

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПЛАТФОРМИ ДЛЯ
ВІДВІДУВАЧІВ ТЕАТРАЛЬНИХ ПОДІЙ
SOFTWARE FOR THE PLATFORM FOR THEATRICAL PERFORMANCE
VISITORS

Виконав: студент(ка) 4 курсу, групи КНТз-219 2
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Євсєєв О.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Євсєєва Олександра Олександровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення платформи для відвідувачів театральних подій
Software for the Platform for Theatrical Performance Visitors

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання « 4 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Проектування програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

(підпис) Євсєєв О.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Льовкін В.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 83 с., 27 рис., 3 табл., 3 дод., 14 джерел.

ТЕАТРАЛЬНА ПОДІЯ, ВІДВІДУВАЧ, РЕЦЕНЗУВАННЯ, ВЕБСИСТЕМА, ASP.NET CORE.

Об'єкт роботи – процес інформування про театральні події. Предмет роботи – програмне забезпечення платформи для відвідувачів театральних подій. Мета роботи – розробити програмне забезпечення для підтримки платформи відвідувачів театральних подій.

Матеріали, методи та технічні засоби: мова програмування C#, вебфреймворк ASP.NET Core, шаблон проєктування Model-View-Controller, система Microsoft SQL Server, фреймворк для роботи з даними Entity Framework Core.

Результати. Аналіз стану театральної галузі в Україні дозволив виявити відсутність міжтеатральної платформи, що разом з аналізом існуючих платформ в інших країнах дозволило визначити функціональні вимоги до програми. Проєктування програми дозволило створити структуру основних сутностей предметної області, самої вебсистеми і стало основою подальшої виконаної реалізації програми.

Висновки. Створена вебсистема дозволила реалізувати можливості перегляду театрів у місті, театральних подій за категоріями, тегами, виконавцем, створення розкладу театральних подій, рецензій про театральні події з можливостями їх коригування, перегляду і пошуку театральних подій у театрі або окремій залу театру, в тому числі всіх або майбутніх, визначення історії відвідування театральних подій, перегляду новин.

Галузь використання. Спільнота любителів театру з різних міст країни, зокрема України.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 pages, 27 figures, 3 tables, 3 appendixes, 14 sources.

THEATER EVENT, VISITOR, REVIEW, WEBSITE, ASP.NET CORE.

The object of the work is the process of informing about theatrical events. The subject of the work is the platform software for visitors of theatrical events. The purpose of the work is to develop software to support the platform of visitors of theatrical events.

Materials, methods and technical means: C# programming language, ASP.NET Core web framework, Model-View-Controller design template, Microsoft SQL Server system, Entity Framework Core data framework.

Results. Analysis of the state of the theater industry in Ukraine revealed the lack of an inter-theater platform, which together with the analysis of existing platforms in other countries allowed to determine the functional requirements for the program. The design of the program allowed to create a structure of the main entities of the subject area, the website itself and became the basis for the further implementation of the program.

Conclusions. The created web system allowed to realize possibilities of viewing theaters in a city, theatrical events by categories, tags, performers, creating a schedule of theatrical events, reviewing theatrical events with the ability to adjust them, viewing and searching for theatrical events in a theater or a separate theater, including all or future events, determining history of attending theatrical events, reading news.

Field of use. A community of theater lovers from different cities of a country, including Ukraine.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	7
Вступ	8
1 Аналіз предметної області	10
1.1 Огляд існуючих програмних аналогів	10
1.2 Порівняння існуючих програм	14
1.3 Висновки за розділом 1	14
2 Проєктування програмного забезпечення	16
2.1 Вибір засобів для проєктування і реалізації	16
2.2 Визначення функціональних вимог до програми	17
2.3 Проєктування бази даних	19
2.4 Визначення структури вебсистеми	25
2.5 Висновки за розділом 2	26
3 Розробка програми	27
3.1 Алгоритм функціонування програми	27
3.2 Опис реалізованих класів	28
3.3 Висновки за розділом 3	37
4 Експлуатація та тестування програми	38
4.1 Призначення програми	38
4.2 Умови виконання програми	38
4.3 Експлуатація програми	38
4.4 Тестування програми	43
4.5 Висновки за розділом 4	45
Висновки	46
Перелік джерел посилання	47
Додаток А Технічне завдання	49
Додаток Б Текст програми	53
Додаток В Слайди презентації	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

LINQ – Language-INtegrated Query;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Звичайно, що театри на даний момент не є найбільш популярним напрямком культури. Тим більше, що телебачення і інтернет впливають на всі сфери культурного життя. Однак цей вплив може бути різним. Це може бути привертання уваги від театру до інтернету, а з іншого боку це може бути застосуванням інформаційних технологій і інтернету для підтримки поціновувачів театру. Саме останній напрямок і є цікавим, адже він є багатосаровим. Ця багатосаровість обумовлена тим, що з одного боку такі інформації є наслідком існування театрів, і відповідно створюють лише певну підтримку театральним подіям, проте з іншого боку, існування єдиної платформи, де можна отримати інформацію про театральні події, створює одночасно і спільноту людей, але групує таку спільноту, якщо вона була роздібною. Сформована єдина спільнота призводить до того, що інші люди починають цікавитися самим фактом її існування, а тоді відповідно спільнота за рахунок цього розширюється. Тож враховуючи все це, можна говорити про синергію театрів і інформаційних технологій.

Синергія дозволяє розширити вплив, активніше розповсюджувати інформацію і таким чином отримувати зиск самим театрам в результаті. Про актуальність взагалі театральної галузі культури, незважаючи навіть на меншу популярність цієї сфери, говорить зокрема наявність колонок, розділів, окремо присвячених театральним подіям у провідних виданнях світу, як наприклад, в New York Times [1]. Проте в Україні такої платформи або певної міжтеатральної інформаційної технології не існує на даний момент. І це додатково до відсутності культурного ринку в Україні в результаті сповільнює розвиток даної сфери, робить її малорозповсюдженою, а театральні події маловідвідуваними. Однак вплив театру на свідомість людей великий, адже це простір, де можна формувати сенси, а не тільки впливати на маси.

У результаті була визначена мета дипломної роботи – розробити програмне забезпечення (ПЗ) для підтримки платформи відвідувачів театральних подій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих програмних аналогів

Програмні аналоги виконуваної розробки виконують підтримку інформування про театральні події у різних країнах світу. Одним з міст, в якому дуже розвинена подібна інфраструктура і де кількість таких платформ є достатньо великою, є Лондон.

Однією з платформ для поціновувачів театральних подій в Лондоні є London Theatre Direct (рис. 1.1) [2]. Загалом це платформа компанії з продажу квитків на театральні події в Лондоні, а оскільки платформа має міжтеатральний характер, то вона цікава в розрізі аналізу цієї предметної області.

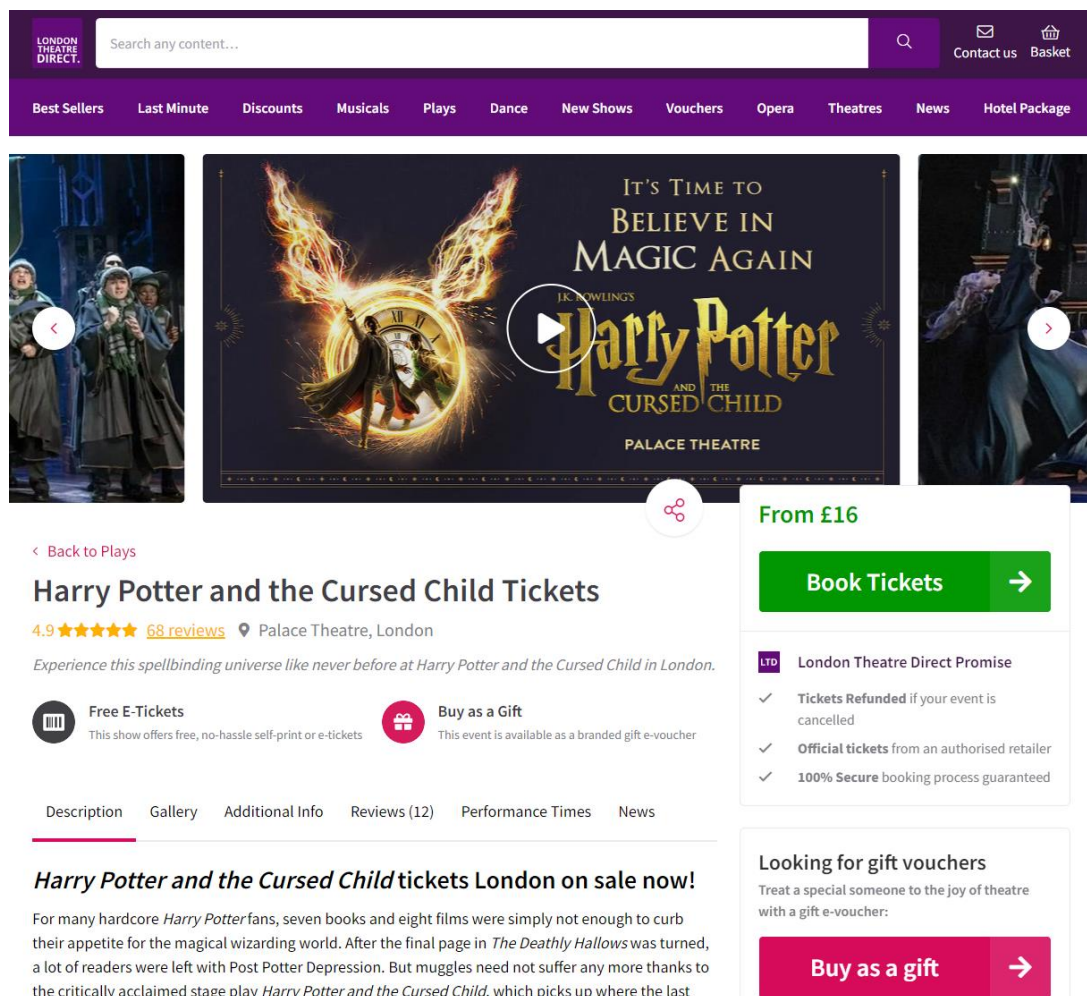


Рисунок 1.1 – London Theatre Direct

На цьому вебсайті надаються великі описи відповідних вистав, мюзиклів, опер, оперет. За кожної з них окрім опису представляється множина фотографій з події, а також інформація про вікові обмеження, тривалість, дати виступів, особливості контенту, інформація про доступ для людей з інвалідністю, а також дуже розлога інформація, що охоплює відгуки відвідувачів, теги, час виступів. Теги дозволяють переходити між виставами і знаходити таким чином швидко те, що цікавить користувача. Також надаються новини, які співвідносяться з театральними подіями.

Вебсайт також надає можливість бронювання квитків на подію, а також реалізовувати пошук за назвою події. Окрім того доступні сторінки з даними відповідних театрів, звідки можна отримати інформацію про перелік подій на майбутнє та таких, які вже відбулись.

При відображенні всієї множини подій можна встановити фільтри, що включають дату виступу, інтервал цін на квитки, теги, рейтинг відгуків відвідувачів.

Вебсайт London Theatre (рис. 1.2) [3] позиціонується як путівник в театальному світі Лондона, що дозволяє купляти квитки на події.

London Theatre дозволяє переглядати всі театри міста, як за алфавітом, так і на мапі, у вигляді путівників за театрами. На вебсайті доступні відгуки експертів про вистави, які окремо виділені в єдиний розділ. Також окремо доступний перелік найбільш популярних вистав, вистав за різними жанрами. Інформація про кожну виставу включає тривалість, час завершення бронювання, перелік категорій, вік, з якого доступна подія, час прем'єри, початку бронювання та місце, де відбувається подія. Також у вигляді таблиці виводиться розклад за днями тижня цієї вистави, а також відео та фотографії з вистави, інформація про розташування театру, способи дістатися до цього театру.

London Theatre

What's On ▾ Tickets ▾ News & Interviews ▾ Reviews Theatres ▾ Stars on Stage ▾ Info ▾

Home / Tickets / **Britannicus**

Britannicus Tickets

London

[Lyric Hammersmith](#)

26 May 2022 - 25 Jun 2022

★ ★ ★ ★ ★

Britannicus Information

Power and succession are in the very bones of Rome. Agrippina, desperate to cling onto power, has ensured her son, Nero, is the Roman Emperor, in place of his half-brother and the rightful heir. But the corridors of power and Nero's own obsession turn a once virtuous ruler into an oppressive tyrant.

Book tickets to *Britannicus*, or [sign up to receive the latest news and show alerts](#).

Opening date
May 26th, 2022

Closing date
June 25th, 2022

Categories
[Play](#)

Venue
Lyric Hammersmith

Show schedule

Sorry, no performance schedule available.

Get the latest updates on Britannicus

To get the latest news, reviews, interviews, new show alerts and ticket offers, sign up to our weekly newsletter

Email

☐ Weekly newsletter - London Theatre news and the latest reviews

☐ Last minute deals, flash sales and new show alerts

Рисунок 1.2 – Вебсайт London Theatre

На вебсайті список виконавців (акторів, режисерів тощо) міститься, але він не дозволяє переглядати інформацію за конкретним актором, наприклад. Тобто не можна дізнатися всі вистави за актором, наприклад, в результаті. Немає також і можливості створювати власні відгуки.

Окремо також було розглянуто вебсайти в Україні, які присвячені окремим театрам. Вебсайт Національного театру імені Марії Заньковецької (рис. 1.3) [4] надає афішу театру, новини театру, можливість перейти до придбання квитків на зовнішньому вебсайті за відкритою виставою. Також на сторінці вистави є інформація про її тривалість, прем'єру, опис, фотографії, дійові особи та виконавців. При цьому імена оформлені у вигляді посилань, що дозволяє відкрити сторінку виконавця з переліком всіх вистав.



Рисунок 1.3 – Вебсайт Національного театру імені Марії Заньковецької

Вебсайт Запорізького академічного обласного українського музично-драматичного театру імені В.Г. Магара (рис. 1.4) [5] має подібну інформацію.



Рисунок 1.4 – Вебсайт Запорізького академічного обласного українського музично-драматичного театру імені В.Г. Магара

1.2 Порівняння існуючих програм

За результатами огляду було виконано порівняння основних існуючих засобів з тією програмою, яку необхідно в роботі розробити (табл. 1.1). В якості особливостей розробки визначена наявність історії відвідування, направленість на театри в Україні, а також можливість концентрації інформації щодо виконавців, які беруть участь в театральних подіях (акторів, сценаристів тощо).

Таблиця 1.1 – Порівняння програмних аналогів з виконуваною розробкою

Критерій	London Theatre Direct	London Theatre	TheatralApp
Країна розташування театрів	Велико-британія	Велико-британія	Потенційно Україна
Пошук подій	Так	Так	Так
Сторінки виконавців	Ні	Ні	Так
Бронювання квитків	Так	Так	Ні
Відгуки користувачів	Так	Ні	Так
Театральні новини	Так	Ні	Так
Історія відвідувань	Ні	Ні	Так
Обліковий запис користувача	Ні	Ні	Так

1.3 Висновки за розділом 1

Огляд ПЗ, що використовується для підтримки роботи театральних платформ, включав London Theatre Direct, London Theatre, вебсайти українських театрів. Вебсайти українських театрів не об'єднані на єдиній платформі. В якості особливостей розробки визначена наявність історії

відвідування, направленість на театри в Україні а також можливість концентрації інформації щодо виконавців, які беруть участь в театральних подіях (акторів, сценаристів тощо).

Завданнями дипломної роботи є:

- аналіз предметної області;
- визначення вимог до ПЗ платформи для відвідувачів театральних подій;
- проєктування ПЗ платформи для відвідувачів театральних подій;
- реалізація і налагодження ПЗ платформи для відвідувачів театральних подій.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір засобів для проєктування і реалізації

Першим питанням, що вирішувалось у даному підрозділі, було питання вибору мови програмування. Враховуючи необхідність веброзробки, було використано дві альтернативи, які включали мови C# [6]-[7] і Java [8]. Проведене порівняння (табл. 2.1) вказало на те, що обидві мови є достатньо якісними для розробки цієї програми, проте обрано було C#, зважаючи на деякі додаткові переваги, що включають кращу оптимізацію швидкості виконання, що дозволяє цим програмам загалом дещо швидше працювати в підсумку, до того ж мова підтримує технологію Language-Integrated Query (LINQ), що дозволяє працювати з даними без необхідності створення постійно відповідних запитів.

Таблиця 2.1 – Вибір мови програмування для розробки

Критерій	C#	Java
Швидкість розробки	+—	+—
Підтримка спрощеної роботи з даними	+	—
Засоби веброзробки на основі фреймворків	+	+
Оптимізація швидкості виконання	+	+—
Можливість створення кросплатформного рішення	+	+

Додатково до даного вибору було також зроблено вибір на користь вебфреймворку ASP.NET Core [9]-[10].

2.2 Визначення функціональних вимог до програми

Повний перелік функціональних вимог до програми включає:

- перегляд переліку театрів у місті;
- перегляд рейтингу подій за категоріями подій;
- створення і редагування розкладу театральних подій;
- створення рецензій про театральну подію;
- коригування рецензії автором;
- перегляд переліку театральних подій у театрі або окремій залу театру, в тому числі всіх або майбутніх;
- перегляд історії відвідування театральних подій;
- внесення обговорень за подіями;
- визначення вартості квитків на подію;
- пошук театральних подій;
- перегляд переліку театральних подій за тегом;
- перегляд стрічки новин, в тому числі за подією або за театром;
- пошук виконавців;
- перегляд ролей виконавця;
- перегляд рецензій за автором;
- деталізація театральної події;
- деталізація театру;
- створення новини;
- створення і редагування театральних подій;
- створення і редагування даних театрів;
- внесення виконавців;
- авторизація і реєстрація.

На основі цих вимог було визначено перелік сценаріїв роботи відвідувача (рис. 2.1) як звичайного користувача програми і експерта як користувача, який може працювати з привілейованими рецензіями, та

редактора (рис. 2.2) як користувача, який може вносити дані в програму, необхідні для роботи відвідувачів.

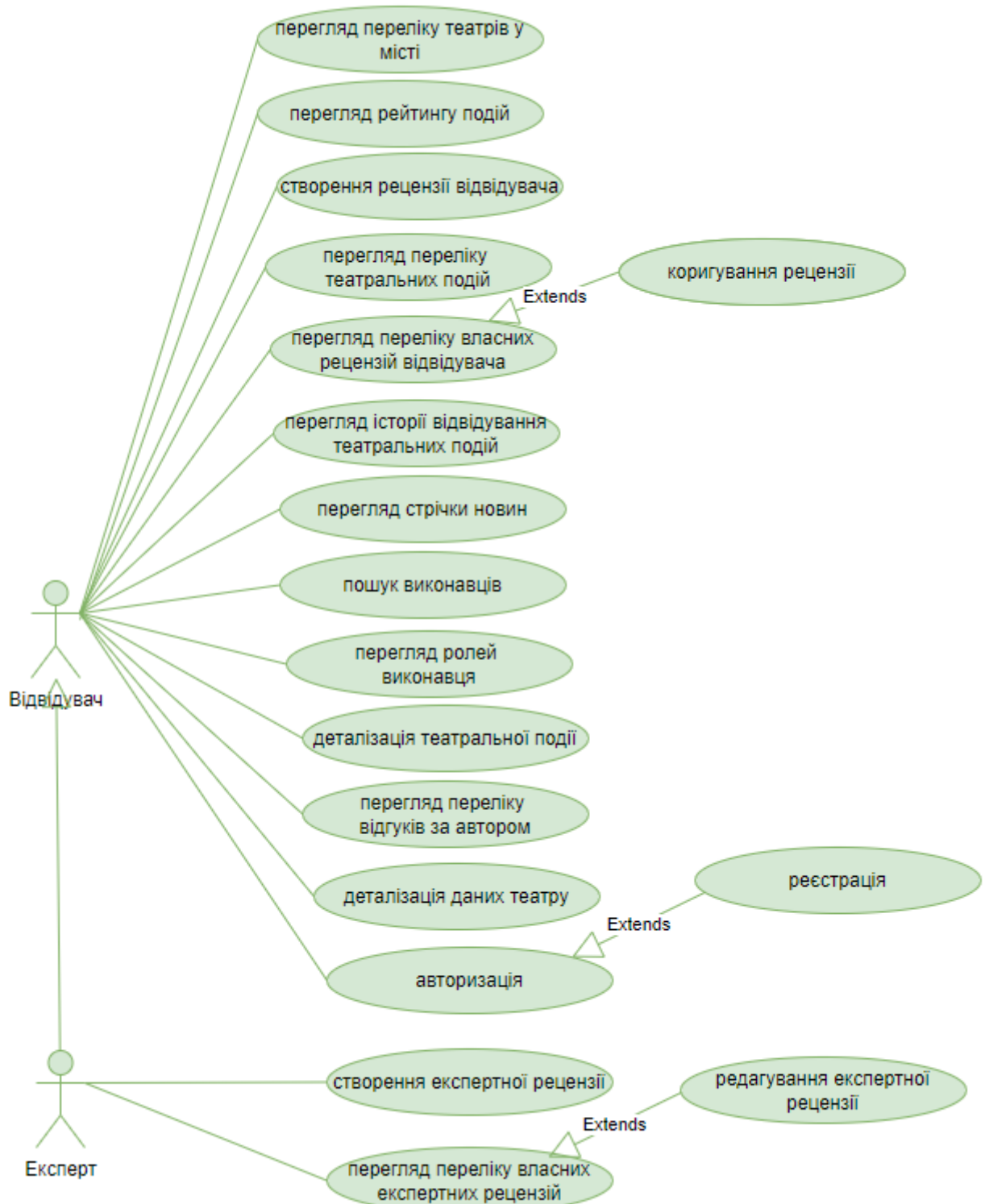


Рисунок 2.1 – Діаграма прецедентів відвідувача і експерта

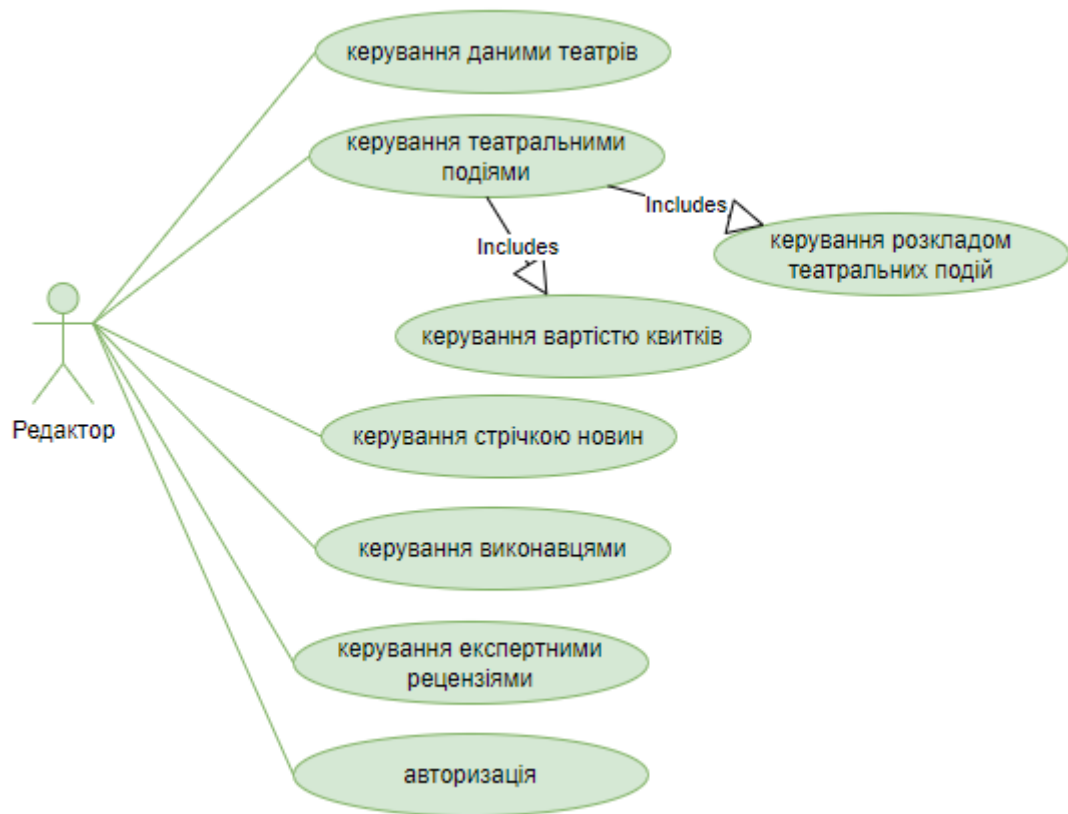


Рисунок 2.2 – Діаграма прецедентів редактора

2.3 Проектування бази даних

Для роботи з базою даних (БД) було використано систему Microsoft SQL Server [11].

Сутність Theater є театром, де і відбувається театральна подія. Відповідна таблиця БД Theater зв'язана з таблицею City, а оскільки в місті може бути декілька театрів, при цьому кожен театр в конкретному місті, то це зв'язок типу один до багатьох.

Сутність Hall є залом театру, оскільки в одному театрі може бути декілька залів. Відповідна таблиця БД Hall зв'язана з таблицею Theater, а оскільки в театрі може бути декілька залів, а при цьому кожна зала належить до конкретного театру, то це зв'язок типу один до багатьох.

Сутність PlaceCategory є категорією місць у залі (наприклад, ложа, партер).

Сутність Staff є особою, задіяною в роботі над певною театральною подією. Це може бути і конкретний актор, і режисер, і будь-який інший учасник вистави.

Сутність StaffRole є роллю, яку може грати учасник в театральній події. Це безпосередньо актор, режисер, сценарист тощо.

Сутність PerformanceStaff є визначенням ролі участі певної особи в певній виставі.

Відповідна таблиця БД PerformanceStaff зв'язана з таблицею Performance, оскільки запис про участь деякої особи у виставі стосується конкретної вистави, а при цьому кожна вистава може включати участь різних осіб, то це зв'язок типу один до багатьох.

Відповідна таблиця БД PerformanceStaff зв'язана з таблицею Staff, оскільки запис про участь особи у виставі стосується конкретної особи, а при цьому кожна особа може брати участь у різних виставах, то це зв'язок типу один до багатьох.

Відповідна таблиця БД PerformanceStaff зв'язана з таблицею StaffRole, оскільки запис про участь особи у виставі стосується конкретної ролі, а при цьому кожна роль може бути використана в різних таких записах, то це зв'язок типу один до багатьох.

Сутність City є містом, в якому розташовано театр. Необхідність цієї сутності в тому, що програма повинна дозволяти працювати з нею різними театрами з однієї країни, адже переважно вона розроблялась саме для України.

Сутність Category є категорією театральної події. Це може бути мюзикл, опера, оперета.

Сутність Performance є безпосередньо театральною подією.

Відповідна таблиця БД Performance зв'язана з таблицею Hall, оскільки театральна подія може відбуватися в одному залі театру, а при цьому кожна зала може використовуватися для різних таких подій, то це зв'язок типу один до багатьох.

Сутність Talk є обговоренням театральної події, що може відбуватися онлайн або як зустріч офлайн.

Відповідна таблиця БД Talk зв'язана з таблицею Performance, оскільки може бути багато обговорень театральних подій, а за правилами застосунку кожне обговорення має стосуватися конкретної події, то це зв'язок типу один до багатьох.

Сутність PerformanceTime визначає час, коли відбувається конкретна театральна подія, тобто час вистави.

Відповідна таблиця БД PerformanceTime зв'язана з таблицею Performance, оскільки може бути багато вистав за театральною подією в різні дати, а кожна вистава стосується тільки однієї театральної події, то це зв'язок типу один до багатьох.

Сутність Expert є експертом. Експерт в системі визначає критичні рецензії про театральні події.

Сутність Visitor є відвідувачем.

Сутність Editor є редактором, який відповідає за загальне керування.

Сутність News є новиною, яка стосується театральної події.

Сутність SearchTag є тегом, який використовується для пошуку між театральними подіями.

Сутність Visit є інформацією про відвідування події відвідувачем.

Відповідна таблиця БД Visit зв'язана з таблицею Performance, оскільки може бути багато відвідувань однієї театральної вистави різними відвідувачами, а кожне відвідування може бути відвідуванням тільки однієї театральної події, то це зв'язок типу один до багатьох.

Відповідна таблиця БД Visit зв'язана з таблицею Visitor, оскільки у одного відвідувача може бути багато відвідувань різних театральних подій, а кожне відвідування може стосуватися тільки одного відвідувача, то це зв'язок типу один до багатьох.

Сутність ExpertReview є рецензією експерта про театральну подію.

Відповідна таблиця БД ExpertReview зв'язана з таблицею Expert, оскільки у одного експерта може бути багато рецензій, а кожна експертна рецензія може бути авторства тільки одного експерта одночасно, то це зв'язок типу один до багатьох.

Відповідна таблиця БД ExpertReview зв'язана з таблицею Editor, оскільки одним редактором може бути внесено багато рецензій, а кожна експертна рецензія може вноситися тільки одним редактором, який за неї відповідає, то це зв'язок типу один до багатьох.

Дана таблиця має зв'язок одразу з двома таблицями, тому що може статися така ситуація, що сам експерт з системою не працює, але надає рецензію, тоді редактор вносить дані рецензії в систему.

Відповідна таблиця БД ExpertReview зв'язана з таблицею Performance, оскільки одна театральна подія може мати багато рецензій експертів, а кожна рецензія експерта стосується однієї конкретної рецензії, то це зв'язок типу один до багатьох.

Сутність PriceScheme визначає схему вартості квитків. Необхідно для того, щоб мати для однієї й тієї самої театральної події різні схеми вартості, щоб на кожен дату заново не визначати вартість.

Сутність PerformancePlaceCategory визначає використання категорій місць у залі для конкретної театральної події. Це необхідно, оскільки, наприклад, не на всі події можуть продаватися одразу всі квитки в залі.

Відповідна таблиця БД PerformancePlaceCategory зв'язана з таблицею HallPlaceCategory, оскільки одна категорія місць у залі для конкретної театральної події стосується конкретної категорії місць у залі, а кожна категорія місць у залі може мати багато категорій місць у залі для конкретної театральної події, то це зв'язок типу один до багатьох.

Відповідна таблиця БД PerformancePlaceCategory зв'язана з таблицею Performance, оскільки одна театральна подія може мати багато категорій місць, а кожна категорія місця визначається саме під конкретну театральну подію, то це зв'язок типу один до багатьох.

Сутність VisitorReview є рецензією відвідувача про театральну подію.

Відповідна таблиця БД VisitorReview зв'язана з таблицею Visitor, оскільки у одного відвідувача може бути багато рецензій, а кожна рецензія відвідувача може бути авторства тільки одного відвідувача одночасно, то це зв'язок типу один до багатьох.

Відповідна таблиця БД VisitorReview зв'язана з таблицею Performance, оскільки одна театральна подія може мати багато рецензій її відвідувачів, а кожна рецензія відвідувача стосується однієї конкретної рецензії, то це зв'язок типу один до багатьох.

Сутність PriceGroup визначає ділення категорій місць, описаних в сутності PriceCategory, на окремі групи, тобто місця на балконі можуть мати різну вартість на різних балконах або в різних рядах.

Відповідна таблиця БД PriceGroup зв'язана з таблицею PriceCategory, оскільки одне ділення категорій місць може мати декілька варіантів вартості, а відповідно кожний такий варіант стосується тільки однієї конкретної такої категорії місць, то це зв'язок типу один до багатьох.

Сутність PerformancePrice є визначенням застосування схеми вартості квитків для конкретного дня і часу, коли виконується театральна подія.

Відповідна таблиця БД PerformancePrice зв'язана з таблицею PriceSchemeGroup, оскільки одне ділення категорій місць може мати декілька варіантів вартості, а відповідно кожний такий варіант стосується тільки однієї конкретної такої категорії місць, то це зв'язок типу один до багатьох.

Сутність PriceSchemeGroup є визначенням вартості конкретного ділення місць за конкретної категорії місць на цій театральній події за конкретною схемою вартості квитків.

Таблиця VisitorPerformance визначає відвідування певної театральної події відвідувачем, забезпечуючи зв'язування таблиць Visitor і Performance.

Результуюча схема БД показана на рис. 2.3.

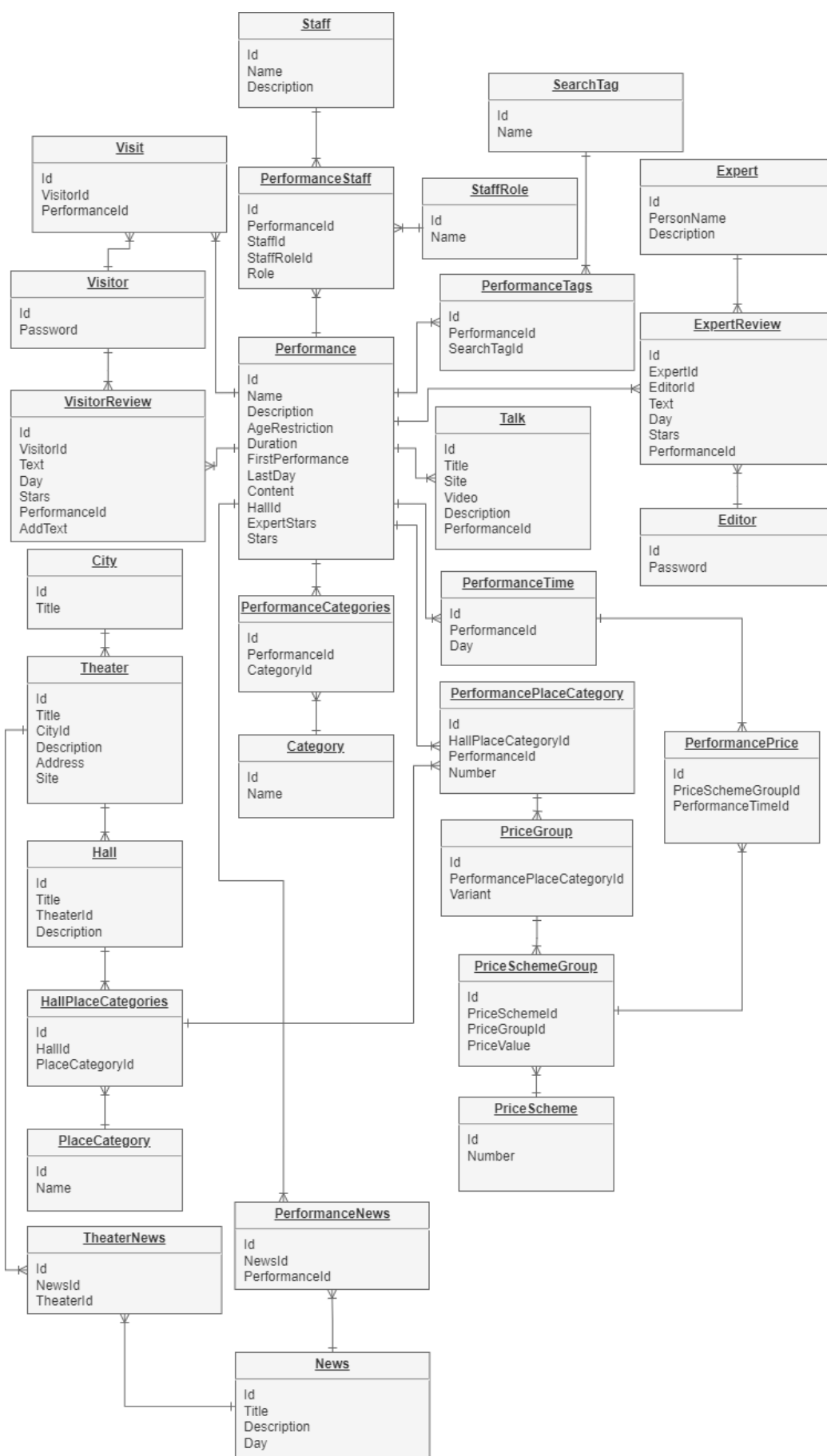


Рисунок 2.3 – Схема БД

Таблиця TheaterNews визначає новини за театром, забезпечуючи зв'язування таблиць Theater і News, тобто вказуючи, що певна новина стосується деякого театру.

Таблиця PerformanceNews визначає новини про театральні події, забезпечуючи зв'язування таблиць Performance і News, тобто вказуючи, що певна новина стосується деякої театральної події.

Таблиця PerformanceCategories визначає відношення театральної події до категорії театральних подій, забезпечуючи зв'язування таблиць Performance і Category.

Таблиця HallPlaceCategories визначає відношення категорій місць в залі, забезпечуючи зв'язування таблиць HallPlace і Category.

Таблиця PerformanceTags визначає теги театральної події, забезпечуючи зв'язування таблиць Performance і Tag.

2.4 Визначення структури вебсистеми

Для визначення структури було використано шаблон Model-View-Controller (рис. 2.4) [12]-[13].

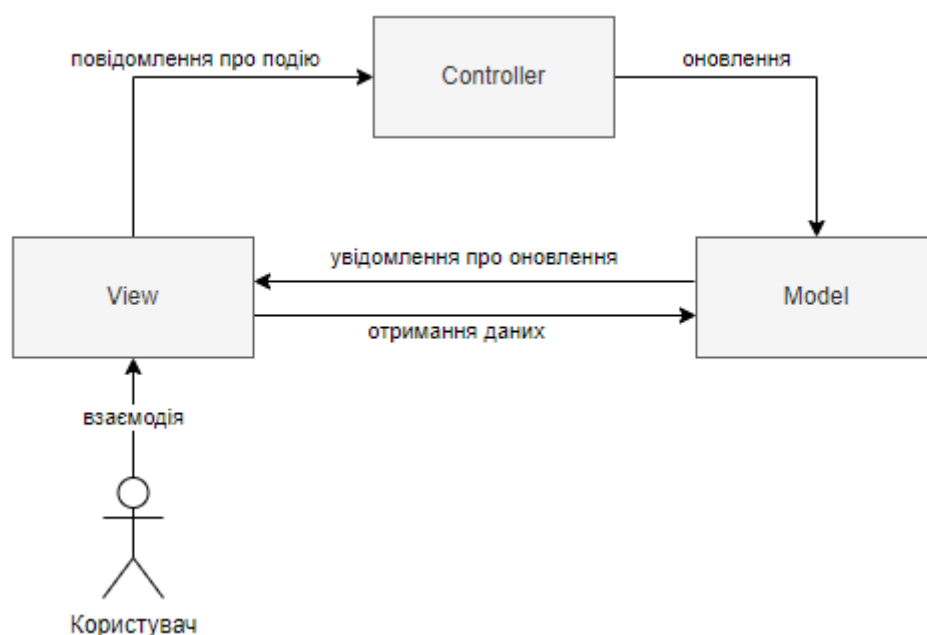


Рисунок 2.4 – Шаблон проєктування Model-View-Controller

Model-View-Controller містить такі складові:

- модель: бекенд, який містить всю логіку даних;
- представлення: інтерфейс або графічний інтерфейс користувача;
- контролер: мозок програми, який контролює відображення даних

[12]-[13].

2.5 Висновки за розділом 2

Вибрані засоби проєктування та подальшої розробки в результаті включили мову програмування C#, вебфреймворк ASP.NET Core, шаблон проєктування Model-View-Controller, систему Microsoft SQL Server. Представлено діаграми прецедентів відвідувача і експерта та редактора. Реалізовано проєктування БД.

3 РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Порядок роботи з функціями програми відвідувача показаний на рис. 3.1.



Рисунок 3.1 – Схема функціонування програми

3.2 Опис реалізованих класів

Реалізацію програми виконано за допомогою використання фреймворку ASP.NET та фреймворку для роботи з даними Entity Framework Core [14]. Використання цих технологій дозволило зменшити кількість коду, яку потрібно для роботи з даними в БД, враховуючи також застосування технології LINQ.

Реалізований клас моделі театральної події Performance складається з полів:

- public int Id – ідентифікатор театральної події;
- public string Name – назва театральної події;
- public string Description – опис театральної події;
- public short AgeRestriction – обмеження віку для події;
- public TimeSpan Duration – тривалість вистави;
- public DateTime FirstPerformance – час, коли відбулась прем'єра;
- public DateTime LastDay – час останнього запланованого виступу;
- public string Content – змістовні особливості театральної події;
- public float ExpertStars – середня оцінка експертів;
- public float Stars – середня оцінка відвідувачів;
- public int HallId – номер зали театру, де має відбуватися подія;
- public virtual Hall Hall – віртуальне поле, що надає доступ до зали театру, де має відбуватися подія;
- public virtual ICollection<PerformanceStaff> PerformanceStaffs – виконавці театральної події;
- public virtual ICollection<Category> PerformanceCategories – категорії театральної події;
- public virtual ICollection<Talk> Talks – обговорення театральної події;
- public virtual ICollection<PerformanceTime> PerformanceTimes – час вистав за театральною подією;

– public virtual ICollection<PerformancePlaceCategory>

PerformancePlaceCategories – категорії місць на театральній події;

– public virtual ICollection<Visit> Visits – історія візитів на подію;

– public virtual ICollection<SearchTag> PerformanceTags – теги події;

– public virtual ICollection<ExpertReview> ExpertReviews – експертні рецензії;

– public virtual ICollection<VisitorReview> VisitorReviews – рецензії відвідувачів;

– public virtual ICollection<News> PerformanceNews – новини про подію.

Клас моделі обговорення Talk складається з полів:

– public int Id – ідентифікатор обговорення;

– public string Title – заголовок обговорення;

– public string Site – посилання на обговорення;

– public string Video – посилання на файл з відео обговорення;

– public string Description – опис обговорення;

– public int PerformanceId – номер театральної події, якої стосується обговорення;

– public virtual Performance Performance – віртуальне поле, що надає доступ до театральної події, якої стосується обговорення.

Клас моделі експертної рецензії ExpertReview складається з полів:

– public int Id – ідентифікатор експертної рецензії;

– public string Text – повний текст рецензії;

– public DateTime Day – час, коли створена рецензія;

– public short Stars – оцінка, виставлена в рецензії;

– public int ExpertId – номер експерта, який написав рецензію;

– public virtual Expert Expert – віртуальне поле, що надає доступ до експерта, який написав рецензію;

– public int EditorId – номер редактора, який вніс рецензію;

- public virtual Editor Editor – віртуальне поле, що надає доступ до редактора, який вніс рецензію;

- public int PerformanceId – номер театральної події, про яку рецензія;

- public virtual Performance Performance – віртуальне поле, що надає доступ до театральної події, про яку створена рецензія.

Клас моделі рецензії відвідувача VisitorReview складається з полів:

- public int Id – ідентифікатор рецензії відвідувача;

- public string Text – повний текст рецензії;

- public DateTime Day – час, коли створена рецензія;

- public short Stars – оцінка, виставлена в рецензії;

- public int VisitorId – номер відвідувача, який написав рецензію;

- public virtual Visitor Visitor – віртуальне поле, що надає доступ до відвідувача, який написав рецензію;

- public int PerformanceId – номер театральної події, про яку створена рецензія;

- public virtual Performance Performance – віртуальне поле, що надає доступ до театральної події, про яку створена рецензія.

Клас моделі театру Theater складається з полів:

- public int Id – ідентифікатор театру;

- public string Title – назва театру;

- public string Description – опис театру;

- public string Address – адреса театру;

- public string Site – посилання на вебсайт театру;

- public int CityId – номер міста, де розташовано театр;

- public virtual City City – віртуальне поле, що надає доступ до міста, де розташовано театр;

- public virtual ICollection<Hall> Halls – перелік залів театру.

Клас моделі зали театру Hall складається з полів:

- public int Id – ідентифікатор зали театру;

- public string Title – назва зали театру;

- public string Description – опис зали театру;
- public int TheaterId – номер театру, де розташовано залу;
- public virtual Theater Theater – віртуальне поле, що надає доступ до театру, де розташовано залу;
- public virtual ICollection<Performance> Performances – перелік театральних подій в цьому залі;
- public virtual ICollection<PlaceCategory> HallPlaceCategories – перелік театральних подій в цьому залі.

Клас моделі особи учасника театральних подій Staff складається з полів:

- public int Id – ідентифікатор особи;
- public string Name – повне ім'я людини;
- public string Description – опис цього учасника;
- public virtual ICollection<PerformanceStaff> PerformanceStaffs – перелік даних про участь цієї особи в різних театральних подіях.

Клас моделі новини News складається з полів:

- public int Id – ідентифікатор новини;
- public string Title – заголовок новини;
- public string Description – опис новини;
- public DateTime Day – час, коли створена новина;
- public virtual ICollection<Performance> PerformanceNews – перелік театральних подій, що відповідають цій новині;
- public virtual ICollection<Theater> TheaterNews – перелік театрів, що відповідають цій новині.

Клас моделі експерта Expert складається з полів:

- public int Id – ідентифікатор експерта;
- public string PersonName – повне ім'я експерта;
- public string Description – опис експерта;
- public virtual ICollection<ExpertReview> ExpertReviews – перелік рецензій цього експерта.

Клас моделі тегу SearchTag складається з полів:

- public int Id – ідентифікатор тегу;
- public string Name – назва тегу;
- public virtual ICollection<Performance> PerformanceTags – перелік

театральних подій за цим тегом.

Клас моделі міста City складається з полів:

- public int Id – ідентифікатор міста;
- public string Title – назва міста;
- public virtual ICollection<Theater> Theaters – перелік театрів з цього

міста.

Клас моделі категорії місця PlaceCategory складається з полів:

- public int Id – ідентифікатор категорії місця;
- public string Name – назва категорії;
- public virtual ICollection<Hall> HallPlaceCategories – перелік залів, у

яких є ця категорія місць.

Клас моделі ролі, яку особа може виконувати в театральній події, StaffRole складається з полів:

- public int Id – ідентифікатор ролі;
- public string Name – назва ролі;
- public virtual ICollection<PerformanceStaff> PerformanceStaffs –

перелік всіх осіб, які брали участь в різних театральних подіях в цій ролі.

Клас моделі категорії подій Category складається з полів:

- public int Id – ідентифікатор категорії події;
- public string Name – назва категорії;
- public virtual ICollection<Performance> PerformanceCategories –

перелік театральних подій, що відносяться до цієї категорії.

Клас моделі часу, коли відбувається конкретна театральна подія, PerformanceTime складається з полів:

- public int Id – ідентифікатор часу, коли відбувається конкретна

театральна подія,;

- public DateTime Day – дата події;
- public int PerformanceId – ідентифікатор театральної події;
- public virtual Performance Performance – віртуальне поле, що надає доступ до театральної події;
- public virtual ICollection<PriceSchemeGroup> PriceSchemeGroups – перелік визначень вартості конкретного ділення місць за конкретної категорії місць на даному виконанні театральної події за конкретною схемою вартості квитків.

Клас сервісу PerformanceService дозволяє виконувати базові дії щодо взаємодії з БД, забезпечуючи для цього ряд методів.

Метод Performance GetById (int perform) повертає повні дані театральної події за її номером perform.

Метод IEnumerable<Performance> GetAll() повертає перелік всіх театральних подій.

Метод IEnumerable<Performance> GetByTheater(int theaterId) повертає театральні події, що відбуваються в театрі theaterId.

Метод IEnumerable<Performance> GetCurrentByTheater(int theaterId) повертає театральні події, що відбуватимуться в театрі theaterId.

Метод IEnumerable<Performance> GetByStaff(int staffId, int? roleId) повертає театральні події, в яких бере участь особа staffId в ролі roleId, а якщо значення roleId пропущено, то за всіма ролями.

Метод IEnumerable<Performance> GetByTag(int tagId) повертає театральні події, відмічені тегом tagId.

Метод IEnumerable<Performance> GetByCategory(int categId) повертає театральні події, що стосуються категорії categId.

Метод IEnumerable<Performance> GetCurrentByHall(int hallId) повертає театральні події, що відбуватимуться в залі hallId.

Метод Task Add(Performance perf) вносить театральну подію perf.

Метод Task Update(Performance perf) оновлює театральну подію perf.

Метод Performance GetLast() повертає останню театральну подію.

Метод `IEnumerable<Performance> Search (string perf_name, short age, TimeSpan minduration, TimeSpan maxduration, int? cityId, int? theaterId, float minstars, float maxstars)` виконує пошук театральних подій, які мають в назві рядок `perf_name`, доступні для людей віку від `age` років, мають мінімальну тривалість `minduration`, максимальну тривалість `maxduration`, відбуваються в місті `cityId` (може бути пропущено), відбуваються в театрі `theaterId` (може бути пропущено), мають оцінку від `minstars` до `maxstars`.

Клас сервісу `TheaterService` дозволяє виконувати базові дії щодо взаємодії з БД з театрами, забезпечуючи для цього ряд методів.

Метод `Theater GetById (int theaterId)` повертає повні дані театру `theaterId`.

Метод `IEnumerable<Theater> GetAll()` повертає перелік всіх театрів.

Метод `IEnumerable<Theater> GetByCity(int cityId)` повертає театри з міста `cityId`.

Метод `Task Add(Theater t)` вносить дані театру `t`.

Метод `Task Update(Theater t)` оновлює дані театру `t`.

Метод `Theater GetLast()` повертає останній внесений театр.

Клас `ExpertReviewController` реалізує зв'язування представлень з сервісами через роботу контролера експертної рецензії, тобто заповнює даними представлення, що стосуються експертних рецензій.

Метод, який повертає представлення перегляду експертної рецензії з номером `erId`, – це метод `Index(int erId)`.

Метод, який обробляє представлення створення експертної рецензії, – це метод `Create([Bind("ExpertId, Text, Stars, PerformanceId")] ExpertReview ereview)`.

Метод, який обробляє представлення редагування експертної рецензії, – це метод `Edit(int ereviewId, [Bind("Id, ExpertId, Text, Stars, PerformanceId")] ExpertReview ereview)`.

Метод, який повертає представлення перегляду експертних рецензій за театральною подією з номером `perf`, – це метод `PerformanceExpertReviews(int perf)`.

Метод, який повертає представлення перегляду експертних рецензій за експертом з номером `expId`, що є автором рецензії, – це метод `ExpertReviews(int expId)`.

Метод, який повертає представлення перегляду експертних рецензій, внесених редактором, який авторизувався в системі, – це метод `EditorReviews()`.

Клас `VisitorReviewController` реалізує зв'язування представлень з сервісами через роботу контролера рецензії відвідувача, тобто заповнює даними представлення, що стосуються рецензій відвідувача.

Метод, який повертає представлення перегляду рецензії відвідувача з номером рецензії `erId`, – це метод `Index(int vrId)`.

Метод, який обробляє представлення створення рецензії відвідувача, – це метод `Create([Bind("ExpertId, Text, Stars, PerformanceId")] ExpertReview ereview)`.

Метод, який обробляє представлення редагування рецензії відвідувача, – це метод `Edit(int vreviewId, [Bind("Id, AddText")] VisitorReview vreview)`.

Метод, який повертає представлення перегляду рецензій відвідувачів за театральною подією з номером `perf`, – це метод `PerformanceVisitorReviews(int perf)`.

Метод, який повертає представлення перегляду власних рецензій авторизованого відвідувача, – це метод `OwnVisitorReviews()`.

Метод, який повертає представлення перегляду рецензій відвідувачів за відвідувачем з номером `visitorId`, – це метод `VisitorReviews(int visitorId)`.

Клас `TalkController` реалізує зв'язування представлень з сервісами через роботу контролера обговорення театральної події, тобто заповнює даними представлення, що стосуються обговорення.

Метод, який повертає представлення перегляду обговорення з номером talkId, – це метод `Index(int talkId)`.

Метод, який обробляє представлення створення обговорення, – це метод `Create([Bind("Title, Site, Video, Description, PerformanceId")] Talk talk)`.

Метод, який обробляє представлення редагування обговорення, – це метод `Edit(int talkId, [Bind("Id, Title, Site, Video, Description, PerformanceId")] Talk talk)`.

Метод, який повертає представлення перегляду обговорення за театральною подією з номером perf, – це метод `PerformanceTalks(int perf)`.

Клас `HallController` реалізує зв'язування представлень з сервісами через роботу контролера зали театру, тобто заповнює даними представлення, що стосуються зали.

Метод, який повертає представлення перегляду зали театру hallId, – це метод `Task<IActionResult> Index(int hallId)`.

Метод, який обробляє представлення створення зали театру, – це метод `Task<IActionResult> Create([Bind("Title, TheaterId, Description, Plan")] Hall hall)`.

Метод, який обробляє представлення редагування зали театру, – це метод `Edit(int hallId, [Bind("Title, TheaterId, Description, Plan")] Hall hall)`.

Клас `PerformanceController` реалізує зв'язування представлень з сервісами через роботу контролера театральної події, тобто заповнює даними представлення, що стосуються цієї події.

Метод, який повертає представлення перегляду зали театральної події perfId, – це метод `Task<IActionResult> Index(int perfId)`.

Метод, який обробляє представлення створення театральної події, – це метод `Task<IActionResult> Create([Bind("Name, Description, AgeRestriction, Duration, FirstPerformance, LastDay, Content, HallId")] Performance perf)`.

Метод, який обробляє представлення редагування театральної події, – це метод `Edit(int perfId, [Bind("Id, Name, Description, AgeRestriction, Duration, FirstPerformance, LastDay, Content, HallId")] Performance perf)`.

Метод, який повертає представлення перегляду театральних подій, що відбуваються в театрі theaterId, – це метод `IActionResult TheaterPerformances (int theaterId)`.

Метод, який повертає представлення перегляду театральних подій, в яких бере участь особа staffId в ролі roleId, – це метод `IActionResult StaffPerformances (int staffId, int? roleId)`.

Метод, який повертає представлення перегляду театральних подій, відмічених тегом tagId, – це метод `IActionResult TagPerformances (int tagId)`.

Метод, який повертає представлення перегляду театральних подій, що стосуються категорії categId, – це метод `IActionResult CategoryPerformances (int categId)`.

Метод, який повертає представлення перегляду театральних подій, що відбуватимуться в залі hallId, – це метод `IActionResult HallCurrentPerformances (int hallId)`.

Метод, який повертає представлення пошуку театральних подій, які мають в назві рядок perf_name, доступні для людей віку від age років, мають мінімальну тривалість minduration, максимальну тривалість maxduration, відбуваються в місті cityId, відбуваються в театрі theaterId, мають оцінку від minstars до maxstars, – це метод `IActionResult SearchPerformances (string perf_name, short age, TimeSpan minduration, TimeSpan maxduration, float minstars, float maxstars, int? cityId, int? theaterId)`.

3.3 Висновки за розділом 3

Реалізацію програми виконано за допомогою використання фреймворку ASP.NET та фреймворку для роботи з даними Entity Framework Core. Описано створені класи. Використання цих технологій дозволило зменшити кількість коду, яку потрібно для роботи з даними в БД, враховуючи також застосування технології LINQ.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма, яка розробляється, призначена для отримання широкого масиву інформації про театральні події в різних театрах спільноти та обмінювання думками про події, які вже відбулись.

4.2 Умови виконання програми

Апаратне забезпечення роботи сервера повинно включати мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

4.3 Експлуатація програми

Програма фактично може взаємодіяти з користувачем в трьох режимах. Два режими мають багато спільного і призначені для відвідувача і експерта. Вони можуть виконувати подібний набір дій, але експерт створює експертні рецензії, маючи таким чином привілейований статус, а відвідувач створює звичайні рецензії, які враховуються на рівні зі всіма іншими рецензіями інших відвідувачів.

Для звернення в цьому режимі потрібно лише ввести стандартну адресу вебсайта, а тоді відобразиться одразу сторінка пошуку театральних подій, яку буде описано нижче. З цієї сторінки, а також з будь-якої іншої, де відображається перелік театральних подій, можна перейти до сторінки деталізації театральної події (рис. 4.1).

На сторінці деталізації театральної події відображається назва події, її опис, основна фотографія (нижче розташовано набір інших фотографій, розташованих в папці п'єси на сервері, якщо вони там є), вікові обмеження,

театр, де відбувається подія, зала театру, дата прем'єри, дата останнього показу, якщо вона визначена, тривалість, експертна оцінка, глядацька оцінка (нижче на сторінці розташовані відгуки відповідно окремо експертів і окремо звичайних відвідувачів), категорія події, інформація про обмеження контенту, послідовність тегів. За допомогою кнопки «Вартість квитків» можна відобразити вартість квитків на театральну подію, обираючи при цьому потрібну дату зі всіх доступних. Також на сторінці окремо відображається таблиця з датами виступів.

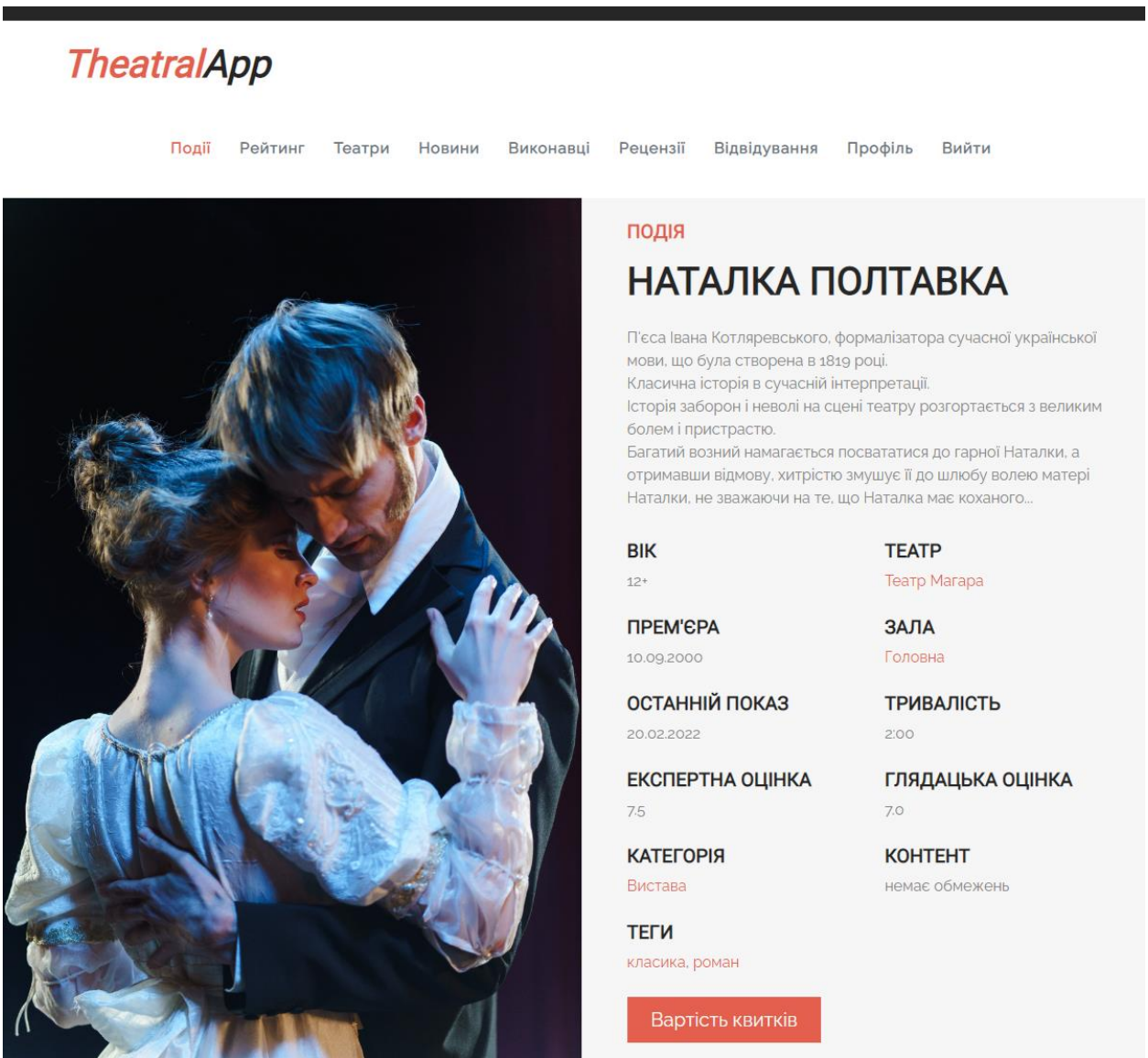


Рисунок 4.1 – Деталізація театральної події

Під фотографією доступні кнопки «Додати рецензію» та «Відвідував», що дозволяють відповідно створити власну рецензію про подію та вказати, що цю подію власноруч відвідав користувач.

Це фактично довга сторінка, яка містить різну інформацію про виставу, якою можна переміщатися за допомогою окремих кнопок. Зокрема на сторінці міститься перелік виконавців (рис. 4.2). Перелік виконавців може включати режисера, сценариста, акторів, технічних працівників, усіх задіяних у виставі і визначених відповідним чином редактором.

ВИКОНАВЦІ

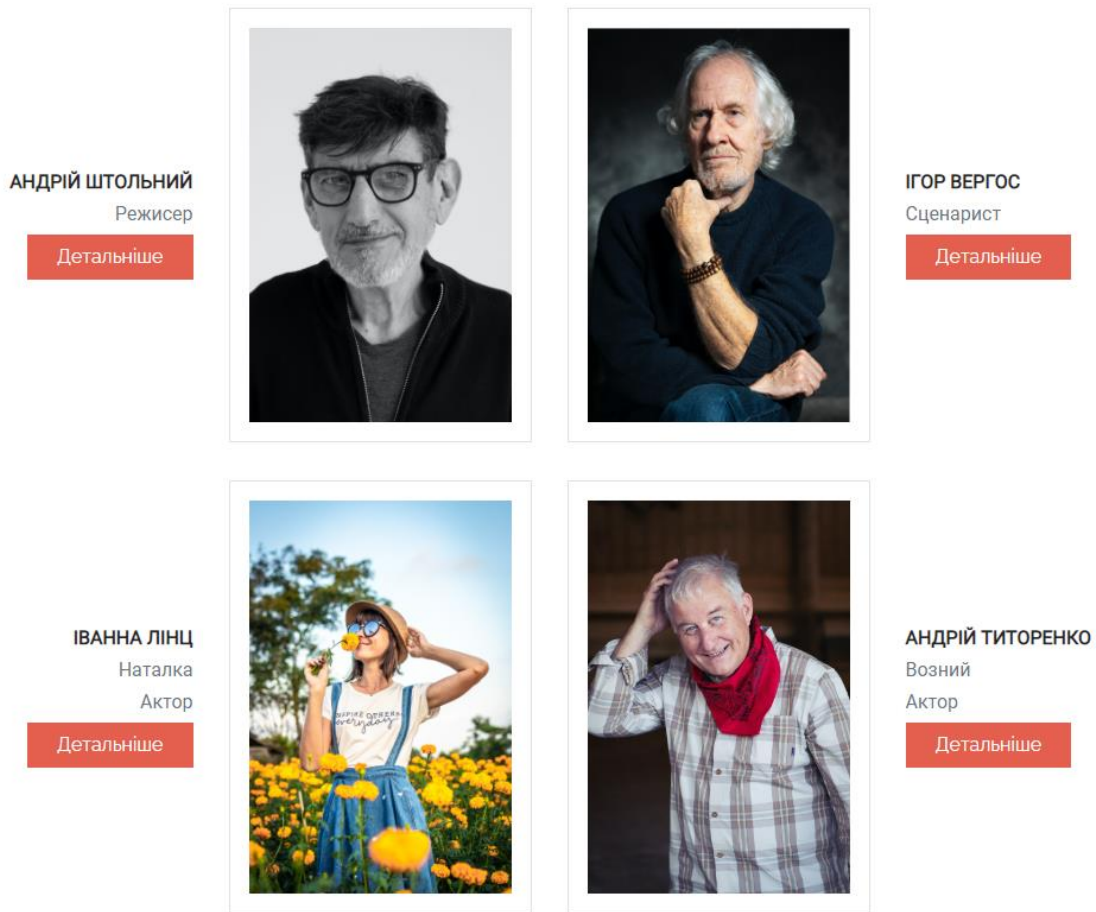


Рисунок 4.2 – Виконавці театральної події

За кожним виконавцем відображається його ім'я, фотографія, роль, яку він виконував у події, а також (для акторів), сценарна роль. Для кожного виконавця доступна кнопка «Детальніше», що дозволяє відкрити сторінку

виконавця і на ній переглянути детальніше інформацію про нього, зокрема про події, в яких він брав участь.

Розділ меню «Виконавці» дозволяє також виконати пошук серед всіх осіб, які могли бути виконавцями в театральних подіях.

Окремий розділ «Рейтинг» надає можливість переглянути рейтинг серед подій.

Розділ «Події» дозволяє виконати пошук серед всіх подій (рис. 4.3).

ПОШУК ПОДІЙ

Місто Не визначено ▾

Вік 15+ ▾

Мінімальна тривалість 30 хвилин ▾

Оцінка від 7 ▾

Театр Не визначено ▾

Максимальна тривалість 2 години ▾

Оцінка до 10 ▾

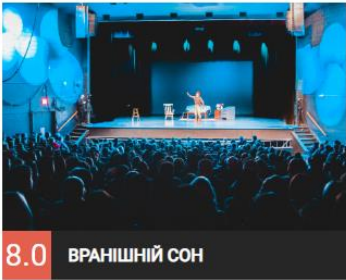
РЕЗУЛЬТАТИ ПОШУКУ



8.0
МАЛЬВІНА

ТЕАТР МАРІЇ ЗАНЬКОВЕЦЬКОЇ

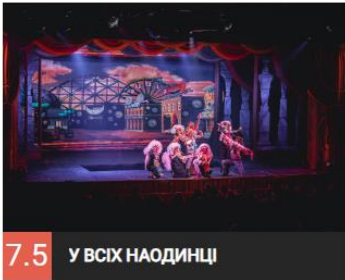
Детальніше



8.0
ВРАНІШНІЙ СОН

ТЕАТР ФРАНКА

Детальніше



7.5
У ВСІХ НАОДИНЦІ

ТЕАТР МАРІЇ ЗАНЬКОВЕЦЬКОЇ

Детальніше

Рисунок 4.3 – Пошук театральних подій

Для пошуку потрібно спочатку визначити параметри пошуку, що включають назву театральної події, місто, в якому відбувається подія, театр,

вказати мінімальний потрібний вік відвідувача, мінімальну та максимальну тривалість, мінімальну та максимальну оцінки, а далі натиснути на кнопку «Шукати».

Після цього відвідувачу відобразяться результати пошуку, що включають зображення події, назву події, театр та середню експертну оцінку події. За допомогою кнопки «Детальніше» можна відкрити сторінку відповідної події.

Розділ «Театри» дозволяє переглянути театри за містом, яке обирається з переліку.

Розділ «Новини» дозволяє переглянути всі доступні новини. Також новини відображаються на деталізованій сторінці театральної події з можливістю переходу до всіх новин події, а так само театру.

Розділ «Рецензії» дозволяє створювати власні рецензії і переглядати власні вже створені рецензії.

Розділ «Відвідування» дозволяє переглянути всі вистави, на яких був користувач. Додавати сюди дані можна зі сторінок самих подій.

Окрім того програма надає окремий інтерфейс для редактора. Він доступний за окремою адресою і не суміщений з інтерфейсом відвідувача. Для роботи з цим інтерфейсом одразу потрібно авторизуватися. Тоді відкриється сторінка, що складається з розділів:

- «Події», звідки є можливість створення подій і редагування існуючих, а зі сторінок окремих подій доступні можливості переходу до редагування вартості, розкладу виступів, учасників події;
- «Новини», що дозволяє вносити нові новини;
- «Виконавці», що дозволяє вносити дані виконавців;
- «Рецензії», звідки можна переглянути рецензії за експертами, а також внести нову рецензію експерта (рис. 4.4).

При внесенні нової рецензії можна задати експерта, якщо він не захотів рецензію вносити вручну самотійно, оцінку та текст рецензії. Потім рецензію можна відредагувати. Відвідувач свою рецензію може

відредагувати тільки шляхом додавання нового коментаря до рецензії, тобто змінити оцінку, текст, на відміну від експерта, він не може.

TheatralApp

Події Театри Новини Виконавці **Рецензії** Профіль Вийти

ЕКСПЕРТНИЙ ВІДГУК
Наталка Полтавка

Віктор Зленко

Оцінка

Текст рецензії

Зберегти

Рисунок 4.4 – Створення експертної рецензії

Розділ «Театри» редактору доступний тільки для перегляду театрів.

4.4 Тестування програми

У відповідності з технічним завданням був створений набір функцій, для кожної з яких було в кінці виконано виклик і перевірку отриманого результату, реалізуючи ці дії щонайменше раз на кожну функцію.

При перевірці співставлявся очікуваний результат з отриманим. Усі результати в кінцевому варіанті співпали. Тому тестування в підсумку вважалось успішним, а значить і реалізація програми в підсумку виявилась успішною.

Чеклист тестування програми представлений в табл. 4.1.

Таблиця 4.1 – Чеклист тестування програми

Перевірка	Результат
Перегляд переліку театрів у місті	+
Перегляд рейтингу вистав за категорією подій	+
Створення розкладу театральної події	+
Редагування розкладу театральної події	+
Створення рецензії відвідувача про театральну подію	+
Створення рецензії експерта про театральну подію	+
Коригування рецензії відвідувачем	+
Перегляд ролей виконавця	+
Внесення виконавців	+
Внесення даних нового театру	+
Редагування даних театру	+
Редагування експертної рецензії	+
Внесення події в історію відвідування	+
Перегляд власної історії відвідування подій	+
Внесення обговорення за подією	+
Визначення вартості квитків на подію	+
Пошук театральних подій	+
Перегляд театральних подій за тегом	+
Перегляд стрічки новин	+
Пошук виконавців	+
Перегляд рецензій за автором	+
Деталізація театральної події	+
Деталізація театру	+
Створення новини	+
Створення театральної події	+
Редагування театральної події	+
Авторизація, реєстрація	+ / +

4.5 Висновки за розділом 4

Програма має окремі інтерфейси для роботи відвідувачів і експертів та для роботи редакторів. Редактори можуть вносити основну інформацію щодо театральних подій. Відвідувачі можуть переглядати, шукати інформацію, мають велику кількість зв'язків між сторінками для зручної навігації.

ВИСНОВКИ

За результатом роботи розроблено ПЗ платформи для відвідувачів театральних подій.

Аналіз стану театральної галузі в Україні дозволив виявити відсутність міжтеатральної платформи, що разом з аналізом існуючих платформ в інших країнах дозволило визначити функціональні вимоги до програми. В якості особливостей розробки визначена наявність історії відвідування, направленість на театри в Україні а також можливість концентрації інформації щодо виконавців, які беруть участь в театральних подіях (акторів, сценаристів тощо). На основі визначених функціональних вимог до програми було створено діаграми прецедентів роботи редактора, відвідувача та експерта.

Проектування програми дозволило створити структуру основних сутностей предметної області, визначити відповідні таблиці реляційної бази даних, визначити зв'язки між ними. До комплексу засобів проектування і розробки було внесено мову програмування C#, вебфреймворк ASP.NET Core, шаблон проектування Model-View-Controller, що дозволив визначити структуру вебсистеми, система Microsoft SQL Server, фреймворк для роботи з даними Entity Framework Core.

Результати проектування були використані під час реалізації ПЗ. Результати реалізації описані представленням відповідних розроблених класів. Реалізація було виконана успішно, що підтверджено тестуванням реалізованих функцій.

Створена вебсистема дозволила реалізувати можливості перегляду театрів у місті, театральних подій за категоріями, тегами, виконавцем, створення розкладу театральних подій, рецензій про театральні події з можливостями їх коригування, перегляду і пошуку театральних подій у театрі або окремій залу театру, в тому числі всіх або майбутніх, визначення історії відвідування театральних подій, перегляду новин.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Theater – The New York Times [Electronic resource]. – Access mode : <https://www.nytimes.com/reviews/theater>.
2. London Theatre Direct – London Shows. London Theatre Direct [Electronic resource]. – Access mode : <https://www.londontheatredirect.com/>.
3. London Theatre Tickets. West End News [Electronic resource]. – Access mode : <https://www.londontheatre.co.uk/>.
4. Театр Заньковецької [Електрон. ресурс]. – Режим доступу : <https://zankovetska.com.ua/>.
5. Головна. Театр Магара [Електрон. ресурс]. – Режим доступу : <https://magara.zp.ua/>.
6. Nagel, Ch. Professional C# and .NET [Text] / Ch. Nagel. – Birmingham : Wrox, 2021. – 1008 p.
7. C# Programming Language [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/csharp-programming-language/>.
8. Schildt, H. Java [Text] : The Complete Reference / H. Schildt. – New York : McGraw Hill, 2021. – 1280 p.
9. Duldulao, D. B. ASP.NET Core and Vue.js [Text] : Build real-world, scalable, full-stack applications using Vue.js 3, TypeScript, .NET 5, and Azure / D. B. Duldulao. – Birmingham : Packt Publishing, 2021. – 478 p.
10. Lock, A. ASP.NET Core in Action, Second Edition [Text] / A. Lock. – New York : Manning, 2021. – 832 p.
11. Introducing Microsoft SQL Server 2019 [Text] : Reliability, scalability, and security both on premises and in the cloud / Kellyn Gorman, Allan Hirt, Dave Noderer et al. – Birmingham : Packt Publishing, 2020. – 488 p.
12. MVC : [Electronic resource] Model, View, Controller. Codecademy. – Access mode : <https://www.codecademy.com/article/mvc>.
13. Hernandez, R. D. The Model View Controller Pattern – MVC Architecture and Frameworks Explained [Electronic resource] / R. D. Hernandez. –

Access mode : <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>.

14. What is Entity Framework? [Electronic resource]. – Access mode : <https://www.entityframeworktutorial.net/what-is-entityframework.aspx>.

ДОДАТОК А
Технічне завдання

Вступ

Театральне життя є одним з найбільш класичних проявів культури, при чому як в її класичному, так і сучасному вираженні. Театри привертають увагу і як об'єкти історії, архітектури, взагалі прояви стану життя в місті, країні. В сучасному світі вони потребують деякої платформи, де в разі її успішності буде створено відповідну спільноту для любителів театру. У різних країнах такі платформи має широке поширення, але в Україні з цим є проблеми. Тому створення такої системи є важливим для розвитку галузі за рахунок синергії.

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Програмне забезпечення платформи для відвідувачів театральних подій» у відповідності з наказом № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма, яка розробляється, призначена для отримання широкого масиву інформації про театральні події в різних театрах спільноти та обмінювання думками про події, які вже відбулись.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Користувач програми повинен бути забезпечений наступними функціями:

- перегляд переліку театрів у місті;

- перегляд рейтингу подій за категоріями подій;
- створення і редагування розкладу театральних подій;
- створення рецензій про театральну подію;
- коригування рецензії автором;
- перегляд переліку театральних подій у театрі або окремій залу театру, в тому числі всіх або майбутніх;
- перегляд історії відвідування театральних подій;
- внесення обговорень за подіями;
- визначення вартості квитків на подію;
- пошук театральних подій;
- перегляд переліку театральних подій за тегом;
- перегляд стрічки новин, в тому числі за подією або за театром;
- пошук виконавців;
- перегляд ролей виконавця;
- перегляд рецензій за автором;
- деталізація театральної події;
- деталізація театру;
- створення новини;
- створення і редагування театральних подій;
- створення і редагування даних театрів;
- внесення виконавців;
- авторизація і реєстрація.

А.3.2 Умови експлуатації

Для експлуатації програмного забезпечення потрібно забезпечити окремий сервер, на якому буде запущено вебсервер і сервер бази даних для постійного доступу користувачів до сервісу бронювання паркувальних місць.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для забезпечення підтримки роботи вебсервера та сервера бази даних необхідно надати апаратне забезпечення, що включає мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

А.4 Порядок контролю та приймання

Для контролю виконання дипломної кваліфікаційної роботи повинен використовуватися відповідний календарний план, який затверджено викладачем у завданні на роботу. Приймання роботи відбувається на основі проведення процедури захисту роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Performance.cs

```

using System;
using System.Collections.Generic;

namespace TheatralApp.Data.Models
{
    public class Performance
    {
        public int Id //ідентифікатор театральної події
        { get; set; }
        public string Name //назва театральної події
        { get; set; }
        public string Description //опис театральної події
        { get; set; }
        public short AgeRestriction //обмеження віку для
        події
        { get; set; }
        public TimeSpan Duration //тривалість вистави
        { get; set; }
        public DateTime FirstPerformance //час, коли
        відбулась прем'єра
        { get; set; }
        public DateTime LastDay //час останнього
        запланованого виступу
        { get; set; }
        public string Content //змістовні особливості
        театральної події
        { get; set; }
        public float ExpertStars //середня оцінка експертів
        { get; set; }
        public float Stars //середня оцінка відвідувачів
        { get; set; }
        public int HallId //номер зали театру, де має
        відбуватися подія
        { get; set; }
        public virtual Hall Hall //зала театру, де має
        відбуватися подія
        { get; set; }
        public virtual ICollection<PerformanceStaff>
        PerformanceStaffs //виконавці театральної події
        { get; set; }
        public virtual ICollection<Category>
        PerformanceCategories //категорії театральної події
        { get; set; }
        public virtual ICollection<Talk> Talks //обговорення
        театральної події
    }
}

```

```

        { get; set; }
        public virtual ICollection<PerformanceTime>
PerformanceTimes //час вистав за театральною подією
        { get; set; }
        public virtual ICollection<PerformancePlaceCategory>
PerformancePlaceCategories //категорії місць на театральній події
        { get; set; }
        public virtual ICollection<Visit> Visits //історія
візитів на подію
        { get; set; }
        public virtual ICollection<SearchTag> PerformanceTags
//теги події
        { get; set; }
        public virtual ICollection<ExpertReview>
ExpertReviews //експертні рецензії
        { get; set; }
        public virtual ICollection<VisitorReview>
VisitorReviews //рецензії відвідувачів
        { get; set; }
        public virtual ICollection<News> PerformanceNews
//новини про подію
        { get; set; }
    }
}

```

Б.2 Текст файла Talk.cs

```

using System;
using System.Collections.Generic;

namespace TheatralApp.Data.Models
{
    public class Talk
    {
        public int Id //ідентифікатор обговорення
        { get; set; }
        public string Title //заголовок обговорення
        { get; set; }
        public string Site //посилання на обговорення
        { get; set; }
        public string Video //посилання на файл з відео
обговорення
        { get; set; }
        public string Description //опис обговорення
        { get; set; }
        public int PerformanceId //номер театральної події,
якої стосується обговорення
    }
}

```

```

        { get; set; }
        public virtual Performance Performance //театральна
        подія, якої стосується обговорення
        { get; set; }
    }
}

```

Б.3 Текст файла ExpertReview.cs

```

using System;
using System.Collections.Generic;

namespace TheatralApp.Data.Models
{
    public class ExpertReview
    {
        public int Id //ідентифікатор експертної рецензії
        { get; set; }
        public string Text //повний текст рецензії
        { get; set; }
        public DateTime Day //час, коли створена рецензія
        { get; set; }
        public short Stars //оцінка, виставлена в рецензії
        { get; set; }
        public int ExpertId //номер експерта, який написав
        рецензію
        { get; set; }
        public virtual Expert Expert //експерт, який написав
        рецензію
        { get; set; }
        public int EditorId //номер редактора, який вніс
        рецензію
        { get; set; }
        public virtual Editor Editor //редактор, який вніс
        рецензію
        { get; set; }
        public int PerformanceId //номер театральної події,
        про яку рецензія
        { get; set; }
        public virtual Performance Performance //театральна,
        подія, про яку рецензія
        { get; set; }
    }
}

```

Б.4 Текст файла VisitorReview.cs

```

using System;
using System.Collections.Generic;

namespace TheatralApp.Data.Models
{
    public class VisitorReview
    {
        public int Id //ідентифікатор рецензії відвідувача
        { get; set; }
        public string Text //повний текст рецензії
        { get; set; }
        public DateTime Day //час, коли створена рецензія
        { get; set; }
        public short Stars //оцінка, виставлена в рецензії
        { get; set; }
        public int VisitorId //номер відвідувача, який
написав рецензію
        { get; set; }
        public virtual Visitor Visitor //відвідувач, який
написав рецензію
        { get; set; }
        public int PerformanceId //номер театральної події,
про яку рецензія
        { get; set; }
        public virtual Performance Performance //театральна,
подія, про яку рецензія
        { get; set; }
    }
}

```

Б.5 Текст файла PerformanceService.cs

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Globalization;
using TheatralApp.Data;
using TheatralApp.Data.Interfaces;
using TheatralApp.Data.Models;

```

```

namespace TheatralApp.Service
{
    public class PerformanceService : IPerformance
    {
        private readonly DataContext _dbTheatral;

        public PerformanceService (DataContext dbTheatral)
        {
            _dbTheatral = dbTheatral;
        }

        //повертає повні дані театральної події за її номером
perform
        public Performance GetById (int perform)
        {
            return _dbTheatral.Performances.Where(perf =>
perform.Id == perform)
                .Include(perf => perf.Hall).ThenInclude(h =>
h.Theater).ThenInclude(c => c.City)
                .Include(perf =>
perform.PerformanceStaffs).ThenInclude(s => s.Staff)
                .Include(perf =>
perform.PerformanceStaffs).ThenInclude(sr => sr.StaffRole)
                .Include(perf => perf.PerformanceCategories)
                .Include(perf => perf.Talks)
                .Include(perf =>
perform.PerformanceTimes).ThenInclude(pt => pt
PerformancePrices).ThenInclude(pps => pps.PriceSchemeGroup)
                .Include(perf =>
perform.PerformancePlaceCategories)
                .Include(perf =>
perform.PerformanceTags).ThenInclude(pt => pt.SearchTag)
                .Include(perf =>
perform.ExpertReviews).ThenInclude(er => er.Expert)
                .Include(perf =>
perform.VisitorReviews).ThenInclude(vr => vr.Visitor)
                .Include(perf =>
perform.PerformanceNews).ThenInclude(pns => pns.News)
                .First();
        }

        //повертає перелік всіх театральних подій
        public IEnumerable<Performance> GetAll()
        {
            return _dbTheatral.Performances
                .Include(perf => perf.Hall).ThenInclude(h =>
h.Theater).ThenInclude(c => c.City)
                .Include(perf => perf.PerformanceCategories)

```

```

        .Include(perf =>
perf.PerformanceTags).ThenInclude(pt => pt.SearchTag);
    }

    //повертає театральні події, що відбуваються в театрі
theaterId
    public IEnumerable<Performance> GetByTheater(int
theaterId)
    {
        return _dbTheatral.Performances
            .Where(perf => perf.Hall.Theater.Id ==
theaterId)
            .Include(perf => perf.Hall).ThenInclude(h =>
h.Theater).ThenInclude(c => c.City)
            .Include(perf => perf.PerformanceCategories)
            .Include(perf =>
perf.PerformanceTags).ThenInclude(pt => pt.SearchTag)
            .OrderByDescending(perf => perf.Stars);
    }

    //повертає театральні події, що відбуватимуться в
театрі theaterId
    public IEnumerable<Performance>
GetCurrentByTheater(int theaterId)
    {
        DateTime limit = DateTime.Now;
        return _dbTheatral.Performances
            .Where(perf => perf.LastDay >= limit)
            .Where(perf => perf.Hall.Theater.Id ==
theaterId)
            .Include(perf => perf.Hall).ThenInclude(h =>
h.Theater).ThenInclude(c => c.City)
            .Include(perf => perf.PerformanceCategories)
            .Include(perf =>
perf.PerformanceTags).ThenInclude(pt => pt.SearchTag)
            .OrderByDescending(perf => perf.Stars);
    }

    //повертає театральні події, в яких бере участь особа
staffId в ролі roleId, а якщо значення roleId пропущено, то за
всіма ролями
    public IEnumerable<Performance> GetByStaff(int
staffId, int? roleId)
    {
        if (roleId)
        {
            return _dbTheatral.Performances

```

```

        .Where(perf =>
perf.PerformanceStaffs.Any(a => a.Staff.Id == staffId,
a.StaffRole.Id == roleId)
        .Include(perf => perf.Hall).ThenInclude(h
=> h.Theater).ThenInclude(c => c.City)
        .OrderByDescending(perf =>
perf.FirstPerformance);
    }
    else
    {
        return _dbTheatral.Performances
        .Where(perf =>
perf.PerformanceStaffs.Any(a => a.Staff.Id == staffId)
        .Include(perf => perf.Hall).ThenInclude(h
=> h.Theater).ThenInclude(c => c.City)
        .OrderByDescending(perf =>
perf.FirstPerformance);
    }
}

//повертає театральні події, відмічені тегом tagId
public IEnumerable<Performance> GetByTag(int tagId)
{
    DateTime limit = DateTime.Now;
    return _dbTheatral.Performances
        .Where(perf => perf.PerformanceTags.Any(a =>
a.Id == tagId)
        .Where(perf => perf.LastDay >= limit)
        .Include(perf => perf.Hall).ThenInclude(h =>
h.Theater).ThenInclude(c => c.City)
        .OrderByDescending(perf => perf.Stars);
}

//повертає театральні події, що стосуються категорії
categId
public IEnumerable<Performance> GetByCategory(int
categId)
{
    DateTime limit = DateTime.Now;
    return _dbTheatral.Performances
        .Where(perf =>
perf.PerformanceCategories.Any(a => a.Id == categId)
        .Where(perf => perf.LastDay >= limit)
        .Include(perf => perf.Hall).ThenInclude(h =>
h.Theater).ThenInclude(c => c.City)
        .OrderByDescending(perf => perf.Stars);
}

```

```

        //повертає театральні події, що відбуватимуться в
зали hallId
        public IEnumerable<Performance> GetCurrentByHall(int
hallId)
        {
            DateTime limit = DateTime.Now;
            return _dbTheatral.Performances
                .Where(perf => perf.LastDay >= limit)
                .Where(perf => perf.Hall.Id == hallId)
                .Include(perf => perf.Hall).ThenInclude(h =>
h.Theater).ThenInclude(c => c.City)
                .Include(perf => perf.PerformanceCategories)
                .Include(perf =>
perf.PerformanceTags).ThenInclude(pt => pt.SearchTag)
                .OrderByDescending(perf => perf.Stars);
        }

        //вносить театральну подію perf
        public async Task Add(Performance perf)
        {
            _dbTheatral.Add(perf);
            await _dbTheatral.SaveChangesAsync();
        }

        //оновлює театральну подію perf
        public async Task Update(Performance perf)
        {
            _dbTheatral.Update(perf);
            await _dbTheatral.SaveChangesAsync();
        }

        //повертає останню театральну подію
        public Performance GetLast()
        {
            return GetAll().Last();
        }

        //виконує пошук театральних подій, які мають в назві
рядок perf_name, доступні для людей віку від age років, мають
мінімальну тривалість minduration, максимальну тривалість
maxduration, відбуваються в місті cityId (може бути пропущено),
відбуваються в театрі theaterId (може бути пропущено), мають оцінку
від minstars до maxstars
        public IEnumerable<Performance> Search (string
perf_name, short age, TimeSpan minduration, TimeSpan maxduration,
int? cityId, int? theaterId, float minstars, float maxstars)
        {
            var prefs = _dbTheatral.Performances

```

```

        .Where(perf
=> (perf.AgeRestriction <= age)
    && (perf.Name.Contains(perf_name))
    && (perf.Duration >= minduration)
    && (perf.Duration <= maxduration)
    && (perf.Stars <= minstars)
    && (perf.Stars >= maxstars)
    && (perf.ExpertStars <= minstars)
    && (perf.ExpertStars >= maxstars);
    if (cityId)
    {
        prefs = prefs.Where(perf =>
(pref.Hall.Theater.City.Id == cityId));
    }
    if (theaterId)
    {
        prefs = prefs.Where(perf =>
(perf.Hall.Theater.Id == theaterId));
    }
    if (perf_name)
    {
        prefs = prefs.Where(perf =>
(perf.Contains(perf_name)));
    }
    }
}
}

```

Б.6 Текст файла ExpertReviewController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Threading.Tasks;
using TheatralApp.Data.Models.ExpertReview;
using TheatralApp.Data.Models.Expert;
using TheatralApp.Data.Models.Editor;
using Microsoft.AspNet.Identity;

namespace TheatralApp.Controllers
{
    public class ExpertReviewController : Controller
    {
        private readonly IExpertReview _expertreviewService;
        private readonly IExpert _expertService;
    }
}

```

```

        private readonly IEditor _editorService;

        public ExpertReviewController (IExpertReview
expertreviewService, IExpert expertService, IEditor editorService)
        {
            _expertreviewService = expertreviewService;
            _expertService = expertService;
            _editorService = editorService;
        }

        //повертає представлення перегляду експертної
рецензії з номером erId
        public async Task<IActionResult> Index(int erId)
        {
            var expertreview = _expertreviewService.GetById
(erId);

            ViewData["review"] = expertreview;
            return View();
        }

        public IActionResult Create()
        //підготовка представлення редагування рецензії
експерта
        {
            ViewData["ExpertId"] = new
SelectList(_expertService.GetAll(), "Id", "PersonName");
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        //обробляє представлення створення експертної
рецензії
        public async Task<IActionResult>
Create([Bind("ExpertId, Text, Stars, PerformanceId")] ExpertReview
ereview)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    DateTime curr = DateTime.Now;
                    vreview.Day = curr;
                    ereview.EditorId =
_expertService.GetById(User.Identity.GetUserId());
                    _expertreviewService.Add(ereview);
                    return RedirectToAction(nameof(Index));
                }
            }
        }
    }

```

```

        catch (Exception ex)
        {
            return View();
        }
    }
    ViewData["ExpertId"] = new
SelectList(_expertService.GetAll(), "Id", "PersonName",
ereview.PerformanceId);
    return View(ereview);
}

public async Task<IActionResult> Edit(int ereviewId)
//підготовка представлення редагування рецензії
експерта
{
    if (ereviewId == null)
    {
        return NotFound();
    }

    var ereview = _expertreviewService.GetById
(ereviewId);

    if (ereview == null)
    {
        return NotFound();
    }
    ViewData["ExpertId"] = new
SelectList(_expertService.GetAll(), "Id", "PersonName");
    return View(ereview);
}

[HttpPost]
[ValidateAntiForgeryToken]
//обробляє представлення редагування експертної
рецензії
public async Task<IActionResult> Edit(int ereviewId,
[Bind("Id, ExpertId, Text, Stars, PerformanceId")] ExpertReview
ereview)
{
    if (ereviewId != ereview.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {

```

```

        ereview.EditorId =
_expertService.GetById(User.Identity.GetUserId());
        _expertreviewService.Update(ereview);
        return RedirectToAction("Index");
    }
    catch (DbUpdateConcurrencyException)
    {
        return NotFound();
    }
}
ViewData["ExpertId"] = new
SelectList(_expertService.GetAll(), "Id", "PersonName",
ereview.PerformanceId);
return View(ereview);
}

//повертає представлення перегляду експертних
рецензій за театральною подією з номером perf
public IActionResult PerformanceExpertReviews(int
perf)
{
    var ereviews =
_expertreviewService.GetByPerformance (perf);
    ViewData["ereviews"] = ereviews.ToList();
    return View();
}

//повертає представлення перегляду експертних
рецензій за експертом з номером expId, що є автором рецензії
public IActionResult ExpertReviews(int expId)
{
    var ereviews = _expertreviewService.GetByExpert
(expId);

    ViewData["ereviews"] = ereviews.ToList();
    return View();
}

//повертає представлення перегляду експертних
рецензій, внесених редактором, який авторизувався в системі
public IActionResult EditorReviews()
{
    var editor =
_editorService.GetById(User.Identity.GetUserId());
    var ereviews = _expertreviewService.GetByEditor
(editor.Id);

    ViewData["ereviews"] = ereviews.ToList();
    return View();
}

```

```
    }
}
```

Б.7 Текст файла VisitorReviewController.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Threading.Tasks;
using TheatralApp.Data.Models.VisitorReview;
using TheatralApp.Data.Models.Visitor;
using Microsoft.AspNet.Identity;

namespace TheatralApp.Controllers
{
    public class VisitorReviewController : Controller
    {
        private readonly IVisitorReview
        _visitorreviewService;
        private readonly IVisitor _visitorService;

        public VisitorReviewController (IVisitorReview
        visitorreviewService, IVisitor visitorService)
        {
            _visitorreviewService = visitorreviewService;
            _visitorService = visitorService;
        }

        //повертає представлення перегляду рецензії
        відвідувача з номером vrId
        public async Task<IActionResult> Index(int vrId)
        {
            var visitorreview = _visitorreviewService.GetById
            (vrId);

            ViewData["review"] = visitorreview;
            return View();
        }

        public IActionResult Create()
        //підготовка представлення створення рецензії
        відвідувача
        {
            return View();
        }
    }
}
```

```

[HttpPost]
[ValidateAntiForgeryToken]
//обробляє представлення створення рецензії
Відвідувача
    public async Task<IActionResult> Create([Bind("Text,
Stars, PerformanceId")] VisitorReview vreview)
    {
        if (ModelState.IsValid)
        {
            try
            {
                DateTime curr = DateTime.Now;
                vreview.Day = curr;
                vreview.VisitorId =
                _visitorService.GetById(User.Identity.GetUserId());
                _expertreviewService.Add(vreview);
                return RedirectToAction(nameof(Index));
            }
            catch (Exception ex)
            {
                return View();
            }
        }
        return View(vreview);
    }

    public async Task<IActionResult> Edit(int vreviewId)
    //підготовка представлення редагування рецензії
Відвідувача
    {
        if (vreviewId == null)
        {
            return NotFound();
        }

        var vreview = _visitorreviewService.GetById
        (vreviewId);

        if (vreview == null)
        {
            return NotFound();
        }
        return View(vreview);
    }

[HttpPost]
[ValidateAntiForgeryToken]
//обробляє представлення редагування рецензії
Відвідувача

```

```

        public async Task<IActionResult> Edit(int vreviewId,
[Bind("Id, AddText")] VisitorReview vreview)
        {
            if (vreviewId != vreview.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _visitorreviewService.Update(vreview);
                    return RedirectToAction("Index");
                }
                catch (DbUpdateConcurrencyException)
                {
                    return NotFound();
                }
            }
            return View(vreview);
        }

        //повертає представлення перегляду рецензій
        відвідувачів за театральною подією з номером perf
        public IActionResult PerformanceVisitorReviews(int
perf)
        {
            var vreviews =
            _visitorreviewService.GetByPerformance (perf);
            ViewData["vreviews"] = vreviews.ToList();
            return View();
        }

        //повертає представлення перегляду власних рецензій
        авторизованого відвідувача
        public IActionResult OwnVisitorReviews()
        {
            var vreviews = _visitorreviewService.GetByVisitor
            (_visitorService.GetById(User.Identity.GetUserId()));
            ViewData["vreviews"] = vreviews.ToList();
            return View();
        }

        //повертає представлення перегляду рецензій
        відвідувачів за відвідувачем з номером visitorId
        public IActionResult VisitorReviews(int visitorId)
        {

```

```

        var vreviews = _visitorreviewService.GetByVisitor
(visitorId);

        ViewData["vreviews"] = vreviews.ToList();
        return View();
    }
}
}

```

Б.8 Текст файла TalkController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Threading.Tasks;
using TheatralApp.Data.Models.Talk;

namespace TheatralApp.Controllers
{
    public class TalkController : Controller
    {
        private readonly ITalk _talkService;

        public TalkController (ITalk talkService)
        {
            _talkService = talkService;
        }

        //повертає представлення перегляду обговорення з
номером talkId
        public async Task<IActionResult> Index(int talkId)
        {
            var talk = _talkService.GetById (talkId);
            ViewData["talk"] = talk;
            return View();
        }

        public IActionResult Create()
        //підготовка представлення створення обговорення
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
    }
}

```

```

        //обробляє представлення створення обговорення
        public async Task<IActionResult> Create([Bind("Title,
Site, Video, Description, PerformanceId")] Talk talk)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    _talkService.Add(talk);
                    return RedirectToAction(nameof(Index));
                }
                catch (Exception ex)
                {
                    return View();
                }
            }
            return View(talk);
        }

        public async Task<IActionResult> Edit(int talkId)
        //підготовка представлення редагування обговорення
        {
            if (talkId == null)
            {
                return NotFound();
            }

            var talk = _talkService.GetById (talkId);
            if (talk == null)
            {
                return NotFound();
            }
            return View(talk);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        //обробляє представлення редагування обговорення
        public async Task<IActionResult> Edit(int talkId,
[Bind("Id, Title, Site, Video, Description, PerformanceId")] Talk
talk)
        {
            if (talkId != talk.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)

```

```

        {
            try
            {
                _talkService.Update(talk);
                return RedirectToAction("Index");
            }
            catch (DbUpdateConcurrencyException)
            {
                return NotFound();
            }
        }
        return View(vreview);
    }

    //повертає представлення перегляду обговорення за
    театральною подією з номером perf
    public IActionResult PerformanceTalks(int perf)
    {
        var talks = _talkService.GetByPerformance (perf);
        ViewData["talks"] = talks.ToList();
        return View();
    }
}

```

Б.9 Текст файла TheaterService.cs

```

using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Globalization;
using TheatralApp.Data;
using TheatralApp.Data.Interfaces;
using TheatralApp.Data.Models;

namespace TheatralApp.Service
{
    public class TheaterService : ITheater
    {
        private readonly DataContext _dbTheatral;

        public PerformanceService (DataContext dbTheatral)
        {

```

```

        _dbTheatral = dbTheatral;
    }

    //повертає повні дані театру theaterId
    public Theater GetById (int theaterId)
    {
        return _dbTheatral.Theaters.Where(t => t.Id ==
theaterId)
            .Include(t => t.City)
            .Include(t => t.Halls).ThenInclude(h =>
h.Performances)
            .Include(t => t.TheaterNews)
            .First();
    }

    //повертає перелік всіх театрів
    public IEnumerable<Theater> GetAll()
    {
        return _dbTheatral.Theaters
            .Include(t => t.City);
    }

    //повертає театри з міста cityId
    public IEnumerable<Theater> GetByCity(int cityId)
    {
        return _dbTheatral.Theaters
            .Include(t => t.City);
    }

    //вносить дані театру t
    public async Task Add(Theater t)
    {
        _dbTheatral.Add(t);
        await _dbTheatral.SaveChangesAsync();
    }

    //оновлює дані театру t
    public async Task Update(Theater t)
    {
        _dbTheatral.Update(t);
        await _dbTheatral.SaveChangesAsync();
    }

    //повертає останній внесений театр
    public Theater GetLast()
    {
        return GetAll().Last();
    }

```

```
    }
}
```

Б.10 Текст файла HallController.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Threading.Tasks;
using TheatralApp.Data.Models.Hall;
using TheatralApp.Data.Models.Performance;

namespace TheatralApp.Controllers
{
    public class HallController : Controller
    {
        private readonly IHall _hallService;
        private readonly IPerformance _performanceService;

        public HallController (IHall hallService,
IPerformance performanceService)
        {
            _hallService = hallService;
            _performanceService = performanceService;
        }

        //повертає представлення перегляду зали театру hallId
        public async Task<IActionResult> Index(int hallId)
        {
            var hall = _hallService.GetById (hallId);
            ViewData["hall"] = hall;
            ViewData["performances"] =
_performanceService.GetCurrentByHall (hallId).ToList();
            return View();
        }

        public IActionResult Create()
        //підготовка представлення створення зали театру
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
```

```

        //обробляє представлення створення зали театру
        public async Task<IActionResult> Create([Bind("Title,
TheaterId, Description, Plan")] Hall hall)
        {
            if (ModelState.IsValid)
            {
                try
                {
                    _hallService.Add(hall);
                    return RedirectToAction(nameof(Index));
                }
                catch (Exception ex)
                {
                    return View();
                }
            }
            return View(hall);
        }

        public async Task<IActionResult> Edit(int hallId)
        //підготовка представлення редагування зали театру
        {
            if (hallId == null)
            {
                return NotFound();
            }

            var hall = _hallService.GetById (hallId);
            if (hall == null)
            {
                return NotFound();
            }
            return View(hall);
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        //обробляє представлення редагування зали театру
        public async Task<IActionResult> Edit(int hallId,
[Bind("Id, Title, TheaterId, Description, Plan")] Hall hall)
        {
            if (hallId != hall.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {

```

```
        try
        {
            _hallService.Update(hall);
            return RedirectToAction("Index");
        }
        catch (DbUpdateConcurrencyException)
        {
            return NotFound();
        }
    }
    return View(hall);
}
}
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота бакалавра

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПЛАТФОРМИ ДЛЯ
ВІДВІДУВАЧІВ ТЕАТРАЛЬНИХ ПОДІЙ
SOFTWARE FOR THE PLATFORM FOR THEATRICAL
PERFORMANCE VISITORS**

Виконав

студент групи КНТз-219_2

О.О. Євсєєв

Керівник роботи

к.т.н., доцент

В.М. Льовкін

2022

Рисунок В.1 – Слайд перший

2

Об'єкт роботи – процес інформування про театральні події.

Предмет роботи – програмне забезпечення платформи для відвідувачів театральних подій.

Мета роботи – розробити програмне забезпечення для підтримки платформи відвідувачів театральних подій.

Завдання роботи:

- аналіз предметної області;
- визначення вимог до програмного забезпечення платформи для відвідувачів театральних подій;
- проектування програмного забезпечення платформи для відвідувачів театральних подій;
- реалізація і налагодження програмного забезпечення платформи для відвідувачів театральних подій.

Рисунок В.2 – Слайд другий

Порівняння програмних аналогів з виконуваною розробкою

Критерій	London Theatre Direct	London Theatre	TheatralApp
Країна розташування театрів	Велико-британія	Великобританія	Потенційно Україна
Пошук подій	Так	Так	Так
Сторінки виконавців	Ні	Ні	Так
Бронювання квитків	Так	Так	Ні
Відгуки користувачів	Так	Ні	Так
Театральні новини	Так	Ні	Так
Історія відвідувань	Ні	Ні	Так
Обліковий запис користувача	Ні	Ні	Так

Рисунок В.3 – Слайд третій

Вибір мови програмування для розробки

Критерій	C#	Java
Швидкість розробки	+-	+-
Підтримка спрощеної роботи з даними	+	-
Засоби веброзробки на основі фреймворків	+	+
Оптимізація швидкості виконання	+	+-
Можливість створення кросплатформного рішення	+	+

Рисунок В.4 – Слайд четвертий

Діаграми прецедентів

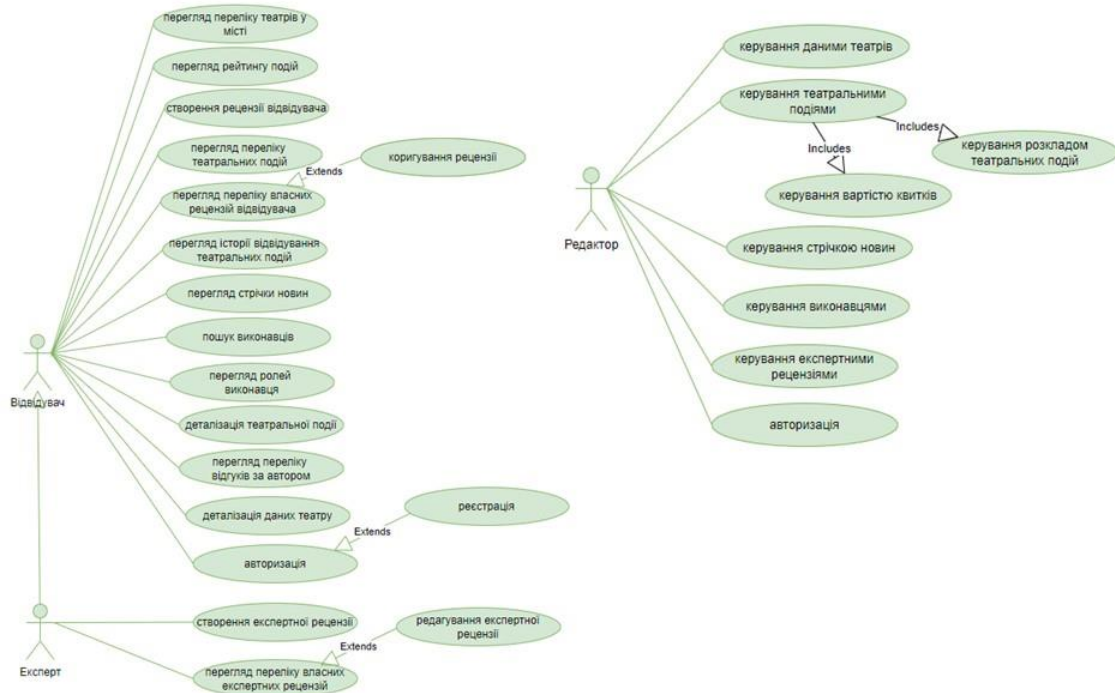


Рисунок В.5 – Слайд п'ятий

Схема бази даних

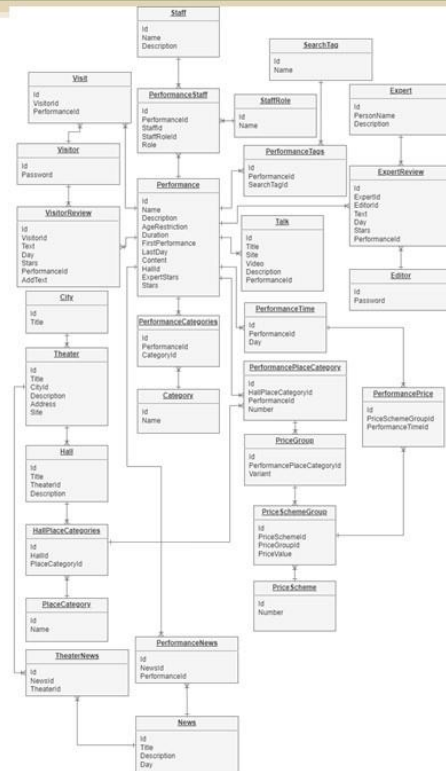


Рисунок В.6 – Слайд шостий

Шаблон проєктування Model-View-Controller

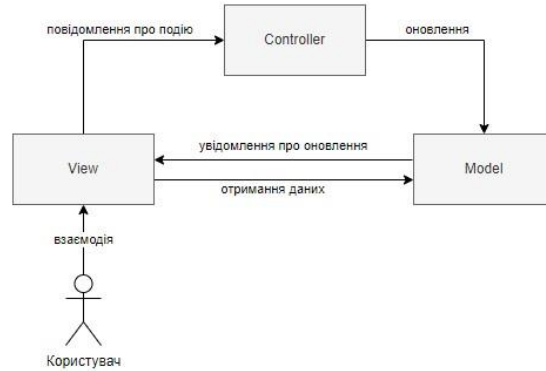


Рисунок В.7 – Слайд сьомий

Схема функціонування програми



Рисунок В.8 – Слайд восьмий

Перегляд театральної події

TheatralApp

Події Рейтинг Театри Новини Виконавці Рецензії Відвідування Профіль Вийти



подія

НАТАЛКА ПОЛТАВКА

Пісся Івана Котляревського, формалізатора сучасної української мови, що була створена в 1819 році.
Класична історія в сучасній інтерпретації.
Історія заборон і неволі на сцені театру розгортається з великим болем і пристрастю.
Багатий воєний намагався посвататися до гарної Наталки, а отримавши відмову, хитрою змусив її до шлюбу волею матері Наталки, не зважаючи на те, що Наталка має коханого...

ВІК

12+

ПРЕМ'ЄРА

10.09.2000

ОСТАННІЙ ПОКАЗ

20.02.2022

ЕКСПЕРТНА ОЦІНКА

7.5

КАТЕГОРІЯ

Вистава

ТЕГИ

класика, роман

Вартість квитків

ТЕАТР

Театр Магара

ЗАЛА

Головна

ТРИВАЛІСТЬ

2:00

ГЛЯДАЦЬКА ОЦІНКА

7.0

КОНТЕНТ

немає обмежень

Рисунок В.9 – Слайд дев'ятий

Перегляд виконавців театральної події

ВИКОНАВЦІ

АНДРІЙ ШТОЛЬНИЙ

Режисер

Детальніше



ІГОР ВЕРГОС

Сценарист

Детальніше



ІВАННА ЛІНЦ

Наталка

Актор

Детальніше



АНДРІЙ ТИТОРЕНКО

Возний

Актор

Детальніше



Рисунок В.10 – Слайд десятий

Пошук театральних подій

ПОШУК ПОДІЙ

Назва події

Місто Театр

Вік

Мінімальна тривалість Максимальна тривалість

Оцінка від Оцінка до

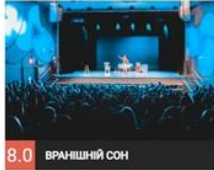
РЕЗУЛЬТАТИ ПОШУКУ



8.0 МАЛЬВИНА

ТЕАТР МАРІЇ ЗАНЬКОВЕЦЬКОЇ

[Детальніше](#)



8.0 ВРАНІШНІЙ СОН

ТЕАТР ФРАНКА

[Детальніше](#)



7.5 У ВСІХ НАОДИНЦІ

ТЕАТР МАРІЇ ЗАНЬКОВЕЦЬКОЇ

[Детальніше](#)

Рисунок В.11 – Слайд одинадцятий

Створення експертного відгуку про театральну подію

TheatralApp

Події Театри Новини Виконавці **Рецензії** Профіль Вийти

ЕКСПЕРТНИЙ ВІДГУК

Наталка Полтавка

Віктор Зленко

Оцінка

Текст рецензії

Рисунок В.12 – Слайд дванадцятий

Чеклист тестування програми

Перевірка	Результат
Перегляд переліку театрів у місті	+
Перегляд рейтингу вистав за категорією подій	+
Створення розкладу театральної події	+
Редагування розкладу театральної події	+
Створення рецензії відвідувача про театральну подію	+
Створення рецензії експерта про театральну подію	+
Коригування рецензії відвідувачем	+
Перегляд ролей виконавця	+
Внесення виконавців	+
Внесення даних нового театру	+
Редагування даних театру	+
Редагування експертної рецензії	+
Внесення події в історію відвідування	+
Перегляд власної історії відвідування подій	+
Внесення обговорення за подією	+
Визначення вартості квитків на подію	+
Пошук театральних подій	+
Перегляд театральних подій за тегом	+
Перегляд стрічки новин	+
Пошук виконавців	+
Перегляд рецензій за автором	+
Деталізація театральної події	+
Деталізація театру	+
Створення новини	+
Створення театральної події	+
Редагування театральної події	+
Авторизація, реєстрація	+ / +

Рисунок В.13 – Слайд тринадцятий

Висновки

- Проєктування програми дозволило створити структуру основних сутностей предметної області, визначити відповідні таблиці реляційної бази даних, визначити зв'язки між ними. До комплексу засобів проєктування і розробки було внесено мову програмування C#, вебфреймворк ASP.NET Core, шаблон проєктування Model-View-Controller, що дозволив визначити структуру вебсистеми, система Microsoft SQL Server, фреймворк для роботи з даними Entity Framework Core.
- Результати проєктування були використані під час реалізації програмного забезпечення. Результати реалізації описані представленням відповідних розроблених класів. Реалізація було виконана успішно, що підтверджено тестуванням реалізованих функцій.
- Створена вебсистема дозволила реалізувати можливості перегляду театрів у місті, театральних подій за категоріями, тегами, виконавцем, створення розкладу театральних подій, рецензій про театральні події з можливостями їх коригування, перегляду і пошуку театральних подій у театрі або окремій залу театру, в тому числі всіх або майбутніх, визначення історії відвідування театральних подій, перегляду новин.

Рисунок В.14 – Слайд чотирнадцятий