

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

**ПОГОДЖЕНО
Декан факультету**

**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри інформаційних
систем і технологій**

(підпис) **Ігор БОЛБОТ**

(підпис) **Михайло ШВИДЕНКО**

« ____ » _____ 2026 р.

« ____ » _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «Інформаційна система онлайн бронювання квитків на художні
виставки України»**

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
д.е.н., професор

(підпис) **Роман РУДЕНСЬКИЙ**

**Керівник бакалаврської
кваліфікаційної роботи**
асистент

(підпис) **Ярослав ПОНЗЕЛЬ**

Консультант
к.п.н., доцент

(підпис) **Тетяна ВОЛОШИНА**

Виконав

(підпис) **Маргарита КРУГЛІК**

Київ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і
технологій

к.е.н., доцент _____ Михайло ШВИДЕНКО
(підпис)

« ____ » _____ 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Круглік Маргариті Ігорівні

(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Інформаційна система онлайн бронювання квитків на художні виставки України»
затверджена наказом від «6» січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «25» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):
вимоги до інформаційної системи бронювання квитків художніх виставок України;
дані про користувачів, виставки, організації, місця проведення, статті, квитки, теги та
рекомендації

Перелік питань, що підлягають дослідженню:
аналіз предметної області та існуючих рішень; визначення вимог до системи;
розробка структури бази даних; проєктування і реалізація модулів вебзастосунку;
тестування системи; підготовка рекомендацій щодо впровадження та експлуатації.

Дата видачі завдання «6» січня 2026 р.

Керівник бакалаврської
кваліфікаційної роботи

(підпис)

Ярослав ПОНЗЕЛЬ

Консультант

(підпис)

Тетяна ВОЛОШИНА

Завдання взяв до
виконання

(підпис)

Маргарита КРУГЛІК

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	4
ВСТУП.....	6
ОСНОВНА ЧАСТИНА.....	9
1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	9
1.1 Постановка завдання.....	9
1.2 Огляд інформаційних джерел та існуючих рішень.....	11
1.3 Моделювання предметної області.....	15
2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ.....	25
2.1 Логічна модель даних.....	25
2.2 Вибір системи управління інформаційною базою.....	31
2.3 Створення інформаційної бази.....	32
2.4 Організація доступу до інформаційної бази.....	34
3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.....	36
3.1 Організаційна структура програмного забезпечення.....	36
3.2 Вибір інструментарію для створення ППЗ.....	39
3.3 Алгоритмізація та програмування програмних модулів.....	40
4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ.....	49
4.1 Тестування системи.....	49
4.2 Вимоги до апаратного та програмного забезпечення.....	53
4.3 Склад інсталяційного пакету.....	55
ВИСНОВКИ.....	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	60
ДОДАТОК А.....	62
ДОДАТОК Б.....	66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

1. API – Application Programming Interface, інтерфейс програмування застосунків.
2. ASP.NET Core – фреймворк для розробки вебзастосунків на платформі .NET.
3. CRUD – Create, Read, Update, Delete, базові операції створення, читання, оновлення та видалення даних.
4. CSS – Cascading Style Sheets, мова опису стилів вебсторінки.
5. EF Core – Entity Framework Core, ORM-технологія для роботи з базою даних у .NET.
6. ER-діаграма – Entity-Relationship Diagram, діаграма сутностей і зв'язків бази даних.
7. HTML – HyperText Markup Language, мова розмітки гіпертекстових документів.
8. HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту.
9. HTTPS – HyperText Transfer Protocol Secure, захищений протокол передачі даних.
10. JSON – JavaScript Object Notation, текстовий формат обміну даними.
11. MVC – Model-View-Controller, архітектурний шаблон побудови вебзастосунку.
12. QR-код – Quick Response Code, двовимірний код для швидкого зчитування інформації.
13. Razor – синтаксис шаблонів ASP.NET Core для створення динамічних HTML-сторінок.
14. SQL – Structured Query Language, мова структурованих запитів до бази даних.
15. SSMS – SQL Server Management Studio, середовище для адміністрування SQL Server.

- 16. UI – User Interface, інтерфейс користувача.
- 17. UML – Unified Modeling Language, уніфікована мова моделювання.
- 18. URL – Uniform Resource Locator, уніфікований покажчик ресурсу.
- 19. БД – база даних.
- 20. СКБД – система керування базами даних.
- 21. ІС – інформаційна система.
- 22. ПЗ – програмне забезпечення.

ВСТУП

Цифрові технології відіграють велику роль у тому, як ми споживаємо культуру у сучасному суспільстві. Більшість дізнаються про художні виставки, використовуючи онлайн-ресурси, але інформаційний простір в Україні для художніх виставок не дуже організований. Щоб знайти виставку, зазвичай доводиться переглядати численні вебсайти, соціальні мережі та афішні платформи. При існуванні такої великої кількості варіантів пошуку виставок знайти достовірну інформацію складно. Через це користувачам важче знаходити актуальні виставки, а туристам – швидко орієнтуватися в українському мистецькому середовищі.

Важливість роботи полягає у створенні єдиної інформаційної системи для пошуку українських мистецьких подій, перегляду виставок, читання тематичних статей, резервування квитків та отримання персоналізованих рекомендацій. Ця система може бути корисною як для звичайних користувачів, які цікавляться мистецтвом, так і для авторів публікацій та організаторів виставок. Крім того, особливе значення має можливість не лише спостерігати за подіями, а й отримувати рекомендації відповідно до своїх інтересів та попередньої взаємодії з контентом.

Об'єктом дослідження є процес організації, поширення та використання інформації, пов'язаної з художніми виставками та мистецькими подіями, у цифровому середовищі.

Предметом дослідження є методи та засоби проектування та впровадження веб-інформаційної системи для перегляду виставок, бронювання квитків, публікації статей, адміністрування контенту та формування рекомендацій для користувачів.

Головною метою цього проєкту є розробка та створення інформаційної системи UArtFan, яка забезпечить централізований спосіб пошуку виставок українського мистецтва, перегляду контенту, бронювання квитків, управління подіями та надання персоналізованих рекомендацій за допомогою тегів та дій користувачів.

Всі перераховані методології, включаючи методи системного аналізу, методи моделювання структур даних, методи моделювання UML, методології проектування реляційних баз даних і методології об'єктно-орієнтованого програмування, використовувалися одночасно для розробки та впровадження інформаційної системи. Серверна частина інформаційної системи створена за допомогою ASP.NET Core MVC мовою програмування C#. Для роботи з базою даних використано Entity Framework Core та SQL Server. Інтерфейс користувача створено з використанням HTML, CSS, Razor Views та JavaScript. Інтерактивна карта використовує Google Maps API для відображення місць проведення виставок. Технічне рішення бронювання квитків адмініструється шляхом зберігання даних, пов'язаних з бронюванням, у базі даних.

Користувачам системи доступні різні ролі. Звичайний користувач має можливість переглядати виставки та читати статті. Він також може бронювати квитки, коментувати матеріали, ставити лайки та отримувати рекомендації на основі своїх уподобань і попередньої взаємодії з контентом. Автори можуть створювати/редагувати створені ними статті, а також переглядати статистику щодо взаємодії з їхніми статтями. Організатори також мають доступ до керування виставками та місцями проведення своєї організації, а також переглядати аналітичні дані щодо бронювання квитків та рівня зацікавленості користувачів, які відвідують їхні виставки. Адміністратори мають повний доступ до всіх облікових записів користувачів, усіх ролей, можливість створювати/редагувати будь-яку організацію в системі, а також можливість створювати/редагувати будь-яку виставку, статтю, коментар чи скаргу; а також переглядати загальну статистику системи.

Рекомендації, що надаються користувачам на платформі рекомендацій UArtFan, формуються за допомогою тегів, які пов'язують користувачів із виставками та статтями. Під час реєстрації користувачі мають можливість вибрати теми чи предмети, які їх цікавлять, і які вони можуть пропустити. Після цього додаток відстежує, як вони взаємодіють із контентом (тобто перегляди, вподобання, бронювання квитків, коментарі та позначка «Не цікавить»), щоб допомогти в наданні рекомендацій пізніше. Якщо система не має достатньо

даних про інтереси користувача, вона надасть популярні виставки, які вимірюються за кількістю вподобань, бронювань квитків та переглядів.

Пояснювальна записка містить перелік умовних позначень, вступ, чотири розділи основної частини, висновки, список використаних джерел та додатки.

У першому розділі досліджується простір цифрових платформ, що дозволяє користувачам шукати мистецькі події. У ньому обговорюються основні труднощі, з якими стикаються користувачі та організатори під час використання цифрових платформ для пошуку мистецьких подій, а також наводиться формулювання проблеми та вимог до системи UArtFan. Також описано існуючі рішення, що вирішують ті самі проблеми користувачів, а також спосіб взаємодії користувачів із системою.

Другий розділ присвячено інформаційному забезпеченню системи. У ньому описано структуру даних, основні сутності предметної області, взаємозв'язки між ними, логічну модель бази даних та обґрунтовано використання реляційної СКБД SQL Server.

У третьому розділі подано опис прикладного програмного забезпечення. Розглянуто архітектуру вебзастосунку, використані інструментальні засоби, реалізацію основних модулів системи, зокрема реєстрації користувачів, резервування квитків, роботи зі статтями, адміністративної панелі та рекомендаційного механізму.

У четвертому розділі наведено результати тестування розробленої системи, вимоги до апаратного й програмного забезпечення, а також рекомендації щодо впровадження та подальшої експлуатації вебплатформи UArtFan.

ОСНОВНА ЧАСТИНА

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Розробка доступних інформаційних систем, що забезпечують швидкий доступ до актуальної інформації про мистецькі події, є одним із важливих завдань у сучасному цифровому культурному середовищі. Більшість інформації про художні виставки в Україні доступна на різних інтернет-ресурсах: вебсайтах галерей, сторінках у соціальних мережах, афішних платформах або інформаційних порталах. Пошук події ускладнюється, оскільки користувачам доводиться переглядати кілька джерел, порівнювати інформацію про подію та переконуватися, що вона є актуальною.

Проблема полягає в тому, що різні джерела не завжди поєднують в одному середовищі визначення місця проведення виставки, перегляд тематичних статей, отримання рекомендацій, резервування квитків і взаємодію з контентом. Це також створює труднощі для організаторів виставок у просуванні своїх заходів, аналізі інтересів аудиторії та поширенні інформації про події. Водночас відсутність інтегрованої платформи ускладнює авторам публікацій поширення матеріалів про українське мистецтво.

У межах виконання бакалаврської кваліфікаційної роботи основним завданням є створення платформи UArtFan, яка має спростити пошук виставок в Україні за допомогою централізованого каталогу, надати можливість читання різних типів статей про мистецтво, резервування квитків на виставки, відображення подій на карті та отримання персоналізованих рекомендацій. Платформа повинна забезпечити зручну взаємодію для різних типів користувачів, включаючи звичайних користувачів, авторів, організаторів та адміністраторів.

Основними користувачами платформи є:

звичайний користувач, який може переглядати виставки, читати статті, бронювати квитки, коментувати матеріали та отримувати рекомендації;

- автор, який може створювати та редагувати власний контент, а також переглядати основну статистику щодо взаємодії користувачів з його статтями;
- організатор, який може керувати виставками та місцями проведення, що пов'язані з його організацією;
- адміністратор, який має можливість виконувати загальні функції управління, зокрема керувати користувачами, ролями, контентом, організаціями, скаргами та аналітичними даними.

Інформаційна система повинна дозволяти користувачам реєструватися та автентифікуватися, а також автоматизувати процес надання доступу до функцій відповідно до ролі користувача. Вона повинна надавати можливість переглядати список художніх виставок, детальну інформацію про окрему експозицію та бронювати квитки. Після бронювання квитка система повинна автоматично генерувати QR-код, пов'язаний із зарезервованим квитком.

Крім того, система повинна автоматизувати такі операції:

- створення, перегляд та модерацію статей, пов'язаних з мистецькою тематикою;
- додавання вподобань і коментарів до статей;
- обробку скарг на недоречні коментарі чи контент;
- керування організаторами виставок та місцями їх проведення;
- відображення місць проведення виставок на карті;
- формування рекомендацій на основі тегів, інтересів та поведінки користувача;
- надання доступу до статистики та звітів для авторів, організаторів виставок і адміністраторів.

Вхідні дані включають як вручну введену, так і автоматично згенеровану інформацію, що стосується користувачів, ролей, організацій, місць проведення, виставок, статей, тегів, коментарів, вподобань, квитків, переглядів та дій, виконаних у системі рекомендацій. Ці дані вводяться користувачами відповідно до їхніх прав доступу або формуються автоматично під час роботи

системи. Вихідними даними є списки виставок і статей, персоналізовані рекомендації, інформація про зарезервовані квитки, QR-коди, аналітичні звіти, статистика взаємодії з контентом, дані для адміністративного керування та повідомлення про результати виконання операцій.

Для забезпечення коректної роботи системи необхідно реалізувати базу даних, яка зберігатиме інформацію про всі основні сутності предметної області. До таких сутностей належать користувачі, ролі, автори, організатори, організації, майданчики, виставки, статті, теги, квитки, коментарі, вподобання, перегляди та рекомендаційні зв'язки. Взаємозв'язки між цими сутностями повинні забезпечувати цілісність даних і можливість подальшого формування звітів та рекомендацій.

Окрему увагу в системі необхідно приділити рекомендаційному механізму. Його завданням є підбір виставок відповідно до інтересів користувача. Для цього використовуються теги, які пов'язують виставки, статті та вподобання користувачів. Якщо даних про інтереси користувача недостатньо, система повинна показувати популярні виставки, визначені за кількістю переглядів, вподобань і резервувань.

Таким чином, поставлене завдання полягає у створенні веб-орієнтованої інформаційної системи, яка об'єднує функції пошуку мистецьких подій, резервування квитків, публікації статей, адміністрування контенту, роботи з картою та формування рекомендацій. Реалізація такої системи спрямована на підвищення доступності інформації про українське мистецтво та покращення взаємодії між відвідувачами, авторами, організаторами й адміністраторами культурних подій.

1.2 Огляд інформаційних джерел та існуючих рішень

На сьогодні користувачі можуть отримувати інформацію про художні виставки та мистецькі події з різних джерел: сайтів галерей і музеїв, соціальних мереж, афішних платформ та сервісів квитків. Однак більшість таких рішень не є спеціалізованими саме для українських художніх виставок або не поєднують в

одному середовищі пошук подій, тематичні статті, резервування квитків, персоналізовані рекомендації та аналітику для організаторів.

У межах дослідження було розглянуто кілька існуючих рішень, які частково виконують подібні функції. Аналіз здійснювався за такими критеріями: наявність інформації про мистецькі події, можливість роботи з квитками, зручність пошуку, підтримка рекомендацій, наявність ролей для організаторів та можливість централізованого керування контентом.

1.1.1 Kontramarka

Kontramarka – це онлайн-платформа для бронювання квитків та пошуку культурних і розважальних подій. На вебсайті доступні різні типи заходів, зокрема концерти, театральні вистави, дитячі шоу, фестивалі та інші події. Користувачі можуть переглядати дати й місця проведення подій або вибирати потрібний захід відповідно до категорії. Також сервіс надає інформацію про можливість бронювання або придбання квитків. Kontramarka має зручний інтерфейс для перегляду списку подій і пошуку за одним або кількома критеріями, а також надає користувачам доступ до широкого спектру заходів та можливість роботи з квитками.

До переваг Kontramarka можна віднести наявність великої кількості подій, зручну систему побудови афіші, пошук за різними параметрами та організацію роботи з квитками. Ці функції дозволяють користувачам сервісу легко знаходити події культурного чи розважального характеру.

Одна з проблем використання Kontramarka полягає в тому, що сервіс не призначений саме для художніх виставок або українського мистецтва. Артподії на сайті подаються поряд із багатьма іншими типами заходів, що ускладнює пошук саме виставок. Крім того, сервіс не надає окремого простору для авторів мистецьких статей, персоналізованих рекомендацій на основі тегів і дій користувача, кабінету організатора виставок з аналітикою та централізованої модерації тематичного контенту.

1.1.2 Karabas

Karabas – це український онлайн-сервіс для пошуку подій та купівлі квитків. Платформа надає список подій з фільтрами за містом, категорією та датою. Сайт підтримує електронні квитки та пропонує концерти, шоу, фестивалі й інші культурно-розважальні заходи, що забезпечує доступ до широкої бази подій.

Перевагою використання сервісу Karabas є значна кількість подій, представлених у базі даних, добре організований каталог, а також можливість використання електронних квитків. Водночас сервіс не зосереджений переважно на художніх виставках, тому не може бути використаний як повноцінний спеціалізований ресурс для представлення українського мистецтва. База подій не містить окремого середовища для мистецьких статей, не надає рекомендацій на основі тегів, не пропонує кабінетів авторів та організаторів, а також не забезпечує модерацію тематичного контенту.

1.1.3 Facebook Events та соціальні мережі

Багато галерей, музеїв та інших культурних закладів використовують соціальні мережі для поширення інформації про виставки. Facebook Events, Instagram-сторінки та інші соціальні платформи дозволяють швидко публікувати афіші, описи подій, фотографії та оновлення для підписників.

Однією з переваг соціальних мереж є простота публікації та швидке поширення інформації серед уже наявної аудиторії. Недоліком є відсутність централізованої структури. Події різних організацій публікуються на окремих сторінках, тому користувач повинен самостійно стежити за сторінками галерей або організаторів. Ці платформи не забезпечують єдиної бази даних виставок, бронювання квитків, аналітики для організаторів або рекомендацій на основі інтересів користувача.

1.1.4 Порівняльний аналіз існуючих рішень

Згадані сервіси частково вирішують завдання пошуку подій, хоча й не повністю задовольняють вимоги системи, пов'язаної з художніми виставками та

українським мистецтвом. Kontramarka та Karabas пропонують зручні каталоги подій і можливість роботи з квитками, однак вони зосереджені на ширшому спектрі культурних та розважальних заходів, тому виставки мають для них другорядне значення.

Facebook Events та соціальні мережі корисні для швидкого поширення інформації, проте їм бракує централізованого каталогу, структурованої бази даних, аналітичних інструментів та повноцінної модерації. UArtFan, з іншого боку, поєднує каталог художніх виставок, тематичні статті, сервіс резервування квитків, карту подій, рекомендації, кабінети організаторів та авторів – усе в межах однієї системи.

Для наочного порівняння функціональних можливостей розглянутих рішень наведено табл. 1.1.

Таблиця 1.1

Порівняльний аналіз існуючих рішень

Критерій	Kontramarka	Karabas	Facebook Events / соцмережі	UArtFan
1	2	3	4	5
Централізований каталог подій	+	+	–	+
Спеціалізація на художніх виставках	–	–	частково	+
Орієнтація на українське мистецтво	–	–	частково	+
Пошук за містом / датою	+	+	частково	+
Дані про організації та майданчики	–	–	частково	+
Резервування квитків без онлайн-оплати	частково	–	–	+
QR-код для квитка	частково	+	–	+
Статті про мистецтво	–	–	частково	+
Кабінет автора	–	–	–	+
Кабінет організатора	частково	частково	частково	+
Аналітика для організатора	частково	частково	частково	+

Таблиця 1.1 (продовження)

1	2	3	4	5
Персоналізовані рекомендації	частково	частково	частково	+
Модерація коментарів і скарг	частково	частково	частково	+
Адміністративне керування	–	–	–	+

ролями				
--------	--	--	--	--

Таблиця 1.1 (закінчення)

Отже, розроблювана система UArtFan відрізняється від розглянутих рішень тим, що поєднує в одному середовищі каталог художніх виставок, перегляд тематичних статей, резервування квитків, подання скарг, рекомендації, кабінети автора й організатора, а також адміністративне керування контентом і користувачами. Це дозволяє розглядати UArtFan не лише як афішний ресурс, а як спеціалізовану інформаційну систему для підтримки українського мистецького середовища.

1.3 Моделювання предметної області

Для опису предметної області інформаційної системи UArtFan було створено набір моделей, який показує функції системи, ролі користувачів та їхню взаємодію між собою. Мета моделей полягає у визначенні меж системи, сценаріїв використання та прав доступу для кожного типу користувачів.

Акторів системи можна поділити на такі категорії:

- Неавторизований користувач – має обмежений доступ до системи: може переглядати виставки, а також перейти до реєстрації або входу.
- Користувач – основний актор системи, який після авторизації може переглядати виставки, вибирати події, резервувати квитки, читати статті, залишати коментарі або вподобайки та подавати скарги на контент.
- Автор – користувач із додатковими правами, який створює статті та переглядає аналітику взаємодії з ними.
- Організатор – користувач із додатковими правами, який створює виставки, заповнює анкету виставки та переглядає аналітику взаємодії з виставками.
- Адміністратор – користувач із розширеними правами, який керує користувачами, переглядає скарги, модерує коментарі та має доступ до загальної аналітики.

1.1.5 Use Case діаграма

Діаграма варіантів використання (Use Case діаграма) (рис. 1.1) відображає взаємодію між основними акторами та інформаційною системою UArtFan. Вона показує, які функції доступні кожній ролі, а також визначає межі доступу до основних можливостей платформи. У системі передбачено п'ять акторів: неавторизований користувач, авторизований користувач, автор, організатор та адміністратор.

Неавторизований користувач має обмежений доступ до системи. Він може переглядати список виставок, а також виконати реєстрацію або вхід у систему.

UC01 «Реєстрація / вхід» – варіант використання, що дозволяє користувачеві створити обліковий запис або увійти в систему для отримання доступу до додаткових функцій.

UC02 «Перегляд виставок» – перегляд загального списку художніх виставок без необхідності авторизації.

Авторизований користувач має доступ до основних функцій платформи. Він може переглядати детальну інформацію про виставки, бронювати квитки, читати статті, залишати коментарі, ставити вподобайки та подавати скарги на контент.

UC03 «Вибір виставки» – перегляд детальної інформації про обрану виставку, зокрема її опису, дати проведення, місця розташування та вартості квитка.

UC04 «Бронювання квитка» – створення резервування квитка на обрану виставку без здійснення онлайн-оплати. Після бронювання у системі формується запис про квиток, який користувач може переглянути у власному кабінеті.

UC05 «Перегляд статей» – перегляд списку тематичних матеріалів, присвячених українському мистецтву, художнім виставкам, авторам та культурним подіям.

UC06 «Читання статті» – відкриття повного тексту статті, перегляд її змісту, зображень і пов'язаної інформації.

UC07 «Додавання коментаря або вподобайки до статті» – взаємодія користувача зі статтею шляхом залишення коментаря або позначення матеріалу як вподобаного.

UC08 «Подання скарги на контент» – надсилання скарги на статтю, виставку або коментар у разі виявлення некоректного, недоречного чи помилкового змісту.

Автор має доступ до функцій, пов'язаних зі створенням і аналізом статей.

UC09 «Створення статті» – додавання нової тематичної статті до системи із зазначенням назви, тексту, зображення та тегів.

UC10 «Перегляд аналітики взаємодії зі статтями» – перегляд статистики щодо взаємодії користувачів із публікаціями автора, зокрема кількості переглядів, вподобань або коментарів.

Організатор має доступ до функцій керування виставками. Він відповідає за створення, заповнення та оновлення інформації про події, а також може переглядати статистику зацікавленості користувачів.

UC11 «Створення виставки» – додавання нової тематичної статті до системи із зазначенням назви, тексту, зображення та тегів.

UC12 «Заповнення анкети виставки» – внесення основної інформації про виставку: назви, опису, дати проведення, місця, вартості квитка, зображення та тегів.

UC13 «Перегляд аналітики взаємодії з виставками» – перегляд статистичних даних щодо зацікавленості користувачів виставками.

Адміністратор має доступ до функцій загального керування системою.

UC14 «Перегляд загальної аналітики» – перегляд узагальнених статистичних даних про роботу системи, активність користувачів, кількість виставок, статей і резервувань.

UC15 «Перегляд скарг» – опрацювання скарг, поданих користувачами на контент системи.

UC16 «Керування користувачами» – перегляд і редагування даних користувачів, а також керування їхніми ролями в системі.

UC17 «Модерація коментарів» – перевірка коментарів користувачів і видалення тих, що порушують правила платформи.

Таким чином, діаграма варіантів використання показує розподіл функціональних можливостей між неавторизованим користувачем, авторизованим користувачем, автором, організатором та адміністратором. Вона визначає основні сценарії роботи із системою UArtFan і межі доступу для кожної ролі.



Рис. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.1 Use Case діаграма

1.1.6 Діаграма послідовності – бронювання квитка

Діаграма послідовності процесу бронювання квитка (рис. 1.2) показує взаємодію між такими учасниками: користувач, вебінтерфейс UArtFan, TicketsController, SQL Server, RecommendationService та QRCoder. Діаграма

містить два альтернативні блоки, що відображають розгалуження сценарію залежно від авторизації користувача та правильності вибору дати бронювання.

Процес бронювання квитка починається, коли користувач натискає кнопку «Забронювати квиток» на сторінці обраної виставки. Вебінтерфейс UArtFan надсилає POST-запит до TicketsController за маршрутом /Tickets/Reserve/{eventId}.

TicketsController перевіряє, чи існує ідентифікатор користувача в сесії. Першою альтернативною умовою є перевірка авторизації: якщо користувач не має активної сесії, система перенаправляє його на сторінку входу. У такому випадку бронювання квитка не виконується.

Якщо користувач авторизований, TicketsController отримує дані обраної виставки з бази даних SQL Server за ідентифікатором виставки. SQL Server повертає інформацію про виставку, зокрема назву, дату початку, дату завершення, місце проведення та ціну квитка.

Після цього TicketsController перевіряє дату бронювання. Дата не повинна передувати поточній даті та має входити в період проведення виставки. Якщо дата бронювання некоректна, користувача перенаправляють назад на сторінку виставки з повідомленням про помилку. Якщо дата коректна, виконання процесу продовжується.

Далі TicketsController застосовує обмеження на кількість квитків від одного до десяти. Для кожного зарезервованого квитка створюється запис у таблиці Tickets, що містить ідентифікатор користувача, ідентифікатор виставки, дату бронювання, ціну квитка, статус Reserved та унікальне значення QR_Code, сформоване за допомогою методу Guid.NewGuid(). Усі записи про квитки зберігаються в базі даних SQL Server.

Після успішного збереження квитків у базі даних TicketsController звертається до RecommendationService для оновлення рекомендаційного профілю користувача. Бронювання квитка на виставку фіксується як позитивна взаємодія з подією. Після завершення операції користувач автоматично перенаправляється на сторінку «Мої квитки», де відображаються зарезервовані ним квитки.

Під час відкриття QR-коду вебінтерфейс надсилає GET-запит до TicketsController за маршрутом /Tickets/QrCode/{ticketId}. TicketsController отримує дані квитка з бази даних і перевіряє, чи має користувач право переглядати відповідний QR-код.

На основі інформації, що міститься у квитку, формується текст QR-коду. Він містить ідентифікатор квитка, ідентифікатор виставки, ідентифікатор користувача та унікальний код бронювання. Система використовує бібліотеку QRCode для створення QR-коду у форматі PNG.

Зображення QR-коду повертається вебінтерфейсу та відображається користувачеві. У межах цього процесу онлайн-оплата не виконується: система створює лише резервування квитка, а QR-код використовується як ідентифікатор цього резервування.

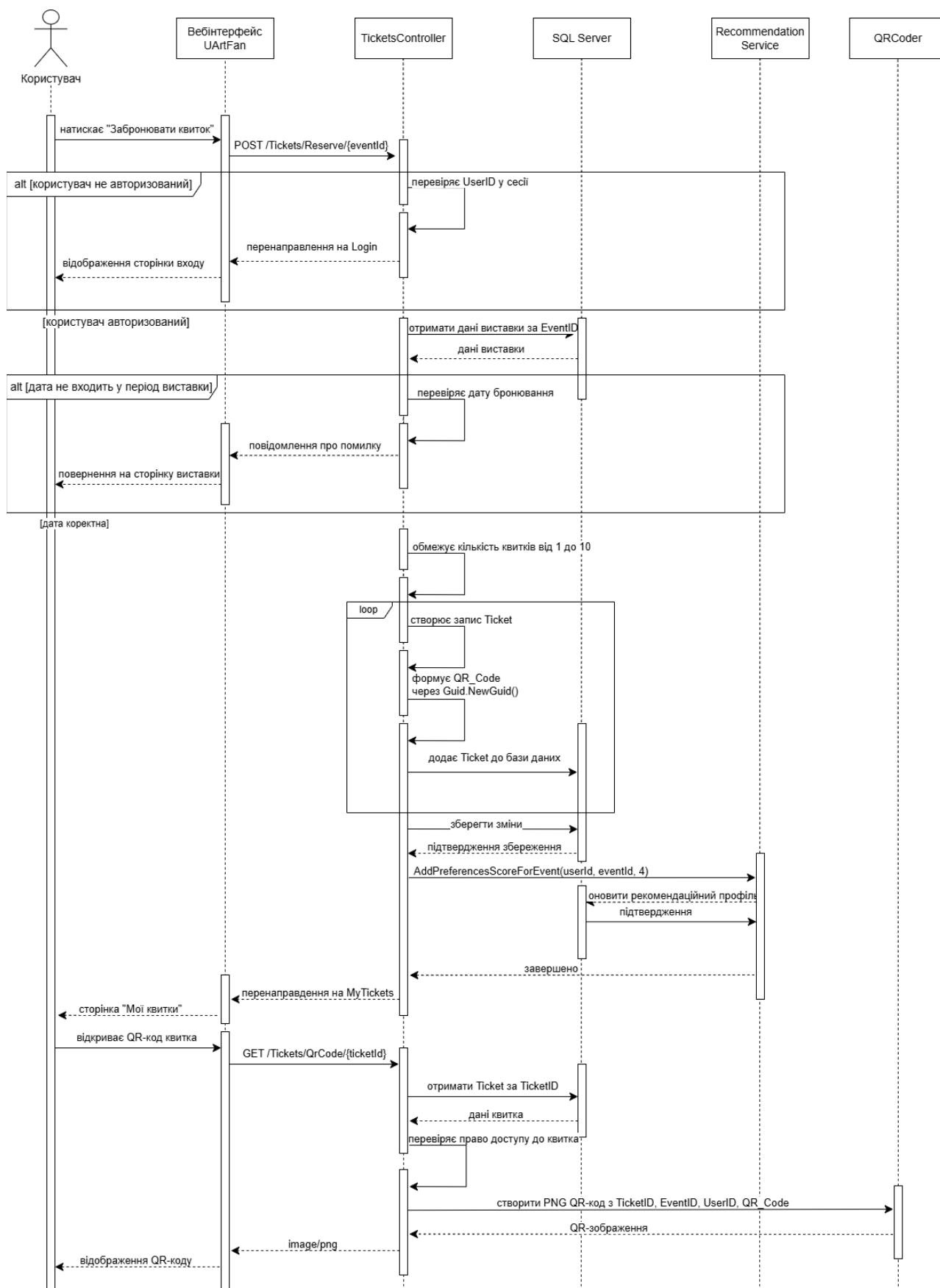


Рис. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.2 Діаграма послідовності процесу бронювання квитка

1.1.7 Діаграма розміщення системи

Схема розміщення компонентів інформаційної системи UArtFan показує логічну організацію вебзастосунку та розташування його основних елементів: клієнтської частини, серверної частини, бази даних і зовнішніх сервісів. Також схема демонструє, як користувач отримує доступ до вебклієнта, як запити надсилаються на сервер та які модулі використовуються для виконання основних функцій системи.

Клієнтська частина представлена вебклієнтом UArtFan, який працює у браузері користувача. Через нього користувач переглядає виставки, читає тематичні статті, бронює квитки, переглядає QR-коди, залишає коментарі, подає скарги та отримує рекомендації.

Серверна частина представлена бекенд-компонентом UArtFan, який отримує запити від клієнта та передає їх до відповідних модулів системи. На сервері розміщено серверну частину UArtFan, яка містить основні функціональні модулі: модуль автентифікації, модуль профілю користувача, модуль рекомендацій, модуль карти, модуль виставок, модуль квитків, модуль статей, модуль коментарів, модуль реакцій на статті та модуль модерації.

Модуль автентифікації виконує функції реєстрації користувачів, входу в систему та перевірки сесій. Модуль профілю користувача забезпечує роботу з персональними даними. Функцією модуля рекомендацій є створення персоналізованих пропозицій на основі тегів та дій користувача. Модуль карти використовується для підготовки даних про місце проведення виставок та взаємодії із зовнішнім Google Maps API.

Модуль виставок підтримує створення, перегляд та редагування інформації про виставки. Модуль квитків пов'язаний з модулем виставок і забезпечує бронювання квитків, скасування бронювання та роботу з QR-кодами. Для створення QR-кодів використовується бібліотека QRCode.

Модуль статей дозволяє створювати, переглядати та редагувати тематичні матеріали, пов'язані з мистецтвом. Модуль коментарів і модуль реакцій на статті забезпечують можливість залишати коментарі, ставити вподобання та дизлайки.

Модуль модерації використовується для обробки скарг і контролю якості контенту.

Сервер бази даних містить базу даних SQL Server, у якій зберігається інформація про користувачів, ролі, профілі, організації, місця проведення, виставки, статті, квитки, коментарі, реакції, скарги, теги та рекомендації. Взаємодія серверної частини з базою даних забезпечує збереження, оновлення та отримання інформації, необхідної для роботи системи.

Отже, діаграма розміщення компонентів поділяє UArtFan на вебклієнт, серверну логіку, базу даних та зовнішні сервіси. Така структура дозволяє відокремити інтерфейс користувача, обробку запитів, збереження даних, роботу з картою та генерацію QR-кодів.

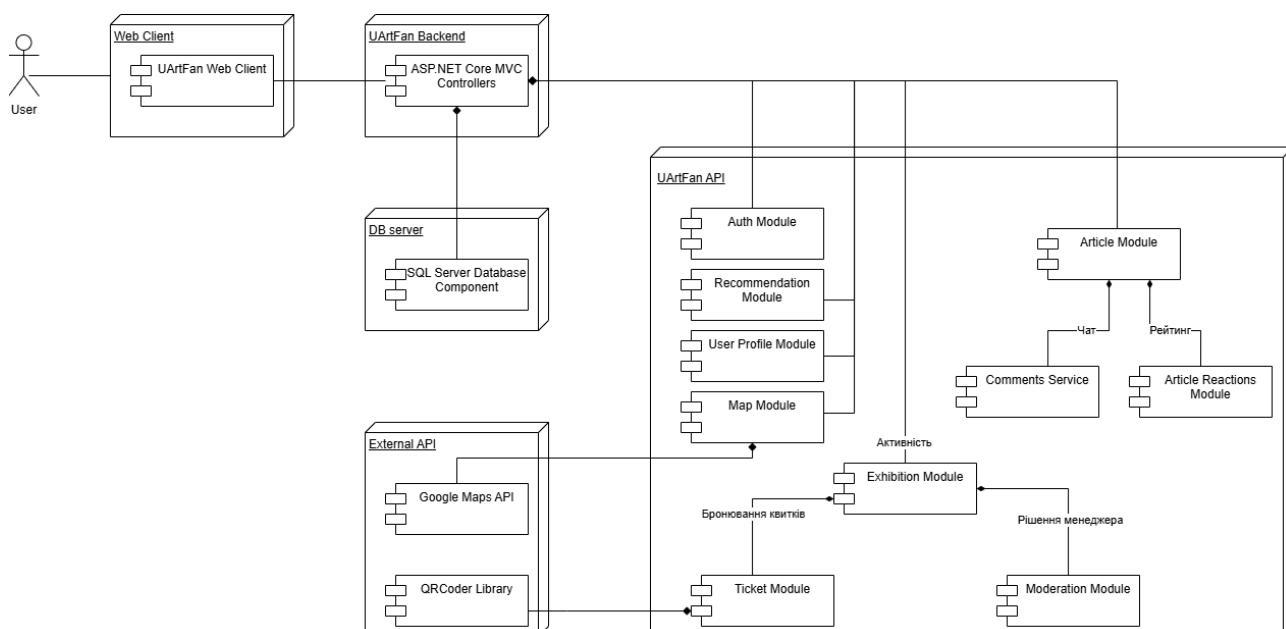


Рис. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.3
Діаграма розміщення компонентів IC UArtFan

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

1.4 Логічна модель даних

База даних UArtFan класифікована за допомогою реляційної моделі класифікації баз даних та містить пов'язані таблиці, зв'язки яких визначаються за допомогою первинних та зовнішніх ключів. Через цю структуру даних можна зберігати та пов'язувати такі записи: користувачі, ролі, профілі, організації, організатори, місця проведення, виставки, статті, квитки, коментарі, вподобання, теги, перегляди, скарги та дані для рекомендацій.

Сутність Users є основною сутністю в системі. Сутність облікового запису користувача містить атрибути, пов'язані з користувачем, включаючи UserID, RoleID, Username, Email, PasswordHash та CreatedAt. Роль, яку має користувач, визначає, який рівень доступу він матиме в системі. Деякі з цих ролей включають звичайного користувача, автора, організатора або адміністратора.

Додаткові дані про користувача зберігаються у сутності UserProfiles. Вона містить ProfileID, UserID, Bio, AvatarUrl та Location. Ця сутність потрібна для того, щоб зберігати не тільки дані для входу, а й додаткову інформацію про профіль користувача, наприклад опис, аватар та місцезнаходження.

Сутність Organizations містить інформацію про організації, які проводять мистецькі заходи. До них можуть належати галереї, музеї, артцентри або інші установи. Вона містить OrganizationID, Name, Type, Description, Email, Phone, Website та LogoUrl. Зв'язок між організацією та користувачем реалізований через сутність Organizers. Вона містить OrganizerID, UserID, OrganizationID та PositionTitle. Тобто за допомогою цієї таблиці система визначає, який користувач є організатором певної організації.

Інформація про місця проведення виставок зберігається у сутності Venues. Вона містить VenueID, OrganizationID, Name, City, Address, Latitude, Longitude, Description та ImageUrl. Через цю сутність можна зберігати дані про місце, де проходить виставка: назву, місто, адресу, координати, опис та зображення.

Сутність Events зберігає інформацію про виставки. Вона містить EventID, OrganizerID, VenueID, Title, City, StartDate, EndDate, Price, Description, ImageUrl та CreatedAt. Через зв'язки з Organizers та Venues можна визначити, хто відповідає за виставку та де саме вона проводиться.

Сутність Writers використовується для зберігання інформації про авторів статей. Вона містить WriterID, UserID та PenName. Сутність Articles містить інформацію про самі статті: ArticleID, WriterID, Title, ImageUrl, ArticleText, PublishedAt та EventID. Завдяки цьому система може визначити, який автор створив конкретну статтю, і зберігати матеріали, пов'язані з українським мистецтвом або конкретною виставкою.

Сутність Tickets використовується для зберігання інформації про заброньовані квитки. Вона містить TicketID, EventID, UserID, ReservationDate, Price, Status та QR_Code. У системі не виконується онлайн-оплата, тому такий запис означає саме резервування квитка, а не його купівлю. QR_Code використовується як унікальний код для ідентифікації цього резервування.

Категоризація контенту реалізована через сутність Tags. Вона містить TagID та TagName. Зв'язок між виставками та тегами реалізований через сутність EventTags, що містить EventID та TagID. Зв'язок між статтями та тегами реалізований через сутність ArticleTags, що містить ArticleID та TagID. Це дозволяє відносити виставки та статті до певних тематичних категорій.

Вподобання користувачів щодо статей зберігаються у сутності ArticleLikes, яка містить ArticleID, UserID та LikedAt. Ця таблиця потрібна для того, щоб фіксувати реакції користувачів на статті.

Коментарі до статей зберігаються у сутності ArticleComments. Вона містить CommentID, ArticleID, UserID, CommentText та PostedAt. За допомогою цієї сутності користувачі можуть залишати коментарі до статей, а система може зберігати ці обговорення.

Перегляди контенту зберігаються у сутностях ArticleViews та EventViews. Вони містять інформацію про те, які статті та виставки переглядав користувач, і можуть використовуватися для аналітики та рекомендацій.

Для роботи рекомендаційного механізму використовується сутність UserTagPreferences, яка зберігає рівень інтересу користувача до певних тегів. Сутність UserOnboardings фіксує, чи пройшов користувач початковий вибір інтересів після реєстрації. Контент, який користувач не хоче бачити в рекомендаціях, зберігається у сутностях UserHiddenArticles та UserHiddenEvents. Вони містять приховані користувачем статті та виставки. Скарги на контент зберігаються у сутності ContentReports. Вона містить інформацію про користувача, об'єкт скарги, її категорію, причину, статус та дату створення.

Таким чином, база даних UArtFan містить основні сутності, необхідні для роботи системи: Users, Roles, UserProfiles, UserOnboardings, Organizations, Organizers, Venues, Events, Writers, Articles, Tickets, Tags, EventTags, ArticleTags, UserTagPreferences, ArticleLikes, ArticleComments, ArticleViews, EventViews, UserHiddenArticles, UserHiddenEvents та ContentReports. Така структура дозволяє зберігати дані про користувачів, виставки, статті, квитки, теги, коментарі, вподобання, перегляди, скарги та рекомендаційні зв'язки.

У табл. 2.1 наведено основні сутності інформаційної бази UArtFan.

Таблиця 2.1

Сутності інформаційної бази UArtFan

Сутність	Призначення	Ключові поля
1	2	3
Users (користувачі)	Зберігання облікових записів користувачів	UserID, RoleID
Roles (ролі)	Зберігання ролей користувачів	RoleID
UserProfiles (профілі користувачів)	Зберігання додаткової інформації про користувача	ProfileID, UserID
UserOnboardings (онбординг користувачів)	Зберігання інформації про проходження початкового вибору інтересів	UserID
Organizations (організації)	Зберігання даних про організації, галереї, музеї або артцентри	OrganizationID
Organizers (організатори)	Зв'язок користувачів з організаціями	OrganizerID, UserID, OrganizationID
Venues (місця проведення)	Зберігання інформації про місця проведення виставок	VenueID, OrganizationID
Events (виставки / події)	Зберігання основної інформації про виставки	EventID, OrganizerID, VenueID

Таблиця 2.1 (продовження)

1	2	3
Writers (автори)	Зберігання даних про авторів статей	WriterID, UserID
Articles (статті)	Зберігання тематичних статей про мистецтво	ArticleID, WriterID, EventID
Tickets (квитки / резервування)	Зберігання інформації про зарезервовані квитки	TicketID, EventID, UserID
Tags (теги)	Зберігання тематичних тегів	TagID
EventTags (теги виставок)	Зв'язок виставок із тегами	EventID, TagID
ArticleTags (теги статей)	Зв'язок статей із тегами	ArticleID, TagID
UserTagPreferences (уподобання користувача за тегами)	Зберігання оцінки інтересу користувача до окремих тегів	UserID, TagID
ArticleLikes (вподобання статей)	Зберігання вподобань користувачів щодо статей	UserID, ArticleID
ArticleComments (коментарі до статей)	Зберігання коментарів користувачів до статей	CommentID, ArticleID, UserID
ArticleViews (перегляди статей)	Зберігання переглядів статей користувачами	ArticleViewID, UserID, ArticleID
EventViews (перегляди виставок)	Зберігання переглядів виставок користувачами	EventViewID, UserID, EventID
UserHiddenArticles (приховані статті користувача)	Зберігання статей, які користувач позначив як нецікаві	UserID, ArticleID
UserHiddenEvents (приховані виставки користувача)	Зберігання виставок, які користувач позначив як нецікаві	UserID, EventID
ContentReports (скарги на контент)	Зберігання скарг користувачів на статті або виставки	ContentReportID, ReporterUserID, ArticleID, EventID

Таблиця 2.1 (закінчення)

Інформаційна база UArtFan визначає кілька основних типів зв'язків між сутностями. Зв'язок «один до багатьох» існує між Roles та Users: одну роль можна надати багатьом користувачам, але один користувач має лише одну роль. Users пов'язана з UserProfiles зв'язком «один до одного», тобто один користувач має один профіль.

Також Users пов'язана з Organizers, Writers, Tickets, ArticleComments та ArticleLikes. Це означає, що користувач може мати додатковий запис

організатора або автора, резервувати квитки, залишати коментарі та ставити вподобання статтям.

Зв'язок «один до багатьох» існує між Organizations та Organizers, Organizations та Venues, Organizers та Events, Venues та Events. Одна організація може мати кількох організаторів і кілька місць проведення. Один організатор може створювати кілька виставок, а одне місце проведення може бути пов'язане з різними подіями. Завдяки цьому система зберігає не лише інформацію про виставку, а й дані про те, хто її організовує і де вона проходить.

Events та Tickets пов'язані зв'язком «один до багатьох», оскільки одна виставка може мати кілька заброньованих квитків. Кожен запис у Tickets належить до конкретного користувача та конкретної виставки. Writers та Articles також мають зв'язок «один до багатьох», оскільки один автор може створити кілька статей. Articles можуть бути пов'язані з Events, якщо матеріал стосується певної виставки.

Теги використовуються для тематичного групування контенту. Зв'язок між Events та Tags є зв'язком «багато до багатьох» і реалізований через EventTags. Зв'язок між Articles та Tags також є зв'язком «багато до багатьох» і реалізований через ArticleTags. Це дозволяє прив'язувати виставки та статті до кількох тематичних категорій і використовувати ці дані під час формування рекомендацій.

Коментарі до статей і вподобання реалізовані через ArticleComments та ArticleLikes. Ці сутності пов'язують Users та Articles і дозволяють зберігати взаємодію користувачів зі статтями. Для рекомендацій використовуються UserTagPreferences, UserHiddenArticles та UserHiddenEvents, які зберігають інтереси користувача та прихований контент.

Цілісність даних забезпечується зовнішніми ключами. Вони не дозволяють створювати записи, пов'язані з неіснуючими користувачами, виставками, статтями, організаціями, місцями проведення або тегами.

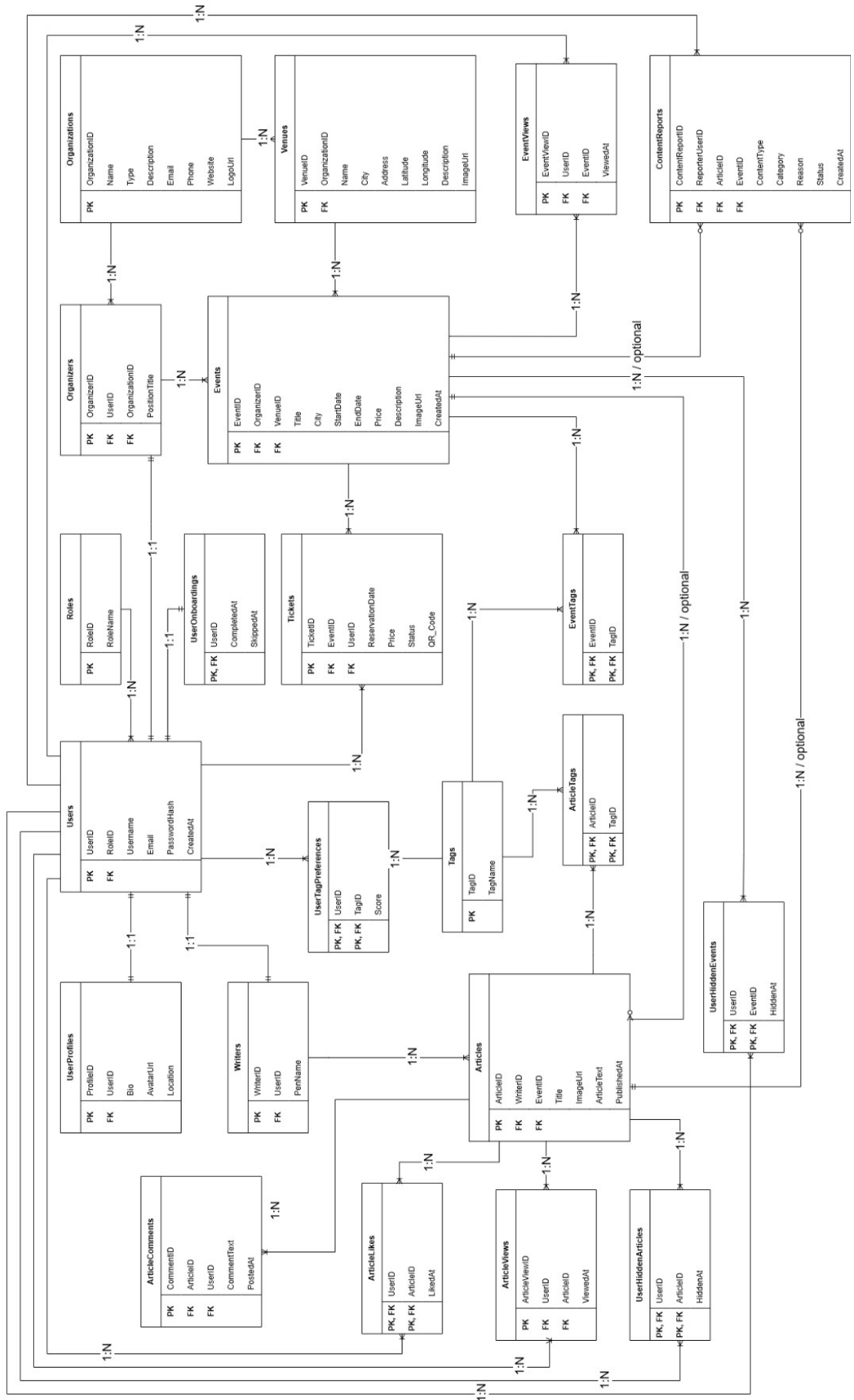


Рис. 2.1 Логічна модель бази даних IC UArtFan

1.5 Вибір системи управління інформаційною базою

Вибір СУБД є важливим етапом у створенні інформаційної системи UArtFan, оскільки СУБД відповідає не лише за збереження даних, необхідних для роботи вебзастосунку, а й за їх обробку та отримання під час виконання основних операцій. Структура даних цієї системи має реляційний характер і складається з кількох взаємопов'язаних сутностей, тому для реалізації інформаційної бази було обрано Microsoft SQL Server.

У базі даних UArtFan зберігається інформація про користувачів, ролі, організації, місця проведення, виставки, статті, квитки, теги, коментарі, вподобання, скарги та рекомендаційні зв'язки. Оскільки ці дані пов'язані між собою за допомогою первинних та зовнішніх ключів, реляційна СУБД є найбільш доцільним варіантом. Microsoft SQL Server підтримує такі зв'язки, забезпечує контроль цілісності даних та запобігає створенню некоректних записів.

Крім того, SQL Server є зручним вибором, оскільки він сумісний зі стеком технологій, використаним для побудови системи. Серверна частина UArtFan була реалізована з використанням ASP.NET Core MVC, а для доступу до даних застосовується Entity Framework Core. У цьому випадку SQL Server добре інтегрується з .NET, моделями C# та механізмами міграцій бази даних. Для написання коду використовувалося середовище Visual Studio Code, а для перегляду структури бази даних, зв'язків між таблицями та виконання SQL-запитів використовувалося SQL Server Management Studio.

Іншою перевагою SQL Server є підтримка транзакцій, що важливо для операцій бронювання квитків. Під час створення резервування потрібно коректно записати дані користувача, інформацію про виставку, статус квитка та QR-код. Використання транзакцій зменшує ризик появи неповних або неправильних записів у базі даних.

SQL Server підтримує зовнішні ключі, обмеження унікальності, обмеження цілісності та каскадні операції. Завдяки цьому квиток можна створити лише для існуючого користувача та існуючої виставки, коментар може існувати лише за

наявності відповідної статті, а зв'язки між тегами, статтями та виставками залишаються логічно коректними.

PostgreSQL, MySQL, SQLite та MongoDB також могли бути можливими варіантами для цього проєкту. PostgreSQL є потужною реляційною СУБД, проте SQL Server краще підходить для даної системи через просту інтеграцію з технологіями Microsoft та ASP.NET Core. MySQL також підтримує реляційну модель, однак SQL Server зручніший для роботи з .NET, Entity Framework Core та інструментами розробки, які використовувалися в цьому проєкті. SQLite є простішим вбудованим варіантом бази даних, але він менш придатний для вебзастосунку з багатьма користувачами. MongoDB є нереляційною СУБД, тому не є найкращим варіантом для UArtFan, оскільки структура системи містить чітко визначені таблиці, зв'язки та зовнішні ключі.

1.6 Створення інформаційної бази

Інформаційна база системи UArtFan побудована на основі Microsoft SQL Server. Початкова структура бази даних була створена за допомогою SQL-запитів у SQL Server Management Studio. У цих запитах були визначені таблиці, первинні ключі, зовнішні ключі та зв'язки між основними сутностями.

Під час створення інформаційної бази спочатку були визначені основні сутності предметної області: Users, Roles, UserProfiles, UserOnboardings, Organizations, Organizers, Venues, Events, Writers, Articles, Tickets, Tags, EventTags, ArticleTags, UserTagPreferences, ArticleLikes, ArticleComments, ArticleViews, EventViews, UserHiddenArticles, UserHiddenEvents та ContentReports. Кожна сутність відповідає окремій таблиці бази даних і зберігає інформацію про певний об'єкт системи.

Зв'язки між таблицями встановлюються за допомогою первинних та зовнішніх ключів. Наприклад, таблиця Tickets містить посилання на Users та Events. Завдяки цьому можна зберігати інформацію про те, який користувач забронював квиток і на яку саме виставку. Крім того, первинні та зовнішні ключі забезпечують цілісність даних і не дозволяють створювати записи, які посилаються на неіснуючі об'єкти.

Бронювання квитків реалізовано через таблицю Tickets. Під час бронювання створюється запис, який містить ідентифікатор користувача, ідентифікатор виставки, дату резервування, статус, ціну та QR_Code. Запис у таблиці Tickets підтверджує резервування квитка, а QR_Code використовується як унікальне значення для подальшої генерації QR-коду.

Для прикладу можна навести логіку отримання зарезервованих квитків користувача. У базі даних для цього використовуються таблиці Tickets та Events. Їх об'єднання дозволяє отримати не лише дані про сам квиток, а й назву виставки та дати її проведення. Відповідний SQL-запит наведено в додатку А у лістингу А.1.

Категоризація виставок і статей здійснюється за допомогою тегів. Для цього використовуються таблиці Tags, EventTags та ArticleTags. Таблиця Tags зберігає самі теги, EventTags пов'язує теги з виставками, а ArticleTags – зі статтями. Завдяки цьому контент можна групувати за тематичними категоріями та використовувати ці дані під час фільтрації і формування рекомендацій. SQL-запит для отримання тегів, пов'язаних із конкретною виставкою, подано в додатку А у лістингу А.2.

Для роботи рекомендаційного механізму використовується інформація про інтереси користувача. Вона зберігається у таблиці UserTagPreferences, де для кожного користувача і тега може зберігатися оцінка зацікавленості. Чим частіше користувач взаємодіє з контентом певної тематики, тим більшим може ставати значення Score для відповідного тега. Також у рекомендаціях можуть враховуватися дані з ArticleViews, EventViews, UserHiddenArticles, UserHiddenEvents та UserOnboardings.

Для роботи зі скаргами використовується таблиця ContentReports. Вона містить інформацію про користувача, який подав скаргу, об'єкт скарги, її категорію, причину, статус і дату створення. Завдяки цьому адміністратор може переглядати скарги та приймати рішення щодо подальших дій.

Таким чином, інформаційна база UArtFan створена засобами Microsoft SQL Server. Її структура відповідає логічній моделі даних системи та дозволяє зберігати дані, необхідні для роботи основних функцій вебзастосунку: перегляду

виставок, публікації статей, бронювання квитків, роботи з тегами, скаргами та рекомендаціями. SQL-запити, наведені у цьому підрозділі, демонструють принцип отримання пов'язаних даних, а в програмному коді така взаємодія реалізована через Entity Framework Core.

1.7 Організація доступу до інформаційної бази

Доступ до інформаційної бази UArtFan організовано через серверну частину вебзастосунку, реалізовану на основі ASP.NET Core MVC. Користувач не звертається до бази даних напряму: усі дії виконуються через вебінтерфейс, після чого відповідний контролер обробляє запит і взаємодіє з базою даних через Entity Framework Core.

Основним компонентом для доступу до даних є `AppDbContext`. Він містить `DbSet` для таблиць системи та дозволяє працювати з ними як із C#-моделями. Загальна схема взаємодії виглядає так: користувач відкриває сторінку в браузері, контролер отримує запит, звертається до `AppDbContext`, Entity Framework Core формує запит до SQL Server, а результат передається у Razor View для відображення на сторінці.

Для доступу до даних у програмному коді використовуються LINQ-запити Entity Framework Core. Вони дозволяють фільтрувати, сортувати та отримувати пов'язані дані з кількох таблиць. Наприклад, під час відкриття сторінки деталей виставки контролер отримує не тільки сам запис `Events`, а й пов'язані дані про місце проведення, організатора, організацію та теги виставки. Приклад LINQ-запиту для отримання деталей виставки наведено в додатку А у лістингу А.3.

У цьому фрагменті метод `Include` використовується для завантаження пов'язаних сутностей разом із виставкою. Метод `ThenInclude` дозволяє отримати вкладені пов'язані дані, наприклад організацію організатора або теги, що належать до виставки.

Бронювання квитків виконується через `TicketsController`. Після натискання кнопки бронювання система перевіряє користувача, отримує дані про виставку, створює запис у таблиці `Tickets` і формує унікальне значення `QR_Code` для подальшої генерації QR-коду.

Доступ до дій системи контролюється на рівні вебзастосунку відповідно до ролі користувача. Це дозволяє обмежити доступ до функцій автора, організатора та адміністратора. Дані про статті, коментарі, вподобання, рекомендації та скарги також отримуються або змінюються через відповідні контролери та моделі EF Core.

Для відображення місць проведення виставок на карті в таблиці Venues зберігаються координати Latitude та Longitude. Ці дані отримуються через серверну частину застосунку та використовуються разом із Google Maps API для показу місця проведення виставки на інтерактивній карті.

Отже, доступ до інформаційної бази UArtFan реалізовано через ASP.NET Core MVC та EF Core. Такий підхід дозволяє відокремити користувацький інтерфейс від прямої роботи з базою даних, контролювати права доступу та виконувати операції з даними через C#-моделі.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

1.8 Організаційна структура програмного забезпечення

Програмне забезпечення UArtFan побудоване за структурою ASP.NET Core MVC, тому основні частини програми розділені між інтерфейсом, контролерами, моделями, сервісами та роботою з базою даних. Такий поділ був зручний для розробки, тому що кожна частина системи відповідає за свою задачу і не змішується з іншими.

Користувацький інтерфейс реалізований через Razor Views. У ньому знаходяться сторінки для перегляду виставок, статей, квитків та інших розділів системи. Для оформлення сторінок і роботи клієнтської частини використовується папка wwwroot, де зберігаються CSS-файли, JavaScript-файли, зображення та інші статичні ресурси.

Запити користувача обробляються контролерами. Наприклад, HomeController використовується для головних сторінок, EventsController – для роботи з виставками, ArticlesController – для статей, коментарів і вподобань, а TicketsController – для бронювання квитків. Окремі адміністративні контролери потрібні для роботи зі скаргами, користувачами та контентом.

Частина логіки винесена в окремий рівень Application / Services. Тут розміщена логіка рекомендацій, бронювання квитків, формування QR-кодів, модерації скарг і відображення місць проведення на карті. Це зроблено для того, щоб контролери не містили всю логіку одразу, а тільки приймали запити та передавали їх на обробку.

Моделі системи зберігаються в частині Domain / Models. Вони описують основні об'єкти UArtFan: користувачів, ролі, профілі, виставки, місця проведення, організації, статті, коментарі, вподобання, квитки, теги, скарги та дані для рекомендацій. Ці моделі пов'язані з таблицями бази даних і використовуються під час роботи контролерів та сервісів.

За роботу з базою даних відповідає рівень Data Access. Основним класом тут є ApplicationDbContext, через який програма звертається до таблиць SQL Server. Для

отримання, фільтрації та оновлення даних використовуються LINQ-запити Entity Framework Core, а підключення до бази задається через рядок підключення. Окремо можна виділити інфраструктурні компоненти, які використовуються в системі. SQL Server відповідає за збереження даних, Google Maps API використовується для показу місць проведення виставок на карті, QRCode – для створення QR-кодів квитків, а QuestPDF – для формування PDF-документів.

Отже, організаційна структура UArtFan дозволяє розділити інтерфейс, обробку запитів, прикладну логіку, моделі та доступ до бази даних. Це спрощує підтримку проєкту й робить структуру вебзастосунку більш зрозумілою. На рис. 3.1 наведено організаційну структуру програмного забезпечення UArtFan.

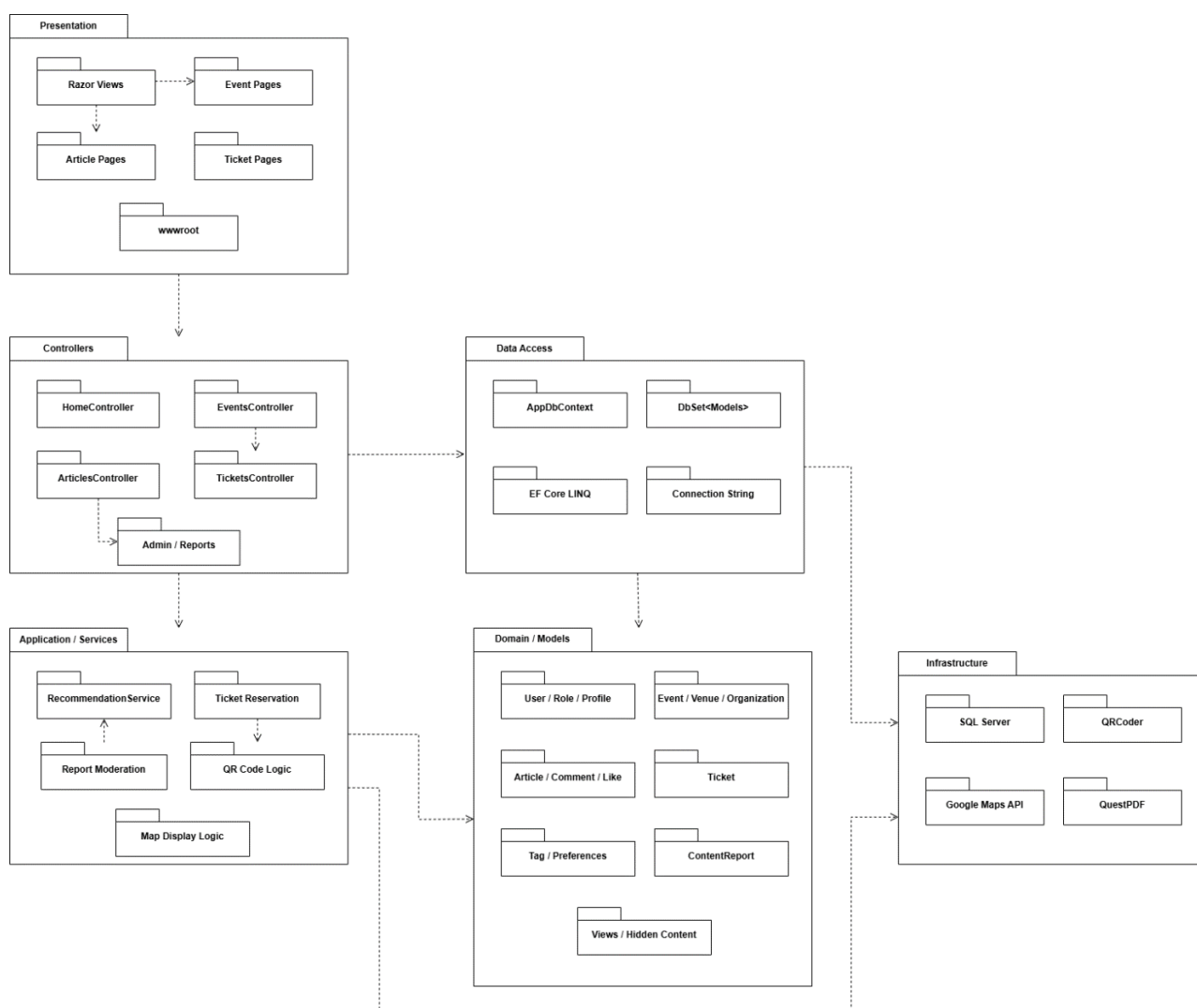


Рис. 3.1 Організаційна структура програмного забезпечення UArtFan

У табл. 3.1 наведено перелік основних компонентів програмного забезпечення UArtFan із зазначенням їх типу та функціонального призначення.

Таблиця 3.1

Компоненти програмного забезпечення UArtFan

Компонент	Тип	Призначення
1	2	3
Program.cs	Конфігураційний файл запуску	Налаштовує запуск вебзастосунку, підключення сервісів, маршрутизацію та роботу з базою даних
AppDbContext	Клас доступу до даних	Забезпечує зв'язок між моделями C# та таблицями SQL Server через Entity Framework Core
HomeController	Контролер	Обробляє запити до головних сторінок вебзастосунку
EventsController	Контролер	Забезпечує перегляд виставок, роботу з деталями подій, місцями проведення та пов'язаними даними
ArticlesController	Контролер	Відповідає за роботу зі статтями, коментарями, вподобаннями та переглядами
TicketsController	Контролер	Реалізує бронювання квитків, збереження QR_Code та перегляд квитків користувача
AdminController / ContentReportsController	Контролер	Використовується для адміністративних функцій, перегляду скарг, керування користувачами та контентом
RecommendationService	Сервіс	Формує персоналізовані рекомендації на основі тегів, переглядів, бронювань і прихованого контенту
Models	Моделі даних	Описують сутності системи: Users, Events, Articles, Tickets, Tags, Venues, ContentReports та інші
Razor Views	Представлення	Формують HTML-сторінки для користувача на основі даних, отриманих від контролерів
wwwroot	Каталог статичних ресурсів	Містить CSS, JavaScript, зображення та інші файли інтерфейсу
SQL Server	СУБД	Зберігає інформаційну базу системи UArtFan
Google Maps API	Зовнішній сервіс	Використовується для відображення місць проведення виставок на карті

Таблиця 3.1 (продовження)

1	2	3
QRCoder	Бібліотека	Генерує QR-коди для заброньованих квитків

QuestPDF	Бібліотека	Використовується для формування PDF-документів, пов'язаних із квитками
----------	------------	--

Таблиця 3.1 (закінчення)

1.9 Вибір інструментарію для створення ППЗ

Для створення UArtFan було використано набір інструментів, який підходить для вебзастосунку з базою даних і різними ролями користувачів. UArtFan побудований на ASP.NET Core MVC, тому проєкт розділений на контролери, моделі, представлення, сервіси та роботу з базою даних.

Основна частина проєкту написана мовою C#. Вона використовується для серверної логіки, роботи з контролерами, моделями та сервісами. Для відображення сторінок застосовуються Razor Views, а зовнішній вигляд інтерфейсу створюється за допомогою HTML, CSS і JavaScript. Статичні файли, зокрема стилі, скрипти та зображення, розміщені у папці wwwroot.

Для роботи з базою даних використовується Entity Framework Core. Через ApplicationDbContext застосунок звертається до таблиць SQL Server і працює з ними через C#-моделі. Це дозволяє виконувати основні операції з даними за допомогою LINQ-запитів. Сама база даних створена в Microsoft SQL Server, а для роботи з її структурою використовувалося SQL Server Management Studio.

У проєкті також застосовуються додаткові інструменти для окремих функцій. Google Maps API використовується для показу місць проведення виставок на карті, оскільки для кожного місця зберігаються координати Latitude та Longitude. QRCoder потрібен для створення QR-кодів квитків на основі значення QR_Code, а QuestPDF використовується для формування PDF-документів, пов'язаних із квитками.

Розробка виконувалася у Visual Studio Code. У ньому редагувалися файли C#, Razor Views, HTML, CSS і JavaScript, а також запускалися команди .NET. У результаті обрані інструменти дозволили реалізувати основні функції UArtFan: перегляд виставок, роботу зі статтями, бронювання квитків, QR-коди, карту, скарги та рекомендації.

У табл. 3.2 наведено відповідність між основними функціями системи UArtFan та інструментами, які були використані для їх реалізації. Такий поділ показує, які технології застосовуються не окремо самі по собі, а для виконання конкретних задач вебзастосунку.

Таблиця 3.2

Відповідність інструментів функціям системи

Функція UArtFan	Використані інструменти
Перегляд виставок і статей	ASP.NET Core MVC, Razor Views, EF Core
Бронювання квитків	TicketsController, EF Core, SQL Server
Генерація QR-кодів	QRCode, QR_Code
Формування PDF-квитка	QuestPDF
Відображення місця проведення	Google Maps API, Latitude, Longitude
Рекомендації	RecommendationService, UserTagPreferences, Views
Адміністрування контенту	AdminController, ContentReports, SQL Server
Робота з базою даних	SQL Server, SSMS, ApplicationDbContext

1.10 Алгоритмізація та програмування програмних модулів

У цьому підрозділі розглянуто алгоритмізацію та програмну реалізацію основних модулів інформаційної системи UArtFan. Основна увага приділяється тим частинам системи, які забезпечують бронювання квитків, роботу рекомендаційного механізму, обробку скарг, коментування статей та відображення виставок на карті. Для реалізації цих функцій використовуються контролери ASP.NET Core MVC, моделі Entity Framework Core та відповідні таблиці бази даних.

1.1.8 Модуль бронювання квитка

Одним із ключових модулів системи є модуль бронювання квитків. Він забезпечує створення запису про резервування квитка для конкретного користувача та конкретної виставки.

Після вибору виставки користувач може перейти до бронювання квитка. Для цього він натискає кнопку бронювання на сторінці конкретної виставки. Запит передається до TicketsController, який перевіряє, чи користувач

авторизований, отримує дані про обрану виставку та створює новий запис у таблиці Tickets.

Під час бронювання в базу даних записуються ідентифікатор користувача, ідентифікатор виставки, дата резервування, ціна, статус квитка та унікальне значення QR_Code. У системі UArtFan бронювання не є онлайн-оплатою, тому запис у таблиці Tickets підтверджує саме резервування місця на виставку.

Фрагмент C#-коду для створення записів квитків наведено в додатку А у лістингу А.4.

Після збереження запису в базі даних користувач може переглянути свій квиток у відповідному розділі системи. Значення QR_Code використовується для подальшої генерації QR-коду, який дозволяє ідентифікувати конкретне резервування.

Якщо користувач не авторизований, система не створює запис квитка, а перенаправляє його на сторінку входу. Якщо виставка не знайдена або дані є некоректними, користувач отримує повідомлення про помилку. Така перевірка потрібна для того, щоб у таблиці Tickets не створювалися неправильні або неповні записи.

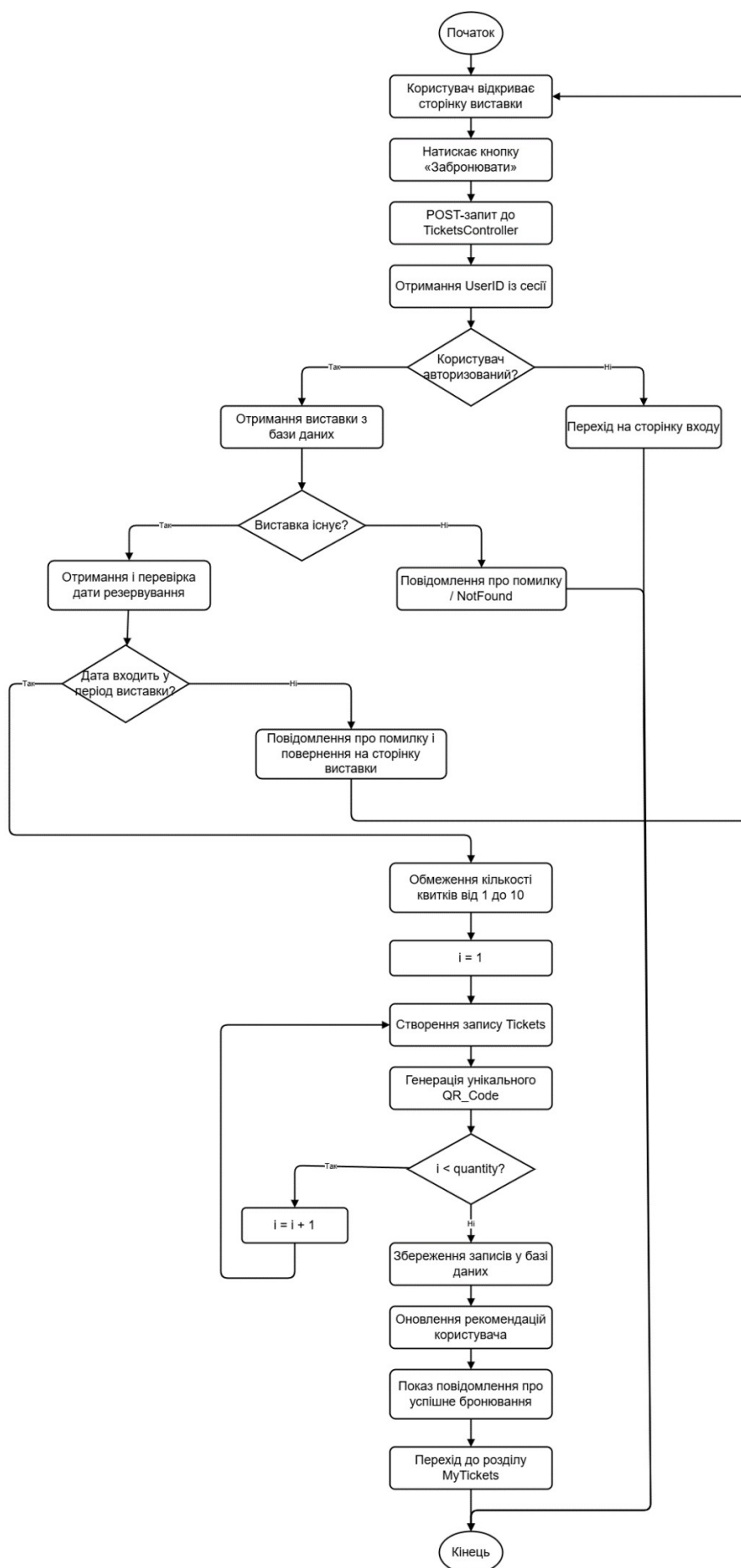


Рис. 3.2 Блок-схема модулю бронювання квитка

1.1.9 Модуль рекомендаційного механізму

Ще одним важливим модулем у UArtFan є рекомендації. Вони потрібні для того, щоб користувач бачив не випадкові виставки або статті, а матеріали, які більше відповідають його інтересам. У цьому проєкті рекомендації реалізовано через систему балів для тегів, які пов'язують користувачів із певними темами.

Основна таблиця для цього – UserTagPreferences. У ній зберігаються UserID, TagID і Score, тобто для кожного користувача система може зберігати оцінку зацікавленості в окремих тегах. Якщо користувач взаємодіє з контентом, пов'язаним із певним тегом, значення Score змінюється. Наприклад, перегляд виставки або читання статті додає 1 бал до відповідних тегів, коментар або вподобання статті додає 2 бали, а бронювання виставки додає 4 бали. Бронювання має більшу вагу, ніж звичайний перегляд, оскільки така дія показує вищий рівень зацікавленості користувача.

Фрагмент C#-коду для оновлення оцінки інтересу користувача за тегами виставки наведено в додатку А у лістингу А.5. У цьому фрагменті метод AddPreferenceScoreForEvent отримує ідентифікатор користувача, ідентифікатор виставки та кількість балів, яку потрібно додати. Система знаходить усі теги, пов'язані з обраною виставкою, після чого для кожного з них створює новий запис у UserTagPreferences або оновлює вже наявне значення Score.

Якщо користувач позначає контент як нецікавий, оцінка інтересу зменшується: для виставки на 4 бали, а для статті на 3 бали. Завдяки цьому система поступово накопичує інформацію про те, які теми користувач частіше переглядає, бронює або приховує.

Коли потрібно показати рекомендації, система спочатку визначає, які теги мають найбільше значення Score для конкретного користувача. Потім за цими тегами підбираються відповідні виставки або статті. Окремо враховується прихований контент: якщо користувач уже позначив певну виставку або статтю як нецікаву, вона не повинна знову з'являтися в рекомендаціях. Для цього використовуються таблиці UserHiddenEvents та UserHiddenArticles. На рис. 3.3 наведено блок-схему алгоритму формування рекомендацій у системі UArtFan.

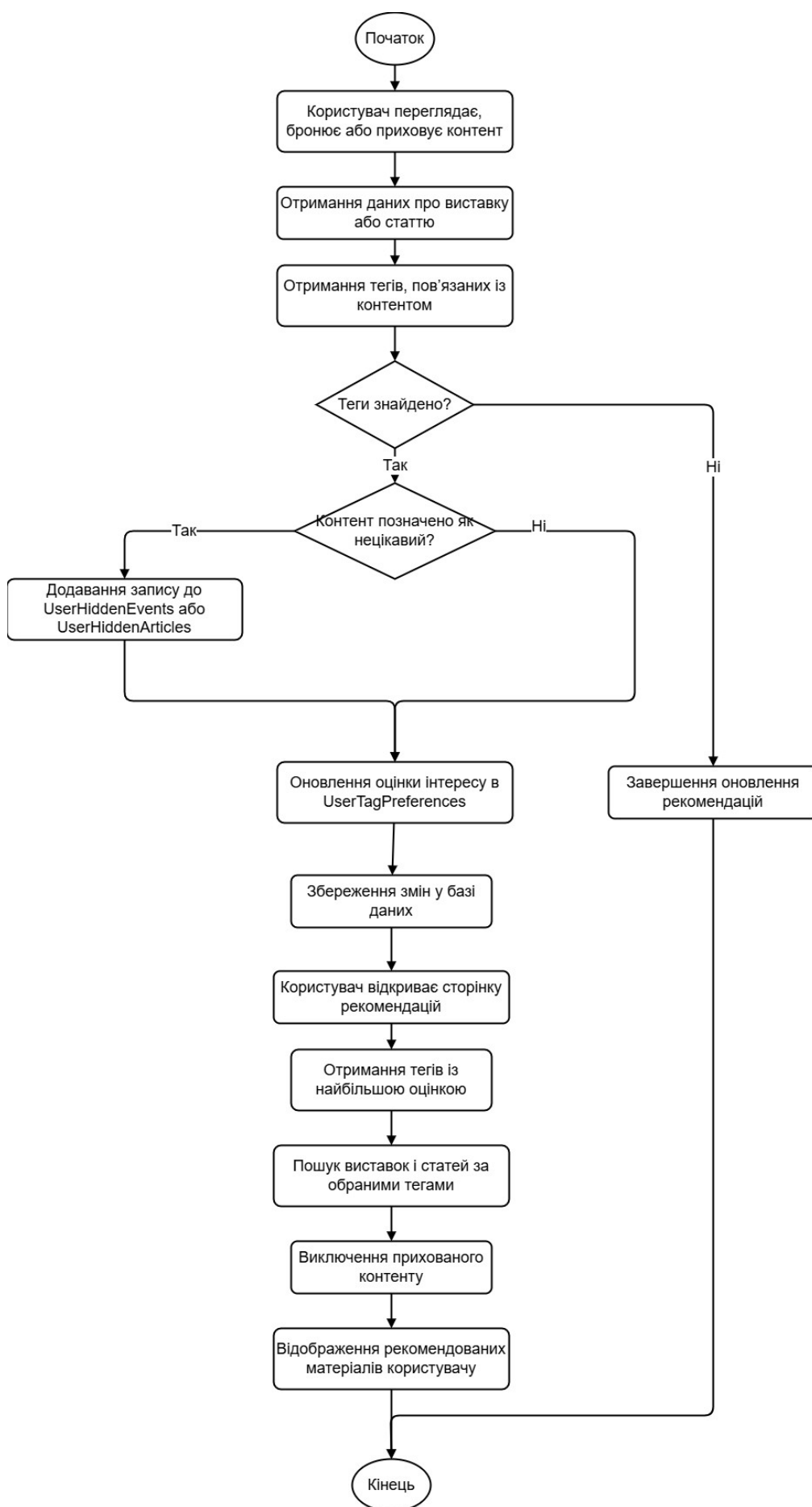


Рис. 3.3 Блок-схема алгоритму формування рекомендацій

У результаті модуль рекомендацій змінює список матеріалів відповідно до дій користувача. Система враховує перегляди, бронювання, взаємодію зі статтями та прихований контент, тому рекомендації стають більш персоналізованими.

1.1.10 Алгоритм подання та обробки скарги на контент

Ще один модуль UArtFan пов'язаний зі скаргами на контент. Він дозволяє користувачу повідомити адміністратору про проблему зі статтею або виставкою, наприклад неправильний опис чи помилкову інформацію. Після натискання кнопки скарги система перевіряє авторизацію користувача. Якщо він увійшов у систему, відкривається форма для вибору категорії та введення причини скарги, після чого дані передаються до контролера.

Скарги зберігаються в таблиці ContentReports. У записі вказується користувач, який подав скаргу, тип контенту, ідентифікатор статті або виставки, категорія, причина, статус і дата створення. Якщо користувач не вказав категорію або причину, система підставляє значення за замовчуванням: Other для категорії та No details provided. для причини. Початковий статус скарги має значення Open, тобто вона ще очікує на перегляд адміністратором.

Фрагмент C#-коду для створення скарги на статтю наведено в додатку А у лістингу А.6.

Схожа логіка використовується і для скарги на виставку. У цьому випадку замість ArticleID записується EventID, а поле ContentType має значення Event.

Після збереження скарги адміністратор може відкрити список скарг в адміністративному розділі, переглянути причину та визначити, що потрібно зробити з відповідним матеріалом. Після розгляду статус скарги може бути змінений. Так у системі зберігається не тільки повідомлення від користувача, а й результат його подальшої обробки.

На рис. 3.4 наведено блок-схему алгоритму подання та обробки скарги на контент.

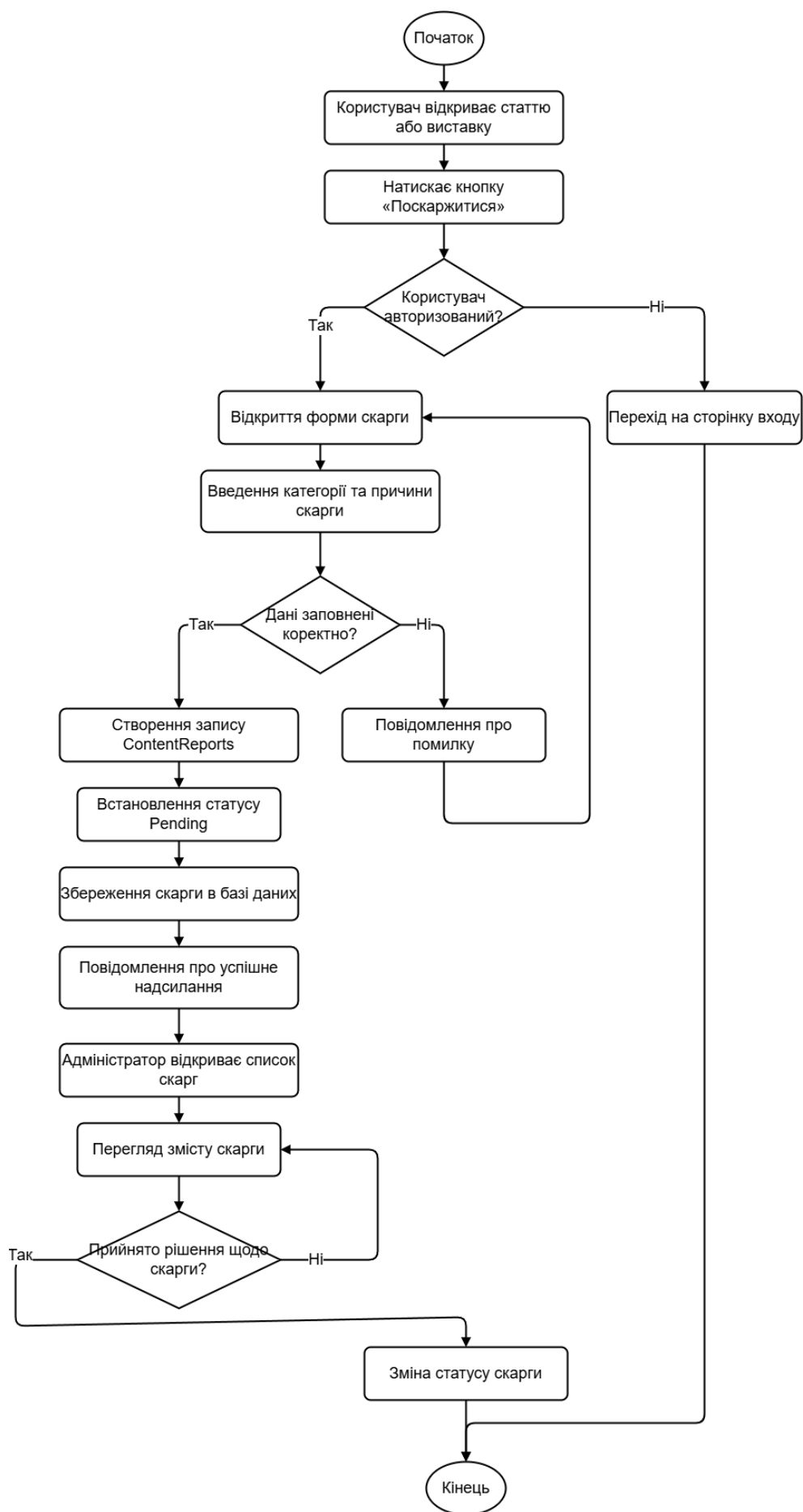


Рис. 3.4 Блок-схема алгоритму подання та обробки скарги

Таким чином, модуль скарг забезпечує зв'язок між користувачами та адміністратором. Користувач може повідомити про проблему, а адміністратор отримує можливість переглянути скаргу й виконати подальші дії щодо контенту.

1.1.11 Модуль коментарів та вподобань статей

Ще одним функціональним модулем UArtFan є модуль коментарів та вподобань до статей. Він потрібен для того, щоб користувачі могли не тільки читати матеріали, а й взаємодіяти з ними. У цьому модулі користувач може залишити коментар до статті або поставити вподобання.

Після відкриття сторінки статті система отримує саму статтю, пов'язані з нею коментарі та інформацію про вподобання. Якщо користувач залишає коментар, текст передається до контролера, після чого створюється запис у таблиці ArticleComments. У цьому записі зберігається ідентифікатор статті, ідентифікатор користувача, текст коментаря та дата публікації.

Фрагмент C#-коду для створення коментаря наведено в додатку А у лістингу А.7.

У наведеному фрагменті створюється об'єкт ArticleComment, у який записуються дані про статтю, користувача, текст коментаря та час публікації. Після цього коментар додається до колекції ArticleComments і зберігається в базі даних за допомогою методу SaveChangesAsync. Вподобання статей зберігаються в таблиці ArticleLikes. Завдяки цьому система може визначити, чи конкретний користувач уже поставив вподобання певній статті. Це потрібно для того, щоб не створювати дублікати та правильно відображати стан кнопки вподобання на сторінці.

Цей модуль є простішим за бронювання або рекомендації, тому для нього не потрібна окрема блок-схема. Його основна логіка полягає в отриманні даних зі сторінки, перевірці користувача та збереженні запису в базі даних..

1.1.12 Модуль відображення місць проведення виставок на карті

Окремим модулем у системі UArtFan є відображення місць проведення виставок на карті. Він використовується для того, щоб користувач міг не тільки прочитати інформацію про виставку, а й побачити її розташування візуально.

Для роботи цього модуля в таблиці Venues зберігаються координати місця проведення: Latitude та Longitude. Під час завантаження головної сторінки система отримує дані про виставки разом із пов'язаними місцями проведення. Для цього у HomeController використовується Include, який дозволяє одразу завантажити Venue разом із Event.

Фрагмент C#-коду для завантаження виставок і отримання координат для карти наведено в додатку А у лістингу А.8.

У наведеному фрагменті спочатку формується запит до таблиці Events із підвантаженням пов'язаного місця проведення Venue та тегів виставки. Після цього під час побудови маркерів для карти з об'єкта Venue беруться значення Latitude та Longitude. Якщо координати відсутні, система використовує резервні значення fallback із невеликим зміщенням, щоб маркер усе одно міг бути відображений.

На сторінці карта показує маркери відповідних місць проведення. Це зручно для користувача, тому що він може одразу побачити, де саме проходить виставка, і краще зорієнтуватися. Для коректної роботи цього модуля важливо, щоб у таблиці Venues були правильно заповнені координати місць проведення.

Цей модуль не потребує окремої блок-схеми, оскільки його логіка є досить прямою: завантажити виставки разом із місцями проведення, отримати координати з Venue та передати їх для відображення маркерів на карті через Google Maps API.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

1.11 Тестування системи

Тестування UArtFan виконувалося вручну через вебінтерфейс. Я перевіряла основні сценарії, які повинен виконувати користувач у системі: реєстрацію, вхід, перегляд виставок і статей, бронювання квитків, перегляд квитків, роботу з коментарями, вподобаннями, рекомендаціями, скаргами та адміністративними функціями.

Такий спосіб тестування підходить для цього проєкту, тому що важливо було перевірити не тільки окремі частини коду, а й повний шлях користувача на сайті. Наприклад, під час бронювання потрібно перевірити не лише натискання кнопки, а й створення запису в базі даних, правильне збереження кількості квитків, формування QR_Code та відображення результату в розділі квитків.

Під час тестування також перевірялася взаємодія вебзастосунку з базою даних SQL Server. Після виконання дій на сайті я перевіряла, чи правильно створюються або змінюються записи у відповідних таблицях: Tickets, ArticleComments, ArticleLikes, UserTagPreferences, ContentReports та інших.

Для перевірки були складені тест-кейси. У кожному з них зазначено назву перевірки, передумови, послідовність дій і очікуваний результат. Це дозволило пройти основні функції системи в одному порядку і перевірити, чи працює сайт так, як було заплановано.

У табл. 4.1 наведено тест-кейси функціонального тестування системи UArtFan. Вони охоплюють основні дії користувача та адміністратора в системі: реєстрацію, вхід, перегляд контенту, бронювання квитків, роботу з коментарями, вподобаннями, рекомендаціями, скаргами та перевірку доступу до адміністративних функцій.

Відповідність інструментів функціям системи

ID	Назва	Передумови	Кроки	Очікуваний результат
1	2	3	4	5
ТС-01	Реєстрація користувача	Вебзастосунок запущено, користувач не має акаунта	1. Відкрити сторінку реєстрації. 2. Ввести ім'я користувача, email і пароль. 3. Натиснути кнопку реєстрації.	Новий користувач створюється в системі, дані зберігаються в таблиці Users, користувач може виконати вхід.
ТС-02	Вхід користувача	Користувач уже зареєстрований	1. Відкрити сторінку входу. 2. Ввести правильний email і пароль. 3. Натиснути кнопку входу.	Користувач входить у систему та отримує доступ до функцій відповідно до своєї ролі.
ТС-03	Вхід з неправильними даними	Користувач відкрив сторінку входу	1. Ввести неправильний email або пароль. 2. Натиснути кнопку входу.	Система не виконує вхід і показує повідомлення про помилку.
ТС-04	Перегляд списку виставок	У базі даних є записи в таблиці Events	1. Відкрити сторінку виставок.	На сторінці відображається список доступних виставок із основною інформацією.
ТС-05	Перегляд деталей виставки	У базі даних є виставка з пов'язаним місцем проведення	1. Відкрити список виставок. 2. Обрати конкретну виставку.	Відкривається сторінка деталей виставки з описом, датами, ціною, місцем проведення та іншою інформацією.
ТС-06	Відображення місця проведення на карті	Для місця проведення задано Latitude та Longitude	1. Відкрити сторінку виставки з координатами місця проведення.	На сторінці відображається карта Google Maps з маркером місця проведення.
ТС-07	Бронювання одного квитка	Користувач авторизований, виставка існує	1. Відкрити сторінку виставки. 2. Обрати бронювання одного квитка. 3. Натиснути кнопку бронювання.	У таблиці Tickets створюється один запис із UserID, EventID, датою резервування, статусом і QR_Code.

Таблиця 4.1 (продовження)

1	2	3	4	5
ТС-	Бронювання	Користувач	1. Відкрити сторінку	У таблиці Tickets

08	кількох квитків	авторизований, виставка існує	виставки. 2. Вказати кількість квитків. 3. Натиснути кнопку бронювання.	створюється відповідна кількість записів, кожен квиток має окремий QR_Code.
ТС-09	Обмеження кількості квитків	Користувач авторизований	1. Спробувати вказати некоректну або надто велику кількість квитків. 2. Натиснути кнопку бронювання.	Система обмежує кількість квитків допустимим діапазоном і не створює некоректні записи.
ТС-10	Перегляд квитків користувача	Користувач авторизований і має заброньовані квитки	1. Перейти до розділу MyTickets.	Відображаються квитки поточного користувача з даними про виставку, статус і QR-код.
ТС-11	Генерація QR-коду квитка	У користувача є заброньований квиток із QR_Code	1. Відкрити сторінку конкретного квитка.	Система генерує та показує QR-код, пов'язаний із відповідним резервуванням.
ТС-12	Формування PDF-квитка	У користувача є заброньований квиток	1. Відкрити квиток. 2. Натиснути кнопку формування або завантаження PDF.	Створюється PDF-документ із даними квитка та QR-кодом.
ТС-13	Перегляд статті	У базі даних є опубліковані статті	1. Відкрити сторінку статей. 2. Обрати статтю.	Відкривається сторінка статті з текстом, зображенням, тегами, коментарями та вподобаннями.
ТС-14	Додавання коментаря до статті	Користувач авторизований, стаття існує	1. Відкрити статтю. 2. Ввести текст коментаря. 3. Натиснути кнопку додавання.	У таблиці ArticleComments створюється новий запис, коментар відображається на сторінці статті.
ТС-15	Додавання вподобання до статті	Користувач авторизований, стаття існує	1. Відкрити статтю. 2. Натиснути кнопку вподобання.	У таблиці ArticleLikes створюється або оновлюється запис про реакцію користувача.
ТС-16	Оновлення рекомендацій після взаємодії	Користувач авторизований, контент має теги	1. Відкрити виставку або статтю з тегами. 2. Виконати дію, яка впливає на рекомендації.	У таблиці UserTagPreferences оновлюється Score для відповідних тегів.

Таблиця 4.1 (продовження)

1	2	3	4	5
ТС-17	Формування рекомендацій	Користувач має записи в	1. Відкрити розділ рекомендацій.	Система показує виставки або статті,

		UserTagPreferences		пов'язані з тегами, які мають більший Score.
ТС -18	Приховування нецікавого контенту	Користувач авторизований, контент відображається в рекомендаціях	1. Натиснути кнопку приховування статті або виставки.	У UserHiddenArticles або UserHiddenEvents створюється запис, прихований контент більше не відображається в рекомендаціях.
ТС -19	Подання скарги на контент	Користувач авторизований, існує стаття або виставка	1. Відкрити статтю або виставку. 2. Натиснути кнопку скарги. 3. Вибрати категорію та ввести причину. 4. Надіслати скаргу.	У таблиці ContentReports створюється запис зі статусом Pending.
ТС -20	Подання скарги без авторизації	Користувач не авторизований	1. Відкрити статтю або виставку. 2. Спробувати подати скаргу.	Система не створює скаргу та пропонує користувачу увійти в акаунт.
ТС -21	Перегляд скарг адміністратором	Користувач має роль адміністратора, у ContentReports є записи	1. Увійти як адміністратор. 2. Відкрити розділ скарг.	Адміністратор бачить список скарг із категорією, причиною, статусом і пов'язаним контентом.
ТС -22	Зміна статусу скарги	Адміністратор відкрив скаргу	1. Переглянути скаргу. 2. Змінити її статус після розгляду.	Статус скарги оновлюється в таблиці ContentReports.
ТС -23	Доступ до адміністративних функцій	Користувач має звичайну роль	1. Спробувати перейти до адміністративного розділу.	Доступ до адміністративних функцій обмежується, користувач не може керувати скаргами, ролями або контентом.
ТС -24	Робота з роллю автора	Користувач має роль автора	1. Увійти в систему як автор. 2. Перейти до розділу роботи зі статтями.	Автор отримує доступ до функцій, пов'язаних зі створенням або редагуванням статей.
ТС -25	Робота з роллю організатора	Користувач має роль організатора	1. Увійти в систему як організатор. 2. Перейти до розділу виставок.	Організатор має доступ до функцій, пов'язаних із виставками та місцями проведення.

Таблиця 4.1 (закінчення)

1.12 Вимоги до апаратного та програмного забезпечення

UArtFan працює як вебзастосунок. Для його роботи потрібен сервер із програмною частиною, база даних SQL Server і пристрій користувача з

браузером. Окрему програму на комп'ютер або телефон встановлювати не потрібно, бо доступ до системи виконується через сайт.

Серверна частина створена на ASP.NET Core MVC. Вона приймає запити від користувачів, звертається до бази даних через Entity Framework Core і повертає готові сторінки у браузер. Тому для роботи системи важливо, щоб були налаштовані і сервер, і база даних, і середовище запуску застосунку.

1.1.13 Вимоги до клієнтської частини

Для роботи з UArtFan користувачу достатньо комп'ютера, ноутбука, планшета або смартфона з доступом до Інтернету. Сайт можна відкривати у Google Chrome, Microsoft Edge, Mozilla Firefox або іншому сучасному браузері, який підтримує HTML, CSS і JavaScript.

JavaScript має бути увімкнений, тому що він використовується для окремих інтерактивних елементів сторінок і для роботи карти. Підключення до Інтернету також потрібне для завантаження сторінок із сервера та відображення Google Maps API.

Основні вимоги до клієнтського пристрою:

- сучасний веббраузер;
- стабільне підключення до Інтернету;
- увімкнена підтримка JavaScript;
- оперативна пам'ять – від 2 ГБ;
- достатньо вільного місця для роботи браузера і кешу;
- екран, на якому зручно переглядати сторінки сайту.

Вимоги до клієнтського пристрою не є високими, тому що основна обробка даних виконується на сервері. Пристрій користувача переважно відповідає за відкриття сторінок, введення даних і надсилання запитів до системи.

1.1.14 Вимоги до серверної частини

Серверну частину UArtFan можна запускати локально на комп'ютері під час розробки. Для постійної роботи системи краще використовувати окремий

сервер або хостинг, щоб сайт був доступний користувачам незалежно від комп'ютера розробника.

Сервер повинен запускати ASP.NET Core MVC-застосунок, обробляти запити користувачів і мати доступ до бази даних SQL Server. Також потрібне підключення до Інтернету, оскільки система використовує Google Maps API для показу місць проведення виставок.

Рекомендована конфігурація серверної частини:

- процесор – 2 ядра або більше;
- оперативна пам'ять – 4 ГБ або більше;
- вільне місце на диску – не менше 5 ГБ;
- операційна система – Windows 10/11 або Windows Server;
- підключення до Інтернету для роботи зовнішніх сервісів.

1.1.15 Програмне забезпечення серверної частини

Для запуску UArtFan потрібно встановити програмне забезпечення, яке забезпечує роботу ASP.NET Core MVC, бази даних і залежностей проєкту. Під час розробки також використовувались інструменти для редагування коду та перевірки бази даних.

Необхідне програмне забезпечення:

- .NET SDK або .NET Runtime – для запуску ASP.NET Core MVC-застосунку;
- Microsoft SQL Server – для збереження даних системи;
- SQL Server Management Studio – для перегляду структури бази даних і виконання SQL-запитів;
- Visual Studio Code – для редагування файлів проєкту;
- NuGet-пакети, зокрема Entity Framework Core, QRCode та QuestPDF.

Також потрібно правильно вказати рядок підключення до бази даних у конфігураційному файлі застосунку. Після запуску серверна частина приймає запити, працює з SQL Server через Entity Framework Core і відображає результат через Razor Views.

1.13 Склад інсталяційного пакету

Інсталяційний пакет системи UArtFan містить файли, необхідні для запуску вебзастосунку, налаштування підключення до бази даних та роботи основних функцій системи. Оскільки UArtFan є вебзастосунком, клієнтська частина не встановлюється окремо на пристрій користувача. Доступ до системи здійснюється через браузер після запуску серверної частини.

1.1.16 Склад програмної частини

Програмна частина UArtFan складається з файлів ASP.NET Core MVC-проєкту. До неї входять контролери, моделі, представлення, клас доступу до бази даних, сервіси та статичні ресурси. Ці файли забезпечують роботу сторінок сайту, обробку запитів користувачів, взаємодію з базою даних і виконання додаткових функцій, таких як рекомендації, бронювання квитків, генерація QR-кодів та формування PDF-документів.

До складу програмної частини входять:

- папка `Controllers` – містить контролери для обробки запитів користувачів;
- папка `Models` – містить моделі основних сутностей системи;
- папка `Views` – містить Razor Views для відображення сторінок вебзастосунку;
- папка `Data` – містить `AppDbContext` для роботи з базою даних;
- папка `Services` – містить окремі сервіси, зокрема логіку рекомендацій;
- папка `wwwroot` – містить CSS-файли, JavaScript-файли, зображення та інші статичні ресурси;
- файл `Program.cs` – містить основні налаштування запуску застосунку;
- файл `appsettings.json` – містить конфігураційні параметри, зокрема рядок підключення до бази даних.

Також до програмної частини належать залежності NuGet, які використовуються в проєкті. Зокрема, Entity Framework Core застосовується для роботи з базою даних, QRCode – для генерації QR-кодів квитків, а QuestPDF – для створення PDF-документів.

1.1.17 Склад бази даних

База даних UArtFan створюється в Microsoft SQL Server. До складу інсталяційного пакету можуть входити SQL-скрипти для створення таблиць або резервна копія бази даних, яку можна відновити в SQL Server. Це потрібно для того, щоб система могла працювати з уже підготовленою структурою інформаційної бази.

База даних містить таблиці для збереження основних даних системи: користувачів, ролей, профілів, організацій, організаторів, місць проведення, виставок, статей, квитків, тегів, коментарів, вподобань, переглядів, рекомендацій, прихованого контенту та скарг. Зв'язки між таблицями забезпечуються первинними та зовнішніми ключами.

Для правильної роботи системи після створення або відновлення бази даних потрібно перевірити рядок підключення в appsettings.json. Він має вказувати на потрібний екземпляр SQL Server і назву бази даних UArtFan.

1.1.18 Порядок запуску системи

Для запуску системи потрібно спочатку підготувати середовище виконання. На комп'ютері або сервері мають бути встановлені .NET SDK або .NET Runtime, Microsoft SQL Server та, за потреби, SQL Server Management Studio для перевірки бази даних.

Порядок запуску системи можна описати так:

- відкрити папку з проєктом UArtFan;
- встановити або відновити необхідні NuGet-пакети;
- створити або відновити базу даних у Microsoft SQL Server;
- перевірити рядок підключення до бази даних у файлі appsettings.json;

- запустити вебзастосунок командою `dotnet run` або через середовище розробки;
- відкрити адресу локального сервера у браузері;
- перевірити роботу основних функцій системи.

Після запуску серверна частина приймає запити користувачів, звертається до бази даних через Entity Framework Core та повертає сторінки вебзастосунку у браузер. Якщо система розгортається не локально, а на сервері або хостингу, додатково потрібно налаштувати доступ до сайту через мережу та переконатися, що сервер має доступ до бази даних і зовнішніх сервісів, наприклад Google Maps API.

Після виконання цих кроків система UArtFan готова до використання. Користувачі можуть відкривати сайт у браузері, переглядати виставки та статті, бронювати квитки, працювати з рекомендаціями, подавати скарги та користуватися іншими функціями відповідно до своєї ролі.

ВИСНОВКИ

Під час виконання бакалаврської кваліфікаційної роботи було розроблено інформаційну систему UArtFan. Це вебзастосунок для перегляду виставок українського мистецтва, читання статей, бронювання квитків і роботи з рекомендаціями. Основна ідея проєкту полягала в тому, щоб зібрати в одному місці інформацію про художні виставки, яка зазвичай розміщується на різних сайтах, афішних платформах або у соціальних мережах.

На початку роботи було визначено, які користувачі будуть працювати із системою і які функції їм потрібні. У системі передбачено кілька ролей: звичайний користувач, автор, організатор та адміністратор. Звичайний користувач може переглядати виставки й статті, бронювати квитки, залишати коментарі, ставити вподобання та подавати скарги. Автор працює зі статтями, організатор – з виставками та місцями проведення, а адміністратор має доступ до керування користувачами, контентом і скаргами.

Для роботи системи було спроектовано інформаційну базу. Вона реалізована в Microsoft SQL Server і містить таблиці для користувачів, ролей, профілів, організацій, місць проведення, виставок, статей, квитків, тегів, коментарів, вподобань, переглядів, рекомендацій, прихованого контенту та скарг. Під час проєктування було важливо правильно зв'язати таблиці між собою, тому в базі використовуються первинні та зовнішні ключі. Завдяки цьому, наприклад, квиток не існує окремо сам по собі, а пов'язаний із конкретним користувачем і конкретною виставкою.

Програмна частина UArtFan була створена на ASP.NET Core MVC мовою C#. Для доступу до бази даних використано Entity Framework Core та ApplicationDbContext. Інтерфейс реалізовано через Razor Views, HTML, CSS і JavaScript. Також у проєкті використовуються додаткові інструменти: Google Maps API для показу місця проведення виставки на карті, QRCode для створення QR-кодів квитків і QuestPDF для формування PDF-документів.

Під час розробки найбільше уваги було приділено модулям, які поєднують роботу інтерфейсу, контролерів і бази даних. До них належать бронювання квитків, формування рекомендацій і подання скарг на контент. Під час

бронювання система створює записи в таблиці Tickets і формує унікальне значення QR_Code. Рекомендації працюють через накопичення балів за тегами, тому система може поступово враховувати, які теми користувач переглядає або бронює частіше. Модуль скарг дозволяє користувачу повідомити адміністратора про проблему зі статтею або виставкою.

Також були реалізовані простіші, але важливі для роботи сайту функції: перегляд статей, додавання коментарів, вподобання статей, відображення місць проведення на карті, робота з ролями користувачів та адміністративні функції. Усі ці частини разом формують основну логіку вебзастосунку й дозволяють користувачам взаємодіяти з контентом, а не лише переглядати його

Після реалізації основних функцій було проведено ручне функціональне тестування через вебінтерфейс. Під час перевірки тестувалися реєстрація та вхід, перегляд виставок і статей, бронювання одного або кількох квитків, перегляд QR-кодів, коментарі, вподобання, рекомендації, подання скарг і доступ до адміністративних функцій. Окремо перевірялося, чи правильно створюються та змінюються записи в базі даних після дій користувача.

Практична цінність UArtFan полягає в тому, що система може бути використана як зручний інструмент для пошуку подій, пов'язаних з українським мистецтвом. Користувач може знайти виставку, прочитати пов'язану статтю, побачити місце проведення на карті та забронювати квиток. Для організаторів система дає можливість працювати з подіями, для авторів – зі статтями, а для адміністратора – контролювати контент і скарги.

Отже, поставлену мету можна вважати виконаною, оскільки в результаті було створено вебзастосунок з основними запланованими функціями. У майбутньому UArtFan можна покращити, додавши онлайн-оплату квитків, сповіщення для користувачів, детальнішу аналітику для організаторів, розширені фільтри пошуку та мобільну версію системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft. C# Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/csharp/> (дата звернення: 01.03.2026).
2. Microsoft. .NET Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/> (дата звернення: 02.03.2026).
3. Microsoft. ASP.NET Core Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core/> (дата звернення: 05.03.2026).
4. Microsoft. Overview of ASP.NET Core MVC [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core/mvc/overview> (дата звернення: 07.03.2026).
5. Microsoft. Razor syntax reference for ASP.NET Core [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core/mvc/views/razor> (дата звернення: 13.03.2026).
6. Microsoft. SQL Server Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/sql/sql-server/> (дата звернення: 20.02.2026).
7. Microsoft. SQL Server Management Studio Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms> (дата звернення: 22.02.2026).
8. Microsoft. Entity Framework Core Documentation [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення: 16.03.2026).
9. Microsoft. Entity Framework Core – DbContext [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/ef/core/dbcontext-configuration/> (дата звернення: 19.03.2026).
10. GeeksforGeeks. SQL Joins [Електронний ресурс]. – Режим доступу: <https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/> (дата звернення: 24.02.2026).

11. GeeksforGeeks. MVC Design Pattern [Электронный ресурс]. – Режим доступа: <https://www.geeksforgeeks.org/mvc-design-pattern/> (дата звернения: 06.03.2026).
12. Mozilla Developer Network. HTML: HyperText Markup Language [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/HTML> (дата звернения: 25.03.2026).
13. Mozilla Developer Network. CSS: Cascading Style Sheets [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата звернения: 27.03.2026).
14. Mozilla Developer Network. JavaScript Guide [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> (дата звернения: 30.03.2026).
15. Google LLC. Maps JavaScript API Documentation [Электронный ресурс]. – Режим доступа: <https://developers.google.com/maps/documentation/javascript> (дата звернения: 07.04.2026).
16. Google LLC. Maps JavaScript API Reference [Электронный ресурс]. – Режим доступа: <https://developers.google.com/maps/documentation/javascript/reference> (дата звернения: 09.04.2026).
17. Shane32. QRCoder Documentation [Электронный ресурс]. – Режим доступа: <https://github.com/Shane32/QRCoder/wiki> (дата звернения: 11.04.2026).
18. Shane32. QRCoder GitHub Repository [Электронный ресурс]. – Режим доступа: <https://github.com/Shane32/QRCoder> (дата звернения: 11.04.2026).
19. QuestPDF. QuestPDF Documentation [Электронный ресурс]. – Режим доступа: <https://www.questpdf.com/> (дата звернения: 12.04.2026).
20. Wasmer. Free Web Hosting [Электронный ресурс]. – Режим доступа: <https://wasmer.io/free-web-hosting> (дата звернения: 17.04.2026).

ДОДАТОК А

Лістинг А.1 – SQL-запит для отримання зарезервованих квитків користувача

Лістинг А.1

```
SELECT
    t.TicketID,
    t.ReservationDate,
    t.Status,
    t.QR_Code,
    e.Title AS EventTitle,
    e.StartDate,
    e.EndDate
FROM Tickets t
INNER JOIN Events e ON t.EventID = e.EventID
WHERE t.UserID = @UserID;
```

Лістинг А.2 – SQL-запит для отримання тегів виставки

Лістинг А.2

```
SELECT
    e.Title AS EventTitle,
    t.TagName
FROM EventTags et
INNER JOIN Events e ON et.EventID = e.EventID
INNER JOIN Tags t ON et.TagID = t.TagID
WHERE e.EventID = @EventID;
```

Лістинг А.3 – LINQ-запит для отримання деталей виставки

Лістинг А.3

```
var selectedEvent = await _context.Events
    Include(e => e.Venue)
    Include(e => e.Organizer)
    ThenInclude(o => o!.Organization)
    Include(e => e.EventTags)
    ThenInclude(et => et.Tag)
    SingleOrDefaultAsync(e => e.EventID == id);
```

Лістинг А.4 – C#-код створення записів квитків

Лістинг А.4

```

for (var i = 0; i < quantity; i++)
{
    _context.Tickets.Add(new Ticket
    {
        EventID = selectedEvent.EventID,
        UserID = userId.Value,
        ReservationDate = selectedTicketDate,
        Price = selectedEvent.Price,
        Status = "Reserved",
        QR_Code = Guid.NewGuid().ToString()
    });
}
await _context.SaveChangesAsync();

```

Лістинг А.5 – C#-код оновлення оцінки інтересу користувача за тегами виставки

Лістинг А.5

```

public async Task AddPreferenceScore(int userId, int tagId, double amount)
{
    var preference = await _context.UserTagPreferences
        .SingleOrDefaultAsync(p => p.UserID == userId && p.TagID == tagId);
    if (preference == null)
    {
        _context.UserTagPreferences.Add(new UserTagPreference
        {
            UserID = userId,
            TagID = tagId,
            Score = amount
        });
    }
}

```

```

else
{
    preference.Score += amount;
}
}

```

Лістинг А.6 – C#-код створення скарги на статтю

Лістинг А.6

```

_context.ContentReports.Add(new ContentReport
{ ReporterUserID = userId.Value,
  ArticleID = articleId,
  ContentType = "Article",
  Category = string.IsNullOrEmpty(category) ? "Other" : category.Trim(),
  Reason = string.IsNullOrEmpty(reason) ? "No details provided." :
reason.Trim(),
  Status = "Open",
  CreatedAt = DateTime.Now
});

```

Лістинг А.7 – C#-код створення коментаря до статті

Лістинг А.7

```

var comment = new ArticleComment
{
    ArticleID = articleId,
    UserID = userId,
    CommentText = commentText,
    PostedAt = DateTime.Now
};
_context.ArticleComments.Add(comment);
await _context.SaveChangesAsync();
await _context.SaveChangesAsync();

```


Лістинг А.8 – C#-код отримання даних для маркерів карти

Лістинг А.8

```
var eventQuery = _context.Events
    .Include(e => e.Venue)
    .ThenInclude(v => v!.Organization)
    .Include(e => e.EventTags)
    .ThenInclude(et => et.Tag)
    .AsQueryable();
var venue = group.First().Venue;
var lat = venue.Latitude ?? fallback.Lat + (index * 0.004);
var lng = venue.Longitude ?? fallback.Lng + (index * 0.004);
```

ДОДАТОК Б

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Круглік Маргарита Ігорівна, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему

«Інформаційна система онлайн бронювання квитків на художні виставки України» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

- ☐ інструменти генеративного ШІ не використовувалися.
- ☐ інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL,

_____ інші (вказати назву)) були використані для:

- ☐ перекладу іншомовних джерел;
- ☐ стилістичного редагування та перевірки граматики;
- ☐ допомоги у написанні/відлагодженні програмного коду;
- ☐ інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

« » 2026 р. **Маргарита КРУГЛІК**