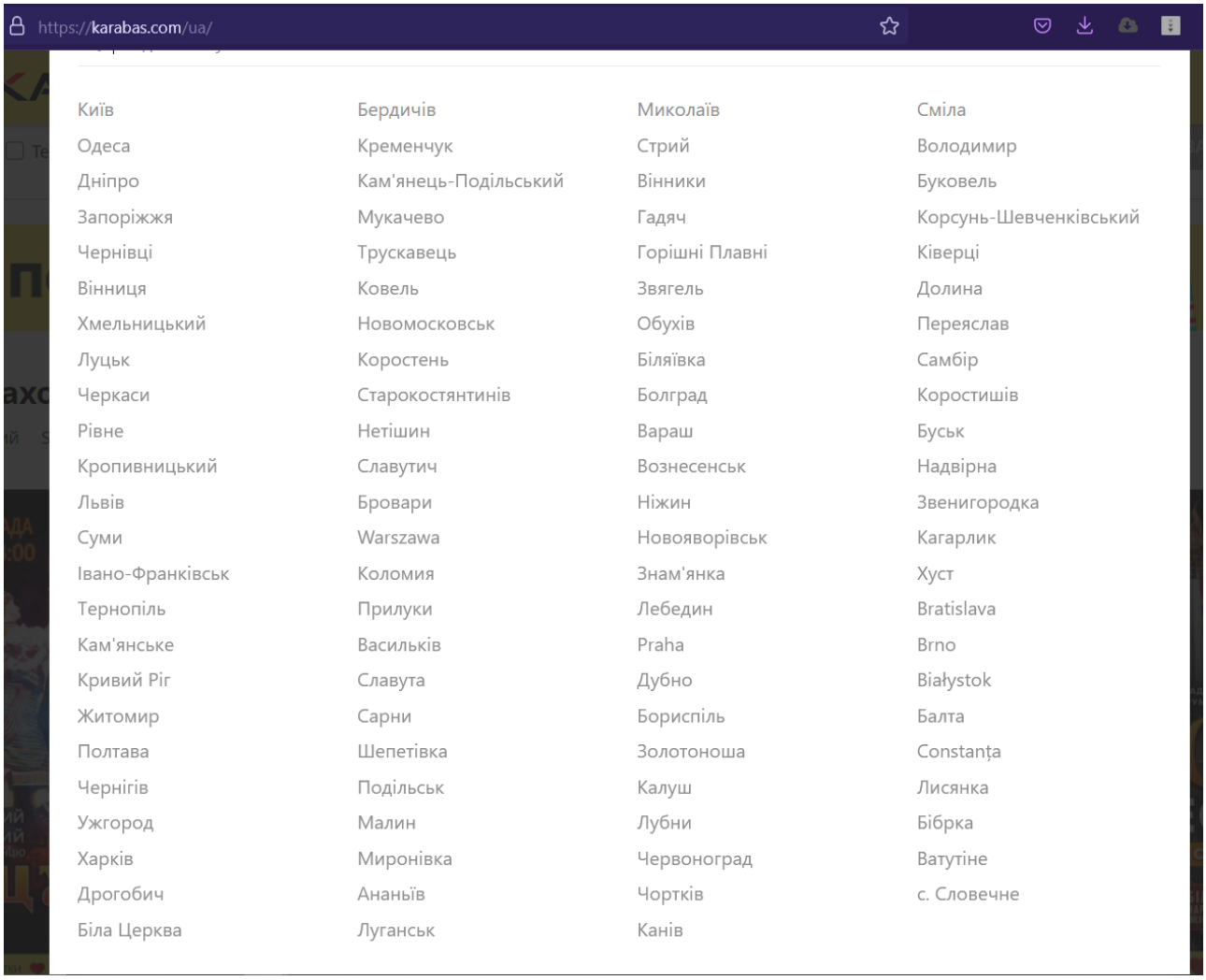


Рисунок 2.2 – можливість вибору міст в сервісі

Як можемо побачити зі списку міст, у переліку присутні й деякі іноземні неукраїнські міста – це означає, що в цих населених пунктах відбуватимуться якісь українські культзаходи, наприклад концерти. Це дозволяє відвідувати українські культурні заходи навіть українцям, які поки знаходяться за кордоном. Назви іноземних міст позначені переважно латинськими літерами, на відмінну від назв українських міст. Серед іноземних міст можемо бачити Брно, Братиславу, Варшаву, Прагу, Констанцу тощо. Тобто в основному це міста у популярних серед українців країн, хоча й не всіх, яких хотілося б. На рисунку 2.3 зображені українські культурні заходи зокрема у Празі.

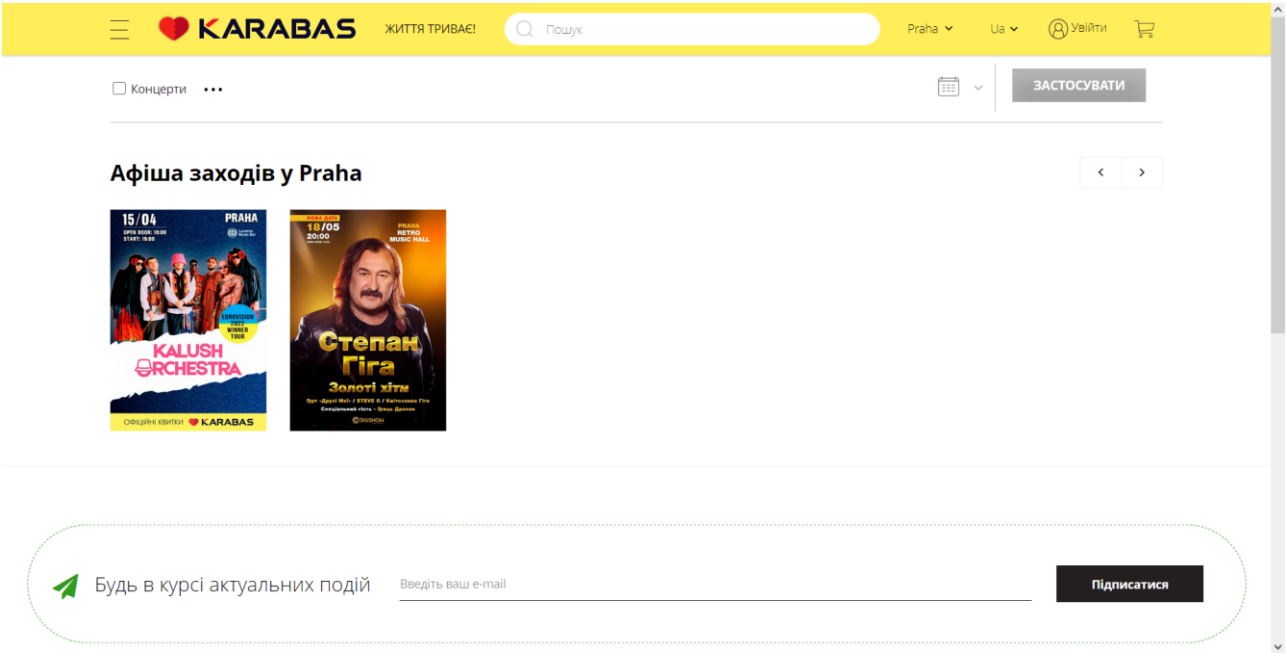


Рисунок 2.3 – афіша українських культзаходів закордоном

При наведенні курсору на якусь подію відображається діапазон цін та можливість купівлі. Це наочно показує рисунок 2.4.

Рисунок 2.4 – відображення додаткової інформації при наведенні на подію

Кожна подія містить опис, а також тур – тобто список міст та місць із вказанням дати та часу, де ще ця подія відбуватиметься, що видно з рисунка 2.5.

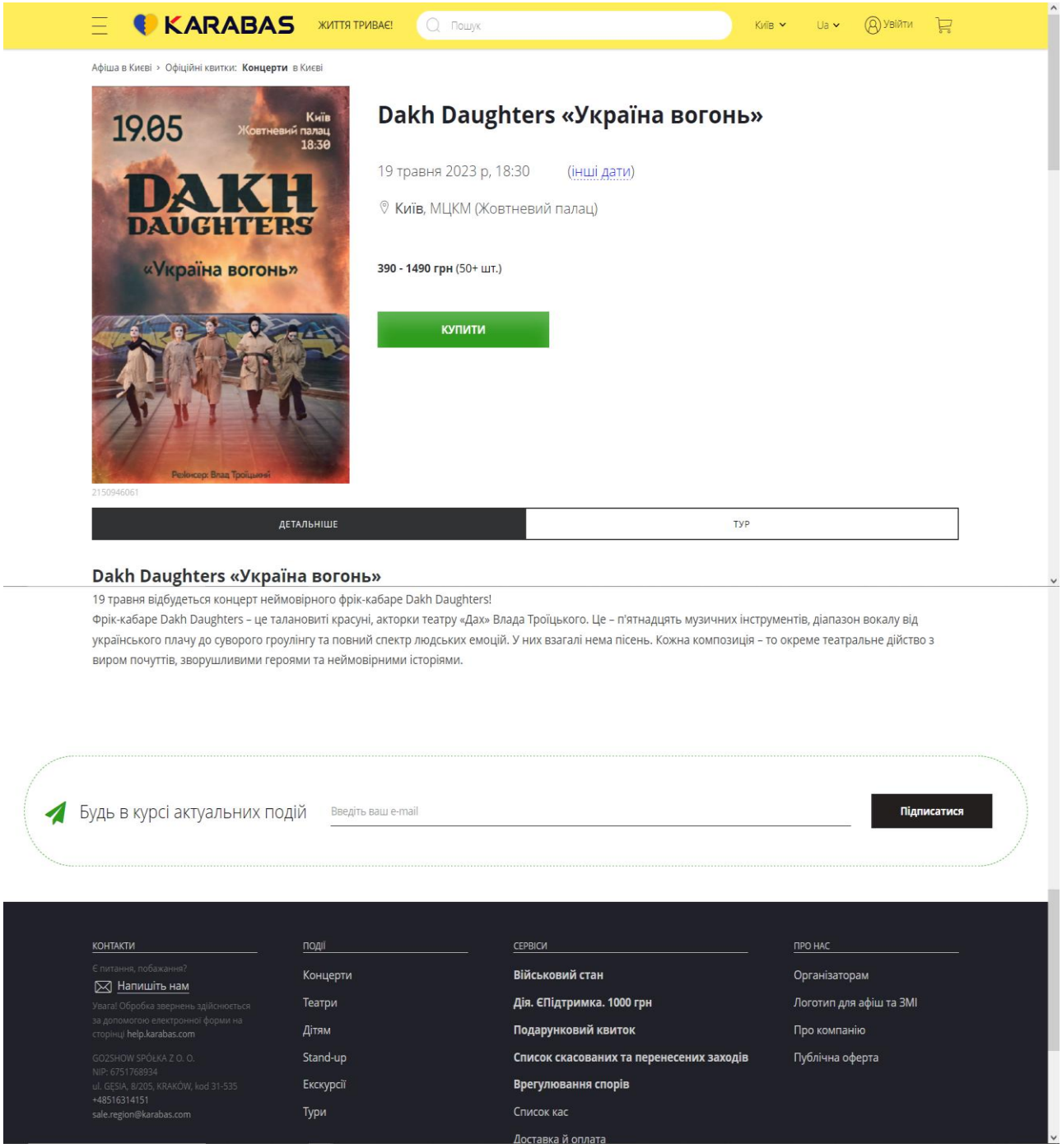


Рисунок 2.5 – сторінка події разом з описом

Під час перегляду заходу можна подивитися де і коли ще він буде проходити. Виходячи з цього, ми можемо реалізовувати таку подію як одну, але з цілим списком міст або ж розглядати це як різні події. Тим паче культурні будівлі і навіть ціни (і звичайно ж кількість вільних квитків на подію) в різних містах можуть дещо відрізнятися і це відображено на рисунку 2.6.

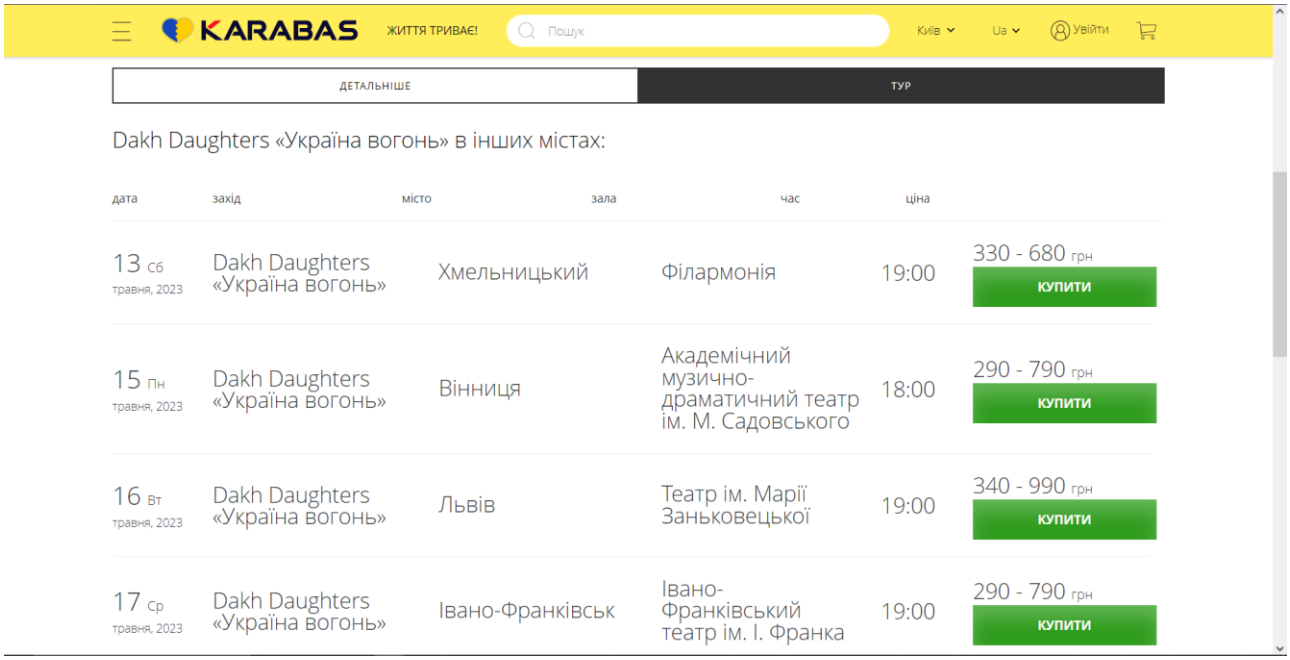


Рисунок 2.6 – тур події

2.2. Програмна система moemisto ua

Дана програмна система пропонує схожий на попередній веб-сайт функціонал. Тут також є різноманітні можливості:

- вибір міста;
- вибір категорії культзаходу;
- перегляд детальної інформації заходу;
- купівля квитка;
- можливість коментувати подію (потрібний вхід через соцмережі: google чи facebook);
- можливість перегляду організаторів та місць проведення, а також інформації про них.

Серед недоліків даного сервісу – обмежений вибір міст у кількості 8 населених пунктів. Адже навіть кількість українських міст невелика, а про якісь іноземні міста, як у випадку з платформою karabas – говорити взагалі не доводиться.

При вході на ресурс, система пропонує обрати місто, вибір досить обмежений. Стартовий екран показаний на рисунку 2.7.

Рисунок 2.7 – початковий екран веб-сервісу перегляду moemisto

Є також можливість вибору рубрики. Зокрема є наступні категорії культурних заходів: кіно, концерти, театр, вистави, спорт, дитяча рубрика, подорожі, відпочинок в іншому місті, розіграші, суспільні події, навчання, вечірки. Екран вибору рубрики представлений на рисунку 2.8.

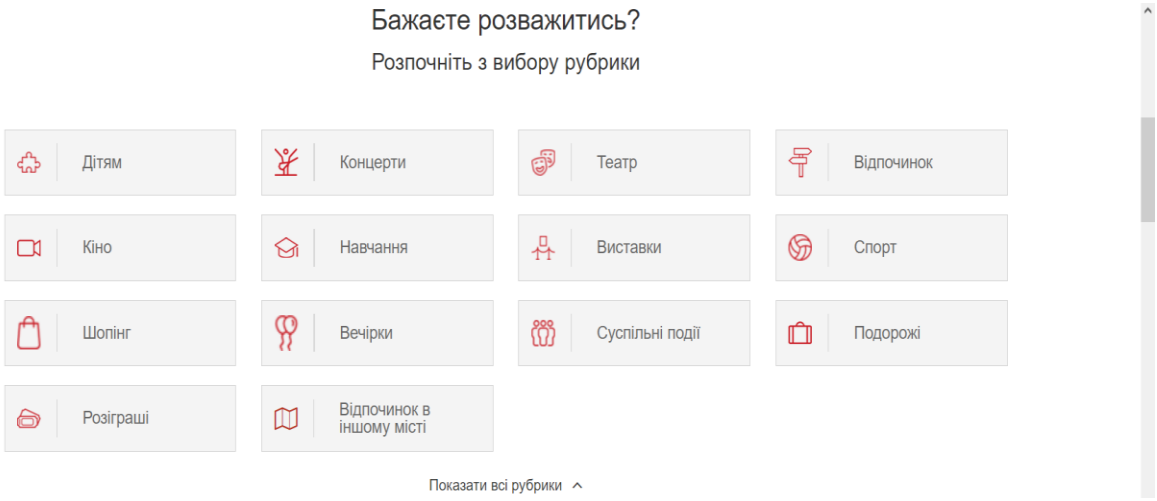


Рисунок 2.8 – вибір рубрики

Сервіс позиціонується як афіша різноманітних подій та заходів у містах для проведення активного способу життя (рисунок 2.9).

Рисунок 2.9 – опис діяльності ресурсу moemisto

При виборі міста на рисунку 2.10 ми отримуємо доступ до заходів різного напрямку в цьому місті

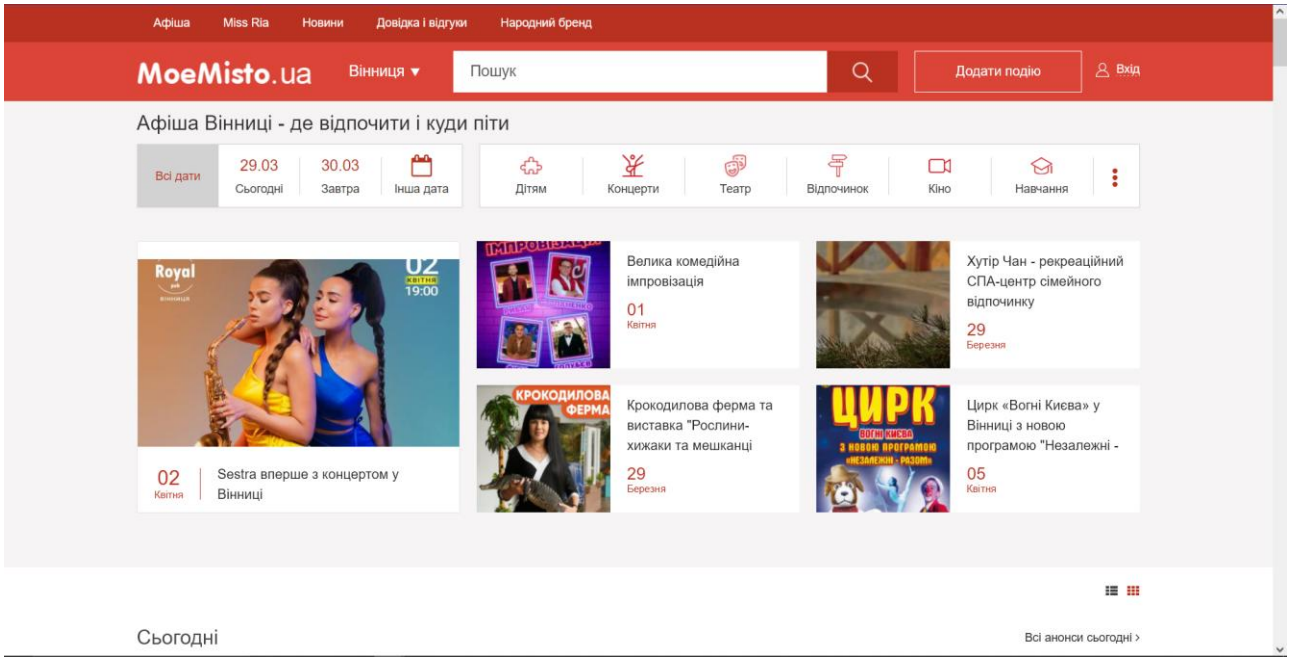


Рисунок 2.10 – афіша конкретного міста

Сервіс пропонує також сортувати обрані події за датою, а також за категоріями. При цьому можна дізнатися організаторів проведення (де відбувається, наприклад, театр, філармонія, тощо), що відображає рисунок 2.11.

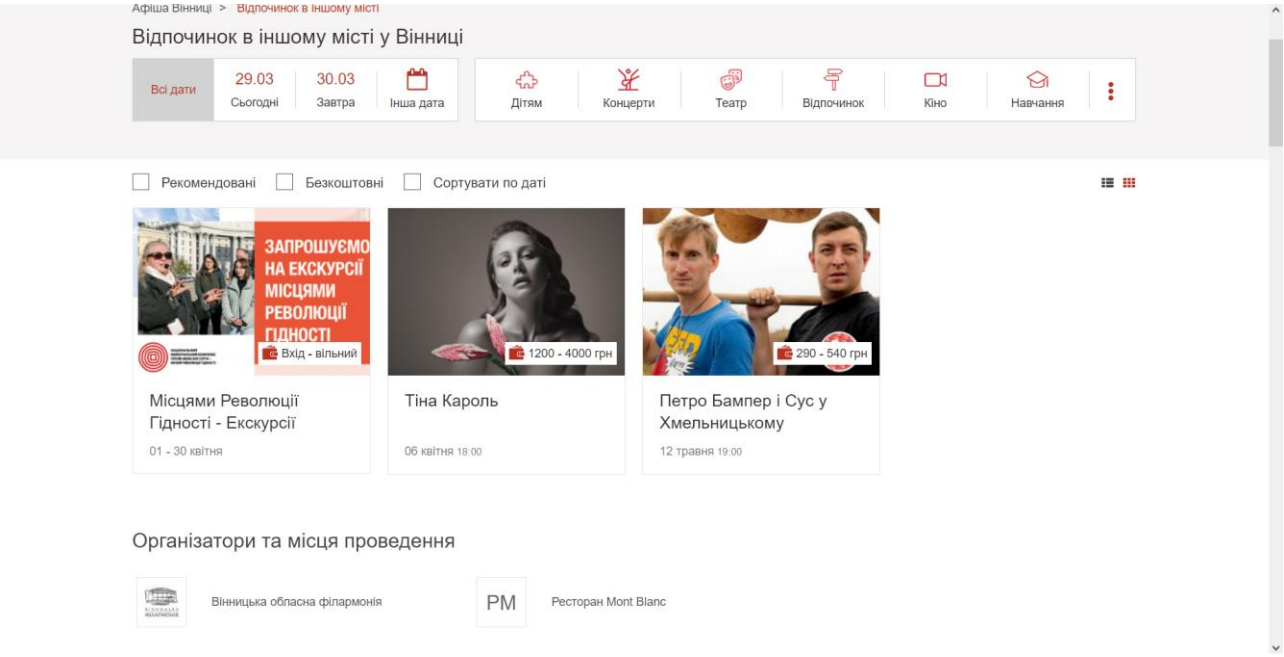


Рисунок 2.11 – вибір категорії заходу у вибраному місті

Якщо натиснути на якогось з організаторів (рисунок 2.12), можна знайти інформацію про нього, зокрема: опис, місцерозташування, контактні телефони, діяльність

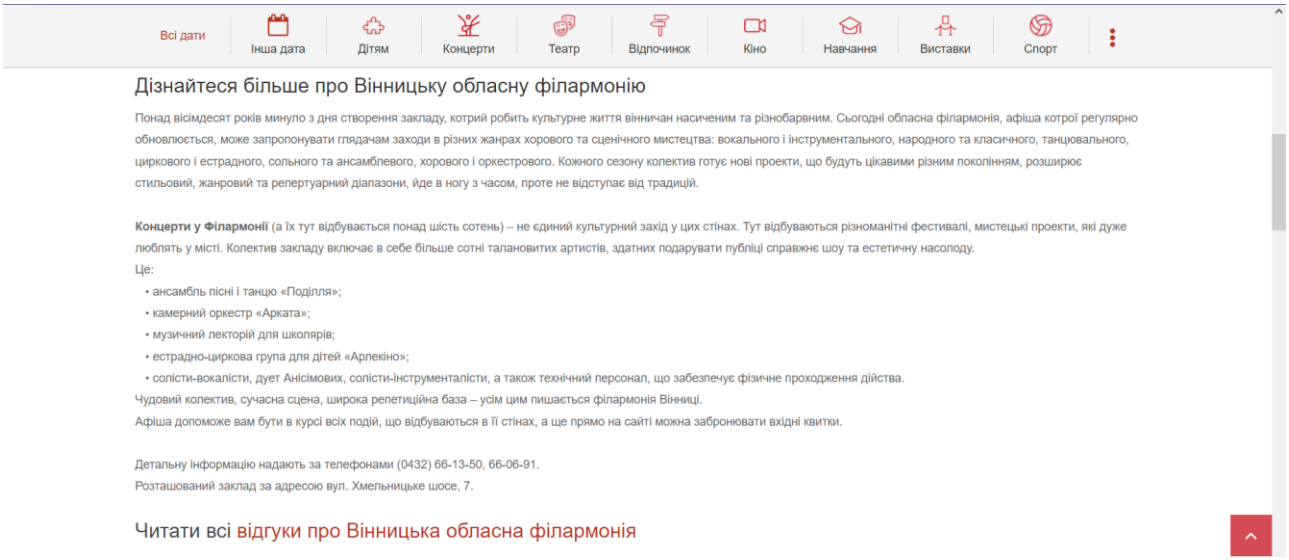


Рисунок 2.12 – Інформація про організатора заходів

РОЗДІЛ 3 ПРОЕКТУВАННЯ, РОЗРОБЛЕННЯ, РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ ПЕРЕГЛЯДУ КУЛЬТУРНИХ ЗАХОДІВ МІСТА

В даному розділі від огляду існуючих рішень перейдемо вже безпосередньо до створення власного веб-сервісу з необхідним йому функціоналом, визначимо який функціонал потрібно забезпечити.

3.1. Інженерія вимог до програмної системи перегляду культурних заходів міста

У даному розділі мова йтиметься про те, який функціонал слід розробити, який функціонал пропонують існуючі веб-сервіси перегляду культурних заходів міст та на скільки їх функціонал відрізняється між собою.

3.1.1. Постановка задачі

Провівши аналіз фундаментальних теоретичних основ проектування подібних систем та виокремивши для себе основні пункти яких варто дотриматися та недоліки яких варто уникати, можемо сформулювати основне завдання цієї роботи: спроектувати та розробити прототип веб-сервісу перегляду культурних заходів міст. Сервіс слід будувати за принципом клієнт-серверної архітектури та з використанням даних, добутих та зібраних від інших подібних сервісів.

Для виконання поставленої задачі необхідно послідовно вирішити наступні завдання:

- Провести аналіз функціоналу, переваг та недоліків схожих ресурсів;
- Провести аналіз формату даних, в якому їх надають сторонні подібні ресурси, зробивши висновок, яким чином можуть бути здобуті дані;
- Розробити компонент доступу до даних (сервер БД) з використанням реляційних баз даних;
- Забезпечити структурованість зібраних даних для зручності їх подальшого аналізу;

- Розробити зручний клієнт-орієнтований інтерфейс взаємодії з користувачами;
- Забезпечити на ресурсі можливість перегляду за містами, за спорудами, за типом культурної події та за найновішими подіями.

При побудові системи дотриматися основних вимог та рекомендацій, наведених в попередніх розділах роботи, щоб забезпечити максимально ефективну та зручну систему перегляду культзаходів через Інтернет.

3.1.2. Аналіз функціоналу схожих систем перегляду культурних заходів

Для розробки актуального та релевантного ресурсу перегляду культзаходів міст необхідно проаналізувати існуючі рішення, адже вони є, а також за можливістю витягнути з них максимально повну кількість даних для занесення у базу даних.

На українському ринку були виявлені 4 сервіси з перегляду та продажу квитків на культурні заходи. Вони відрізняються між собою переліком міст, доступними подіями та категоріями подій, візуальним інтерфейсом користувача. Деякі пропонують і квитки на українські події у іноземних містах Тому гарним варіантом буде об’єднання всіх доступних подій та міст зі всіх цих аналогів у єдину та структуровану базу даних. На сучасному українському ринку найпопулярнішими платформами для перегляду (а деколи і для купівлі) квитків на культзаходи міста є такі ресурси як karabas.com, moemisto.ua, internet-bilet.ua, concert.ua. Для кращого розуміння функціонування цих ресурсів було вирішено провести порівняльну характеристику їх особливостей, наведену у Таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика

Назва ресурсу	Karabas.com	Moemisto.ua	Internet-bilet.ua	Concert.ua
Доступність іноземних	-	-	-	-

виконавців				

Продовження таблиці 3.1

Наявність іноземних міст (в яких відбуваються українські заходи	+	-	-	+
Наявність багатьох категорій заходів	+	+	+	+
Можливість отримання структурованих даних (для запису до своєї бази даних)	-	-	-	-

Порівняння основного функціоналу ресурсів з перегляду та продажу квитків на культурні заходи, наведене в Таблиці 3.1, дає можливість прослідкувати найважливіші характеристики, які варто передбачити, а також особливості роботи з даними ресурсами.

Також у ході порівняльного аналізу було здійснено спостереження про відсутність у функціоналі всіх наведених ресурсів можливості отримання даних у json-форматі. Це змушує добувати дані за технологією веб-скрепінгу, а також ускладнює роботу з даними з цих ресурсів, адже на сторінках знаходиться дуже багато html та js-коду. Це змушує виділення додаткового часу на обробку даних.

3.1.3. Функціональні можливості проектованої системи

Проектована програмна система передбачає реалізацію фундаментального функціоналу, який є затребуваним на платформах зі здійснення перегляду культурних заходів.

Важливо зазначити, що тематика дипломної роботи не передбачає реалізації можливості продажу квитків чи авторизації користувача. Користувач може переглядати актуальні та найповніші дані. За можливістю може бути лише надане посилання на платформу, де можна знайти цей квиток в продажу і придбати.. Для виокремлення функціональних можливостей, які треба було реалізувати на ресурсі було проаналізовано декілька платформ зі схожою тематикою, такі як karabas.com, moemisto.ua, intetnet-bilet.ua, concert.ua. На основі аналізу наявних в них можливостей, а також знайдених недоліків, було сформульовано перелік функцій, які варто передбачити, та наведено у Таблиці 3.2.

Таблиця 3.2 – Функціонал програмної системи

Назва функціоналу	Опис функціоналу
Перегляд всіх міст	Можливість переглядати всі міста, в яких бувають українські культурні заходи, в тому числі іноземні міста
Перегляд всіх подій	Перегляд всіх культурних заходів, незалежно від їх категорії
Перегляд будівель, в яких можуть відбуватися культзаходи	Можливість ознайомлення зі всіма будівлями в місті для проведення культурних заходів, а також перегляду подій, що відбуваються в цих будівлях
Сортування елементів	Сортування переліку міст, подій за зростанням, спаданням тощо
Пошук елементів, фільтрація	Можливість вводити назву елементу (міста, події, будівлі) для швидкого пошуку)

Продовження таблиці 3.2

Отримання даних зі сторонніх існуючих веб-сервісів	Наповнення бази даних даними з подібних ресурсів, але вже структурованими

Веб-сервіс не передбачає поділ на користувацькі ролі, адже це саме веб-сервіс перегляду. Купівля квитків безпосередньо на даному сервісі не здійснюватиметься.

Під час проектування системи були враховані переваги характеристик з аналізу споріднених за характером та тематикою систем з перегляду культзаходів міста, а також виправлені їх недоліки.

3.2. Архітектурні рішення програмної системи перегляду культурних заходів міст

В даному підрозділі мова йтиметься про архітектуру системи, складові цієї системи, та буде здійснено опис кожного компоненту веб-сервісу та його призначення.

3.2.1. Тип архітектури

Завданням дипломної роботи є розроблення системи за принципом клієнт-серверного типу архітектури. Проектована система являє собою приклад сервісу клієнт-серверної архітектури. Особливістю такого типу архітектури є розділення на логіку роботи програми та її візуальне представлення.

Frontend програми побудований з допомогою технології react, він звертається до серверної частини застосунку та бази даних для отримання даних. Ця технологія дозволяє впроваджувати візуалізацію програми поступово та поелементно, що дуже зручно, адже можна побачити результат після кожного невеликого доповнення функціоналу.

Backend програми надає дозвіл конкретному клієнту для доступу до своїх даних і нікому більше, зазвичай для цього вказується url клієнта та

доступні йому методи (лише зчитування, читання й модифікація, видалення тощо) й заголовки. Отримуються ці дані з існуючих веб-сервісів за допомогою добування даних через веб-скрепінг або ж зчитування даних у форматі json зі сторонніх api (однак останніх на задану тематику знайдено не було).

Запис даних до бази даних здійснюється через міграції. Вони дозволяють записувати певні дані вже при створенні бази даних, а також змінювати структуру бази даних в майбутньому без втрати даних, зокрема додавати нові стовпці до таблиць, додавати чи видаляти нові таблиці тощо. Це зручно, адже вимоги до проєкту мають властивість змінюватися. Але при цьому ми маємо можливість не втрачати свої дані, які вже є у базі даних.

Обрана архітектура відображена в діаграмі розгортання. Діаграма розгортання системи представлена на рисунку 3.1.

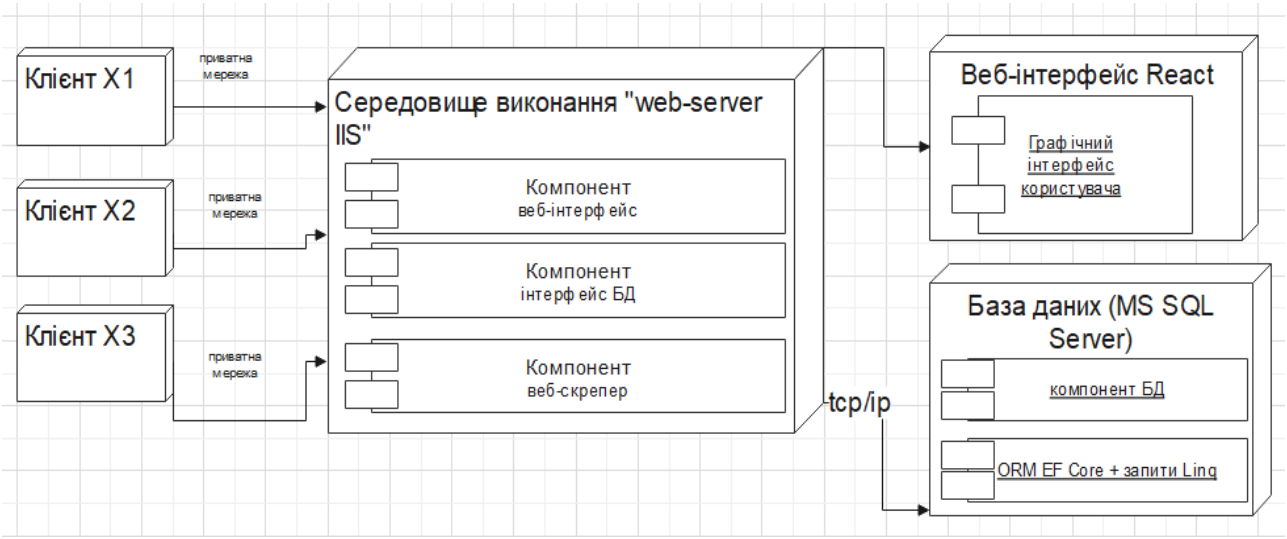


Рисунок 3.1 – Діаграма розгортання системи

Центральним елементом розробленої клієнт-серверної архітектури є веб-сервер, що являє собою високопродуктивну комп'ютерну систему, яка може розміщувати кілька веб-ресурсів.

На сервері встановлений тип програмного засобу веб-серверу Microsoft IIS, що забезпечує доступ до розміщених веб-ресурсів в Інтернеті. Сервер пов'язаний з Інтернетом швидкісним з'єднанням, що забезпечує високу швидкість передачі даних. У свою чергу рівень додатку по протоколах

передачі даних зв’язується із серверною машиною, яка представляє рівень бази даних, посилає до неї запити та отримує відповідні відгуки, які інтерпретуються проміжним рівнем та передаються клієнтові. Рівень даних проектованої системи репрезентований компонентом бази даних Microsoft SQL Server.

3.2.2. Опис структури програми

Структура системи заснована на використанні WebApi, який дозволяє здійснювати основні операції з даними та моделями і дозволяє працювати як сервіс. Такий вид програм може і не містити візуального інтерфейсу користувача, але дозволяти здійснювати певні запити до даних. Такі запити можуть бути здійснені через веб-інтерфейси як-от Swagger, Postman. В сучасних версіях WebApi програм для C# вже за замовчуванням пропонується підтримка Swagger, крім того він може застосовуватися як своєрідна документація програми.

Однак ми не обмежені використанням лише веб-інтерфейсів. До програм WebApi може бути доданий сторонній графічний користувацький інтерфейс, який буде взаємодіяти з бізнес-логікою та даними програми. Це може бути використання будь-якої технології для створення FE програми: react, angular, mvc. Зокрема в даному проєкті було прийняте рішення використання react-у, оскільки він може бути впроваджений до комплексної системи невеличкими частинами, а також дозволяє написання коду у різних стилях, зокрема у функціональному та класовому.

У WebApi-частині системи ми маємо деякі важливі структурні компоненти:

- Моделі – звичайні C# класи, які складаються з опису сутностей та їх властивостей. Зазвичай такі класи в подальшому конвертуватимуться в таблиці у базі даних, а їх властивості у відповідні властивості у базі даних. Використовуючи анотації, ми можемо налаштовувати нашу модель. Зокрема вказувати первинний та зовнішній ключ, вказувати

властивості що не будуть записані до бази даних при потребі, при бажанні змінювати назву властивості чи моделі для запису у базу даних;

- Контролери – базові компоненти, які дозволяють здійснювати певні дії над нашими сутностями. Представляють собою спеціальні класи, які обробляють вхідні запити URI-адреси. Контролер містить методи, які обробляють вхідні запити браузера, а також отримують необхідні дані від моделі та повертають результати. У C# контролери мають містити в своєму імені приставку “Controller” в кінці або ж/і наслідуватися від класу ControllerBase. Для контролерів вказується шлях (uri-адреса), за якими здійснюватиметься робота з ними. Методи контролера можуть бути налаштовані на роботу отримання, відправлення, модифікації (часткової або повної) та видалення даних: HttpGet, post, put, patch, delete);

- Контекст – спеціальний клас, який наслідується від DbContext та містить перелік моделей, що конвертуватимуться в таблиці у базі даних (для цього треба для відповідної моделі призначити DbSet). Тут також можуть вказуватися перевизначені дії, які здійснюватимуться при створенні моделей чи якихось інших діях. Зокрема ми тут можемо вказати, що кожна таблиця бази даних матиме якісь початкові дані, які ми візьмемо з існуючих веб-сервісів культзаходів у містах. Ми також маємо змогу налаштовувати тут наші моделі та зв’язки між ними за допомогою FluentAPI, зокрема вказувати обмеження для стовпців, назву моделі при перенесенні в таблицю бази даних, спосіб генерації для певного стовпця тощо;

- Файл appsettings.json, що містить налаштування програми, зокрема рядка підключення до бази даних.

Дотримуючись принципів наведеної технології, отримуємо структуру системи, наведену у Таблиці 3.3, що представляє основні папки та файли, що містять інформацію додатка.

Таблиця 3.3 – Системні папки та файли програми

Models	Цей каталог містить звичайні C# класи з нашими моделями-сутностями, які представляють таблиці в базі даних. Ці класи містять властивості сутностей, можуть містити і обмеження на ці властивості або їх конфігурацію зі встановленням зв’язків з іншими моделями (але дані дії можна виконувати і в класі контексту за допомогою FluentAPI)
Controllers	Каталог для контролерів. Тут містяться контролери, які дозволяють нам здійснювати запити до даних, зокрема вибірка всіх елементів, вибірка елементу за ідентифікатором, вибірка відсортованих елементів. Контролери створюються для кожної моделі (місто, будівля, захід)
Migrations	Каталог міграцій. Цей каталог створюється автоматично при створенні першої міграції. Міграції нам необхідні для створення бази даних, а також можливості подальшої зміни її структури. При цьому ми маємо можливість зберегти вже записані дані при внесенні змін до структури бази даних
Properties	Містить файл launchsettings.json, в якому можна змінювати налаштування запуску програми, зокрема номер порту, за яким здійснюється запуск
appsetting.json	Файл з налаштуваннями програми. Тут ми вказуємо connectionString – рядок підключення до бази даних, зокрема сервер, назву базу даних, безпеку. Так як це json файл, то це дозволяє проводити нам заміну даних гнучко і без перекомпілювання, на відмінну від варіанту з прописанням параметрів жорстко в C# коді

Продовження таблиці 3.3

Context.cs	Клас, в якому відбувається перетворення моделей-сутностей у таблиці в базі даних. Тут також відбувається налаштування цих таблиць і зв'язків між ними. Налаштування таблиць може здійснюватися і безпосередньо в класі-сутності. Також даний клас може використовуватися для наповнення бази даних (чи якоїсь однієї її таблиці) початковими даними, якщо це потрібно. Нам це потрібно, адже дані для нашого веб-сервісу добуваються з різних існуючих веб-сервісів. Однак дана робота буде здійснюватися не безпосередньо в цьому файлі, а в допоміжному, який тут буде лише викликатися
Program.cs	Основний файл, точка входу в програму. Містить налаштування різних middleware (проміжних компонентів). Зокрема тут встановлюється клас контексту та провайдер баз даних, підключаються контролери, веб-інтерфейс Swagger. Враховуючи необхідність використання FE, доступ до BE йому надається та налаштовується теж тут, зокрема вказується uri кому надаємо, які дії та хедери дозволяємо виконувати. В більш старих версіях C# також додатково окремо був файл Startup.cs для таких дій (Program.cs теж був), однак в новіших версіях він прибраний, а всі початкові налаштування відбуваються виключно у файлі Program.cs

Продовження таблиці 3.3

ModelBuilderExtensions	Допоміжний клас для організації заповнення таблиць бази даних початковими даними. Використовується для того, щоб не засмічувати клас контексту. Саме тут дані за допомогою веб-скрепінгу добуваються з різних існуючих веб-сервісів і записуються для подальшого занесення у базу даних

Таким чином у Таблиці 3.3 був наведений перелік базових та основних структурних елементів проектованої системи та наведене призначення кожного з них.

3.2.3. Архітектура сервера застосувань системи

Архітектура системи є реалізацією технології ASP.NET Core WebApi, яка є новітнім, а найголовніше кросплатформеним випуском платформи веб-розробки ASP.NET від компанії Microsoft. Вона дозволяє реалізувати модель розробки з високою продуктивністю, а також сприяє написанню зрозумілої та структурованої архітектури коду, підтримуючи розробку через тестування та полегшену масштабованість системи. Технологія ASP.NET Core пропонує можливість розробки мовою C#, а також функціональною F# та Visual Basic, на відмінну від старої версії (не Core) пропонує можливість розробки та взаємодії з ОС Linux. Ця платформа динамічно розвивається та постійно відбувається випуск нових версій ASP.NET Core та C#. Для розробки використовується ASP.NET Core 6 спільно з C#10, однак це вже не найсучасніші версії. З іншого боку, дана версія платформи належить до LTS-версій, що надає їй більш довгу підтримку. Спільно з цим були застосовані технології WebApi, Entity Framework Core (Code First).

Проєктована система розділена на компоненти, які взаємодіють з базою даних, які відповідають за загальне налаштування та взаємодію з FE. Ці компоненти є пов’язаними між собою за допомогою засобів ASP.NET Core.

Опрацювання запитів починається від головного модуля системи, що являє собою інтерфейс взаємодії з клієнтом, на якому він бачить всю інформацію. Послідовність на діаграмі відбувається з ліва на право та зверху вниз. Початком послідовності є запит клієнта до системи.

Сутності у програмній реалізації представлені класами на мові C#, що реалізовані за принципом технології WebAPI: контролерами (Controllers) та моделями (Models), а також контекстом. Кожна сутність та процес системи представленні своїм набором пов’язаних об’єктів системи: моделями, контролерами та представленнями. Опис та перелік головних та найважливіших елементів проектованої програмної системи наведений у Таблиці 3.4.

Таблиця 3.4 – Системні елементи

Назва елементу	Тип елементу	Опис
Building.cs	Модель	Опис характеристик культурних будівель
Category.cs	Модель	Опис характеристик категорій подій
City.cs	Модель	Опис характеристик міста
Event.cs	Модель	Опис характеристик, що властиві культурним подіям (місце, час проведення, назва, категорія тощо)
CitiesController.cs	Контролер	Контролер, що дозволяє здійснювати вибірку міста або міст з врахуванням будівель, що в ньому знаходяться

Продовження таблиці 3.4

BuildingsController.cs	Контролер	Контролер, що дозволяє здійснювати вибір будівель з врахуванням подій, що там відбуваються, а також окремої будівлі
EventsController.cs	Контролер	Контролер для перегляду інформації про події в місті
CategoriesController.cs	Контролер	Контролер для роботи з категоріями культурних заходів
launchSettings.json	Файл конфігурації	Слугує для налаштування параметрів запуску програми, зокрема порта, uri-адреси, режиму запуску тощо
Context.cs	Контекст	Використовується для взаємодій з базою даних, налаштування таблиць зі зв’язками між ними, заповнення таблиць початковими даними. Тут під’єднується розширення з вбудованим веб-скрепінгом даних
Program.cs	Основна програма	Точка входу в програму, що містить налаштування запуску і роботи самої програми
Appsettings.json	Файл конфігурації	Використовується для конфігурації рядка підключень до бази даних та інших конфігурацій
ModelBuilderExtensions.cs	Файл розширень	Використовується для веб-скрепінгу даних та заповнення ними власної бази даних

У Таблиці 3.4, що представляє системні елементи, наведений опис та призначення основних елементів програмної системи, що реалізовані у вигляді класів та розміток. Для розробки були використані доступний FE-код існуючих систем для отримання колекції даних.

3.3. Проектування, кодування, реалізація веб-сервісу перегляду культурних заходів міста

В даному підрозділі розберемося з технологіями побудови інтерфейсу системи, зі структурою бази даних, в якій ми зберігатимемо стурктуровані дані з інших подібних веб-сервісів, а також зі сторінок, які бачитиме користувач при використанні веб-сервісу.

3.3.1. Структура візуального представлення системи

Інтерфейс програмної системи відповідає тематиці сайту для ознайомлення з культурними заходами міст. Ресурс містить елементи інтерфейсу для можливості клієнта взаємодіяти з ним. Адміністративних панелей, враховуючи тематику системи, не передбачено. Відсутність адміністративного блоку пояснюється ще й тим, що немає змісту вручну здійснювати або коригувати чи видаляти наповнення контентом веб-сервісу. Адже всі дані беруться за допомогою веб-скрепінгу сторонніх ресурсів та можуть бути автоматично змінені.

Детальна структуризація з урахуванням ієрархії сторінок ресурсу наведена нижче:

Блок «Клієнт»:

- Головна сторінка
  - Відображення списку міст з врахуванням культурних закладів у кожному з них;
  - Пошуковий рядок з можливістю пошуку міста або культурного закладу за назвою;

- Можливість сортування міст за назвою чи за замовчуванням у прямому й зворотному порядку.

- Сторінка культурної будівлі
  - Можливість перегляду додаткової інформації про будівлю та її опису;
  - Список доступних культурних заходів, які буде проведено в даній споруді.
- Сторінка списку культурних заходів
  - Відображення загального списку культурних заходів;
  - Можливість пошуку та фільтрації інформації;
  - Можливість сортування культурних заходів в прямому та зворотньому порядку.
- Сторінка обраної культурної події
  - Загальний опис та більш повна інформація про обрану культурну подію (місце проведення, опис суті події, кількість квитків що залишилася, ціна квитка, час та дата проведення заходу).

Для демонстрації функціоналу розробленої системи були створені сторінки, які надають інформацію про події з категоріями, а також про міста з вказанням списку доступних будівель для проведення культурних заходів у кожному з міст.

Сторінки передбачають можливість фільтрувати, сортувати та здійснювати пошук інформації. Це спрощує взаємодію користувача з ресурсом, а також дозволяє пришвидшувати пошук необхідної інформації користувачем, що позитивно впливає на враження від користування ресурсом.

Сторінки веб-ресурсу передбачають використання певної кількості стилів для реалізації більш красивого й логічного вигляду елементів інтерфейсу, їх розташування та взаєморозташування.

3.3.2. Технології розробки інтерфейсу системи

Розроблення користувацього інтерфейсу системи відбувалася з використанням технології React. Ця технологія дозволяє впроваджувати користувацький інтерфейс невеличкими частинами, що зручно, в тому числі коли відбувається перехід з якоїсь іншої технології FE (наприклад, MVC) на react.

React побудований на базі js. Він пропонує 2 стилі написання програм – класовий та функціональний. Функціональний є новішим та більш простим в освоєнні, саме тому було вибрано його.

Зазвичай при розробці цією технологією надається стандартна базова uri-адреса зі стандартним номером порту. Для react це “localhost:3000”. Саме для цієї адреси необхідно надати дозвіл у BE на доступ до даних з використанням технології CORS.

Крім того, враховуючи специфіку системи та існуючих аналогів, а саме наявність існуючих даних, було здійснено роботу з FE кодом існуючих систем. FE код аналогів є досить об’ємним, тож його було проаналізовано та досліджено для подальшого використання його даних у BE. Зокрема було досліджено використовувані html-теги, css-селектори та класи html-елементів. Саме за допомогою цих елементів є можливість витягнути потрібні дані.

3.3.3. Структура бази даних

Для реалізації роботи з даними був обраний стек технологій на основі використання реляційного типу баз даних, який був описаний у розділі 2 цієї роботи. У базі даних роботи, відповідно до реляційних принципів роботи з даними, заздалегідь визначені зв’язки між її елементами. Елементи упорядковані у вигляді таблиць зі стовпцями та рядками. Таблиці є

уособленням об’єктів, які використовуються у роботі системи. Стовпці таблиці містять певний тип даних, а поле зберігає фактичне значення атрибута. Рядки в таблиці представляють набір пов'язаних значень одного об'єкта або сутності. Кожен рядок таблиці позначений унікальним ідентифікатором (PK – primary key), який називається первинним ключем, а рядки між кількома таблицями пов'язані за допомогою зовнішніх ключів (FK – foreign key). Доступ до цих даних здійснюється запитами на мові SQL (Structured Query Language – Структурована мова запитів) без реорганізації самих таблиць бази даних.

Для зберігання даних таблиць можна використати програмний додаток Microsoft SQL Server - SQL Server Management Studio – студія управління доступом до баз даних. Студія управління SQL Server (SSMS) – це середовище розробки, яке забезпечує графічний інтерфейс для підключення та роботи з сервером MS SQL. Він був випущений з Microsoft SQL Server 2005 і використовується для налаштування, управління та адміністрування всіх компонентів у Microsoft SQL Server. Однак такий варіант не є обов’язковим, адже працювати з базою даних можна і з середовища розробки Visual Studio, яке фактично використовувалося для розробки системи – з лівого боку міститься панель “SQL server object explorer”, яка дозволяє взаємодіяти з базою даних

Робота з даними додатка заснована на застосуванні технології ORM (object relational mapping), а саме Entity Framework Core. Технологія Entity Framework – це об’єктно-реляційна проекція (ORM), яка дозволяє розробникам .NET працювати з базою даних за допомогою об’єктів .NET. Це усуває необхідність у великій кількості коду для доступу до даних. Тобто робота з базою даних відбувається у більш об’єктно-орінтованому стилі, що звично для C# розробників. Версія Core цієї ORM передбачає кросплатформенність та може застосовуватися у ASP.NET Core, який передбачає можливість виконання не лише на Windows, але й у Linux та MacOs.

Рівень Entity Framework розміщується між рівнем класів доменів та базою даних. Він зберігає дані, що зберігаються у властивостях об’єктів, а також отримує дані з бази даних і автоматично перетворює їх на об’єкти. Технологія дозволяє використовувати LINQ-запити для отримання даних із головної бази даних. Програма керування баз даних перекладе ці запити LINQ на мову запитів SQL. Технологія також дозволяє виконувати необроблені SQL-запити безпосередньо по відношенню до бази даних.

При розробці та роботі з базою даних та системою в цілому, можливі такі варіанти підходи розробки:

- Database First – до існуючої бази даних пишеться програмна частина. В C# для спрощення такого написання існує можливість “скафолдингу” – автоматичної генерації коду (моделей, контексту тощо);
- Code First – спочатку пишеться програмний C# код. Створювати базу даних вручну не потрібно, вона створиться на основі моделей та контексту, написаних нами. Цей підхід люблять багато розробників і саме його використовуватимемо, тим паче що окремої бази даних на момент початку написання диплому не було.

В проектуванні системи використовувався підхід Code First, який полягає у написанні класів моделей, що описують властивості та характеристики об’єктів, що будуть зберігатися у базі даних та створенні, а також створенні контексту, в який заносяться моделі, які потрібно перетворити у таблиці у базі даних. У коді спочатку створюється конструктор моделей, який визначає класи, які керують контекстом, а потім використовує набір правил або умов, що визначають, як ці описують модель, і як ця модель повинна зберігатися у базі даних.

Діаграма бази даних системи (рисунок 3.2) зображає взаємодію основних сутностей програми; дані, які вони зберігають, тип зв’язку між ними тощо.

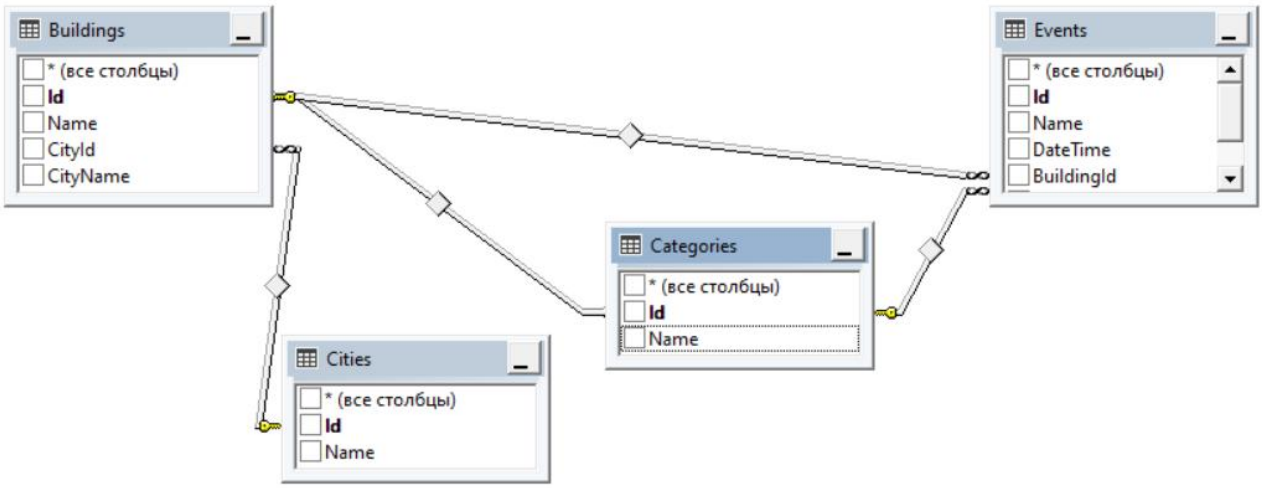


Рисунок 3.2 – Діаграми бази даних системи

Програмна система перегляду культурних заходів передбачає набір таблиць, які відображають реальні сутності, що використовуються у даній предметній сфері. Зокрема можна виділити такі сутності (а отже і таблиці бази даних) для роботи у даній предметній сфері з допомогою:

- Місто – місто, в якому відбувається захід. В основному українські міста, але можливі деякі іноземні, якщо у них проходять виступи українців. Містить такі властивості
    1. Ідентифікатор;
    2. Назва міста;
    3. Список будівель.
- Культурний захід – подія, яку можна відвідати. Містить такі властивості
    1. Ідентифікатор;
    2. Назва заходу;
    3. Категорія;
    4. Час та місце проведення;
    5. Будівля, місто;
    6. Опис.
- Будівля – споруда, в якій відбувається культурний захід (театр, філармонія, аквапарк тощо). Містить такі властивості
    1. Ідентифікатор;

2. Ім’я;
3. Місто;
4. Міське ім’я;
5. Список подій в даній будівлі.

- Категорія – категорія заходу: концерт, театральна вистава, для дітей тощо. Містить такі властивості
    1. Ідентифікатор;
    2. Ім’я;
    3. Список подій даної категорії.

- EfMigrationsHistory – допоміжна таблиця, яка ніяк не використовується нами безпосередньо для роботи з базою даних. Ця таблиця створюється автоматично, коли ми починаємо використовувати міграції бази даних

Для використання певних зв’язків між таблицями потрібно визначитися з можливими типами зв’язків між таблицями. Доступні такі:

- 1:1 (один до одного) – коли одному об’єкту з першої таблиці може відповідати лише один об’єкт з іншої таблиці. В даному проекті таке співвідношення не є потрібним. Але прикладом такого співвідношення може бути наступний приклад: номер паспорта людини – людей багато як і доступних номерів паспортів, але в однієї людини не може бути багато номерів, а 1 номер паспорту може належати лише 1 людині;

- 1:N / N:1 (один до багатьох, багато до одного) – коли для одного об’єкту однієї таблиці можливий лише 1 відповідник іншої таблиці, але в свою чергу той об’єкт другої таблиці може містити цілий список відповідників з першої таблиці. Наприклад, 1 будівля може фізично знаходитися в одному місті, але в місті може бути велика кількість культурних будівель;

- N:M (багато до багатьох) – у об’єкту з першої таблиці може бути безліч значень з іншої. При цьому таке твердження справедливе в обидва боки. Це може бути наприклад співвідношення між категоріями та подіями,

при умові що одна подія може відноситися зразу до декількох категорій (але її відношення до 1-єї або декількох категорій вже залежить від того, як нам потрібно будувати бізнес-логіку і наших потреб. Тому для наведених двох сутностей може використовуватися як варіант багато до багатьох, так і 1 до багатьох).

Враховуючи наведений вище матеріал, зв’язки між таблицями бази даних були реалізовані за допомогою типу «один-до-багатьох», що означає, що один екземпляр першого елемента може бути пов'язаний з безліччю екземплярів другого елементу, а один екземпляр другого елементу може бути пов'язаний лише з одним екземпляром першого елемента.

Якщо потрібно все ж використовувати зв’язок “N:M”, то треба ознайомитися з його особливостями у Ef Core. Раніше цей засіб вимагав створювання додаткової таблиці, в яку просто заносилися та співставлялися первинні ключі з обох таблиць. Зараз нова версія цієї ORM цього не потребує. Допоміжна таблиця може бути створена самостійно, однак нам з нею не потрібно буде з нею взаємодіяти.

Для підключення до бази даних через Entity Framework використовувався контекст даних. Контекст даних – це клас, похідний від класу DbContext. Контекст даних містить одну або кілька властивостей типу DbSet <T>, де T представляє тип об'єкта, що зберігається в базі даних.

3.3.4. Тестування програмної системи

Для перевірки коректної роботи функціоналу розробленої програмної системи був проведений процес аудиту системи засобами мануального тестування [65]. Мануальне тестування полягає у ручній перевірці діяльності ресурсу. Тестувальник відіграє роль користувача ресурсу та взаємодіє із ним так, як це робив би клієнт.

Звичайно для тестування системи може використовуватися і автоматизоване тестування. Для цього створюється в середині проєкту інший проєкт спеціально для тестів. Однак написання таких тестів може бути

досить складним та об’ємним процесом, хоча й виправдає себе в довгостроковій перспективі.

Результатом тестування є або вдале виконання передбачених функцій або помилка системи. Тобто у нас є очікуваний результат та фактичний результат. Для успішного проходження тесту ці 2 результати мають співпасти.

Задачею тестування є знаходження слабких місць системи та внесення даних про них до звіту помилок. Тестування відбувається за планом, складеним заздалегідь та оформленим у звіт під назвою «тестовий випадок». Тестування, особливо якщо воно є автоматичним, надає можливість впевнюватися в коректності роботи програми навіть після численних її змін, власне для цього воно і застосовується.

Для перевірки правильності діяльності системи було перевірено декілька тестових випадків. Так як фактичний результат відповідає очікуваному, то тестування є успішним.

Отже, за результатами виконання тестування можна говорити про коректне функціонування програмної системи.

Таким чином, в результаті виконання третього розділу кваліфікаційної роботи було проведено детальне проектування, планування, реалізація та тестування програмної системи. Результатом є розроблений веб-сервіс культурних заходів міста.

ВИСНОВОК

У результаті виконання кваліфікаційної роботи бакалавра розроблено веб-сервіс перегляду культурних заходів міст, зокрема:

- досліджено загально-теоретичні засади отримання, збору та структуризації інформації з різних веб-ресурсів у мережі Інтернет;
- здійснено аналіз програмно-технологічних рішень перегляду культурних заходів міста;
- спроектовано, реалізовано, впроваджено систему перегляду культзаходів міста з урахуванням технічних вимог;

А саме, були досліджені характеристики, особливості, формат подання даних та список доступних веб-сервісів подібної тематики. Було проведене їх загальне порівняння між собою.

З’ясований вплив наявності подібних веб-сервісів на відвідуваність культурних заходів. Вивчена географія діяльності подібних веб-сервісів, враховуючи що він є загальноукраїнським, однак українські заходи нерідко відбуваються і закордоном.

Були вивчені технологій побудови архітектури подібних систем; класичні та інноваційні підходи до організації роботи з даними користувачів; способи забезпечення безпеки транзакцій та захисту конфіденційних даних користувачів у межах системи. Досліджені програмно-технологічні засоби побудови інтерфейсу користувача та серверної сторони систем перегляду культурних заходів.

Для проектування та розробки системи кваліфікаційної роботи був застосований стек технологій на основі ASP.NET Core мовою C# для модулів програми та Microsoft SQL Server база даних для роботи з даними програми. Проведено порівняльний аналіз існуючих систем перегляду культурних заходів міста та виокремлений необхідний функціонал на розробку. З дотриманням вимог до функціональних можливостей системи була

спроектована клієнт-серверна архітектура застосунку. З використанням технології React спроектований клієнтоорієнтований інтерфейс користувача.

Розроблена структура системи, включаючи файлову систему класів, базу даних та модулі програми. Робота доповнена детальними діаграма послідовності, взаємодії, діяльності, розгортання системи та діаграми бази даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Плескач В. Л., Затонацька Т.Г. Електронна комерція: підручник. Київ: Знання, 2007. 535 с;
2. Andrew Troelsen, Philip Japikse. Pro C# 7 with .NET and .NET Core Eight edition: навч. посіб. Вид. 8-е. New York. 2017. 1372 с;
3. Сервіс продажу електронних квитків на культурні заходи міста karabas.com : веб-сайт. URL: https://karabas.com/ua/ (дата звернення: 10.02.2023);
4. Сервіс продажу електронних квитків на культурні заходи міста moemisto.ua : веб-сайт. URL: https://moemisto.ua/ (дата звернення: 10.02.2023);
5. Сервіс продажу електронних квитків на культурні заходи міста concert.ua : веб-сайт. URL: https://concert.ua/uk/ (дата звернення: 13.04.2023);
6. Сервіс продажу електронних квитків на культурні заходи міста internet-bilet.ua : веб-сайт. URL: https://internet-bilet.ua/uk/ (дата звернення: 12.03.2023);
7. Посібник взаємодії з платформою ASP.NET Core версії 7.0 : веб-сайт. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0 (дата звернення: 10.04.2023);
8. Посібник технології об’єктно-реляційного мапінгу EF Core : веб-сайт. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 20.04.2023);
9. Посібник для роботи з технологією ASP.NET WebAPI : веб-сайт. URL: https://learn.microsoft.com/enus/aspnet/core/fundamentals/apis?view=aspnetcore-7.0 (дата звернення: 11.04.2023);
10. React документація : веб-сайт. URL: https://uk.legacy.reactjs.org/tutorial/tutorial.html (дата звернення: 10.02.2023);
11. Anglesharp tutorial : веб-сайт. URL: https://scrapeshark.com/blog/web-scraper-csharp-dotnet-anglesharp (дата звернення: 13.05.2023);
12. Документація парсеру AngleSharp : веб-сайт. URL: https://anglesharp.github.io/ (дата звернення: 18.03.2023);

13. Що таке веб-сервіс : веб-сайт. URL: https://www.javatpoint.com/what-is-web-service (дата звернення: 11.03.2023);
14. Опис веб-сервісів SOAP та REST : веб-сайт. URL: https://www.reply.com/solidsoft-reply/en/content/webservices-soap-and-rest-a-simple-introduction (дата звернення: 15.03.2023);
15. RESTfull веб-сервіси : веб-сайт. URL: https://www.javatpoint.com/restful-web-services (дата звернення: 17.03.2023);
16. Заповнення зв’язаних таблиць бази даних початковими даними : веб-сайт. URL: https://learn.microsoft.com/en-us/ef/core/modeling/data-seeding (дата звернення:22.04.2023);

ДОДАТКИ

ДОДАТОК А

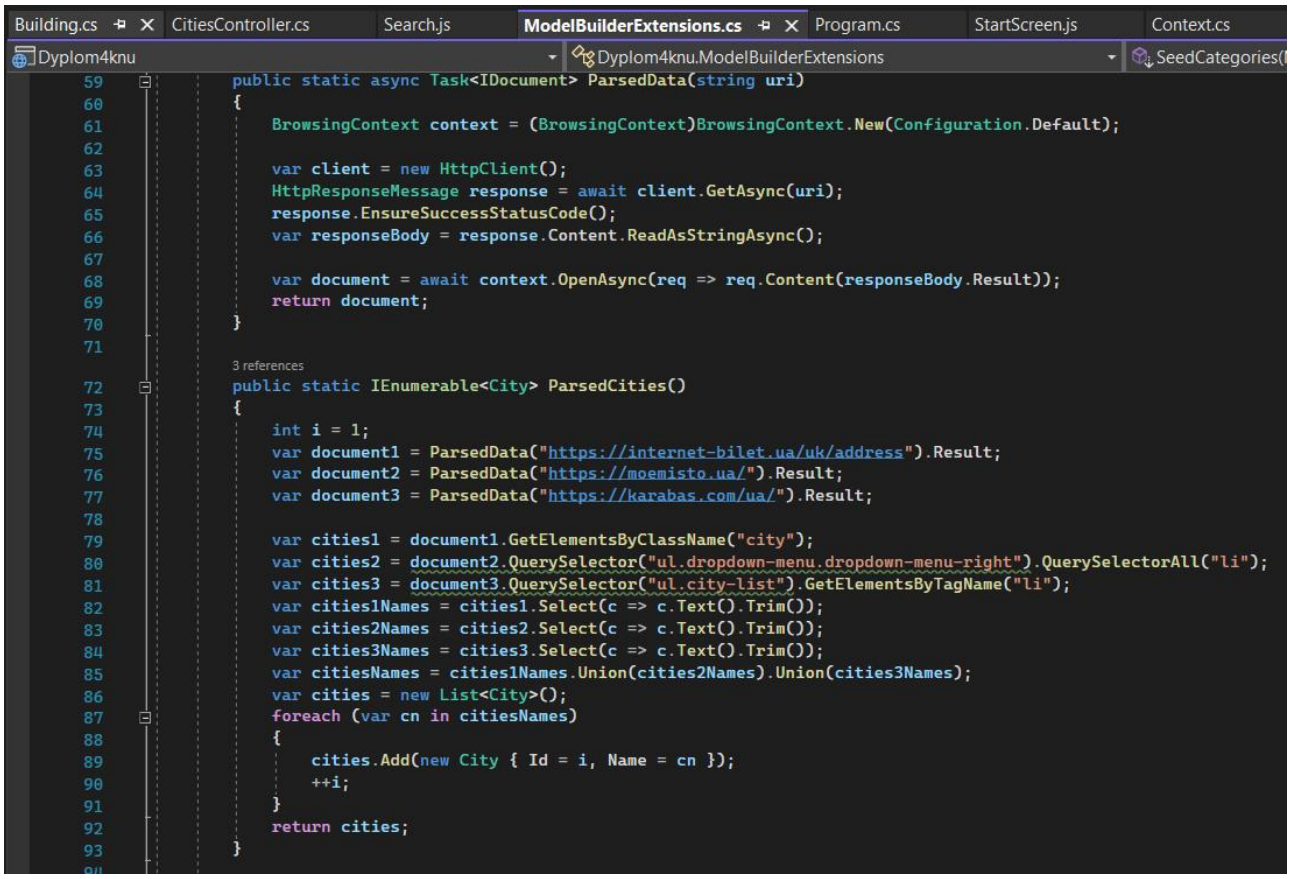


Рисунок 1А – приклад добування даних зі сторонніх ресурсів за допомогою технології веб-скрепінгу

ДОДАТОК Б

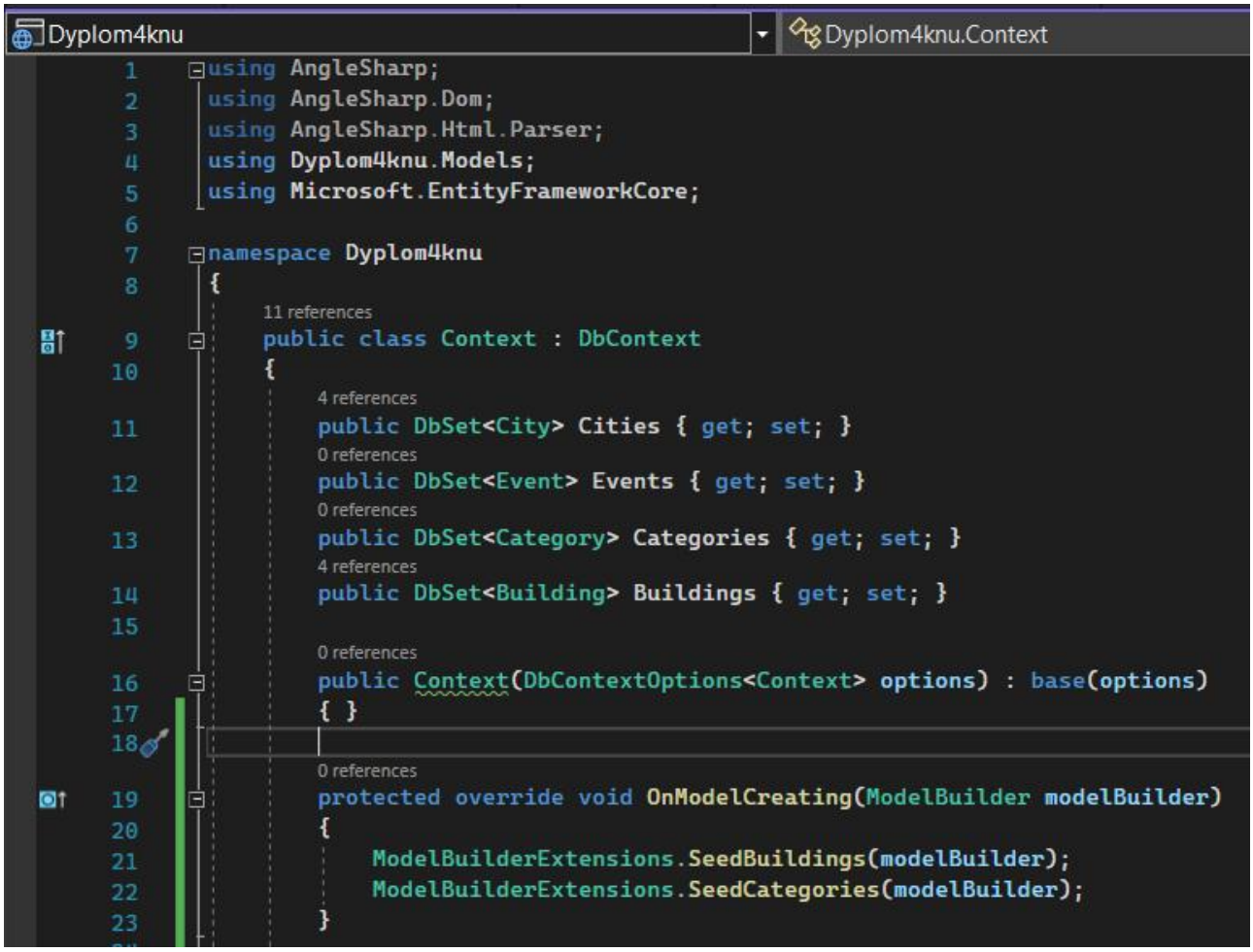


Рисунок 1Б – Контекст для перетворення моделей у таблиці бази даних та налаштування цих таблиць та їх даних

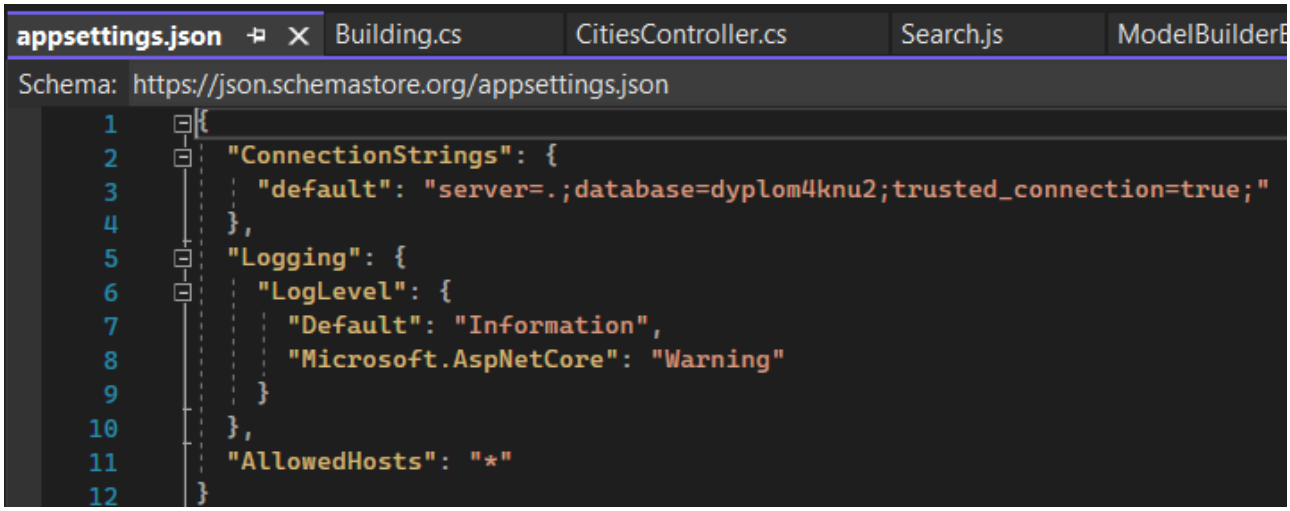


Рисунок 2Б – Файл конфігурації для налаштування підключення до бази даних

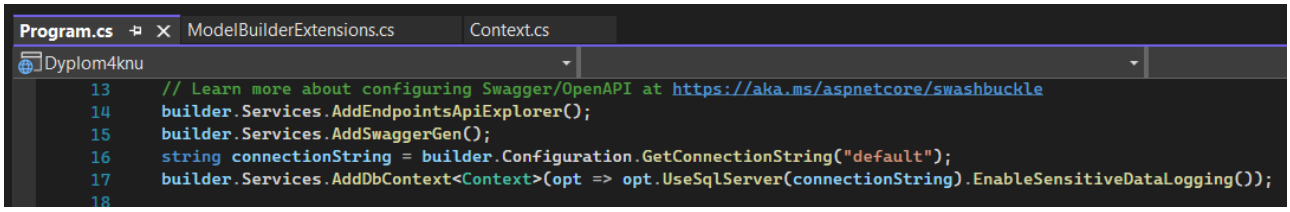


Рисунок 3Б – налаштування провайдера бази даних та зазначення контексту й рядка підключення до бази даних у основній програмі

ДОДАТОК В

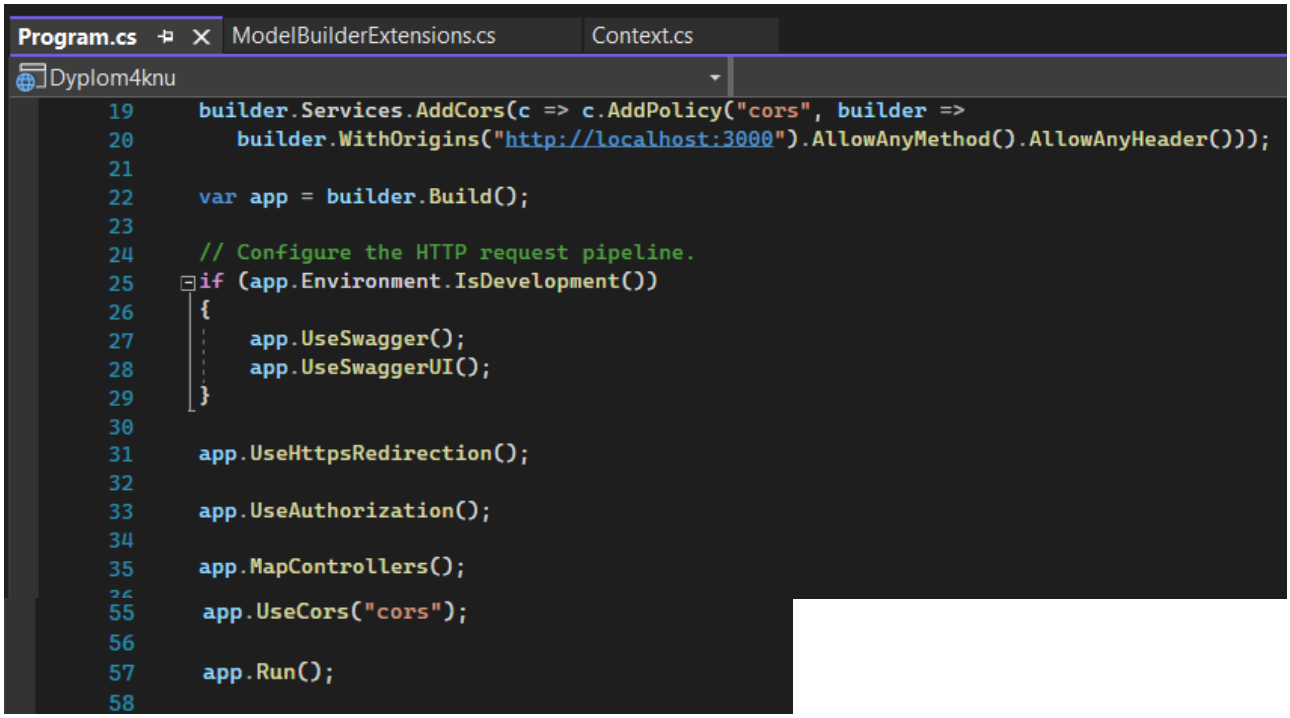


Рисунок 1В – Налаштування доступу FE до даних BE

ДОДАТОК Г

index.js StartScreen.js

react JavaScript Content Files { } "index"

```
1   import React from 'react';
2   import ReactDOM from 'react-dom/client';
3   import './index.css';
4   import App from './App';
5   import reportWebVitals from './reportWebVitals';
6   import StartScreen from "./Entities/StartScreen";
7
8   const root = ReactDOM.createRoot(document.getElementById('root'));
9   root.render(
10      <StartScreen/>
11  );
12
13  // If you want to start measuring performance in your app, pass a function
14  // to log results (for example: reportWebVitals(console.log))
15  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
16  reportWebVitals();
17
```

Рисунок 1Г – Вказання стартової сторінки при завантаженні для FE

index.js | StartScreen.js

react JavaScript Content Files | { } "StartScreen"

```
1   import { useEffect, useState } from "react";
2   import Search from "../Searching/Search";
3   import "./StartScreen.css";
4
5   function StartScreen() {
6       const [cities, setCities] = useState([])
7       var url = 'https://localhost:7176/api/cities/';
8
9       const loadScreen = () => {
10          fetch(url)
11              .then(response => {
12                  return response.json();
13              })
14              .then(data => {
15                  setCities(data)
16              })
17      }
18
19      useEffect(() => {
20          loadScreen()
21      }, [])
22
23      return (
24          <div>
25              <ul>
26                  {cities.map(city => (
27                      <li key={city.id}>
28                          {city.name}
29                          <p className="spoiler">
30                              {city.buildings.map(building => (
31                                  <button key={building.id}>
32                                      {building.name}<br/>
33                                  </button>
34                              ))}
35                          </p>
36                          <br/>
37                      </li>
38                  ))}
39              </ul>
40          </div>
41      )
42  }
43
44  export default StartScreen
```

Рисунок 2Г – Стартова сторінка з відображенням списку міст та підвантаженням списку культурних будівель для кожного з цих міст