Державний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Програмна реалізація алгоритму для придбання квитків до закладів культури»

Студента 2 курсу, 4мз групи,
спеціальності
F3 «Комп'ютерні науки» _____ Стеценко Андрій Олександрович
підпис студента

Науковий керівник
доктор педагогічних наук, доцент _____ Підгорна Тетяна Володимирівна
підпис керівника

Гарант освітньої програми
доктор фізико-математичних наук, професор _____ Пурський Олег Іванович
підпис керівника

Київ 2025

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність F3 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«30 грудня 2024р.

Завдання
на кваліфікаційну роботу студенту

Стеценко Андрію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
«Програмна реалізація алгоритму для придбання квитків до закладів культури»
Затверджена наказом ректора від *«27» грудня 2024 р. № 4253 із змінами «11» грудня 2025 р. № 4000)*

2. Строк здачі студентом закінченої роботи *9 грудня 2025 року*

3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка програмного алгоритму бронювання і придбання квитків до закладів культури у вигляді завершеного додатку.
Об'єктом дослідження: програмні алгоритми та автоматизація процесів бронювання і придбання квитків до закладів культури .
Предметом дослідження: моделі, методи та інформаційні технології бронювання і придбання квитків.

4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Підгорна Т.В.	30.12.2024 р.	30.12.2024 р.
2	Підгорна Т.В.	30.12.2024 р.	30.12.2024 р.
3	Підгорна Т.В.	30.12.2024 р.	30.12.2024 р.

6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ РОЗРОБКИ ТЕХНОЛОГІЙ ДЛЯ ПРИДБАННЯ КВИТКІВ ДО ЗАКЛАДІВ КУЛЬТУРИ

1.1. Роль і місце технологій для придбання квитків у сучасній культурній сфері

1.2. Аналіз існуючих рішень на ринку

1.3. Аналіз існуючих методів та технологій для розробки додатку

РОЗДІЛ 2. АНАЛІЗ ТА МОДЕЛЮВАННЯ ПРОЦЕСІВ ПРИДБАННЯ КВИТКІВ ДО ЗАКЛАДІВ КУЛЬТУРИ

2.1. Моделювання процесу придбання квитків, постановка задачі

2.2. Вибір методів та інструментів для реалізації проекту

2.3 Візуалізація алгоритмів роботи основних функцій додатку

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ДЛЯ ПРИДБАННЯ КВИТКІВ

3.1. Програмна реалізація додатку

3.2. Оцінка отриманих результатів та тестування

РЕЗУЛЬТАТИ І ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*02.11.2024*	*02.11.2024*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*30.12.2024*	*30.12.2024*
3	*Вступ*	*01.05.2025*	*01.05.2025*
4	*РОЗДІЛ 1. Теоретичне дослідження розробки технологій для придбання квитків до закладів культури*	*17.06.2025*	*17.06.2025*
5	*Підготовка статті*	*24.06.2025*	*24.06.2025*
6	*РОЗДІЛ 2. Аналіз та моделювання процесів придбання квитків до закладів культури*	*05.09.2025*	*05.09.2025*
7	*РОЗДІЛ 3. Практична реалізація алгоритму для придбання квитків*	*17.10.2025*	*17.10.2025*
8	*Результати і висновки*	*21.10.2025*	*21.10.2025*
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*25.11.2025*	*25.11.2025*
10	*Попередній захист випускної кваліфікаційної роботи*	*28.11.2025*	*28.11.2025*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.12.2025*	*02.12.2025*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*09.12.2025*	*09.12.2025*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	*За розкладом роботи ЕК*

8. Дата видачі завдання «30» грудня 2024 р.

9. Керівник кваліфікаційної роботи _____ Підгорна Т. В.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми _____ Пурський О. І.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент _____ Стеценко А. О.
(прізвище, ініціали, підпис)

12. Відгук керівника кваліфікаційної роботи

Керівник кваліфікаційної роботи

_____

(підпис, дата)

13. Висновок про кваліфікаційну роботу

Кваліфікаційна робота студента Стеценко А.О.

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми Пурський О.І.

(підпис, прізвище, ініціали)

Завідувач кафедри Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2025 р.

Анотація

У кваліфікаційній роботі здійснено програмну реалізацію алгоритму для придбання квитків до закладів культури. Проведено аналіз сучасних підходів до цифровізації культурної сфери та визначено ключові вимоги до онлайн-систем продажу квитків. Побудовано UML-моделі, що описують основні процеси взаємодії користувача, адміністратора та організатора подій, та ключові функції веб-додатку. Реалізовано функціональний веб-додаток, який забезпечує перегляд подій, вибір місць, оформлення покупки, оплату та генерацію електронного квитка з QR-кодом. Також увагу приділено інтеграції адміністративного інтерфейсу та забезпеченню захисту персональних даних користувачів.

Ключові слова: алгоритм, електронний квиток, моделювання процесів, онлайн-оплата, бази даних.

Anotation

The qualification work developed a software implementation of the algorithm for purchasing tickets to cultural institutions. An analysis of modern approaches to the digitalization of the cultural sphere was conducted and key requirements for online ticket sales systems were identified. UML models were built that describe the main processes of interaction between the user, administrator and event organizer, and key functions of the web application. A functional web application was implemented that provides viewing of events, selection of seats, purchase, payment and generation of an electronic ticket with a QR code. Attention was also paid to the integration of the administrative interface and ensuring the protection of users' personal data.

Keywords: algorithm, electronic ticket, process modeling, online payment, databases.

ЗМІСТ

ВСТУП

Актуальність обраної теми викликана швидким розвитком комп'ютерних технологій, що широко використовуються для створення та функціонування інтернет-додатків, зокрема в культурній галузі. Застосування сучасних платформ для розробки веб-додатків (наприклад, Flask, Django), мови програмування (Python, JavaScript, HTML, CSS), системи управління базами даних (PostgreSQL, SQLite) та інтеграція платіжних систем (Stripe, LiqPay) забезпечують ефективне й безпечне придбання квитків до закладів культури в онлайн-середовищі. В умовах цифровізації культури, наявність зручних та доступних інструментів для купівлі квитків є важливою складовою забезпечення комфортного доступу до культурних подій для широкого кола відвідувачів. Інтернет-додатки такого типу дозволяють організаторам ефективно управляти продажами квитків, а користувачам — швидко та зручно планувати відвідування заходів.

У науковій літературі існують дослідження, що розглядають різні аспекти проблематики реалізації таких систем. Наприклад, у статті А. Аззаарі аналізує дизайн веб-сайтів музеїв і інтереси різних категорій користувачів [1]. О.Ткаченко і О. Тищура розглядають особливості процесу розробки веб-орієнтованої системи COFFEE++, зокрема в ній розглянуто питання архітектури веб-орієнтованих додатків [2]. Проте серед цих робіт недостатньо уваги приділено специфіці розробки веб-додатків для продажу квитків, зокрема алгоритму бронювання місць. Саме ці питання буде розглянуто в цій роботі.

На сучасному ринку існує чимало рішень для онлайн-продажу квитків до театрів, музеїв, концертних залів та інших закладів культури. Популярність таких сервісів зумовлена потребою модернізації системи продажу квитків та підвищення рівня сервісу для користувачів. Отже, тема цієї випускної кваліфікаційної роботи є **актуальною** та практично важливою.

Мета і завдання дослідження. Метою даного дослідження є розробка програмного алгоритму бронювання і придбання квитків до закладів культури у вигляді завершеного додатку. Для досягнення поставленої мети необхідно вирішити такі з**авдання**:

1. Проаналізувати теоретичні та практичні підходи для практичної реалізації алгоритму для продажу квитків у культурній сфері.

2. Визначити особливості алгоритмів роботи основних функцій додатку для придбання квитків.

3. Обґрунтувати вибір засобів для програмної реалізації алгоритму.

4. Провести практичну реалізацію алгоритму на основі визначених технічних засобів з урахуванням зібраної теоретичної бази.

5. Виконати тестування реалізованого алгоритму для підтвердження його функціональності та надійності.

Об'єктом дослідження є програмні алгоритми та автоматизація процесів бронювання і придбання квитків до закладів культури.

Предметом дослідження є моделі, методи та інформаційні технології бронювання і придбання квитків.

Методи дослідження включають теоретичний аналіз літератури з галузі веб-розробки, електронної комерції.

Для вирішення поставлених завдань були використані такі **методи**:

- аналітичний метод для дослідження проблематики предметної області;
- методи розробки веб-додатків;
- метод UML-моделювання;
- методи теорії алгоритмів.

Наукова новизна дослідження полягає у розробці авторських підходів до реалізації алгоритмів придбання квитків до закладів культури з метою побудови веб-ресурсу.

Практичне значення дослідження полягає у можливості застосування розробленого додатку для придбання квитків у культурній сфері, що сприятиме підвищенню доступності культурних подій для широкого загалу та підвищенню ефективності продажів для організаторів.

Публікації: Результати дослідження опубліковано в:

1. Програмна реалізація додатку для придбання квитків до закладів

культури // Modern Perspectives on Science and Economic Progress : Collection of Scientific Papers. – Vilnius, Lithuania, 4–6 June 2025. – P. 145–149.

2. Стаття «Програмна реалізація додатку для придбання квитків до закладів культури» // Прикладні комп'ютерні технології : зб. наук. ст. студ. / відп. ред. А.В. Селіванова — Київ : Держ. торг.-екон. ун-т, 2025. – С. 132-136.

Структура та обсяг кваліфікаційної роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 31 найменування, додатків і містить 58 сторінок основного тексту та 48 рисунків.

РОЗДІЛ 1.
ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ РОЗРОБКИ ТЕХНОЛОГІЙ ДЛЯ ПРИДБАННЯ КВИТКІВ ДО ЗАКЛАДІВ КУЛЬТУРИ

1.1. Роль і місце технологій для придбання квитків у сучасній культурній сфері

У сучасному світі комп’ютерних технологій стала невід'ємною частиною розвитку майже всіх сфер суспільного життя, і культурна індустрія не є винятком. Культурні заклади, такі як театри, музеї, концертні зали, виставкові центри, все активніше впроваджують інформаційні технології для оптимізації взаємодії з відвідувачами. Одним з ключових напрямів цифрової взаємодії з відвідувачами є автоматизація процесу продажу квитків шляхом створення веб-додатків.

Традиційна система розповсюдження квитків, тобто придбання їх в касах або через посередників, поступово втрачає актуальність. Це пов'язано з низкою переваг, які має придбання квитків через онлайн сервіси: немає необхідності фізичного відвідування каси, не потрібно стояти в чергах за квитками, є можливість оперативного перегляду інформації про наявність вільних місць та подій. Електронні сервіси дозволяють швидко та зручно здійснювати покупку в будь-який час, використовуючи комп'ютер або мобільний пристрій [3].

Для закладів культури застосування веб-додатків також має велику важливість, адже вони використовуються не лише як технічний інструмент для оптимізації продажів, але й як засіб розвитку закладу культури, його інтеграція у цифрове середовище та розширення цільової аудиторії. Для таких закладів інтеграція веб-додатків до їх діяльності відкриває низку переваг:

- збільшення відвідувачів завдяки спрощенню процесу покупки квитків;
- зменшення витрат на друк паперових квитків та роботу касирів;
- формується імідж сучасного, інноваційного закладу;
- зменшення черг та адміністративного навантаження;
- автоматизація процесу.

Тепер розглянемо переваги використання мобільних та веб-додатків для

придбання квитків з боку відвідувачів закладу культури:

- цілодобова доступність до сервісу, придбання квитків можливе в будь який час;
- можливість вибрати місце у залі у зручному інтерфейсі;
- безпечні та швидкі онлайн-платежі за допомогою платіжних систем;
- відсутність черг. Покупки квитків через онлайн сервіси дозволяє уникнути ситуацій, коли квитки в музеї будуть розпродані або повністю заброньовані в момент перебування в черзі та уникнути негативного враження від відвідування [4];
- швидке отримання квитка у форматі PDF або доступ до нього через QR-код;
- зручне зберігання квитків у смартфоні без ризику втратити їх;
- екологічний аспект - перехід на електронні квитки зменшує використання паперу.

Продаж квитків онлайн допомагає музеям забезпечити кращий доступ для відвідувачів, це дає їм змогу краще передбачати коливання відвідувачів та приймати найкращі стратегічні рішення з точки зору комунікації та маркетингу. Електронні сервіси культури сприяють формуванню та аналізу бази даних відвідувачів, що відкриває нові маркетингові можливості [5].

Додатковою перевагою онлайн-продажу квитків є те, що це найкращий спосіб спостерігати, як база даних клієнтів/відвідувачів самостійно поповнюється. Тому що з кожною покупкою безпосередньо збираються цінні дані, які маркетингові та членські команди зможуть використовувати для кращого розуміння аудиторії [6].

Держава заохочує інноваційні проекти у сфері культури, спрямовані на підвищення доступності культурних послуг для населення. Впровадження електронних квитків сприяє популяризації культурних подій серед молодого покоління, яке частіше взаємодіє з установами через електронні пристрої.

Тому роль веб-додатків для продажу квитків у сучасній культурній сфері є досить важливою. Ці додатки виступають не лише технічним інструментом для

покращення продажів, а й засобом для розвитку культурного закладу, інтеграція та популяризація закладу в цифровому середовищі та розширення цільової аудиторії. Використання веб-додатків формує новий спосіб взаємодії культурних закладів та відвідувачів, що відповідає стандартам сучасного інформаційного суспільства.

1.2. Аналіз існуючих рішень на ринку

В Україні застосування веб-додатків для придбання квитків набирає все більшої популярності. Сучасні квиткові системи – це вже не просто онлайн резервування та купівля квитків. Це частина цифрової B2B-інфраструктури, яка поєднує маркетинг, аналітику, клієнтський сервіс та контроль доступу [7]. На ринку зараз існує велика кількість сервісів, серед найпопулярніших можна виокремити Kontramarka (kontramarka.ua), Karabas (karabas.com) та TicketsBox (ticketsbox.com). Ці платформи охоплюють широкий спектр культурних закладів – від концертів і театрів до спортивних подій та виставок. Загалом всі ці сервіси позиціонують себе як якісний та функціональний інструмент, який збільшує популярність культурних закладів в сучасному суспільстві.

Kontramarka є одним з найстаріших та найвідоміших сервісів в Україні. Вона має велику базу користувачів і впізнаваність бренду, що робить платформу популярною серед організаторів заходів. Також, Kontramarka використовує Maestro Ticket System.

Maestro Ticket System — системний інтегратор з автоматизації продажу квитків і системами платіжно-пропускного доступу для спортивних і культурно-розважальних установ [8].

Kontramarka має простий та інтуїтивний інтерфейс, користувач може легко орієнтуватися в структурі сайту між різними категоріями об’єктів, фільтрами, тощо. На головній сторінці розташовані необхідні інструменти для ефективного пошуку подій за відповідними критеріями (рис. 1.1.).

Також, сайт містить такий функціонал:

1. Особистий кабінет.

2. Каталог подій.
3. Онлайн оплата.
4. Електронні квитки.
5. Вибір конкретного місця.
6. Пошук та сортування.
7. Додавання подій.

Серед особливостей сервісу, які варто відзначити, можна виокремити наявність промокодів та бонусів як стимул для користувачів та мобільний додаток на Android.

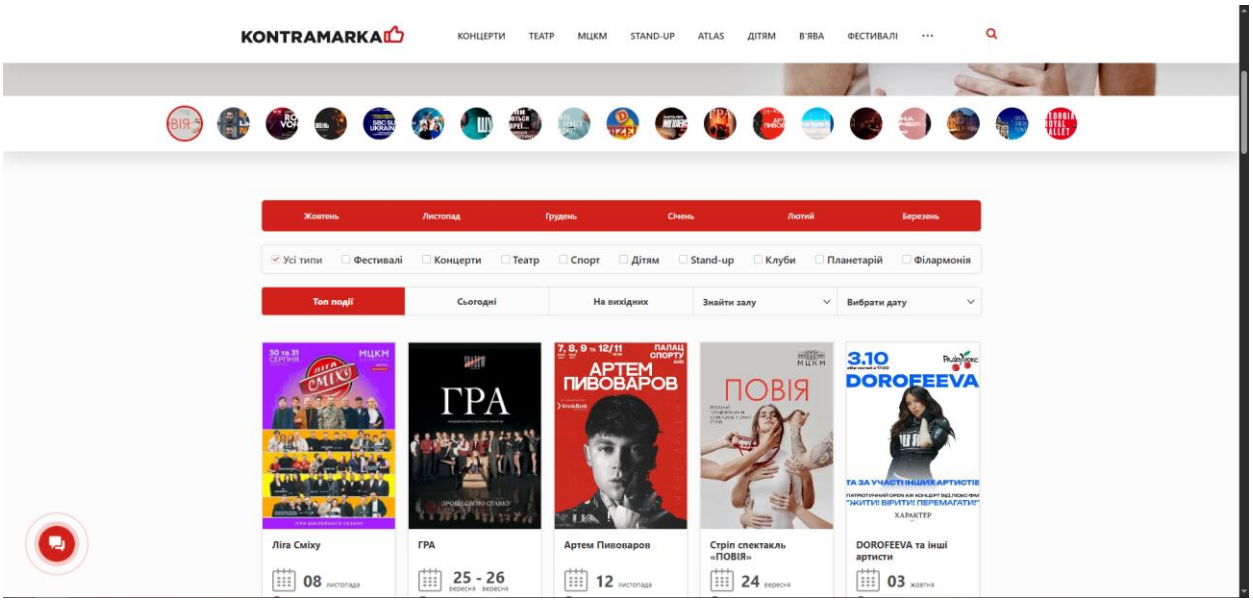


Рис. 1.1. Головна сторінка Kontramarka

Karabas активно розвиває мобільний сегмент та функціонал персоналізації. Сервіс робить акцент на зручному пошуку подій, рекомендаціях та швидкому оформленні замовлень. Karabas активно співпрацює з організаторами великих концертів та фестивалів, що дозволяє йому конкурувати на міжнародному ринку. Сайт містить інтуїтивно зрозумілий інтерфейс та необхідний функціонал для комфортного пошуку та вибору подій.

TicketsBox це сучасний квитковий сервіс з акцентом на прозорість цін і зручність для користувачів. Він надає можливість вибору конкретного місця та

підтримує різні платіжні системи.

Для сервісу TicketsBox також характерні аналогічні функції, як і для Kontramarka та Karabas. Однак, TicketsBox виділяється сучаснішим, більш футуристичним дизайном інтерфейсом, який надає йому перевагу у порівнянні з Kontramarka та Karabas (рис. 1.2.).

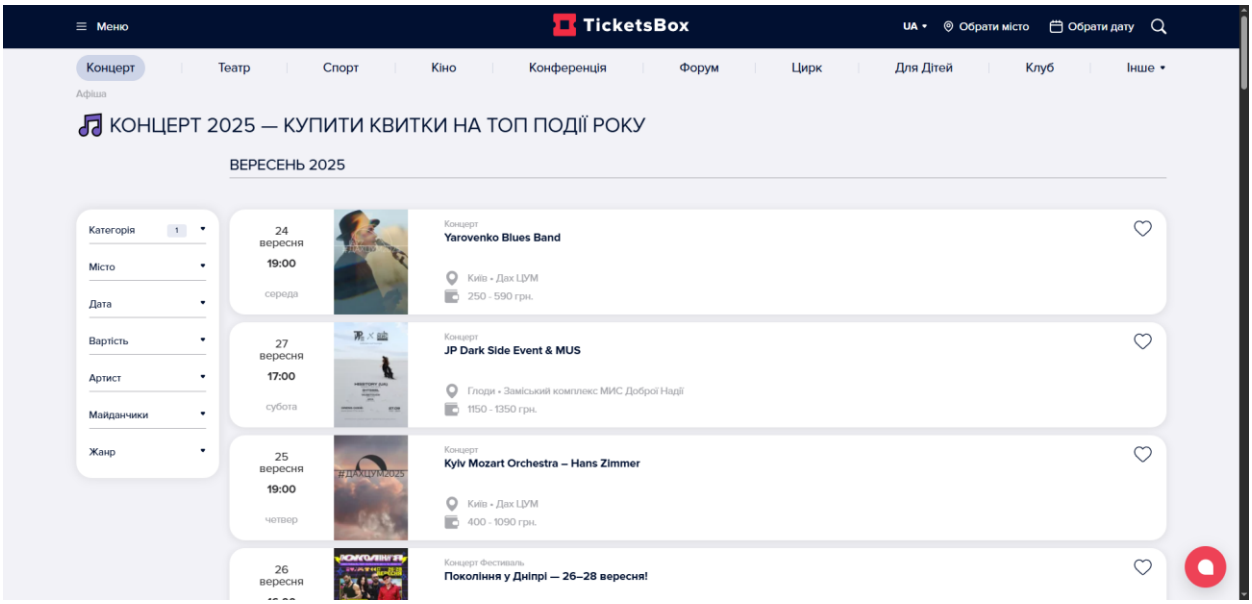


Рис. 1.2. TicketsBox

Також, із особливостей TicketsBox варто відзначити голосовий пошук, який відсутній у раніше згаданих сервісів (рис. 1.3.).

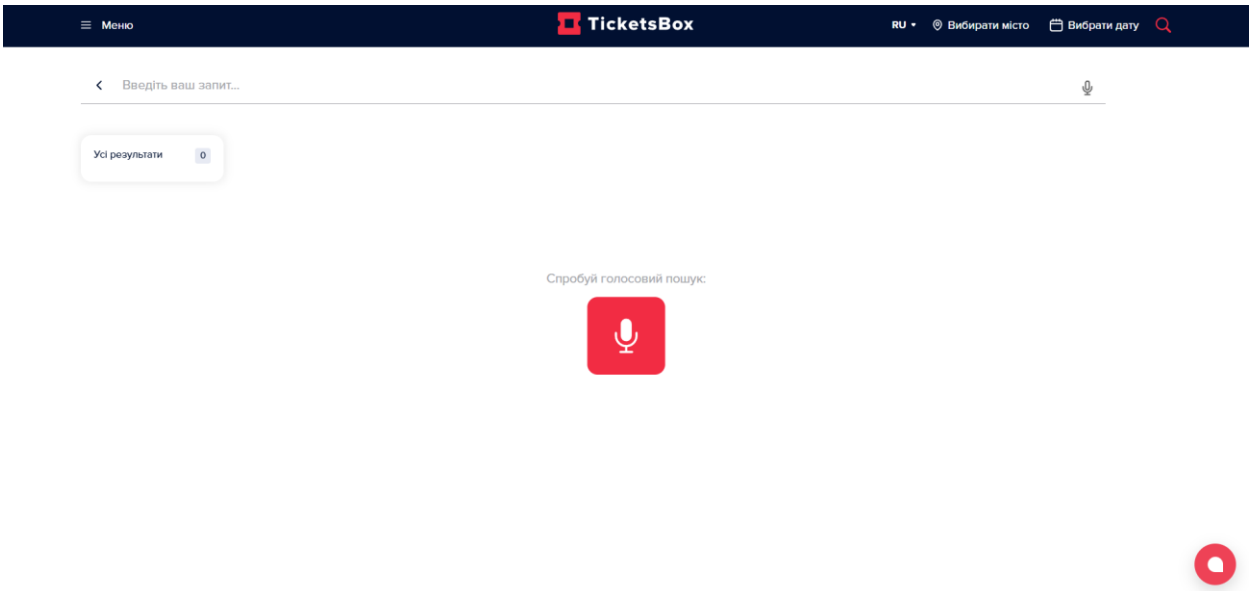


Рис. 1.3. Голосовий пошук TicketsBox

Сучасні сервіси для придбання квитків до закладів культури відзначаються широким функціональним наповненням, яке орієнтоване не лише на зручність використання відвідувачів, а й на потреби організаторів подій:

1. Каталог подій: присутній поділ за критеріями, пошук подій за назвою та фільтри за жанром події.
2. Вибір квитка та місця.
3. Купівля та оплата: підтримка банківських карток, інтеграція з локальними сервісами, можливість отримати електронний квиток на email чи в особистому кабінеті.
4. Електронні квитки.
5. Особистий кабінет користувача: завантаження електронного квитка, можливість повернення або обміну квитка.
6. Функціонал для організаторів: створення та управління подіями, аналітика продажів.

Сучасні сервіси поєднують у собі зручність для користувача, автоматизацію процесів для організатора та ефективні маркетингові інструменти. Вони пропонують широкий набір функцій: від простого продажу квитків для повнофункціональних систем керування відвідувачами, аналітики, абонементів. Недоліком Kontramarka є застарілий дизайн в порівнянні з конкурентами, це також стосується Karabas. TicketsBox же має сучасніший інтерфейс, однак система пошуку виглядає менш розширеною. У TicketsBox та Karabas досить складна система додавання подій для організаторів, тоді як Kontramarka має окрему кнопку на головній сторінці «Додати подію».

1.3. Аналіз існуючих методів та технологічних рішень

Системи для придбання квитків можуть мати різні форми, наприклад веб-сайт чи мобільний додаток. Кожна з цих реалізацій має свої переваги та недоліки. Саме класичним варіантом є веб-сайт.

Веб-сайт – це сукупність веб-сторінок, доступних в інтернеті через протокол

http/https [9]. Будь-який сайт, Інтернет-магазин чи інший online-ресурс складається з певних елементів, одним із яких є веб-сторінка [2]. Структура веб-сторінки повинна поєднувати елементи дизайну та інформативний контент для забезпечування зручності користувачів. Структурна схема веб-сторінки показана на рисунку 1.4.

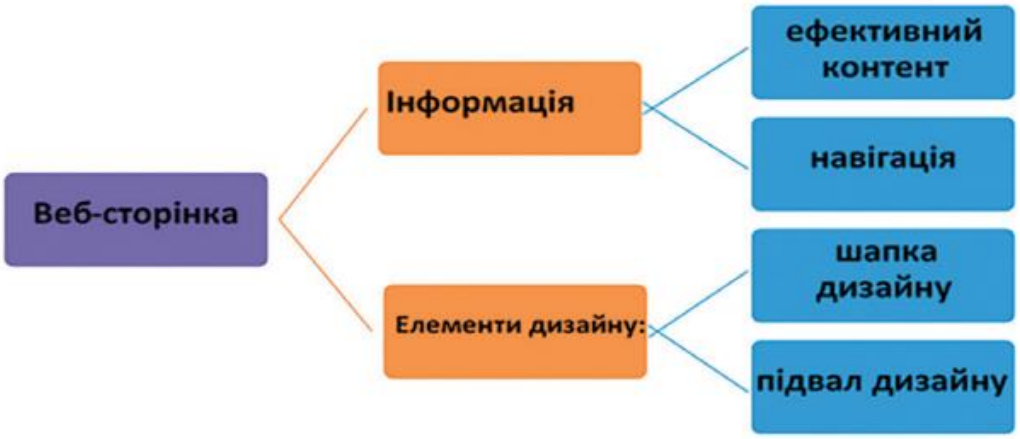


Рис. 1.4. Структурна схема веб-сторінки [2].

Переваги використання:

- доступ з будь-якого пристрою з браузером;
- просте розгортання та оновлення. Оновлення відбувається на сервері, а не на пристроях користувачів;
- легше інтегрувати платіжні системи, генерацію квитків та аналітику;
- веб-сайт легше просувати.

Недоліки використання:

- залежність від інтернет-з’єднання;
- залежність від продуктивності серверу.

Веб-сайт став надзвичайно ефективним засобом просування продуктів і послуг компаній та магазинів. Для створення повноцінних веб-сайтів використовується широкий спектр технологій веб-розробки. Вибір конкретних методів та технологій залежить від функціональних вимог до додатку, його масштабу та очікуваному навантаженні.

Ще одним рішенням для продажів квитків в інтернеті є мобільний

застосунок. Мобільний застосунок – програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях [10]. В наш час, коли майже кожен має смартфон, мобільні застосунки користуються великою популярністю. Це також пояснюється такими перевагами в порівнянні з веб-сайтом:

- висока продуктивність;
- можливість використання апаратних можливостей пристрою (камера для сканування QR, доступність Apple Wallet / Google Wallet, offline-режим та інші);
- залучення користувачів (повідомлення, персональні пропозиції).

Мобільні застосунки також мають і недоліки, такі як:

- більш висока вартість та складність розробки;
- мобільний застосунки потрібно розміщувати в магазинах (App Store / Google Play);
- потребують завантаження оновлення на пристрій користувача.

Розробка веб-орієнтованої системи передбачає використання технологій front-end та back-end [2].

Для створення інтерфейсу найчастіше використовуються такі технології front-end, як HTML5, CSS3 та JavaScript.

HTML – основа кожної сторінки, мова розмітки гіпертексту, допомагає структурувати та відобразити вміст сторінки в браузерах. Він вважається найпопулярнішою технологією зовнішнього інтерфейсу [11].

Основними характеристиками HTML є:

- Теги;
- Атрибути;
- Списки;
- Таблиці;
- Форми.

Теги в HTML служать для визначення елементів веб-сторінки та

відокремлення елементів один від одного. Більшість тегів можуть має атрибути. Їх задача надавати додаткові інформацію про елемент сторінки. Кожен HTML-документ має свою структуру, елементи, без яких сторінка функціонувати не буде. Такими елементами є теги <html>, <head>, <title> та <body>.

CSS – це мова таблиць стилів, яка використовується для створення веб-сайтів та веб-застосунків. Задача CSS робити сторінки веб-сайту, написаних мовою розмітки HTML, покращувати зовнішній вигляд сторінок, робити їх презентабельними, тобто підключати до елементів сторінки стилі. Підключати стилі CSS можна прямо в HTML документі, використовуючи елемент <style>, але для великих проектів краще створювати окремий файл CSS та зробити посилання на нього в HTML документі. На один CSS-файл можна посилатися з декількох веб-сторінок сайту. Браузер, аналізуючи HTML-код, звернеться за вказаним шляхом і, виявивши вказаний файл стильового оформлення, відобразить елементи сторінки відповідно до певних правил CSS [12].

На практиці широко використовуються сучасні фреймворки. Фреймворк у мовах програмування – це комплекс компонентів, бібліотек та інструментів, які пропонують структуру та готові рішення для роботи над певними завданнями [13]. Їх використання є важливим аспектом, оскільки ці технології дозволяють розробникам швидко створювати складні компоненти інтерфейсу та ефективно керувати станом додатку. Вони надають набір інструментів, які спрощують процес розробки:

- React.js;
- Vue.js;
- Angular.

Завдяки цим технологія інтерфейс стає інтерактивним та зручним для користування.

Для серверної частини додатку найчастіше використовуються такі технології back-end:

- Python (Flask, Django) – зручний для швидкої розробки веб-додатків, має великий набір бібліотек;

- JavaScript (Node.js, Express.js) – дозволяє будувати високонавантажені сервіси з підтримкою асинхронних операцій;
- PHP (Laravel, Symfony) – популярний у веб-розробці завдяки простоті та поширеності;
- Java (Spring Boot) – застосовується для масштабованих та безпечних корпоративних систем.

Квиткові сервіси потребують зберігання великої кількості даних, такі як інформація про події, користувачів, квитки, транзакції. Для цього застосовуються бази даних.

База даних – це впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначена для задоволення інформаційних потреб користувачів [14].

Одним з типів бази даних є реляційна. Реляційна модель бази даних базується на теоретико-множинному поняття відношення, тобто на множині кортежів фіксованої довжини. В такій базі даних дані організовують колекцію даних у формі таблиць, що складаються з рядків і стовпців, і зв’язаних між собою за допомогою ключів.

Одним із важливіших частин для повноцінного функціонування додатку, через який відбувається продаж товару, є інтеграції платіжної системи. З розвитком ринкової економіки зростає роль платіжної системи як одного з найважливіших елементів фінансової інфраструктури економіки [15]. Платіжні системи дозволяють веб-сайтам приймати платежі онлайн через кредитні та дебетові карти, банківські перекази, електронні гаманці та інші методи онлайн оплати. Зазвичай інтеграція платіжної системи до веб-сайту відбувається через API.

Перевагами використання інтегрованої платіжної системи є:

- Автоматизація процесів розрахунку: за допомогою інтегрованої платіжної системи відбувається автоматизований процес розрахунків клієнтів з організацією;
- Зручність для клієнтів;

- Безпека: захист особистих даних користувача під час проведення транзакцій;
- Контроль та гнучкість [16].

Також варто згадати про такий спосіб створення веб-сайтів, як система управління контентом (CMS). CMS – це складне програмне забезпечення, призначене для оптимізації процесу створення, редагування, організації та публікації цифрового контенту [17]. CMS дозволяють створити функціональний та привабливий без-сайт за максимально короткі терміни. Найпопулярнішими CMS в світі є WordPress, Joomla та Drupal.

Аналіз існуючих методів і технологій показує, що створення сучасного додатку для придбання квитків є багатокомпонентним процесом. Вибір технологій має ґрунтуватися на балансі між продуктивністю, масштабованістю та зручністю користування. Сучасні технології веб-розробки дозволяють обрати оптимальні рішення для ефективної розробки різноманітних типів веб-додатків. Таким чином, ефективне поєднання сучасних технологій розробки та методів програмування дозволяють створити додаток, який відповідає потребам та стандартам сучасного світу.

РОЗДІЛ 2.
АНАЛІЗ ТА МОДЕЛЮВАННЯ ПРОЦЕСІВ ПРИДБАННЯ КВИТКІВ ДО ЗАКЛАДІВ КУЛЬТУРИ

2.1. Моделювання процесу придбання квитків, постановка задачі

Сучасний ринок квитків до закладів культури потребує автоматизації процесів, які забезпечують швидку та комфортну покупку квитків для користувачів. Тому створюються веб-додатки, які задовольняють ці потреби. Створення додатків досить складний процес з багатьма етапами. Одним з таких етапів є моделювання різних процесів, які відбуваються в додатку. Моделювання бізнес-процесів стає головним інструментом для компаній, які прагнуть підвищити ефективність, а також зберегти конкурентоспроможність на ринку [18]. Також моделювання бізнес-процесів дозволяє виявити вузькі місця системи ще на стадії проектування та зменшити ризик помилок при реалізації програмного забезпечення [19]. Моделювання є важливим етапом при створенні будь-якого додатку, оскільки воно дозволяє наглядно відобразити логіку роботи тої чи іншої системи, виявити ключові взаємодії між програмним забезпеченням та користувачем. Для системи придбання квитків до закладів культури моделювання дозволяє описати дії користувача від моменту реєстрації в сервісі до отримання електронного квитка на подію, або описати взаємодію користувача, організатора та адміністратора з додатком.

Основні учасники та складові процесу придбання квитків:

1. користувач – шукає подію, вибирає квиток, здійснює оплату;
2. організатор – створює подію, задає кількість квитків, ціну, місце проведення;
3. адміністратор – створює подію, перевіряє подані на розгляд події;
4. додаток – зберігає дані про події, користувачів, забезпечую взаємодію між організатором, користувачем та адміністратором;
5. зовнішні сервіси (платіжні системи, генератор електронних квитків) – обробляють платежі, генерують електронні квитки.

Моделювання процесів відбувається за допомогою UML-діаграм. Уніфікована мова моделювання (Unified Modeling Language - UML) - це мова моделювання загального призначення, яка призначена для забезпечення стандартного способу візуалізації проектування системи [20].

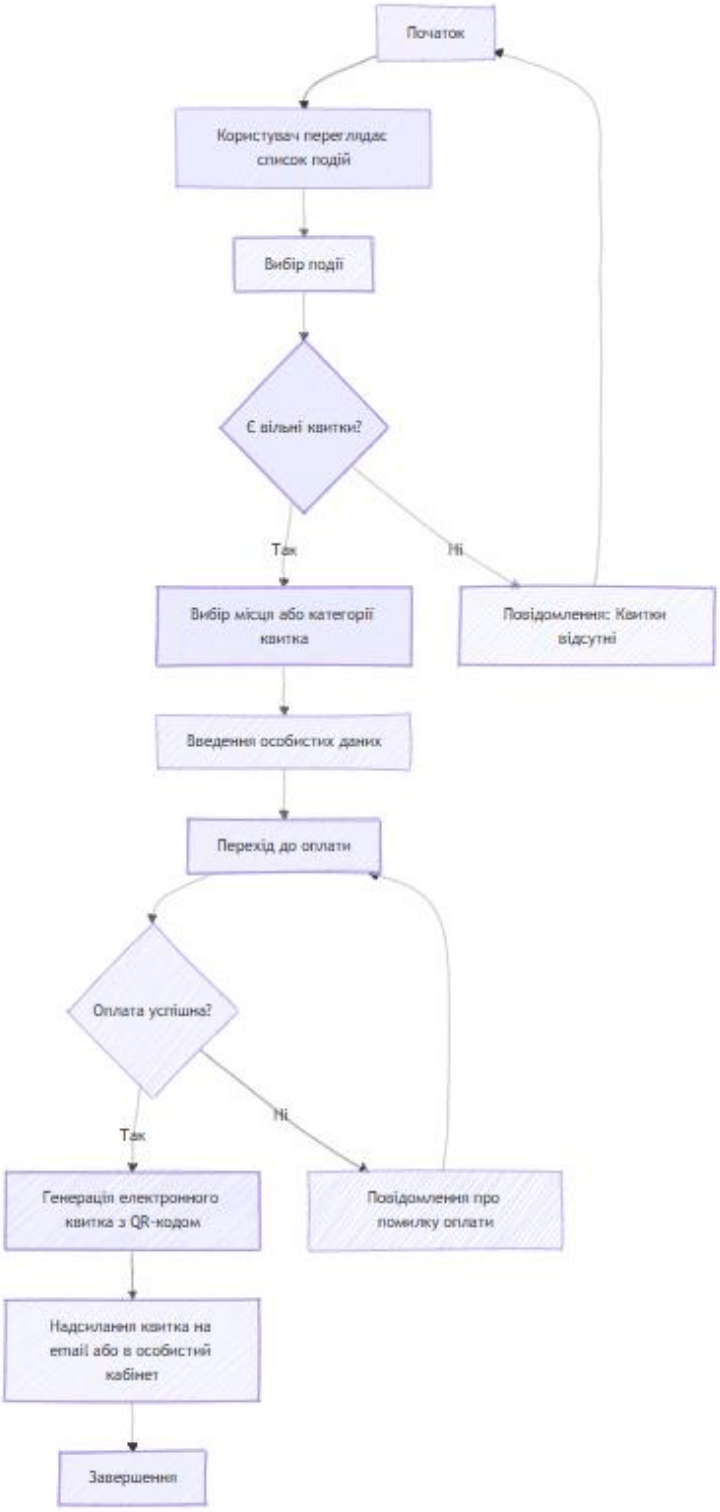


Рис. 2.1. Діаграма діяльності.

Для моделювання процесу виконання операцій в мові UML використовуються так звані діаграми діяльності.

Діаграма діяльності розглядуваної системи (рис. 2.1.) деталізує процес придбання квитка до закладу культури. Процес починається з перегляду списку подій, після чого користувач обирає конкретну поді. Система перевіряє наявність вільних квитків. У разі відсутності квитків користувач отримує відповідне повідомлення. При наявності здійснюється вибір конкретного місця або категорії квитка, введення особистих даних та перехід до оплати. Після успішної оплати генерується електронний квиток з QR-кодом, який надсилається на електронну пошту або зберігається в особистому кабінеті користувача.

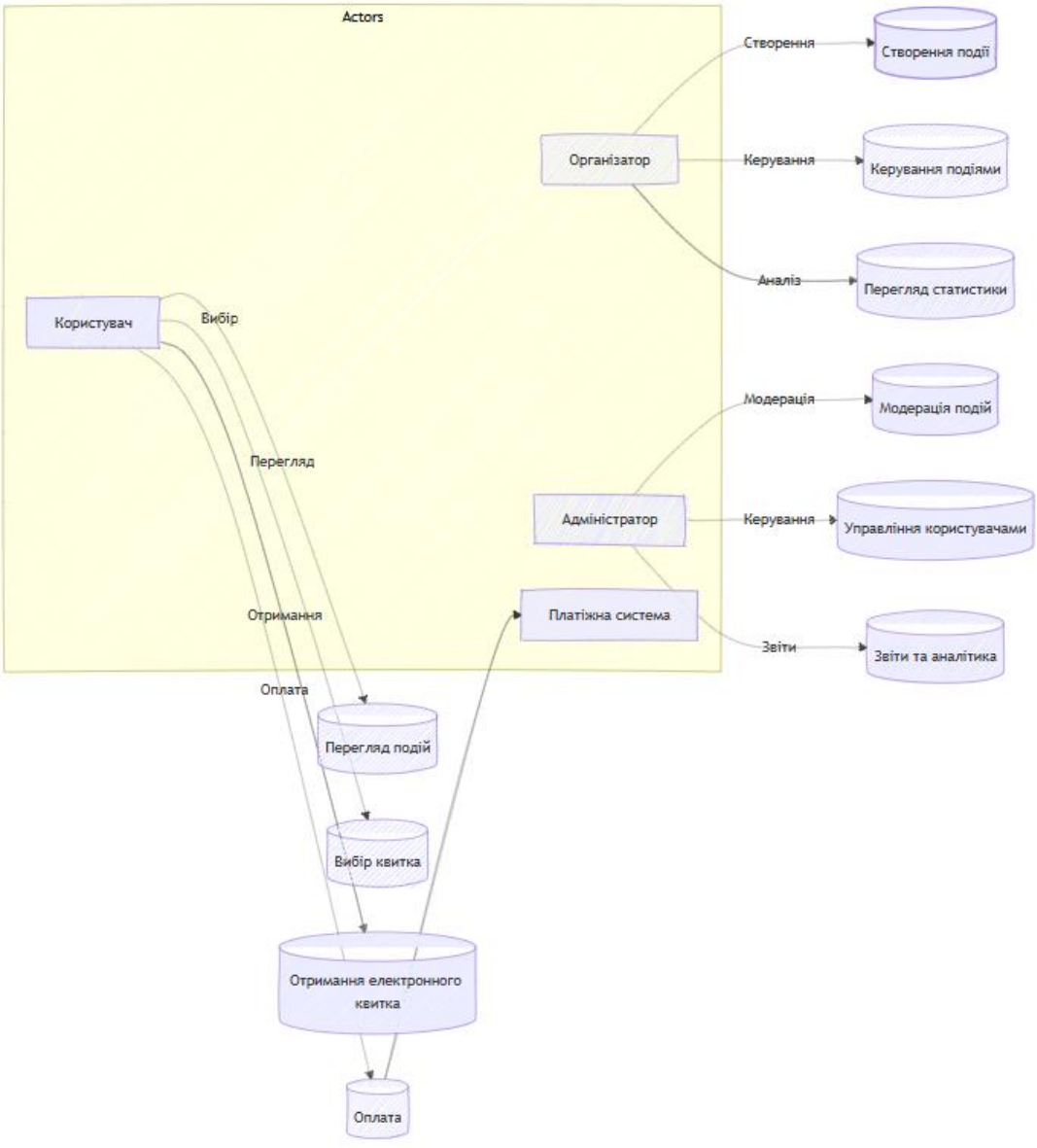


Рис. 2.2. Діаграма варіантів використання.

Діаграма варіантів використання, яка представлена на рисунку 2.2, демонструє основні функції додатку. Для користувача реалізовано можливості перегляду подій, вибору квитка, отримання електронних квитків та здійснення оплати. Адміністратор має розширені права доступу, які надає адміністративна панель, включаючи модернізацію та створення подій, управління користувачами, перегляд статистики та аналітики. Організатор подій має можливість створювати події, керувати ними та переглядати статистику проданих квитків. Описані функції є ключовими для повноцінного функціонування додатку.

На основі описаних процесів формуються задачі розробки:

- зручний пошук та перегляд інформації про події;
- можливість онлайн оплати;
- генерація QR-квитків;
- адміністративний інтерфейс для керування подіями;
- забезпечити організаторам створювати культурні події.

Розробка додатку дозволить суттєво зменшити час обслуговування користувача та усуне необхідність фізичної присутності для придбання квитків. Моделювання процесу придбання квитків дозволяє чітко визначити взаємодії та основні вимоги до додатку.

2.2. Вибір методів та інструментів для реалізації проекту

Вибір методів та інструментів для реалізації веб-додатку для придбання квитків до закладів культури надзвичайно важлива частина розробки, адже від вибору залежить ефективність функціонування системи, зручність в її розробці, подальшої підтримки, масштабованості та безпеки. В обчислювальних системах веб-додаток – це клієнт-серверна комп'ютерна програма, з якою клієнт (включно із інтерфейсом користувача та логікою на стороні клієнта) працює засобами веб-браузерів [21].

Для даного проекту було обрано стек технологій, який відповідає сучасним вимогам до веб-розробки, забезпечує гнучкість у налаштування та зручність у використанні.

В якості середовища розробки було обрано Visual Studio Code (VS Code). Visual Studio Code (VS Code) – це безкоштовний, відкритий і крос-платформний редактор коду, розроблений компанією Microsoft. Вперше він був випущений у квітні 2015 року і з тих пір набув широкої популярності серед розробників [22]. Зараз це один з найпопулярніших сучасних редакторів коду. Його популярність зумовлена зручним інтерфейсом, підтримкою великої кількості мов програмування та ще низкою переваг, такі як:

1. Крос-платформна сумісність. VS Code доступний для Windows, macOS та Linux, що дозволяє розробникам використовувати один і той самий редактор незалежно від операційної системи [22]
2. Широкий вибір розширень, що дозволяє розробникам налаштувати середовище розробки для максимально зручної та ефективної розробки.
3. Зручний попередній перегляд веб-сторінок, що дозволяє оперативно перевірити зміни в інтерфейсі.
4. Наявність вбудованого терміналу.
5. Легкість та оптимізація, що дозволяє розробникам кодувати навіть на машинах х низькою продуктивністю.

Таким чином, VS Code забезпечує баланс між функціональністю та продуктивністю.

Основною мовою розробки серверної частини було обрано Python з його фреймворком Flask.

Python — це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ [23].

Python став однією з найпопулярніших інструментів для створення веб-додатків, різноманітних систем, штучного інтелекту завдяки великій кількості бібліотек та фреймворків [24].

Flask являється одним із найбільш ефективних рішень для створення веб-додатків на Python. Flask надає розробнику необхідний набір для створення функціонального веб-додатку.

Основні переваги Flask:

1. Простота та швидкість розробки. Flask ідеально підходить для навчальних та наукових проектів.

2. Інтеграція з бібліотекам. Flask має функцію інтеграції з бібліотеками Python.

3. Гнучка інтеграція із базами даних. Flask легко взаємодіє з SQLite, PostgreSQL, MySQL та іншими СУБД.

4. Вбудований сервер та налагодження. Дебагер (налагоджувач) Flask спрощує процес тестування та забезпечує надійне налагодження [25].

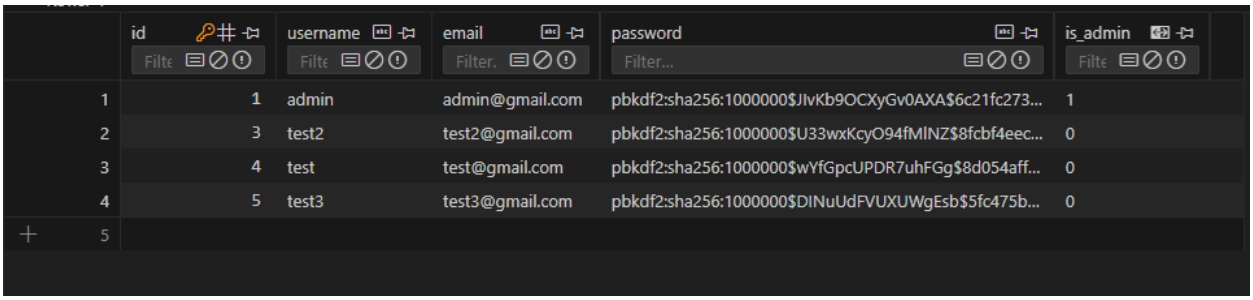
Для front-end частини сайту було обрано HTML та CSS. Використання HTML та CSS у поєднанні дозволяє створити зрозумілий, інтуїтивний та візуально приємний інтерфейс. Крім того, їх використання забезпечить швидке завантаження сторінок, сумісність із усіма основними браузерами та легкість підтримки та оновлень дизайну.

Базу даних для проекту було обрано SQLite. SQLite — це компактна вбудована реляційна база даних з відкритим кодом. Вона одна з найпопулярніших у світі, має нагороду Google-O’Reilly Open Source Awards і широко використовується в додатках та системах, де потрібно організувати зберігання даних [26]. SQLite не потребує окремого встановлення та налаштування серверу бази даних, адже вона вбудовується безпосередньо в додаток, що і робить її ідеальним вибором для невеликих і середніх проектів.

Вся база даних SQLite зберігається в одному файлі з розширенням .sqlite або .db і містить всі таблиці, індекси, схему та дані. База даних містить одну чи кілька таблиць, які визначаються схемою, яка включає імена стовпців, тип даних та інші параметри [26].

У структурі бази даних проекту для продажу квитків до закладів культури передбачено кілька таблиць:

- user – зберігає інформацію про користувачів;
- event – містить дані про події;
- ticket – фіксує дані про придбані квитки.

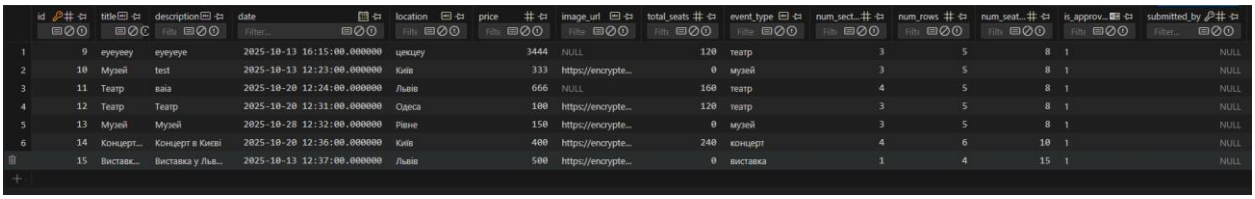


	id	username	email	password	is_admin
1	1	admin	admin@gmail.com	pbkdf2:sha256:1000000$JlvKb9OCXyGv0AXA$6c21fc273...	1
2	3	test2	test2@gmail.com	pbkdf2:sha256:1000000$U33wxKcyO94fMlNZ$8fcbf4eec...	0
3	4	test	test@gmail.com	pbkdf2:sha256:1000000$wYfGpcUPDR7uhFGg$8d054aff...	0
4	5	test3	test3@gmail.com	pbkdf2:sha256:1000000$DINuUdFVUXUWgEsb$5fc475b...	0
5					

Рис. 2.3. Таблиця user

На рисунку 2.3. представлена таблиця user. Після кожної реєстрації користувача в таблиці створюється запис, який містить такі параметри:

- id – унікальний ідентифікатор користувача;
- username – ім’я користувача для входу в систему;
- email – електронна пошта користувача;
- password – хеш пароль користувача;
- is_admin – прапорець адміністратора системи.



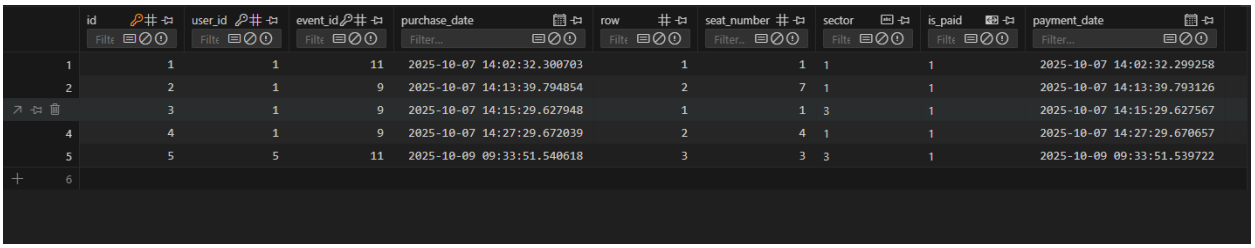
	id	title	description	date	location	price	image_url	total_seats	event_type	num_sect...	num_rows	num_seat...	is_approv...	submitted_by
1	9	eyeyeey	eyeyeye	2025-10-13 16:15:00.000000	цекцеу	3444	NULL	120	театр	3	5	8	1	NULL
2	10	Музей	test	2025-10-13 12:23:00.000000	Київ	333	https://encrypte...	0	музей	3	5	8	1	NULL
3	11	Театр	ваіа	2025-10-20 12:24:00.000000	Львів	666	NULL	160	театр	4	5	8	1	NULL
4	12	Театр	Театр	2025-10-20 12:31:00.000000	Одеса	100	https://encrypte...	120	театр	3	5	8	1	NULL
5	13	Музей	Музей	2025-10-28 12:32:00.000000	Рівне	150	https://encrypte...	0	музей	3	5	8	1	NULL
6	14	Концерт...	Концерт в Києві	2025-10-20 12:36:00.000000	Київ	400	https://encrypte...	240	концерт	4	6	10	1	NULL
7	15	Виставк...	Виставка у Льв...	2025-10-13 12:37:00.000000	Львів	500	https://encrypte...	0	виставка	1	4	15	1	NULL

Рис. 2.4. Таблиця event

В таблиці event (рис. 2.4.) записи з’являються після створення події адміністратором або організатором події, який надіслав заповнену заявку інформації про подію на модерацію адміністратору, та містить такі параметри:

- id – унікальний ідентифікатор події;
- title – назва події;

- description – детальний опис події;
- date – дата проведення події;
- location – місце проведення події;
- price – вартість квитка;
- image_url – посилання на зображення події;
- total_seats –загальна кількість місць;
- event_type – тип події;
- num_sectors – кількість секторів у залі;
- num_rows – кількість рядів у залі;
- num_seats_per_row – кількість місць у ряду;
- is_approved – статус модерації події.



	id	user_id	event_id	purchase_date	row	seat_number	sector	is_paid	payment_date
1	1	1	11	2025-10-07 14:02:32.300703	1	1	1	1	2025-10-07 14:02:32.299258
2	2	1	9	2025-10-07 14:13:39.794854	2	7	1	1	2025-10-07 14:13:39.793126
3	3	1	9	2025-10-07 14:15:29.627948	1	1	3	1	2025-10-07 14:15:29.627567
4	4	1	9	2025-10-07 14:27:29.672039	2	4	1	1	2025-10-07 14:27:29.670657
5	5	5	11	2025-10-09 09:33:51.540618	3	3	3	1	2025-10-09 09:33:51.539722
6									

Рис. 2.5. Таблиця ticket

В таблиці ticket запис з’являється після покупки квитка. Вона містить такі параметри:

- id – унікальний ідентифікатор квитка;
- user_id – id користувача, який придбав квиток;
- event_id – id події, на яку придбали квиток;
- purchase_date – дата та час покупки квитка;
- row – номер ряду;
- seat_number – номер місця;
- sector – номер сектору у залі;
- is_paid – статус оплати;
- payment_date – дата оплати.

На рисунку 2.6. представлена схема бази даних. Вона демонструє як таблиці в базі даних пов’язані між собою.

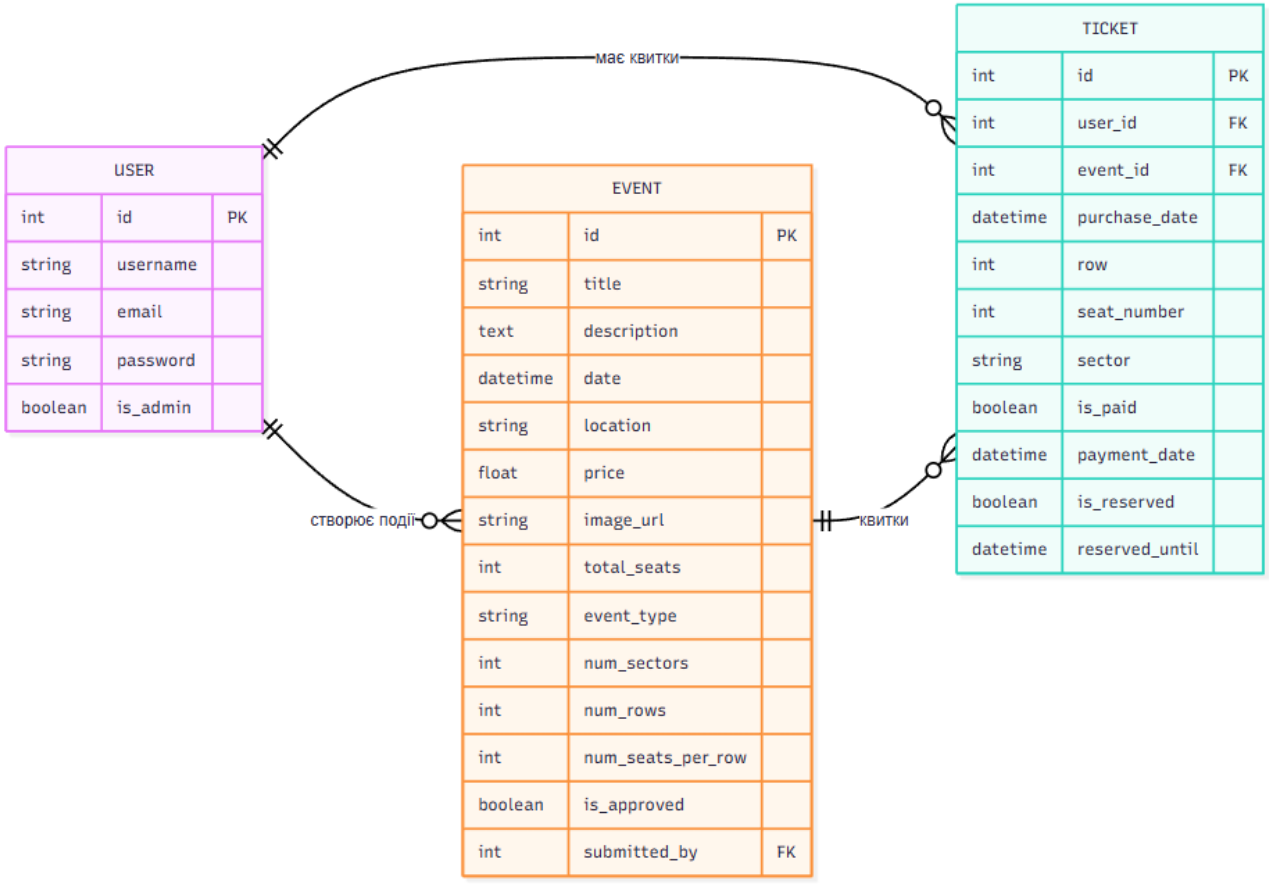


Рис. 2.6. Схема бази даних

Обрана архітектура та стек технологій дозволять ефективно реалізувати всі функціональні можливості, які необхідні для повноцінного функціонування додатку для придбання квитків до закладів культури.

2.3. Візуалізація алгоритмів роботи основних функцій додатку

Використання діаграм дозволяє наочно продемонструвати логіку системи, послідовність дій користувача та адміністратора та взаємодію окремих компонентів між собою. Вони дозволяють пояснити кроки для виконання поставленого завдання та сформувати структуру системи [27].

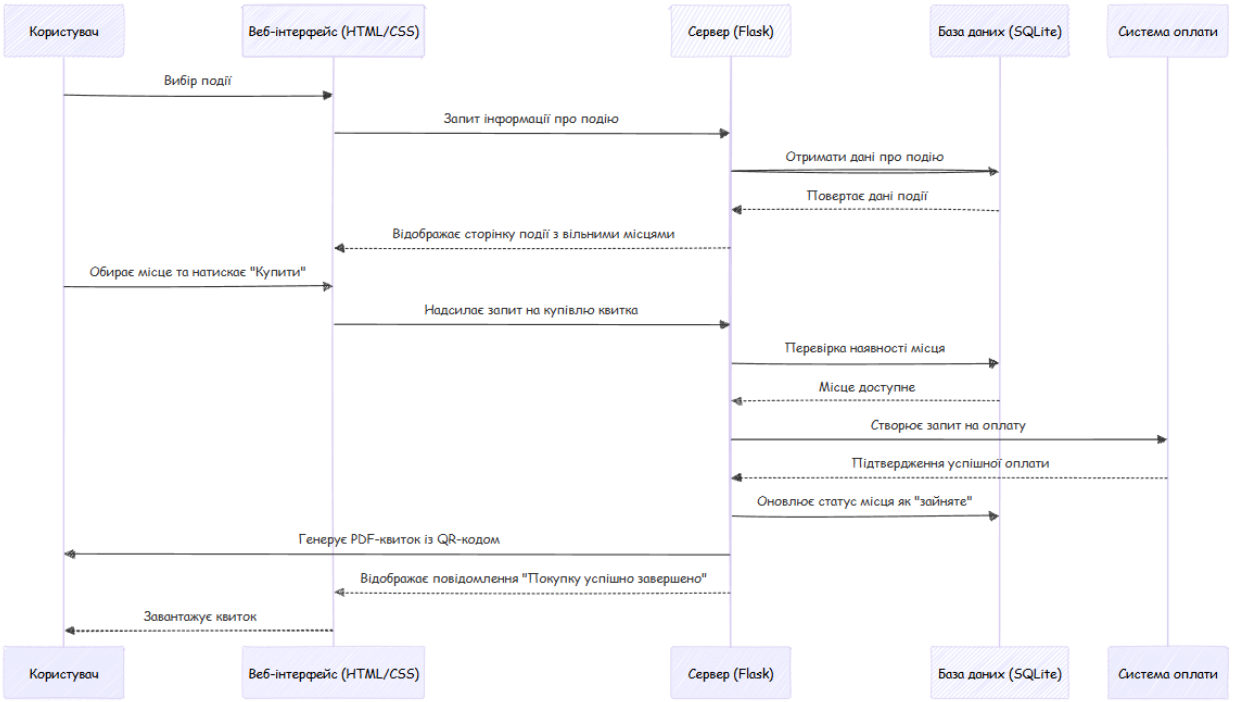


Рис. 2.7. Діаграма послідовності придбання квитка

Діаграма послідовності (рис. 2.7.) відображає порядок взаємодії між основними учасниками процесу придбання квитка. На початку користувач через веб-інтерфейс обирає подію. Інтерфейс надсилає запит до серверної частини сайту, яка звертається до бази даних для отримання інформації про подію, після чого отримані дані повертаються користувачу у веб-інтерфейс у вигляді сторінки з інформацією про подію.

Після вибору місця користувач натискає кнопку «Придбати квиток», та цією дією він ініціює надсилання запиту до серверної частини для перевірки доступності обраного місця. Якщо місце недоступне, то користувачу виводиться відповідне повідомлення. Якщо обране місце не зайняте, то сервер формує запит на оплату для здійснення платежу. Після успішної оплати статус місця у базі даних змінюється на «зайняте» та додається запис в базі даних у таблицю з придбаними квитками.

Останнім етапом у придбанні квитка є генерація PDF-квитка з QR-кодом, який з’являється в особистому кабінеті користувача.

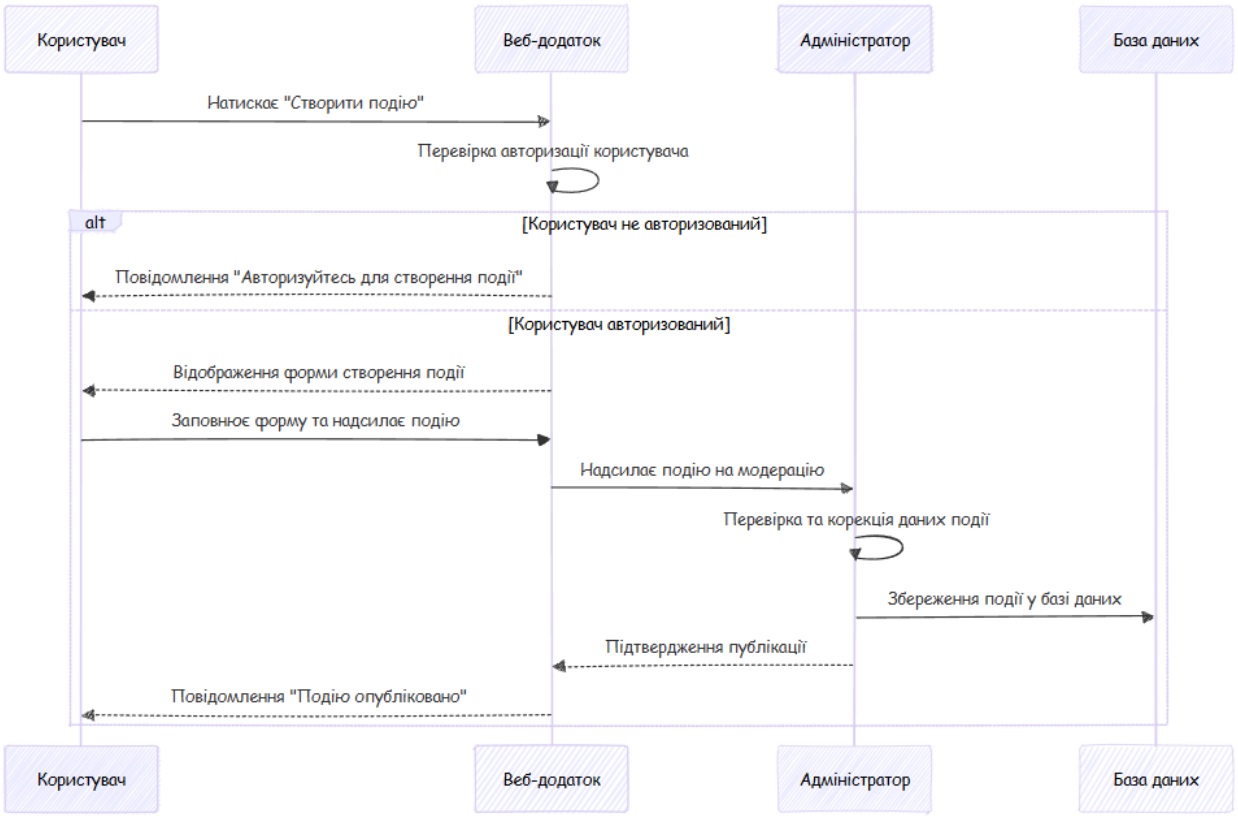


Рис. 2.8. Діаграма послідовності створення події

На рисунку 2.8. представлена діаграма послідовності для процесу створення події. Для створення події користувачу необхідно натиснути на кнопку «Створити подію» на головні сторінці сайту, після чого перевіряється авторизація користувача. Створити подію неавторизованому користувачу неможливо. Наступним етапом авторизований користувач заповнює форму події та надсилає її на модерацію адміністратору. Після цього в базі даних з’являється запис з неопублікованою подією. Адміністратор перевіряє та при необхідності корегує подію, після чого підтверджує її та публікує.

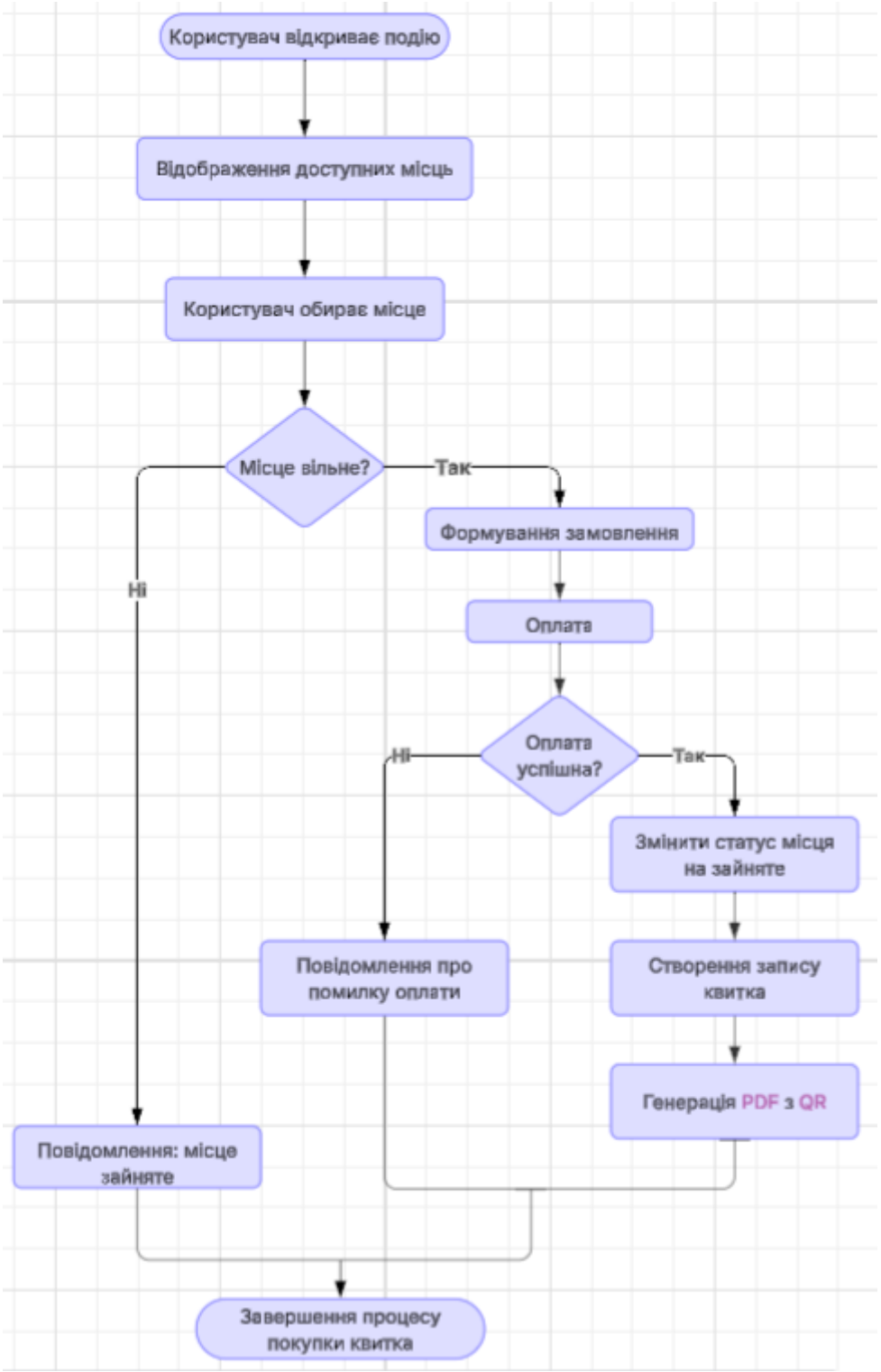


Рис. 2.9. Схема алгоритму придбання квитка

Окрім діаграми послідовності, у рамках даного підрозділу доцільно представити схему алгоритму придбання квитка. Вона відображає повну логіку обробки запиту: від вибору події користувачем до генерації електронного квитка. Така схема дозволяє узагальнити процес та продемонструвати основні етапи

роботи системи при оформленні покупки.

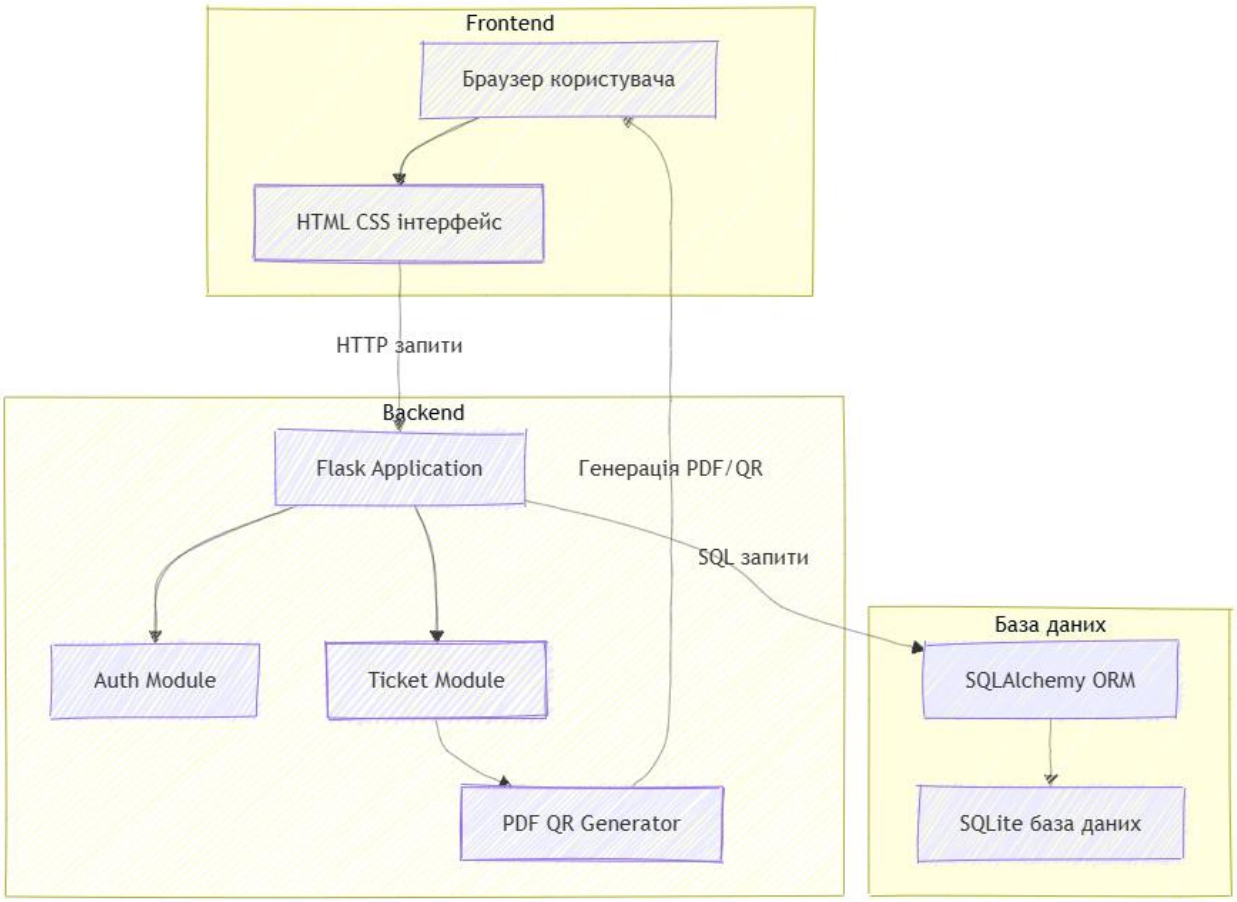


Рис. 2.10. Діаграма компонентів

Далі розглянемо діаграму компонентів (рис. 2.10.). Ця діаграма демонструє архітектуру веб-додатку на рівні основних компонентів, показуючи, як вони взаємодіють між собою.

Система складається з трьох основних рівнів:

1. Рівень представлення (Frontend) – відповідає за відображення контенту сайту. Реалізований за допомогою HTML та CSS. Frontend розробка – це робота зі створення публічної частини web-додатку, з якою безпосередньо контактує користувач, і функціоналу, який зазвичай виконується на стороні клієнта [28].

2. Серверний рівень (Backend) – відповідає за функціональність додатку. Реалізований за допомогою Python з використання фреймворку Flask. Backend — це серверна частина сайту/додатку, прихована від очей користувача. Вона

відповідає за логіку обробки запитів, збереження та отримання даних, управління обліковими записами, авторизацію, взаємодію з зовнішніми API тощо [29].

3. Рівень даних – база даних SQLite. База даних – це засіб для збирання та впорядкування інформації [30]. Відповідає за зберігання інформації про користувачів, події, квитки. Взаємодія між Flask та SQLite здійснюється через SQLAlchemy ORM.

Серверний рівень містить такі модулі:

1. Flask Application – основний компонент серверної частини, який відповідає за обробку всіх запитів від користувача, маршрутизацію запитів до потрібних модулів.
2. Auth Module – модуль авторизації та аутентифікації користувачів.
3. Ticket Module – модуль обробки подій та квитків.
4. PDF QR Generator – модуль генерації електронних квитків.

Таким чином, було розглянуто ключові алгоритми роботи сайту для придбання квитків до закладів культури за допомогою трьох діаграм. Діаграма послідовності придбання квитка відобразила логіку взаємодії користувача з веб-додатком при вибору та покупці квитка. Діаграма послідовності створення події продемонструвала процес створення нових подій організатором події та модерацію адміністратором. Діаграма компонентів показала взаємодію основних модулів системи.

Отримані діаграми слугують основою для практичної реалізації додатку та подальшого тестування його функціональності.

РОЗДІЛ 3.
ПРАКТИЧНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ДЛЯ ПРИДБАННЯ КВИТКІВ

3.1. Програмна реалізація алгоритму

Алгоритм програмно реалізований за допомогою середовища VSCode та розгорнуто на локальному сервері за адресом http://localhost:5000. Даний підхід обраний для цілей розробки, тестування та демонстрації функціоналу. В будь-який момент розробки можна відкрити додаток за вище згаданим адресом та перевірити внесені зміни.

Після встановлення Python на персональний комп'ютер, створюємо проектну дирекцію командами:

mkdir ticket_app

cd ticket_app

Далі створюємо та активуємо віртуальне середовище:

python -m venv venv

venv\Scripts\activate

В створеному віртуальному середовищі встановлюємо Flask:

pip install flask

Наступним кроком буде створення базової структури проекту на Flask, яка представлена на рисунку 3.1.

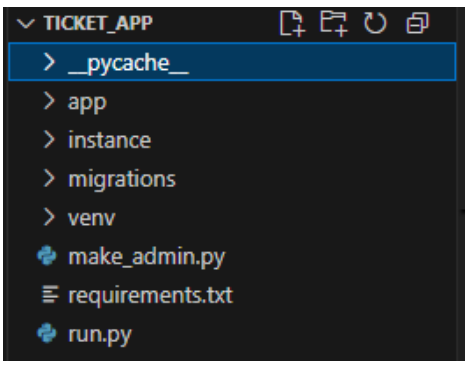


Рис. 3.1. Базова структура проекту.

На даному етапі маємо таку структуру:

- TICKET_APP – головна директорія проекту;
- _pycache_ – службова папка Python;
- app – основна папка додатку Flask, де знаходяться логіка, маршрути, моделі тощо;
- instance – папка для файлів конфігурацій (секретні ключі, бази даних);
- migrations – папка для міграцій бази даних (оновлення бази даних після редагування моделей);
- venv – папка з віртуальним середовищем;
- make_admin.py – скрипт для створення облікового запису адміністратора;
- requirements.txt – файл зі списком бібліотек Python та їх версій, які використовуються в проекті, необхідний для коректної роботи проекту;
- run.py – головний файл для запуску проекту.

Після створення файлу run.py та написання для нього коду, який представлений на рисунку 3.2, проект готовий до першого запуску.

```
run.py > ...
1   from app import create_app
2
3   app = create_app()
4
5   if __name__ == '__main__':
6       app.run(debug=True)
7
```

Рис. 3.2. Вміст файлу run.py

Після запуску файлу run.py проект буде доступний за адресом http://localhost:5000 в браузері. Тепер проект готовий до повноцінної розробки логіки, моделей, маршрутів, бази даних та html сторінок.

Наступним кроком розглянемо структуру папки app (рис.3.3).

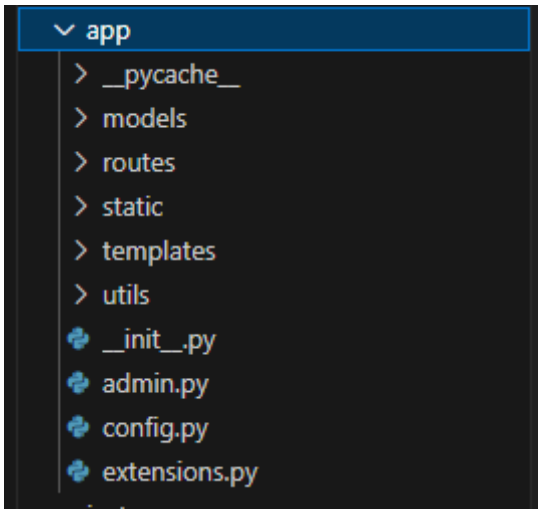


Рис. 3.3. Структура папки app

Папка models є однією з ключових у веб-додатку, побудованому на Flask із використанням бібліотеки SQLAIchemy. У ній зберігаються класи-моделі, які відповідають за структуру та логіку даних у базі даних.

На рисунку 3.4. видно три основні моделі:

- event.py – модель події;
- ticket.py – модель квитка;
- user.py – модель користувача.

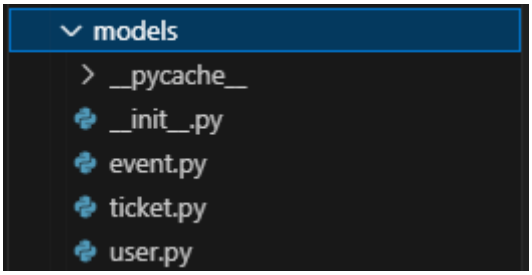


Рис. 3.4. Папка models

Папка routes – це логічний центр Flask-додатку. Він відповідає за обробку запитів користувачів та навігацію між сторінками.

У цій директорії (рис. 3.5.) маршрути поділені за функціями:

- auth.py – реєстрація, вхід/вихід, доступ до особистого кабінету користувача;
- events.py – створення та управління подіями;
- main.py – головна сторінка, перегляд усіх подій;
- payment.py – процес оплати квитків;
- ticket.py – перегляд, вибір, завантаження квитків.

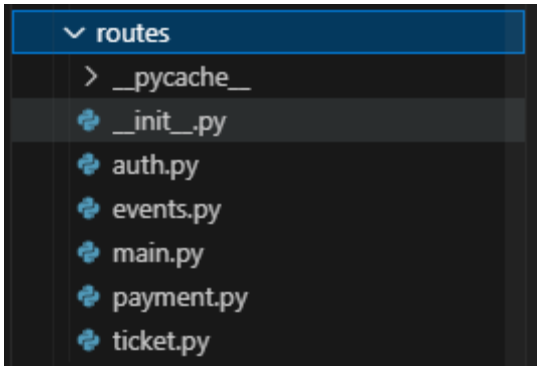


Рис. 3.5. Папка routes

Папка static містить усі статичні ресурси, які використовуються інтерфейсом:

- CSS-файли;
- зображення (іконки, логотипи);
- JavaScript-файли;
- шрифти, мультимедіа.

Папка templates (рис. 3.6.) містить HTML-шаблони, що відповідають за відображення інтерфейсу користувача. Вона складається зі сторінок для користувачів та окремої директорії для адміністратора:

- base.html – головний шаблон із загальними елементами (header, footer);
- confirm_seat.html – підтвердження вибору квитка;
- dashboard_events.html – список подій створених організатором;
- dashboard_statistics.html – сторінка статистики проданих квитків;

- dashboard.html – особистого кабінету користувача;
- edit_event.html – сторінка з можливістю редагувати події організатора;
- event_detail.html - детальна інформації про подію;
- index.html – головна сторінка з переліком всіх подій;
- login.html – сторінка авторизації;
- payment.html – сторінка оплати;
- register.html – сторінка реєстрації;
- select_seat.html – сторінка вибору квитка;
- submit_event.html – сторінка з формою для створення нової події організатором.

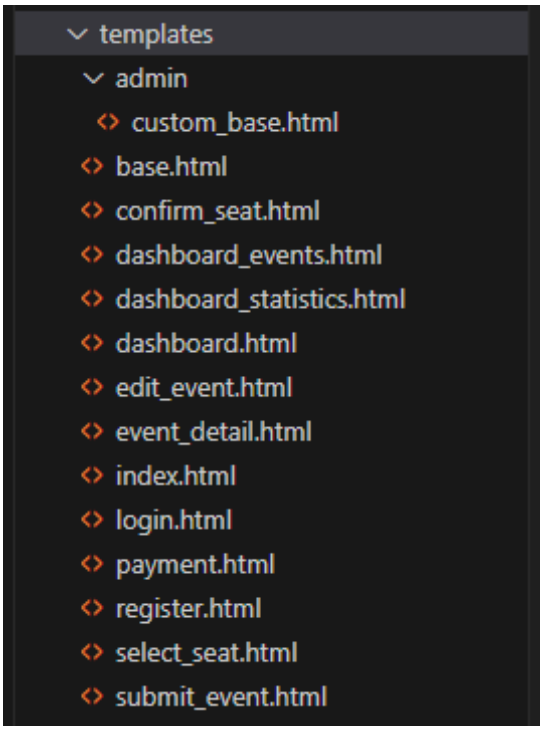


Рис. 3.6. Папка templates

Для кожної групи маршрутів створено власний Blueprint. Blueprint – це спосіб організувати маршрути в окремі модулі, щоб не тримати весь код в одному файлі. Він дозволяє розділити логіку на частини: функції авторизації окремо, функції події окремо, функції покупки окремо тощо. Цей спосіб дозволяє запобігти хаосу, що в свою чергу спрощує орієнтування в коді.

Папки models та routes є основними структурними елементами всієї логіки системи. Разом ці дві частини забезпечують цілісність і функціональність системи.

Кожен клас в файлах папки models відповідає одній таблиці, а кожний атрибут класу – одному стовпцю в таблиці. Flask-SQLAIchemy автоматично перетворює ці класи на SQL-запити та створює відповідні таблиці в базі даних. Код файлу ticket.py представлений на рисунку 3.7, де можна побачити атрибути класу Ticket, які потім сформують таблицю в базі даних.

app > models > ticket.py > Ticket

```python
from datetime import datetime
from app.extensions import db

class Ticket(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    event_id = db.Column(db.Integer, db.ForeignKey('event.id'), nullable=False)
    purchase_date = db.Column(db.DateTime, default=datetime.utcnow)

    row = db.Column(db.Integer, nullable=True)
    seat_number = db.Column(db.Integer, nullable=True)
    sector = db.Column(db.String(20), nullable=True)

    is_paid = db.Column(db.Boolean, nullable=False, default=False)
    payment_date = db.Column(db.DateTime, nullable=True)

    user = db.relationship('User', back_populates='tickets')
    event = db.relationship('Event', back_populates='tickets')
```

Рис. 3.7. Код файлу ticket.py

В класах описується логіка зв’язків між сутностями, наприклад, квиток прив’язується до конкретного користувача, або квиток прив’язується до конкретної події.

Таким чином, models забезпечує опис основних сутностей веб-додатку. Моделі дають змогу структуровано зберігати інформацію про користувачів, події та придбані квитки, а також реалізувати логіку взаємодії між сутностями.

Папка routes реалізує логіку обробки запитів користувачів, керує маршрутизацією та взаємодіє між фронтендом і бекендом. У ній зосереджений

весь функціонал, який має веб-додаток.

Створення події на сайті можливе за двох варіантах:

- перший - подію створює адміністратор системи. У цьому випадку подія не прив’язується до жодного користувацького акаунта, а її редагування доступне лише адміністратору;
- другий – подію створює зареєстрований користувач.

```python
# Подання події користувачем
@ticket_bp.route('/submit_event', methods=['GET', 'POST'])
@login_required
def submit_event():
    if request.method == 'POST':
        title = request.form.get('title')
        description = request.form.get('description')
        date = request.form.get('date')
        location = request.form.get('location')
        price = request.form.get('price')

        if not title or not description or not date or not location or not price:
            flash('Будь ласка, заповніть усі поля.', 'danger')
            return render_template('submit_event.html')

        try:
            date = datetime.strptime(date, "%Y-%m-%d")
            price = float(price)
        except ValueError as e:
            flash(f'Невірний формат дати або ціни: {str(e)}', 'danger')
            return render_template('submit_event.html')

        new_event = Event(
            title=title,
            description=description,
            date=date,
            location=location,
            price=price,
            is_approved=False,
            submitted_by=current_user.id
        )

        db.session.add(new_event)
        db.session.commit()
        flash('Подію подано на розгляд адміністрації.', 'success')
        return redirect(url_for('auth.dashboard_events'))

    return render_template('submit_event.html')
```

Рис. 3.8. Створення події користувачем

На рисунку 3.8. представлений код який дозволяє створити подію користувачу. Користувач через інтерфейс заповнює форму створення події. Після натискання кнопки «Створити подію» система перевіряє коректність введених даних. Якщо не всі поля заповнені, або вони заповнені некоректно, то виводиться відповідне повідомлення про помилку. Якщо всі поля заповнені правильно, подія зберігається у базі даних з атрибутом is_approved=False, тобто вона спочатку потрапляє на розгляд адміністратору. Після чого адміністратор в адміністративній панелі редагує її та публікує на сайт.

Далі розглянемо фрагмент коду, який реалізує логіку вибору та бронювання місць у залі під час купівля квитка (рис. 3.9).

```python
# ВИБІР МІСЦЯ
@ticket_bp.route('/select_seat/<int:event_id>', methods=['GET', 'POST'])
@login_required
def select_seat(event_id):
    event = Event.query.get_or_404(event_id)
    # Очищуємо прострочені бронювання
    _clean_expired_reservations(event_id)
    event_type = event.event_type.strip().lower() if event.event_type else ''
    # Для музеїв/виставок — одразу йдемо на підтвердження покупки
    if event_type in ['музей', 'виставка']:···

    if request.method == 'GET':
        # Отримуємо всі квитки (куплені та заброньовані)
        taken_tickets = Ticket.query.filter_by(event_id=event.id).all()

        occupied_keys = { f"{t.sector}:{t.row}:{t.seat_number}" for t in taken_tickets }
        from_confirm = request.args.get('from_confirm', False)
        return render_template('select_seat.html', event=event, occupied_keys=occupied_keys, from_confirm=from_confirm)
    # Обробка старої ручної форми вибору місця
    try:···
    except (ValueError, TypeError):
        flash('Некоректні дані місця.', 'danger')
        return redirect(request.url)
    # Перевіряємо чи місце вже зайняте або заброньоване
    existing = Ticket.query.filter_by(
        event_id=event.id,
        row=row,
        seat_number=seat_number,
        sector=sector
    ).first()
    if existing:
        flash('Це місце вже зайняте. Оберіть інше.', 'danger')
        return redirect(request.url)
    # Бронюємо місце
    reservation = Ticket(
        user_id=current_user.id,
        event_id=event.id,
        row=row,
        seat_number=seat_number,
        sector=sector,
        is_reserved=True,
        reserved_until=datetime.utcnow() + timedelta(minutes=10)  # Бронювання на 10 хвилин
    )
    db.session.add(reservation)
    db.session.commit()

    return render_template(···
```

Рис. 3.9. Вибір та бронювання місця

Щоб уникнути ситуацій, коли два користувачі одночасно намагаються зайняти одне й те саме місце, було створено даний алгоритм. Перед тим як користувачу показується схема місця, викликається допоміжна функція _clean_expired_reservations, вона виконує очищення заброньованих місць, термін яких минув. Система вважає бронювання простроченим, якщо минуло більше 10 хвилин він його створення. Такі записи видаляються з бази даних, щоб місця знову були доступними для інших користувачів.

Далі система перевіряє тип події. Якщо подія музей чи виставка, то система не пропонує вибір місця, а відразу переходить до підтвердження покупки. Якщо

подія має зал та місце, то виводиться схема залу, де користувач чітко бачить вільні та зайняті місця.

Рис. 3.10. Вибір місця

На схемі залу (рис. 3.10.) місця відображаються у різних станах. Вільні місця позначені темно-синім кольором і є доступними для натискання. Зайняті місця відображаються сірим кольором та є неклікабельними, що унеможливлює їх вибір. Обране користувачем місце автоматично підсвічується оранжевим, що дозволяє чітко бачити, яке місце обране.

Після вибору місця система перевіряє чи місце не заброньоване. Якщо місце вільне – створюється бронювання. Запис в базі даних з квитками створюється в момент коли користувач обрав місце на схемі та натиснув кнопку «Підтвердити вибір». Заброньований квиток у базі даних відрізняється від купленого тим, що для нього поле is_reserved має значення 1, а reserved_until містить дату та час закінчення бронювання, тоді як у куплених квитків ці поля мають значення NULL. Також у забронюваного квитка відсутні значення у полях is_paid та payment_data.

Після створення бронювання користувач перенаправляється на сторінку підтвердження покупки, де може перейти до оплати. На сторінці оплати система

знаходить в базі даних необхідний заброньований квиток та перевіряє чи бронювання в 10 хвилин не закінчилось. Після чого система перевіряє валідність даних введених користувачем та додає квиток зі створеним PDF-файлом в особистий кабінет користувача.

Для підвищення рівня безпеки веб-додатку застосовувалися заходи для захисту персональних даних користувача. До них належить хешування паролю перед збереженням у базі даних, перевірка введених даних на стороні сервера та обмеження прав доступу залежно від ролі користувача (адміністратор, організатор, звичайний користувач).

Окрім захисту даних користувачів, важливою частиною безпеки системи є правильна організація адміністрування системи. У процесі розробки веб-додатку важливо забезпечити можливість простого та зручного керування даними системи. Адмініструвати систему через редактор коду та взаємодіяти з базою даних напряму було б не зручно. Крім того, надання доступу до коду іншому адміністратору було б ризиковано, адже виникає велика вірогідність випадкових помилок або несанкціонованих втручання в логіку системи, що може призвести до великих проблем під час відновлення працездатності веб-додатку.

Щоб уникнути цих проблем, у додаток було інтегровано систему адміністрування Flask-Admin. Flask-Admin - це розширення для Flask, яке дозволяє створювати адміністративні панелі для роботи з базами даних, включаючи SQLAlchemy [31]. Вона надає зручний інтерфейс для управління даними без потреби доступу до програмного коду. Flask-Admin автоматично генерує форми для редагування та відображення даних на основі моделей [31]. За допомогою Flask-Admin адміністратор може додавати, редагувати та видаляти події. Також, за необхідністю, може керувати іншими сутностями додатку, користувачами чи придбаними квитками. Після створення адміністративного інтерфейсу та запуску додатку, його можна відкрити у браузері за адресом http://localhost:5000/admin/.

Для того щоб надати користувачу права адміністратора, у базі даних передбачено поле is_admin. За замовчування при реєстрації нового користувача це

поле має значення 0, тобто користувач не має права адміністратора. Щоб зробити певного користувача адміністратором, в першу чергу, потрібно зареєструватися на сайті з тім електронним адресом, який буде використовуватися як адміністративний. Після успішної реєстрації у таблиці User з'являється запис про користувача.

Далі потрібно запустити код, який представлений на рисунку 3.11, вказавши в змінній admin_email ту адресу, якій хочемо надати права адміністратора.

```python
from app import create_app
from app.extensions import db
from app.models.user import User

app = create_app()

with app.app_context():
    admin_email = 'admin@gmail.com'
    user = User.query.filter_by(email=admin_email).first()
    if user:
        user.is_admin = True
        db.session.commit()
        print(f"Користувач {admin_email} тепер адміністратор.")
    else:
        print(f"Користувача з email {admin_email} не знайдено.")
```

Рис. 3.11. Код файлу make_admin.py

Після запуску скрипт знаходить користувача за email та змінює поле is_admin з 0 на 1 та зберігає зміни у базі даних (рис. 3.12).

	id	username	email	password	is_admin
1	1	admin	admin@gmail.com	pbkdf2:sha256:1000000$JlvKb9OCXyGv0AXA$6c21fc273...	1

Рис. 3.12. Користувач з адмін-правами у таблиці User

Якщо користувача з таким email знайдено, то в консолі виводиться відповідне повідомлення. Якщо ж email не існує в базі даних, то виводиться повідомлення з помилкою.

Таким чином надання адміністративних прав користувачу виконується лише розробником через програмний код.

Після реалізації функціоналу системи важливою складовою стає розробка інтерфейсу, який забезпечить зручність взаємодії користувача з веб-додатком. Інтерфейс має бути інтуїтивно зрозумілим та структурованим.

Головна сторінка (рис. 3.13.) веб-додатку є першим екраном, який бачить користувач після переходу на сайт. Основним призначенням цієї сторінки є швидке надання доступу до переліку подій та основних функцій системи.

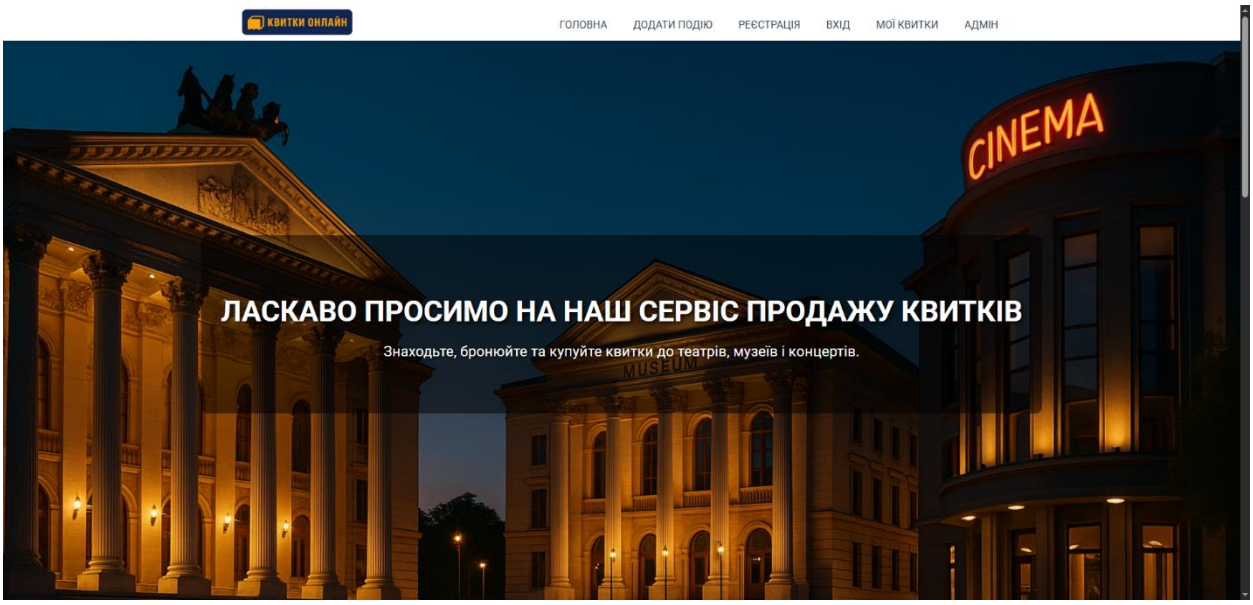


Рис. 3.13. Головна сторінка додатку

У верхній частині сторінки розташоване навігаційне меню, яке включає кнопки «Головна», «Додати подію», «Реєстрація», «Вхід», «Мої квитки» та «Адмін». Також зліва від навігаційного меню присутній логотип додатку, який має таку ж функцію як кнопка «Головна», перенаправляти користувача на головну сторінку з будь якої іншої сторінки. Під хедером з навігацією розташований банер з надписами.

Нижче розташована секція фільтрів подій, яка дозволяє користувачам швидко знаходити заходи за такими критеріями:

- тип події (театр, музей, виставка, концерт);

- місто;
- дата;
- максимальна ціна.

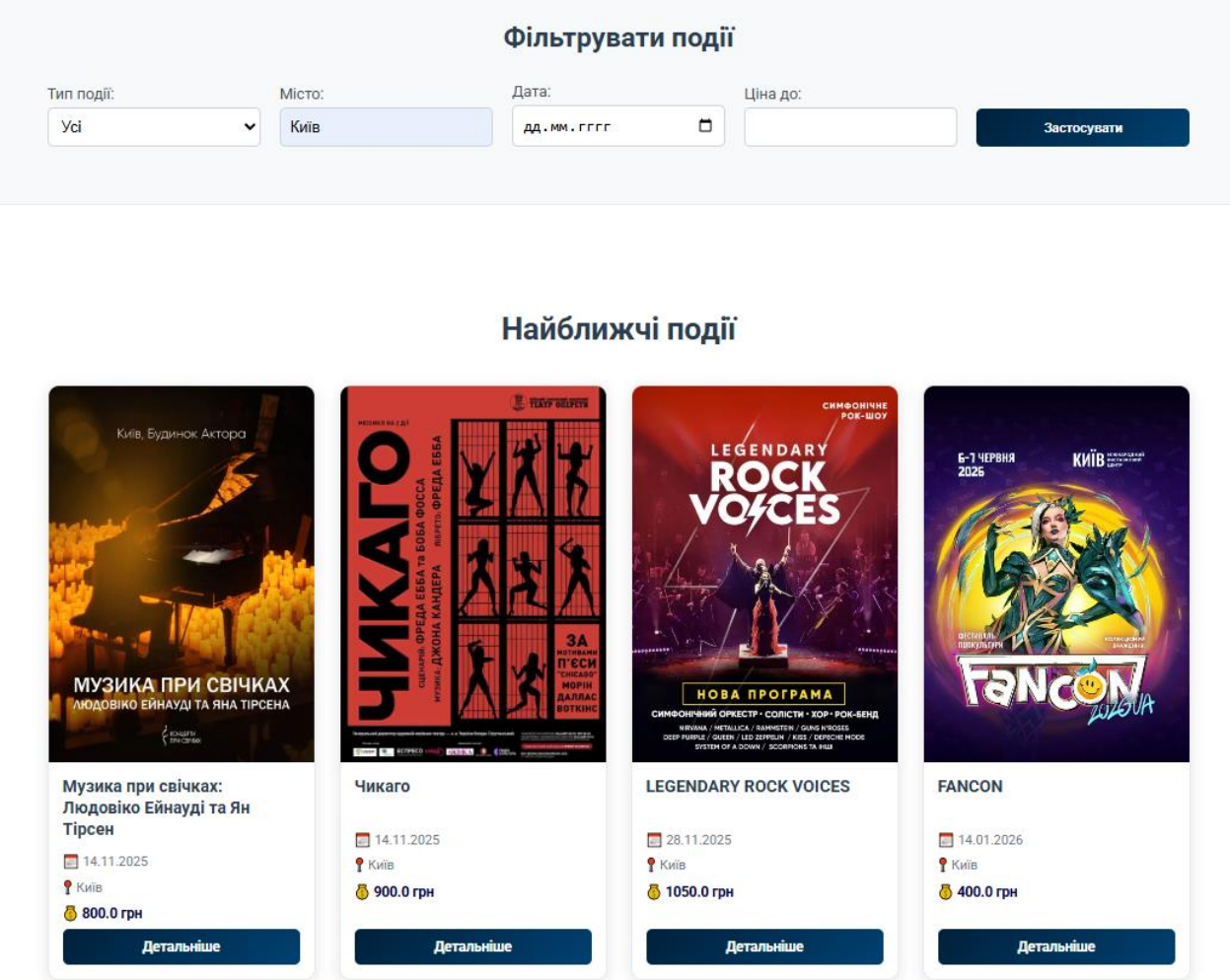


Рис. 3.14. Фільтри

На рисунку 3.14. можна побачити застосування фільтру за містом, в даному випадку відображені всі події в місті Київ.

Далі йде блок з переліком всіх подій (рис. 3.15.), які присутні на сайті. Події в списку відображаються по даті, тобто подія, яка буде проведена найближчим часом, знаходиться в самому верху списку.

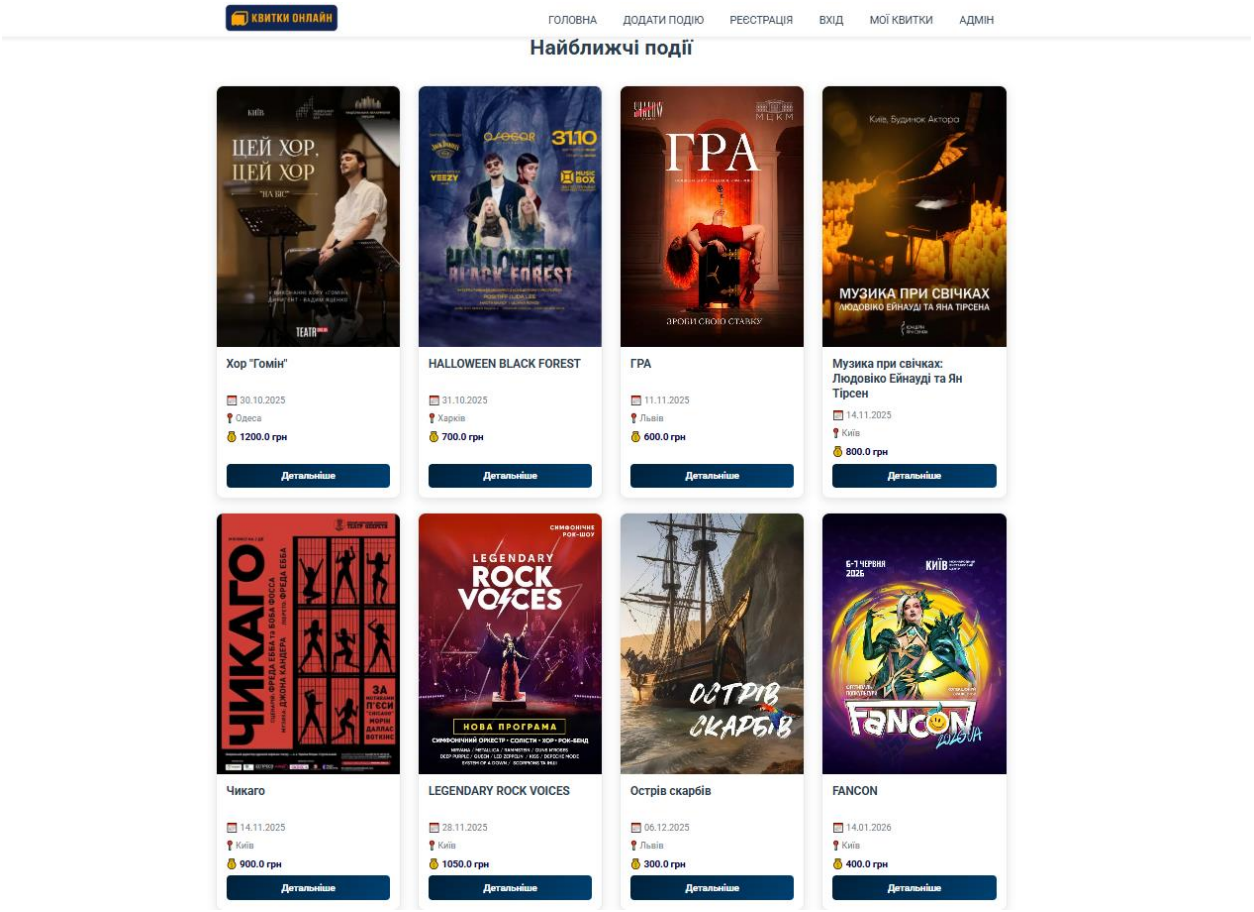


Рис. 3.15. Найближчі події

На картках подій відображені головні дані: постер, назва, дата проведення, місто, ціна квитка та кнопка «Детальніше» для переходу на сторінку опису події.

Після вибору бажаної події користувач переходить на сторінку з детальної інформацією про подію. На ній користувач може побачити окрім інформації з картки події також детальний опис події та кількість вільних квитків якщо тип події театр або концерт, що і продемонстровано на рисунку 3.16. Для типу події музей або виставка кількість необмежена, тому поле з кількістю вільних місць буде відсутнє.

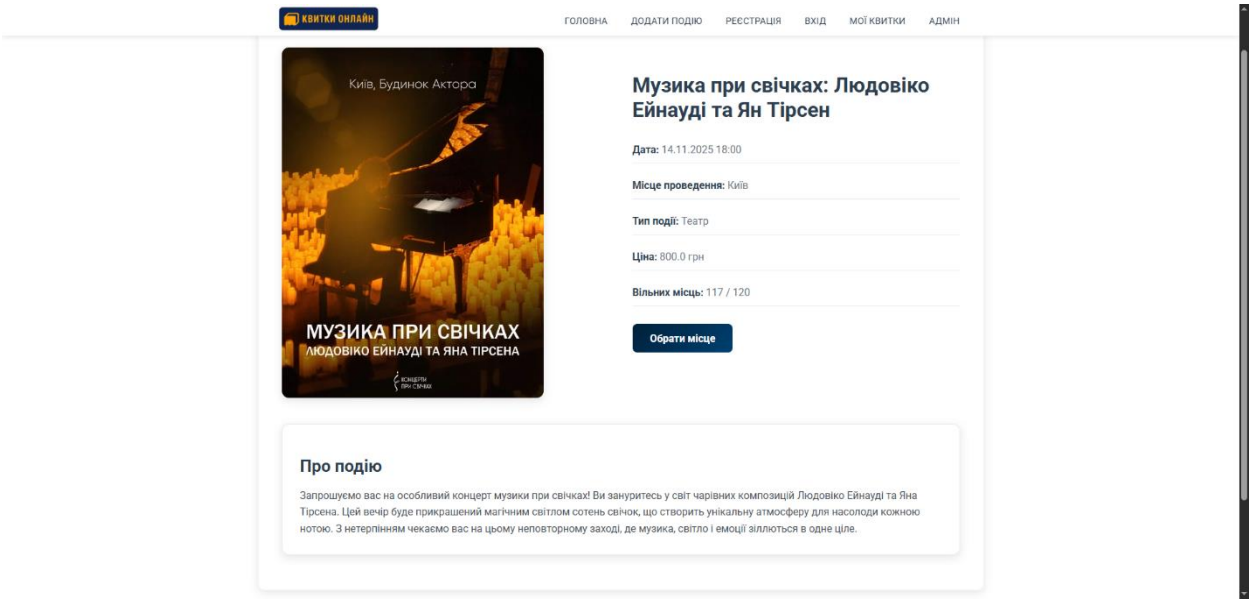


Рис. 3.16. Детальна інформація про подію

3.2. Оцінка отриманих результатів та тестування

Після завершення програмної реалізації виникає необхідність у підтвердженні функціональності та стабільної роботи веб-додатку. Тому наступним етапом буде оцінка отриманих результатів та тестування веб-додатку.

Тестування є ключовим етапом верифікації, тому буде проведено тестування ключових функцій додатку, включаючи:

- реєстрація та авторизація користувача;
- вибір місця;
- процес покупки квитка;
- формування електронного квитка;
- панель адміністратора та управління подіями.

Перед покупкою квитка користувачу необхідно зареєструватися або авторизуватися. В випадку якщо користувач не авторизований замість кнопки «Обрати місце» буде надпис «Увійдіть, щоб купити квиток».

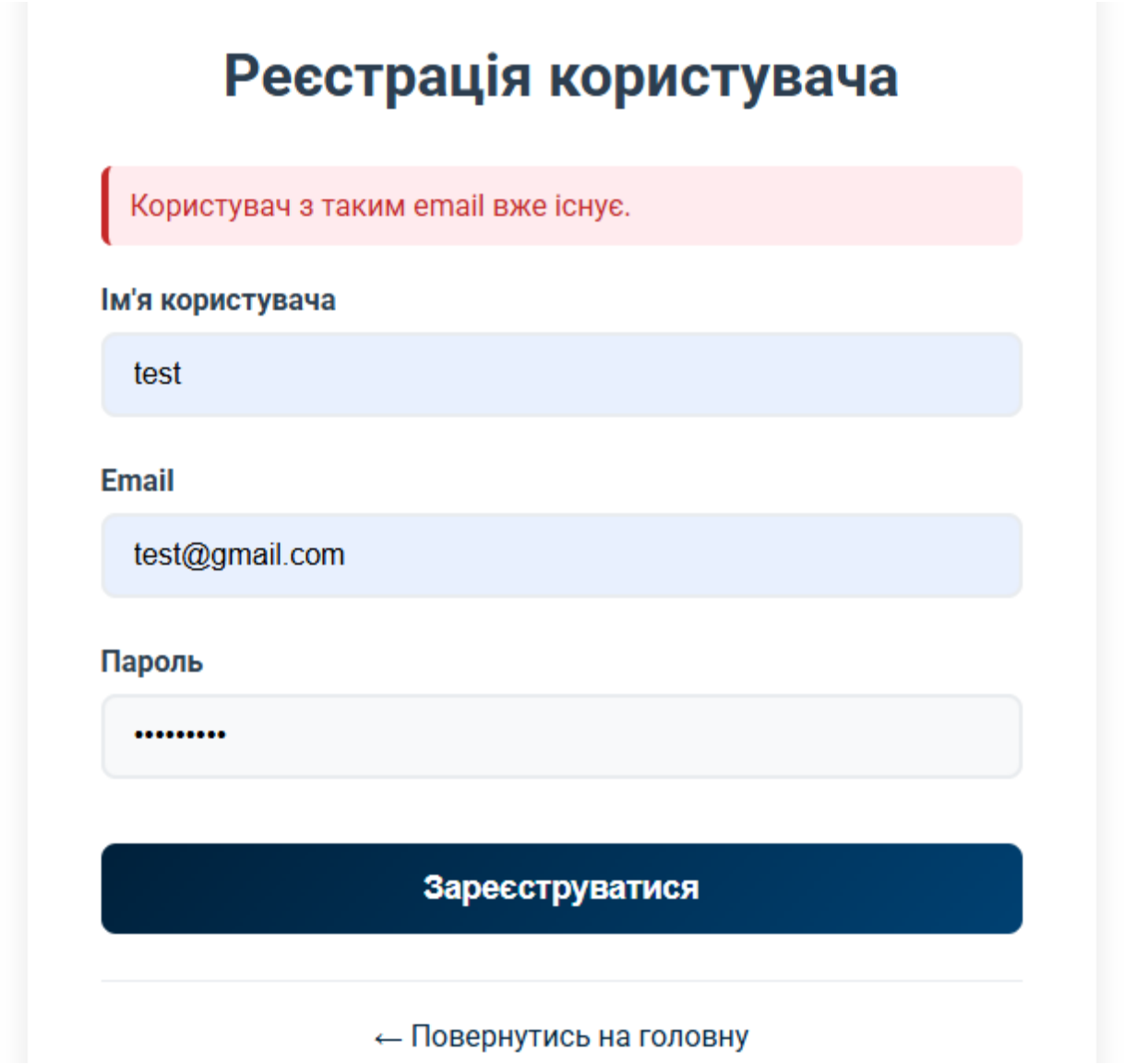


Рис. 3.17. Реєстрація користувача

Форма реєстрації представлена на рисунку 3.17. Форма містить такі поля:

- ім’я користувача;
- email;
- пароль.

Після заповнення даних та надсилання форми застосунок перевіряє, чи існує вже користувач із введеним email. У разі виявлення такого ж email система повідомляє користувача про помилку flash-повідомленнями як продемонстровано на рисунку 3.15. Якщо всі дані коректні, пароль шифрується та новий користувач зберігається в базі даних.

Після успішної авторизації користувач може перейти до обрання місця в

залі. Вибір місця представлений в вигляді інтерактивної схеми залу (3.18).

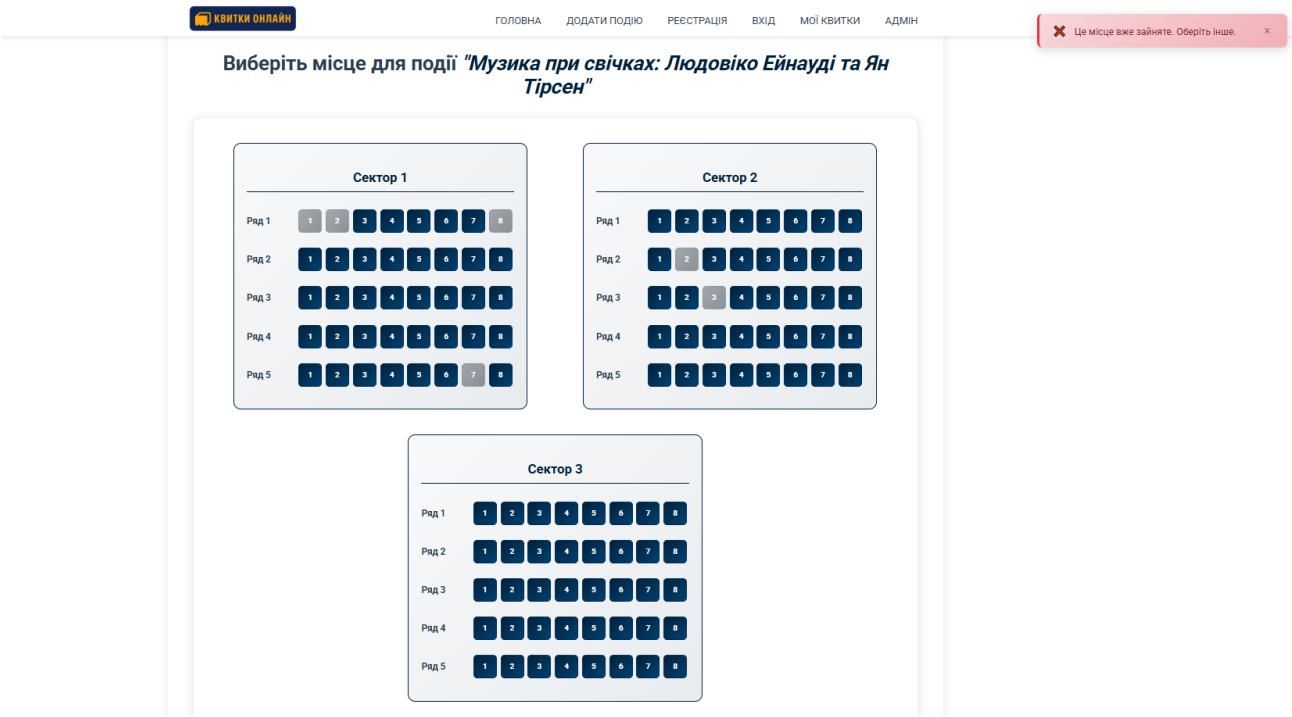


Рис. 3.18. Вибір місця

На даному рисунку представлена ситуація, коли два користувачі одночасно відкрили сторінку вибору місць. Перший користувач вибрав конкретне місце та забронював його, у результаті чого воно стало зайнятим у базі даних. Другий користувач в цей час також переглядав схему залу, і обране їм місцем ще відображалося як вільне, оскільки сторінка не оновлювалася в реальному часі.

Після натискання кнопки «Обрати місце» система миттєво створює запис у базі даних з параметрами місця та статусом is_paid = 0 — квиток вважається забронюваним, але ще не оплаченим. Це гарантує, що інший користувач уже не може обрати те саме місце.

Навіть якщо двоє користувачів натиснуть кнопку одночасно, їхні запити обробляються по черзі на рівні бази даних.

- Перший успішно створює бронювання.
- Другий отримує помилку «місце вже зайняте», і інтерфейс оновлюється відповідним повідомленням.

Таким чином, алгоритм виключає можливість подвійної броні або створення двох однакових квитків.

Після вибору бажаного місця користувач потрапляє на сторінку з підтвердженням покупки квитка (рис. 3.19), на якому зазначено назву події, дату проведення, місце та вартість квитка. Ці дані дозволяють покупцю перевірити коректність вибору перед переходом до оплати квитка.

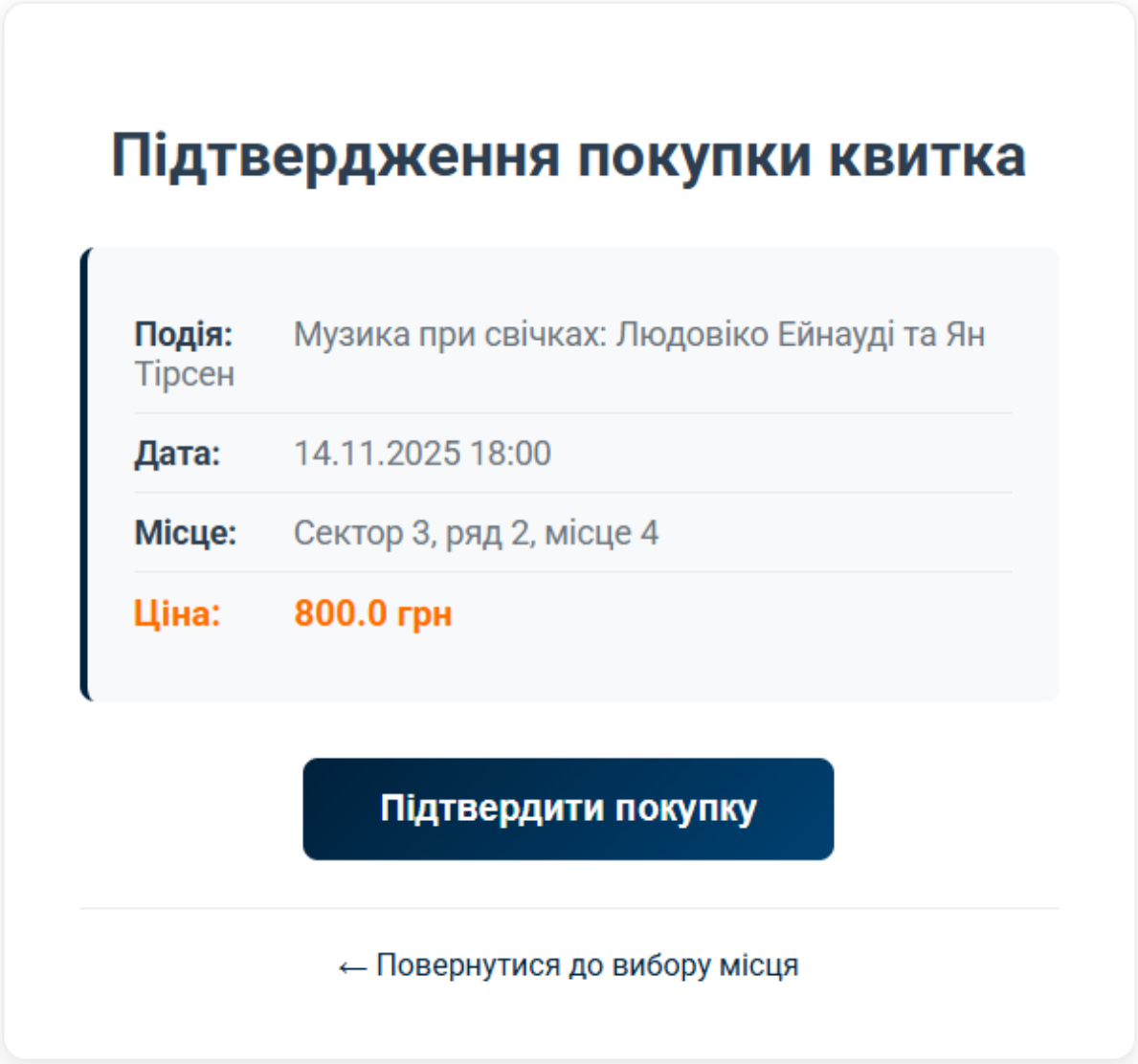


Рис. 3.19. Підтвердження покупки квитка

Після перевірки даних та натискання кнопки «Підтвердити покупку» користувач потрапляє до сторінки оплати (рис. 3.20). Сторінка оплати квитка є

заключним етапом процесу придбання. На цій сторінці відображається загальна інформація про замовлення та форма для внесення даних карти.

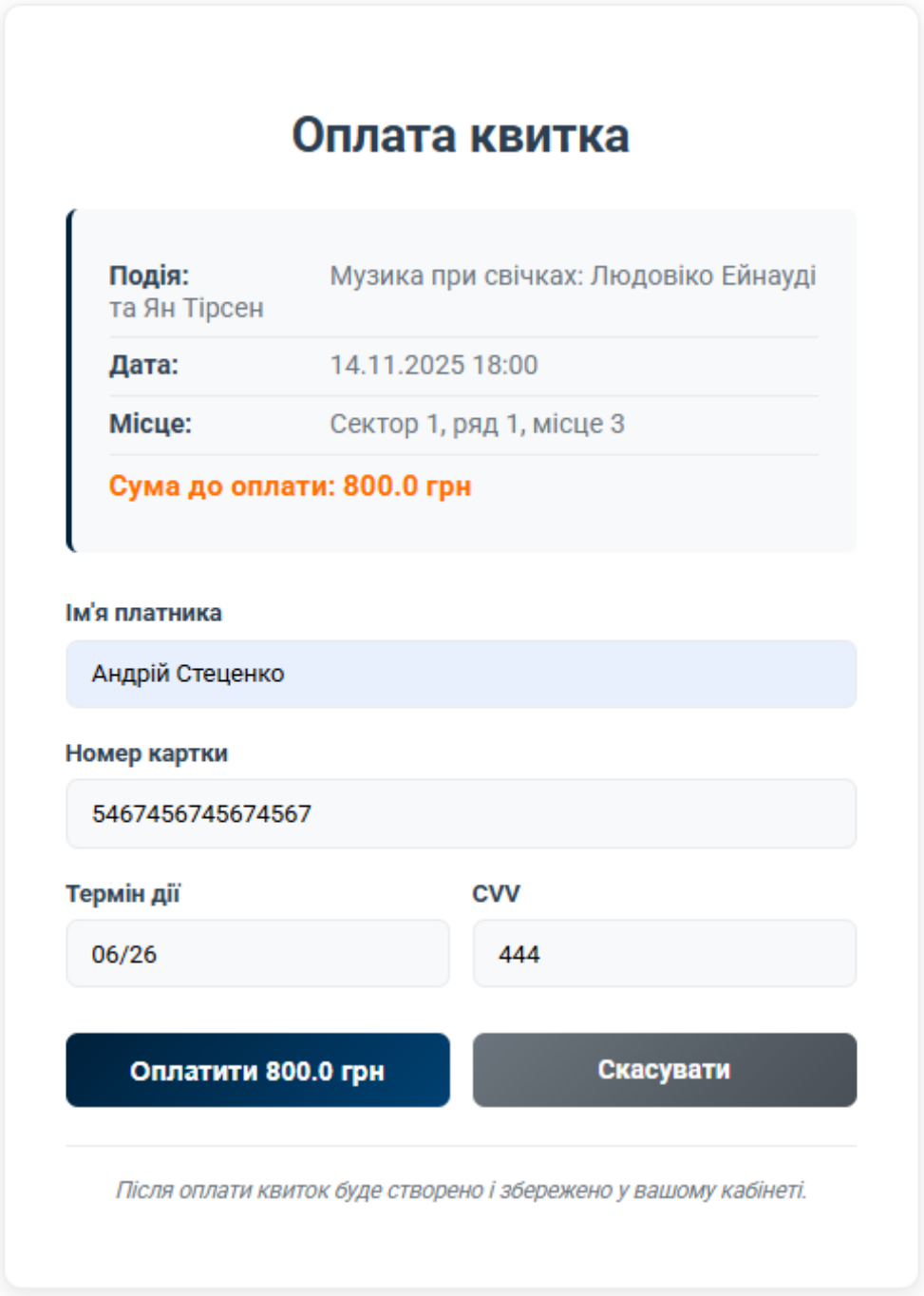


Рис. 3.20. Сторінка оплати квитка

Сторінка оплати квитка призначена для введення платіжних даних користувача та підтвердження покупки. На цій сторінці реалізовано перевірку валідності введених даних, що запобігає можливості помилки під час здійснення

платежу.

Система не дозволяє продовжити оплату, якщо введено некоректні або неповні дані, зокрема:

- у полі «Номер карти» користувач повинен ввести рівно 16 цифр;
- у полі «CVV» необхідно ввести 3 цифри;
- поле «Ім’я платника» не може бути порожнім;
- у полі «Термін дії» перевіряється коректність формату та актуальність введеної дати.

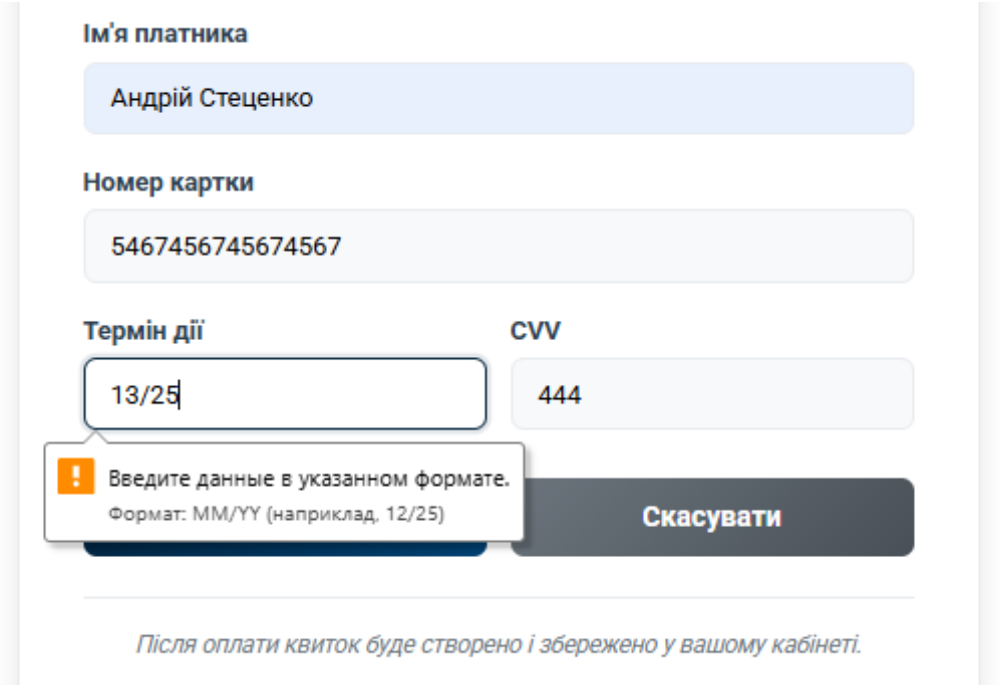


Рис. 3.21. Помилка «Невірний формат даних»

На рисунку 3.21. представлено приклад ситуації, коли користувач ввів невалідний термін дії карти – 13/25, тобто 13-й місяць, якого не існує. У результаті система виводить відповідне повідомлення.

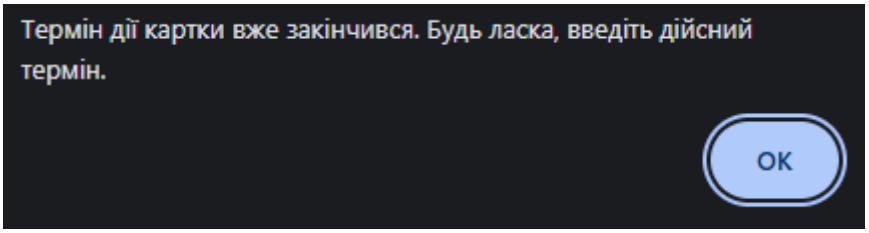


Рис. 3.22. Помилка «Термін дії карти закінчився»

На рисунку 3.22. показано інший випадок, коли термін дії карти вже минув.

Завдяки цим перевіркам забезпечується надійний процес оплати та захист від помилок користувача.

Після успішної оплати створюється запис в базі даних про куплений квиток та автоматично формується електронний квиток з QR-кодом, який зберігається в особистому кабінеті користувача та доступний для завантаження у форматі PDF.

Таким чином, модуль оплати забезпечує логічне завершення процесу купівлі квитку.

Особистий кабінет користувача містить дві вкладки. Перша вкладка «Мої подій», яка представлена на рисунку 3.23, призначена для відображення усіх квитків придбаних користувачем.

На сторінці представлений перелік куплених квитків із зазначенням базової інформації. Для кожного квитка передбачено дві кнопки, для взаємодії з квитком:

- «Скасувати» для скасування квитка;
- «Завантажити PDF» для завантаження PDF-файла з QR-кодом.

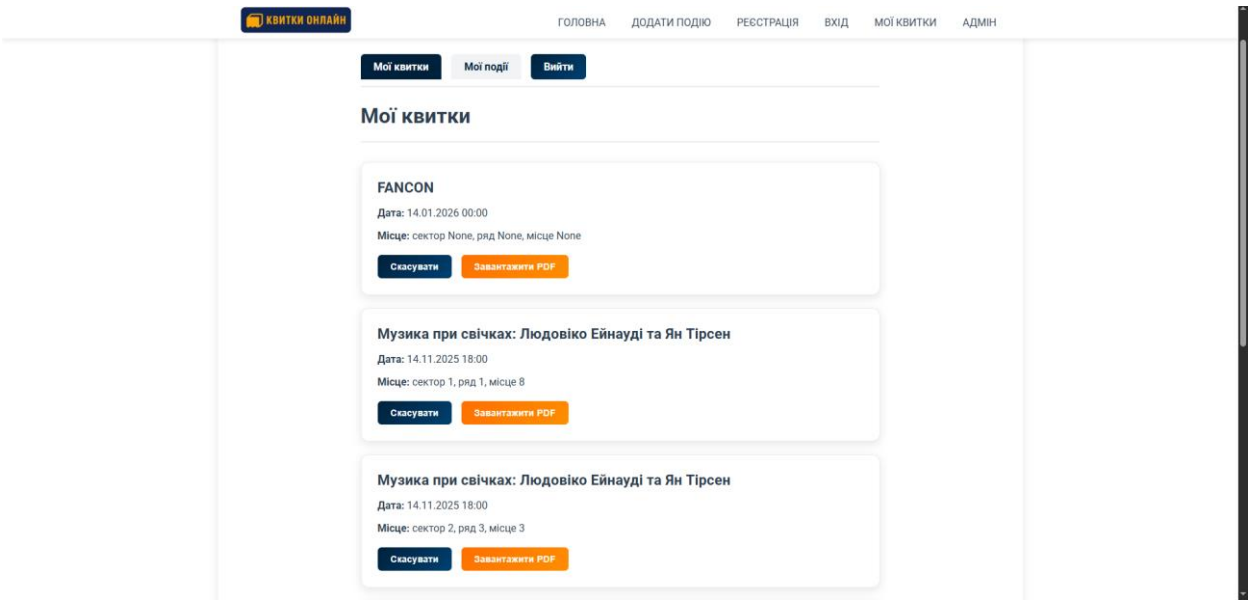


Рис. 3.23. Особистий кабінет «Мої квитки»

Після завершення оплати користувач може завантажити свій квиток у вигляді PDF-файлу. Структуру квитка можна побачити на рисунку 3.24. Документ містить всю необхідну інформацію про замовлення: назву події, дату й час проведення, обране місце, ID квитка та ім’я покупця.

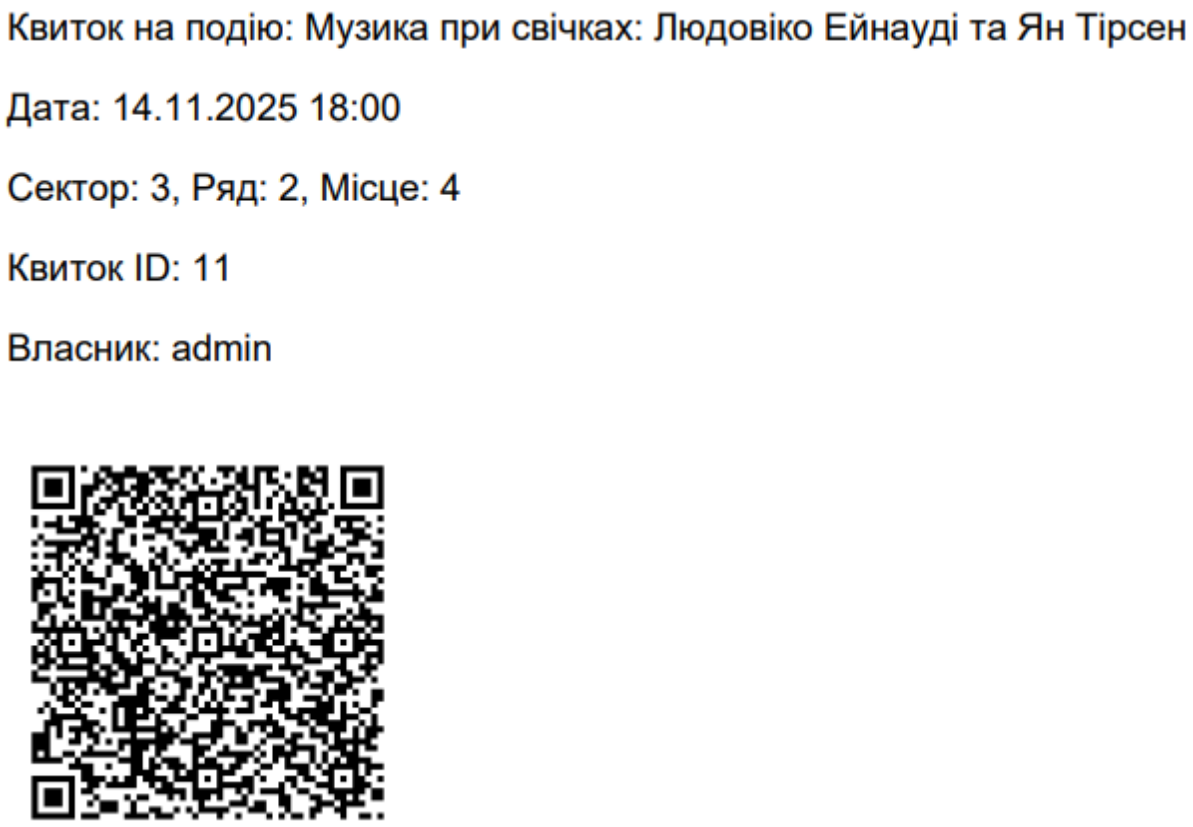


Рис. 3.24. Структура PDF-квитка

Основним елементом PDF-квитка є QR-код, який генерується автоматично під час формування документу. Цей код закодовує унікальні дані квитка, які зазначені в самому PDF-файлі.

Для користувачів, які мають можливість створювати власні культурні заходи, в особистому кабінеті користувача реалізовано вкладку «Мої події» (рис. 3.25). На цій сторінці відображаються всі створені події користувача незалежно від опублікована події на сайті чи ні. Для події, яка створена організатором події та надіслана на розгляд адміністратору, але яка поки не опублікована на сайті, в полі статусу буде надпис «На розгляді».

Для кожної події користувача реалізовано дві кнопки: «Переглянути

статистику» та «Редагувати».

Окрім списку наявних подій, у верхній частині сторінки є кнопка «Створити нову подію», яка також присутня в хедері сторінок в навігаційному меню.

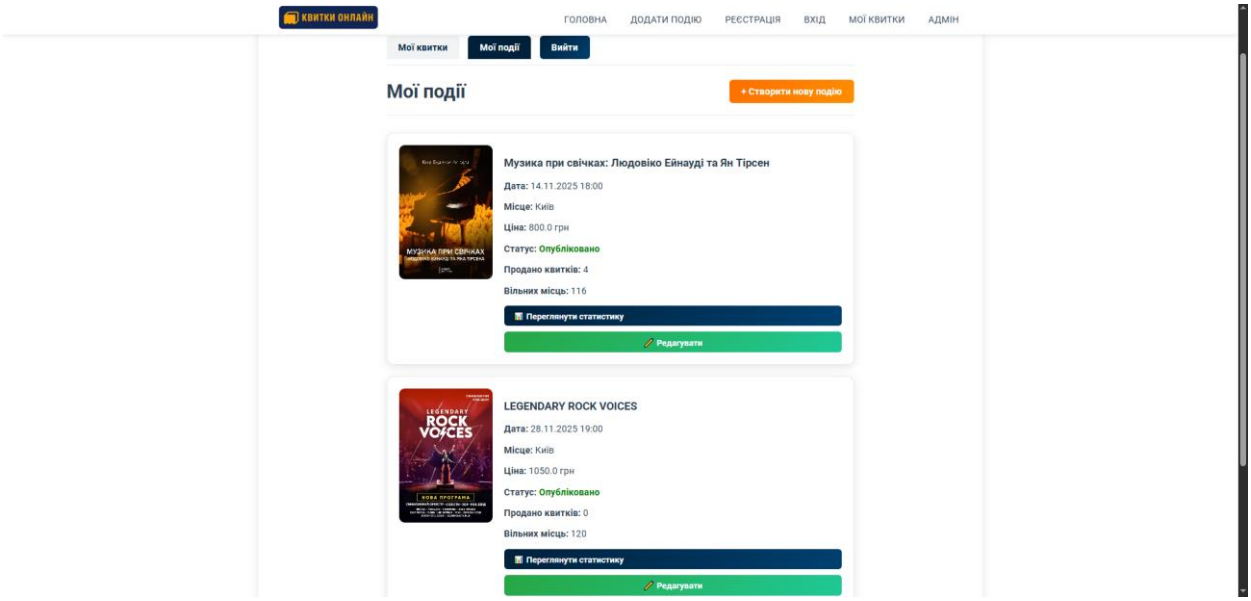


Рис. 3.25. Особистий кабінет «Мої події»

На сторінці «Редагувати подію» (рис. 3.26.) користувач може змінювати основну інформацію: опис події, назву події, дату та час проведення події, місце проведення події та вартість квитка.

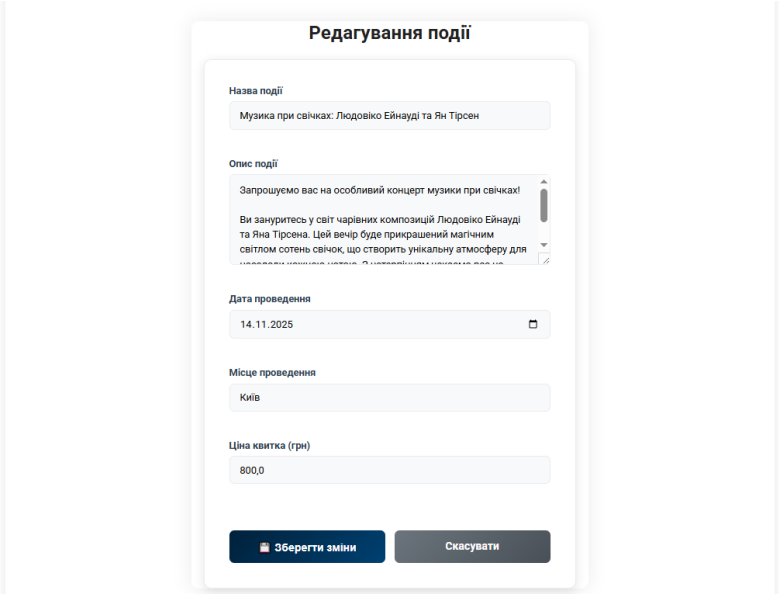


Рис. 3.26. «Редагувати подію»

Сторінка статистики проданих квитків (рис. 3.27.) надає організатору комплексну інформацію про результати продажів для конкретної події. Ця сторінка містить основні показники ефективності події.

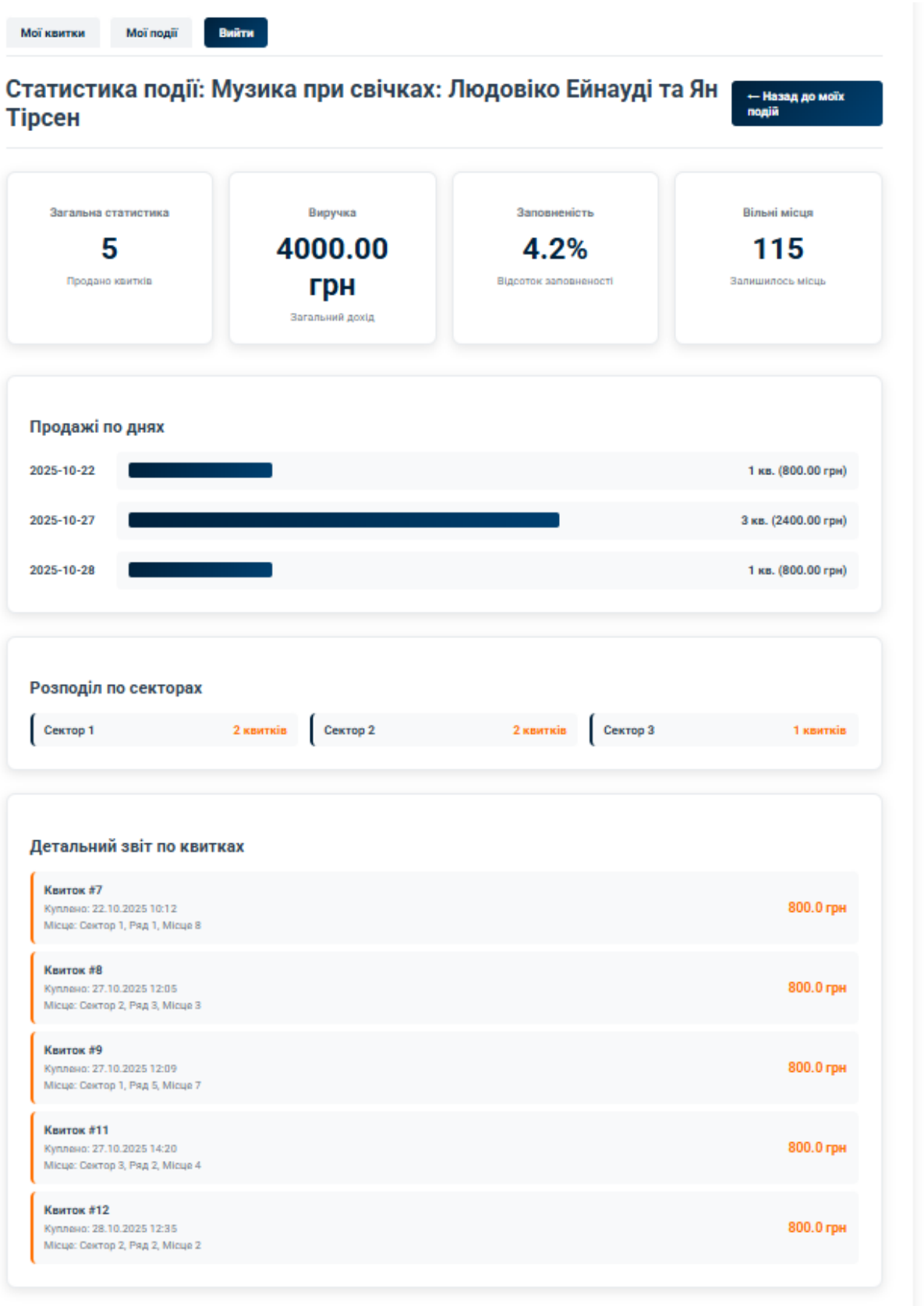


Рис. 3.27. Статистика проданих квитків

На сторінці відображаються наступні аналітичні показники:

- загальна кількість проданих квитків;
- загальна виручка з продажу – сума всіх успішних оплат;
- заповненість залу у відсотках – частка проданих квитків від загальної кількості місць в залі;
- кількість вільних місць в залі;
- продажі по днях – статистика проданих квитків за конкретний день;
- розподіл продажів по секторах – аналіз, який сектор користується більшою популярністю у користувачів;
- детальний звіт по квитках – таблиця з повною інформацією про кожен проданий квиток.

Завдяки формуванню таких аналітичних даних організатор отримую інструмент контролю продажів та прогнозування подальшого попиту.

Для створення нової події користувач має натиснути кнопку «Створити нову подію», після чого користувачу відкривається форма (3.28), де він може заповнити основну інформацію про подію та надіслати її на розгляд адміністратору.

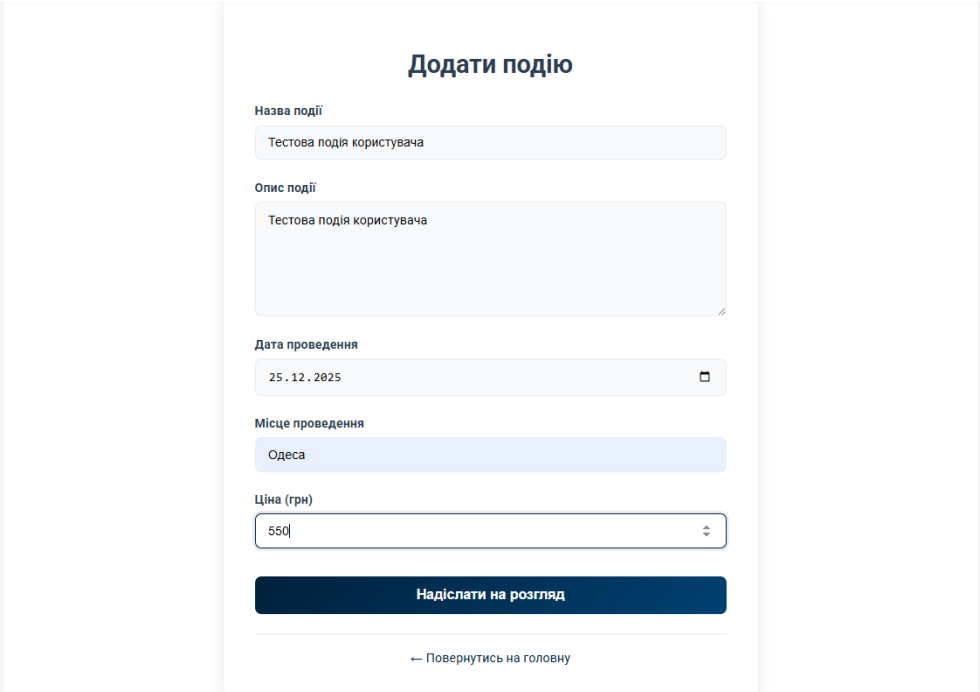
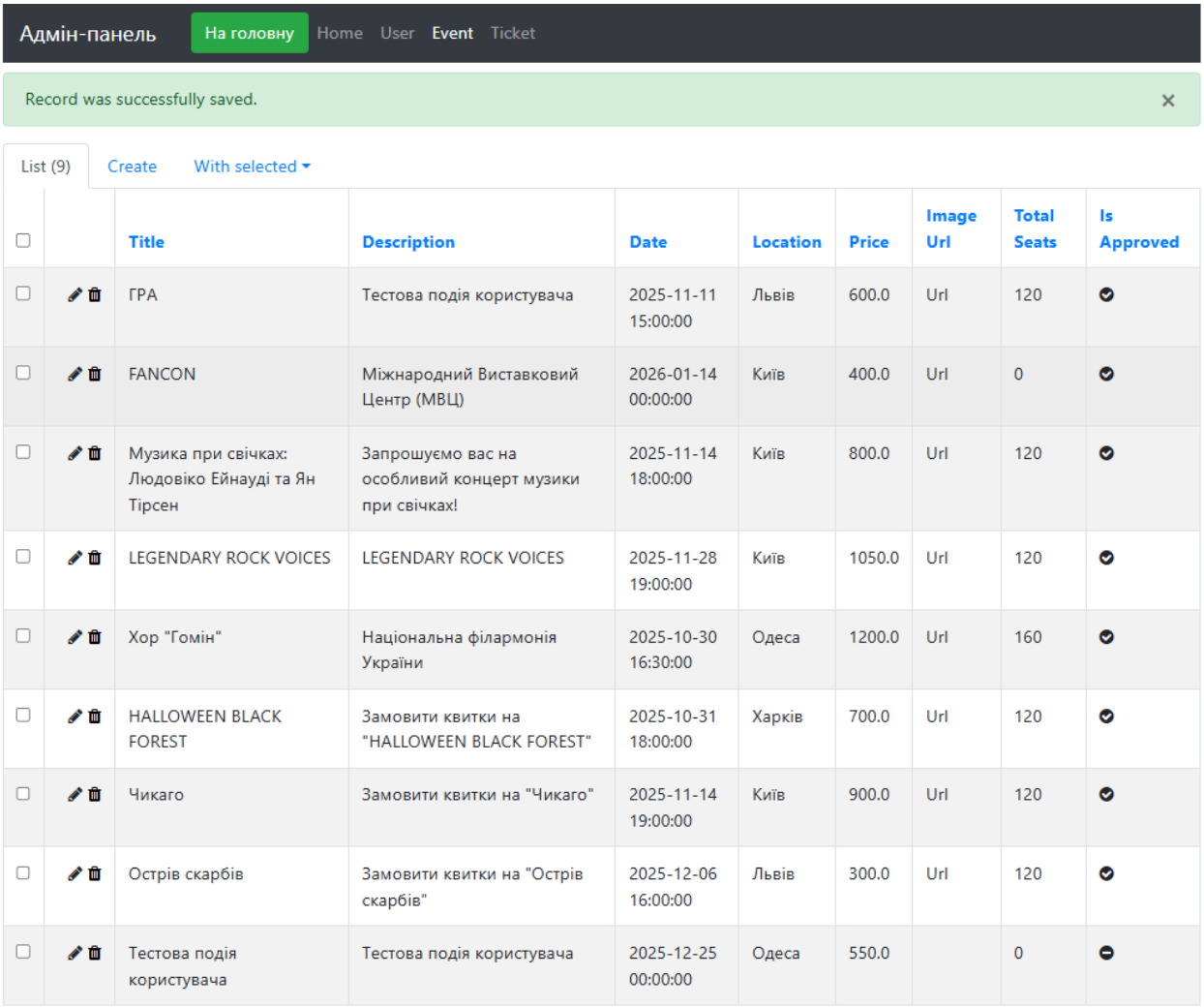


Рис. 3.28. Сторінка створення події користувачем

Після того як організатор створює нову подію через особистий кабінет, вона не одразу публікується на сайті. Спочатку інформація про подію надходить на модерацію адміністратору. На рисунку 3.29. можна побачити список всіх подій які присутні на сайті, а також події, які надіслані на модерацію та очікують публікацію. У неопублікованих подій в полі «is Approved» відображається знак «-».



		Title	Description	Date	Location	Price	Image Url	Total Seats	Is Approved
☐	✎ 🗑	ГРА	Тестова подія користувача	2025-11-11 15:00:00	Львів	600.0	Url	120	✔
☐	✎ 🗑	FANCON	Міжнародний Виставковий Центр (МВЦ)	2026-01-14 00:00:00	Київ	400.0	Url	0	✔
☐	✎ 🗑	Музика при свічках: Людовіко Ейнауді та Ян Тірсен	Запрошуємо вас на особливий концерт музики при свічках!	2025-11-14 18:00:00	Київ	800.0	Url	120	✔
☐	✎ 🗑	LEGENDARY ROCK VOICES	LEGENDARY ROCK VOICES	2025-11-28 19:00:00	Київ	1050.0	Url	120	✔
☐	✎ 🗑	Хор "Гомін"	Національна філармонія України	2025-10-30 16:30:00	Одеса	1200.0	Url	160	✔
☐	✎ 🗑	HALLOWEEN BLACK FOREST	Замовити квитки на "HALLOWEEN BLACK FOREST"	2025-10-31 18:00:00	Харків	700.0	Url	120	✔
☐	✎ 🗑	Чикаго	Замовити квитки на "Чикаго"	2025-11-14 19:00:00	Київ	900.0	Url	120	✔
☐	✎ 🗑	Острів скарбів	Замовити квитки на "Острів скарбів"	2025-12-06 16:00:00	Львів	300.0	Url	120	✔
☐	✎ 🗑	Тестова подія користувача	Тестова подія користувача	2025-12-25 00:00:00	Одеса	550.0		0	⛔

Рис. 3.29. Адміністративна панель (таблиця Event)

Адміністратор перевіряє надані дані (опис, назва, дата, місце проведення, ціну квитка) та за потреби має можливість відкоригувати інформацію перед публікацією. На рисунку 3.30. представлена панель, де адміністратор редагує та опубліковує подію.

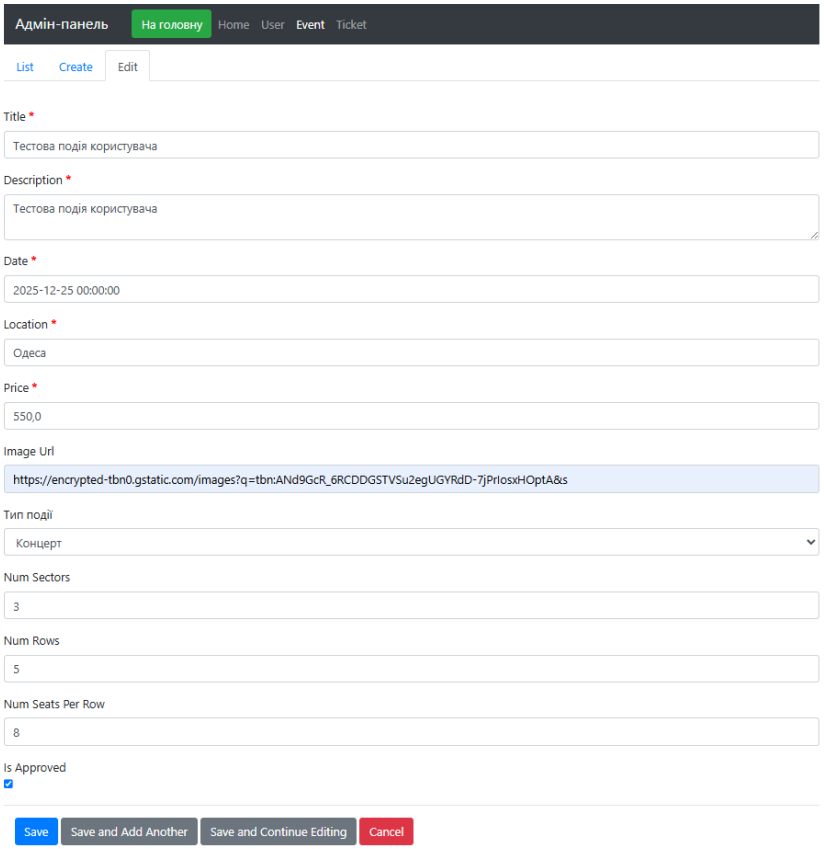


Рис. 3.30. Адміністративна панель (редагування події)

Після схвалення подія відображається на головній сторінці та стає доступною для купівлі квитків.

Окрім керування подіями, в адміністративній панелі також є можливість перегляду та редагування даних з таблиць User та Ticket, що зберігаються у базі даних додатку.

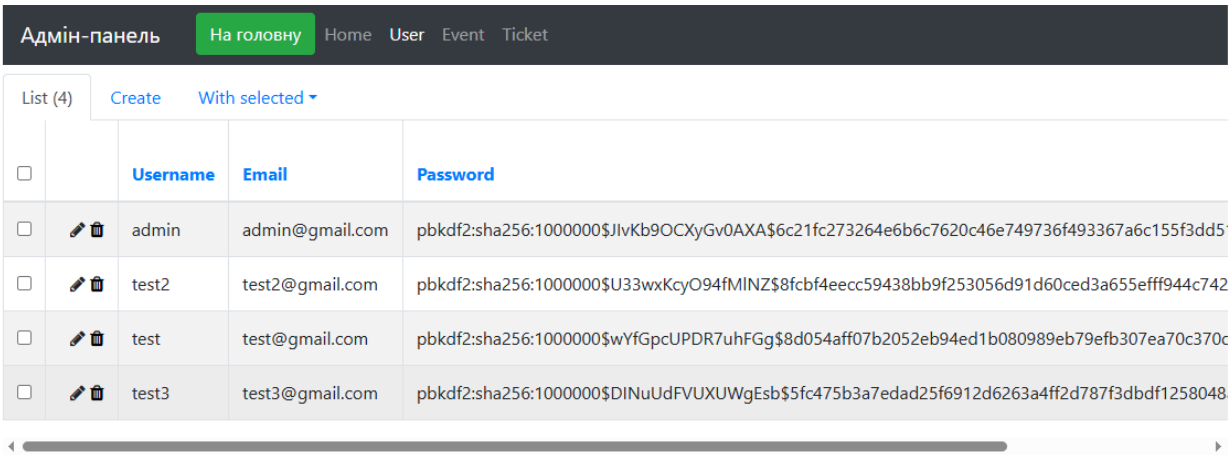
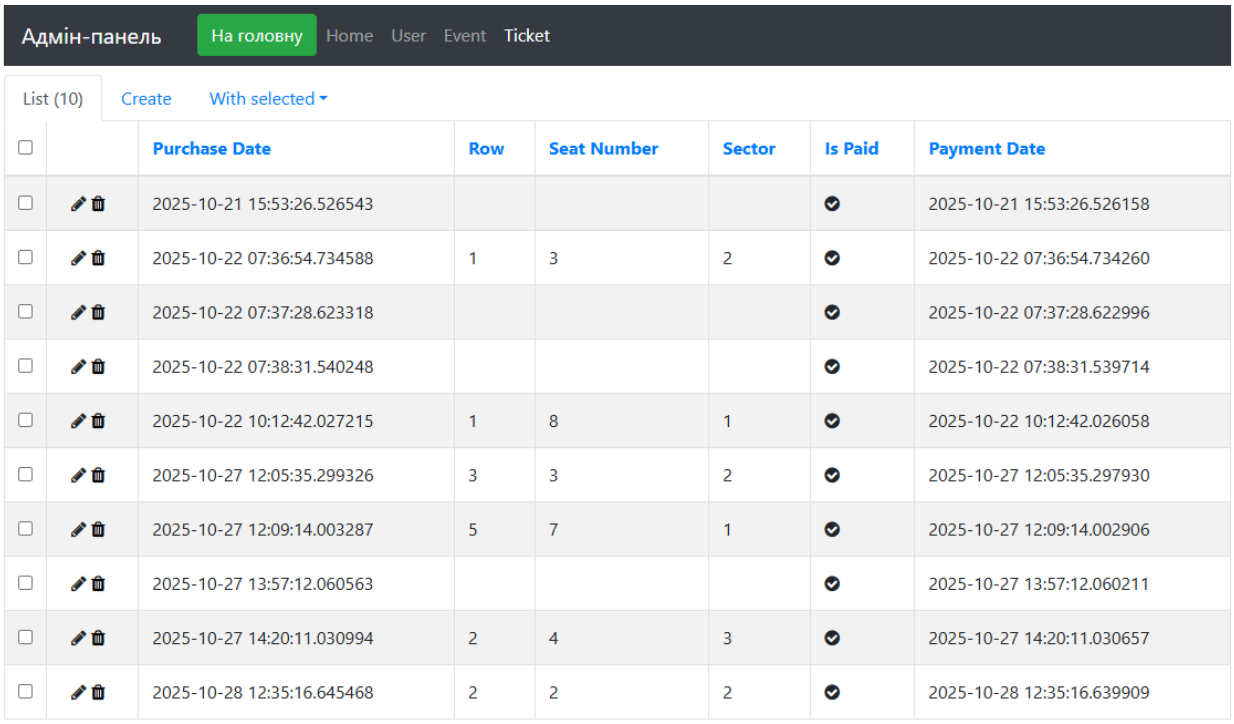


Рис. 3.31. Адміністративна панель (таблиця User)

Таблиця User (рис. 3.31.) містить інформацію про зареєстрованих

користувачів у системі. За потреби адміністратор може редагувати, створювати або видаляти облікові записи.



	Purchase Date	Row	Seat Number	Sector	Is Paid	Payment Date
	2025-10-21 15:53:26.526543				✓	2025-10-21 15:53:26.526158
	2025-10-22 07:36:54.734588	1	3	2	✓	2025-10-22 07:36:54.734260
	2025-10-22 07:37:28.623318				✓	2025-10-22 07:37:28.622996
	2025-10-22 07:38:31.540248				✓	2025-10-22 07:38:31.539714
	2025-10-22 10:12:42.027215	1	8	1	✓	2025-10-22 10:12:42.026058
	2025-10-27 12:05:35.299326	3	3	2	✓	2025-10-27 12:05:35.297930
	2025-10-27 12:09:14.003287	5	7	1	✓	2025-10-27 12:09:14.002906
	2025-10-27 13:57:12.060563				✓	2025-10-27 13:57:12.060211
	2025-10-27 14:20:11.030994	2	4	3	✓	2025-10-27 14:20:11.030657
	2025-10-28 12:35:16.645468	2	2	2	✓	2025-10-28 12:35:16.639909

Рис. 3.32. Адміністративна панель (таблиця Ticket)

Таблиця Ticket (рис. 3.32.) відображає всю інформацію про продані квитки, які також, за потреби, адміністратор може редагувати.

У межах тестування веб-додатку було проведено кросплатформне тестування, метою якого є перевірка коректного відображення та роботи інтерфейсу на різних пристроях.

Під час перевірки було протестовано роботу сайту на різних розмірах екранів. В результаті було підтверджено, що всі сторінки веб-додатку є адаптивними. Елементи інтерфейсу підлаштовуються під розмір вікна браузера.

На рисунку 3.33. представлена головна сторінка з відкритою панеллю навігаційних кнопок, які залишаються доступними та зрозумілими навіть на невеликих екранах.

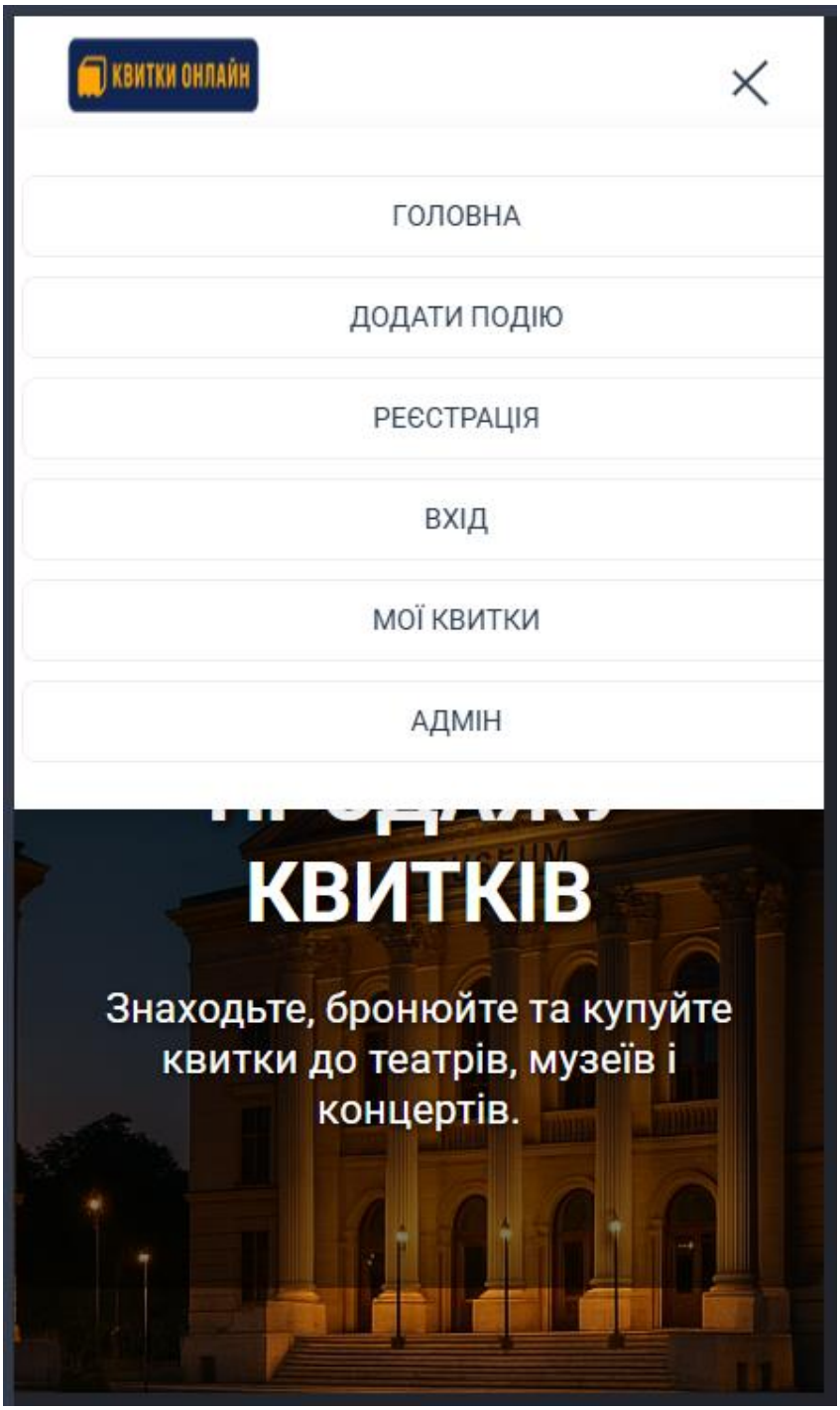


Рис. 3.33. Адаптивна головна сторінка з навігаційним меню

Ще один приклад адаптивної сторінки представлений на рисунку 3.34. На ньому відображена сторінка особистого кабінету.

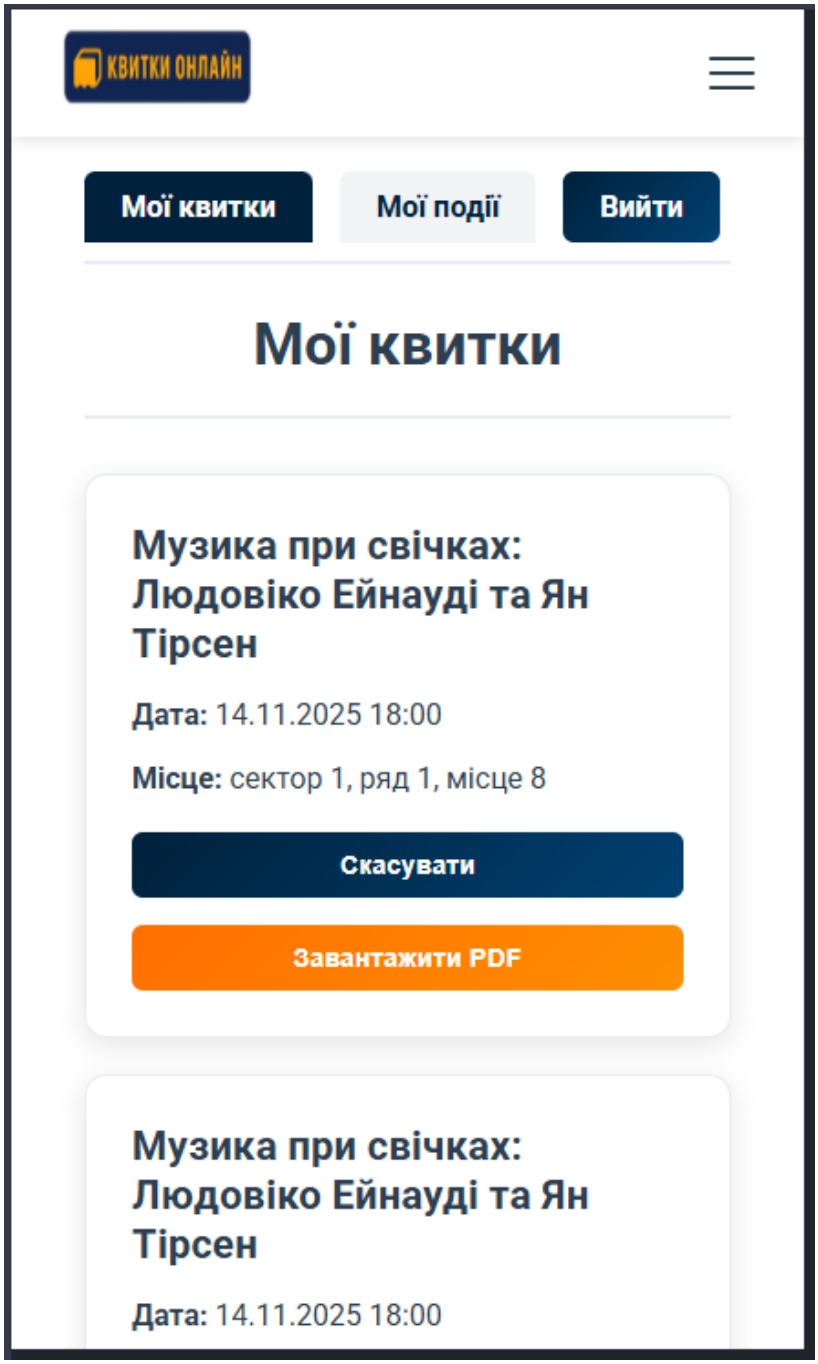


Рис. 3.34. Адаптивна сторінка особистого кабінету

Таким чином, результати тестування підтвердили, що розроблений веб-додаток є зручним, адаптивним та придатним для використання на різних пристроях.

Розроблений додаток продемонстрував працездатність основних функціональних модулів, інтеграцію з базою даних, коректною взаємодію між компонентами інтерфейсу та логіку серверної частини.

РЕЗУЛЬТАТИ І ВИСНОВКИ

Відповідно до поставлених завдань, у ході виконання випускної кваліфікаційної роботи було досягнуто таких результатів:

1. Проаналізовано теоретичні та практичні підходи до розробки алгоритмів для продажу квитків у культурній сфері. У результаті аналізу з'ясовано, що сучасні системи електронного продажу квитків орієнтовані на автоматизацію процесів вибору, бронювання та оплати квитків. Важливим аспектом є забезпечення зручної взаємодії з користувачем та підвищення ефективності процесу придбання. Розглянуто існуючі платформи електронного продажу квитків, визначено їхні переваги та недоліки. Для реалізації алгоритму було обрано підхід побудови серверної логіки з використанням мови програмування Python та фреймворку Flask, що надало гнучкість в архітектурі та ефективну роботу з даними.

2. Визначено особливості алгоритмів реалізації основних функцій процесу придбання квитка. Докладно описано алгоритм створення події, алгоритм вибору місця та алгоритм оформлення покупки. Сформовано діаграми, що відображають ключові етапи та логіку взаємодії користувача з системою.

3. Обґрунтовано вибір засобів програмної реалізації алгоритму. Для реалізації серверної частини застосовано фреймворк Flask, який забезпечує необхідну маршрутизацію та інтеграцію з базою даних SQLite через бібліотеку SQLAlchemy. Використання HTML та CSS дозволило зручно візуалізувати результати роботи алгоритмів у вигляді веб-інтерфейсу. Обраний технологічний стек гарантував простоту реалізації, надійність та можливість масштабування.

4. Проведено практичну реалізацію алгоритму для придбання квитків. Було реалізовано всі необхідні етапи — створення та модерацію подій, побудову схеми залу, вибір місця, бронювання та оформлення покупки. Алгоритм також включає генерацію PDF-квитка з унікальним QR-кодом, що забезпечує можливість швидкої перевірки квитка при вході на подію. Додатково реалізовано адміністративні механізми керування подіями та користувачами, необхідні для повноцінної роботи алгоритму у реальному середовищі.

5. Виконано тестування реалізованого алгоритму для підтвердження його коректності та надійності. Проведено функціональне тестування основних етапів: авторизації, вибору місця, оплати та генерації PDF-квитка. Перевірено роботу алгоритму в різних браузерах і на різних пристроях. Тестування показало, що алгоритм працює стабільно, коректно обробляє помилки та повністю виконує своє функціональне призначення.

Отже, розроблений алгоритм придбання квитків забезпечує автоматизацію ключових етапів вибору, бронювання та покупки квитка, підвищуючи ефективність та зручність цього процесу для користувачів та організаторів подій. Реалізований алгоритм може слугувати надійною основою для подальшого розвитку повноцінних систем електронного продажу квитків та впровадження їх у діяльність закладів культури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алі Аззаарі. Сучасні тенденції в дизайні веб-сайтів музеїв: особливості графічних елементів / А. Аззаарі // Актуальні проблеми сучасного дизайну. – Київ, КНУТД, 27 квітня 2022. – С. 219-222.

2. Ткаченко О. І., Тишура О. М. Деякі аспекти розробки веб-орієнтованої системи COFFEE++. IT SYNERGY. 2023. Вип. 2 (5). С. 115–122

3. Every Ticket. https://everyticket.in/blog/why-every-museum-needs-an-online-ticketing-solution-in-2025?utm_source=chatgpt.com

4. cm.com. https://www.cm.com/blog/10-reasons-online-ticketing-museum-park/?utm_source=chatgpt.com

5. Орел Л. М., Фененко А. О. «Особливості впровадження електронного обліку музейних фондів в практику». Харків: Матеріали конференції «Музей як соціокультурний інститут в умовах інформаційного суспільства», 2012. URL: https://museum.kh.ua/academic/publications.html?n=862&utm_source=chatgpt.com

6. veevart.com. https://www.veevart.com/why-more-museums-should-have-an-online-ticketing-application?utm_source=chatgpt.com

7. Мобільний додаток для продажу квитків на базі квиткової системи «PartyTicket». URL: https://avada-media.ua/services/mobile-app-for-ticket-sales/

8. mticket. https://mticket.com.ua/

9. Веб-технології. Їх різновиди та функції. https://sites.znu.edu.ua/webdevelopment/lect/1417.ukr.html

10. Мобільний застосунок. https://uk.wikipedia.org/wiki/Мобільний_застосунок

11. Ключові інструменти та технології Front-end, що використовуються для розробки веб-сайтів. https://webbookstudio.com/ua/articles/key-tools-and-technologies-used-in-front-end-website-development/

12. Трофименко О. Г. Веб-дизайн та HTML-програмування: навчально-методичний посібник / Трофименко О. Г., Козін О. Б. – Одеса, 2017. – 220 с.

13. Фреймворки у мовах програмування: навіщо служать та як їх вибрати. URL: https://cloud.itstep.org/blog/frameworks-in-programming-languages-what-are-

they-for-and-how-to-choose-them

14. Бутрин Л. Реляційна модель бази даних. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2021. С. 119-120.

15. Назаренко В. До питання визначення поняття «платіжна система» в Україні / В. Назаренко // Науковий журнал. — 2023. — № X. — С. 151-155.

16. Переваги використання платіжних систем. URL: https://detector.media/withoutsection/article/222827/2024-02-12-perevagy-vykorystannya-platizhnykh-system/

17. Що таке Content Management System (CMS). URL: https://smart-seo.com.ua/ua/shho-take-content-management-system-cms/

18. «Моделювання бізнес-процесів — це систематичний підхід до візуалізації, аналізу, а також вдосконалення роботи компанії» // EDILO: блог про бізнес-процеси, 2025. URL: https://edilo.com.ua/blog/modelyuvannya-ta-optymizacziya-biznes-proczesiv-na-pidpryyemstvi/?utm_source=chatgpt.com

19. Станкевич І. В. «Ідентифікація та моделювання бізнес-процесів системи управління якістю освітньої організації: теорія та практика». Проблеми економіки, № 1, 2017, с. 259-268.

20. Уніфікована мова моделювання (Unified Modeling Language - UML) https://www.maxzosim.com/unifikovana-mova-modeluvannia/

21. Мичуда Л. З. Web Applications Technological Stack технологічний стек роботи веб-додатків : навчальний посібник / Мичуда Л. З., Коробейнікова Т. І., Непийвода М. В., Рекало О. В. – Львів ; Вінниця, 2023. – 180 с.

22. Що таке Visual Studio Code https://top-keis.com.ua/shho-take-visual-studio-code/

23. Підручник з Python https://docs.python.org/uk/3/tutorial/index.html

24. Що таке Python і де він використовується. URL: https://dan-it.com.ua/uk/blog/python-chto-jeto-za-jazyk-programmirovanija-i-gde-ego-ispolzujut/

25. Легкість та гнучкість: основи Flask для веброзробки програмного забезпечення https://pnn.com.ua/ua/blog/detail/lightweight-and-flexible-flask-fundamentals-for-software-web-development

26. SQLite https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/

27. Радкевич О. Діаграми, графіки та схеми як інструментарій представлення проєктної інформації // Професійна педагогіка. — 2021. — № 1(22). — С. 197 212.

28. Розробка з боку Front end – що це таке і чим відрізняється від Back end? URL: https://dan-it.com.ua/uk/blog/rozrobka-z-boku-front-end-shho-ce-take-i-chim-vidriznjaietsja-vid-back-end/

29. Що таке Back-end розробка. URL: https://wezom.com.ua/ua/blog/chto-takoe-back-end-razrabotka

30. Основні відомості про бази даних. URL: https://support.microsoft.com/uk-ua/topic/%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%96-%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BE%D1%81%D1%82%D1%96-%D0%BF%D1%80%D0%BE-%D0%B1%D0%B0%D0%B7%D0%B8-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85-a849ac16-07c7-4a31-9948-3c8c94a7c204

31. Flask-Admin https://corp2.eu/p/docs/item/12bc8d329c6bd2cc3037921da3d28c99d

ДОДАТОК

models/event.py

```
from app.extensions import db

class Event(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=False)
    date = db.Column(db.DateTime, nullable=False)
    location = db.Column(db.String(100), nullable=False)
    price = db.Column(db.Float, nullable=False)
    image_url = db.Column(db.String(255))

    total_seats = db.Column(db.Integer, default=0)
    event_type = db.Column(db.String(50), nullable=False, default='концерт')

    num_sectors = db.Column(db.Integer, nullable=True, default=3)
    num_rows = db.Column(db.Integer, nullable=True, default=5)
    num_seats_per_row = db.Column(db.Integer, nullable=True, default=8)

    is_approved = db.Column(db.Boolean, nullable=False, default=False)
    submitted_by = db.Column(db.Integer, db.ForeignKey('user.id'))

    tickets = db.relationship('Ticket', back_populates='event', cascade='all, delete-orphan')

    def available_seats(self):
        return self.total_seats - len(self.tickets)

    def set_seats_defaults(self):
```

```
        if self.event_type.lower().strip() in ['концерт', 'театр']:
            self.num_sectors = self.num_sectors or 3
            self.num_rows = self.num_rows or 5
            self.num_seats_per_row = self.num_seats_per_row or 8
            self.total_seats = self.num_sectors * self.num_rows * self.num_seats_per_row
        else:
            self.num_sectors = None
            self.num_rows = None
            self.num_seats_per_row = None
            self.total_seats = 0
```

routes/events.py

```
from flask import Blueprint, render_template, redirect, url_for, flash, request
from flask_login import login_required, current_user
from app.models.event import Event
from app.models.ticket import Ticket
from app.extensions import db

events_bp = Blueprint('events', __name__)


# Сторінка деталей події
@events_bp.route('/event/<int:event_id>')
def show_event(event_id):
    event = Event.query.get_or_404(event_id)

    sold_tickets = Ticket.query.filter_by(event_id=event.id).count()

    available_seats = event.total_seats - sold_tickets
```

```
    return render_template('event_detail.html', event=event, available_seats=available_seats)


# Купівля квитка
@events_bp.route('/buy_ticket/<int:event_id>', methods=['POST'])
@login_required
def buy_ticket(event_id):
    event = Event.query.get_or_404(event_id)

    if event.event_type in ['музей', 'виставка']:
        return redirect(url_for('tickets.confirm_purchase', event_id=event.id))
    else:
        return redirect(url_for('event.select_seat', event_id=event.id))


# Скасування придбаних квитків
@events_bp.route('/cancel_ticket/<int:ticket_id>', methods=['POST'])
@login_required
def cancel_ticket(ticket_id):
    ticket = Ticket.query.get_or_404(ticket_id)

    if ticket.user_id != current_user.id:
        flash('Ви не можете скасувати цей квиток.', 'danger')
        return redirect(url_for('auth.dashboard'))

    db.session.delete(ticket)
    db.session.commit()

    flash('Квиток скасовано.', 'success')
```

```
    return redirect(url_for('auth.dashboard'))
```

admin.py

```
from flask import Blueprint, render_template, redirect, url_for, flash, request
from flask_admin import Admin, expose, AdminIndexView
from flask_admin.contrib.sqla import ModelView
from flask_login import login_required, current_user
from wtforms import SelectField
from wtforms.validators import Optional
from app.extensions import db
from app.models.user import User
from app.models.event import Event
from app.models.ticket import Ticket

admin_bp = Blueprint('admin_bp', __name__, url_prefix='/admin')

@admin_bp.route('/pending_events')
@login_required
def pending_events():
    if not current_user.is_admin:
        flash('Доступ лише для адміністратора', 'danger')
        return redirect(url_for('events_bp.index'))

    events = Event.query.filter_by(is_approved=False).all()
    return render_template('admin/pending_events.html', events=events)

@admin_bp.route('/approve/<int:event_id>')
@login_required
def approve_event(event_id):
    if not current_user.is_admin:
        flash('Доступ лише для адміністратора', 'danger')
```

```
        return redirect(url_for('events_bp.index'))

    event = Event.query.get_or_404(event_id)
    event.is_approved = True
    db.session.commit()
    flash('Подію підтверджено', 'success')
    return redirect(url_for('admin_bp.pending_events'))

@admin_bp.route('/reject/<int:event_id>')
@login_required
def reject_event(event_id):
    if not current_user.is_admin:
        flash('Доступ лише для адміністратора', 'danger')
        return redirect(url_for('events_bp.index'))

    event = Event.query.get_or_404(event_id)
    db.session.delete(event)
    db.session.commit()
    flash('Подію відхилено', 'info')
    return redirect(url_for('admin_bp.pending_events'))

#Адмін-панель

class MyAdminIndexView(AdminIndexView):
    def is_accessible(self):
        return current_user.is_authenticated and current_user.is_admin

    def inaccessible_callback(self, name, **kwargs):
        return redirect(url_for('auth.login', next=request.url))
```

```
class SecureModelView(ModelView):
    def is_accessible(self):
        return current_user.is_authenticated and current_user.is_admin

    def inaccessible_callback(self, name, **kwargs):
        return redirect(url_for('auth.login', next=request.url))

class EventAdminView(SecureModelView):
    form_overrides = {
        'event_type': SelectField
    }

    form_args = {
        'event_type': {
            'label': 'Тип події',
            'choices': [
                ('музей', 'Музей'),
                ('виставка', 'Виставка'),
                ('концерт', 'Концерт'),
                ('театр', 'Театр')
            ]
        },
        'num_sectors': {'validators': [Optional()]},
        'num_rows': {'validators': [Optional()]},
        'num_seats_per_row': {'validators': [Optional()]},
    }

    column_list = ('title', 'description', 'date', 'location', 'price', 'image_url', 'total_seats',
'is_approved')
    form_columns = (
```

```
        'title', 'description', 'date', 'location', 'price', 'image_url',
        'event_type', 'num_sectors', 'num_rows', 'num_seats_per_row',
        'is_approved'
    )

    def on_model_change(self, form, model, is_created):
        if model.event_type in ['концерт', 'театр']:
            model.total_seats = (model.num_sectors or 0) * (model.num_rows or 0) *
(model.num_seats_per_row or 0)
        else:
            model.total_seats = 0

#Ініціалізація адмінки

admin = Admin(
    name='Адмін-панель',
    template_mode='bootstrap4',
    index_view=MyAdminIndexView(),
    base_template='admin/custom_base.html'
)

def init_admin(app):
    admin.init_app(app)
    admin.add_view(SecureModelView(User, db.session))
    admin.add_view(EventAdminView(Event, db.session))
    admin.add_view(SecureModelView(Ticket, db.session))
```

templates/dashboard_statistics.html

```
{% extends "base.html" %}
```

```
{% block title %}Статистика — {{ event.title }}{% endblock %}

{% block content %}
<div class="statistics-container">
  <!-- Вкладки -->
  <div class="dashboard-tabs">
    <a href="{{ url_for('auth.dashboard') }}" class="dashboard-tab">Мої квитки</a>
    <a href="{{ url_for('auth.dashboard_events') }}" class="dashboard-tab">Мої події</a>
    <a href="{{ url_for('auth.logout') }}" class="dashboard-logout-btn">Вийти</a>
  </div>

  <div class="statistics-header">
    <h1 class="statistics-title">Статистика події: {{ event.title }}</h1>
    <a href="{{ url_for('auth.dashboard_events') }}" class="statistics-back-btn">← Назад до моїх подій</a>
  </div>

  <!-- Основна статистика -->
  <div class="statistics-overview">
    <div class="stat-card">
      <h3>Загальна статистика</h3>
      <div class="stat-value">{{ sold_tickets|length }}</div>
      <div class="stat-label">Продано квитків</div>
    </div>

    <div class="stat-card">
      <h3>Виручка</h3>
      <div class="stat-value">{{ "%.2f"|format(total_revenue) }} грн</div>
      <div class="stat-label">Загальний дохід</div>
```

```
    </div>

    <div class="stat-card">
        <h3>Заповненість</h3>
        <div class="stat-value">
            {% if event.total_seats and event.total_seats > 0 %}
            {% set occupancy = (sold_tickets|length / event.total_seats * 100) %}
            {{ "%.1f"|format(occupancy) }}%
            {% else %}
            —
            {% endif %}
        </div>
        <div class="stat-label">Відсоток заповненості</div>
    </div>

    <div class="stat-card">
        <h3>Вільні місця</h3>
        <div class="stat-value">
            {% if event.total_seats %}
            {{ event.available_seats() }}
            {% else %}
            —
            {% endif %}
        </div>
        <div class="stat-label">Залишилось місць</div>
    </div>
</div>

<!-- Детальна статистика -->
<div class="statistics-details">
```

```
<!-- Продажі по днях -->
<div class="statistics-section">
    <h3>Продажі по днях</h3>
    <div class="sales-chart">
        {% for day, count in sales_by_day.items() %}
        <div class="chart-bar">
            <div class="bar-label">{{ day }}</div>
            <div class="bar-container">
                {% if sold_tickets|length > 0 %}
                {% set percentage = (count / sold_tickets|length) * 100 %}
                <div class="bar-fill" data-width="{{ percentage }}"></div>
                {% else %}
                <div class="bar-fill" style="width: 0%"></div>
                {% endif %}
                <span class="bar-value">{{ count }} кв. ({{ "%.2f"|format(revenue_by_day[day]) }} грн)</span>
            </div>
        </div>
        {% endfor %}
    </div>
</div>

<!-- Статистика по секторах -->
{% if sector_stats %}
<div class="statistics-section">
    <h3>Розподіл по секторах</h3>
    <div class="sector-stats">
        {% for sector, count in sector_stats.items() %}
        <div class="sector-item">
            <span class="sector-name">{{ sector }}</span>
```

```
                <span class="sector-count">{{ count }} квитків</span>
            </div>
            {% endfor %}
        </div>
    </div>
    {% endif %}

    <!-- Список проданих квитків -->
    <div class="statistics-section">
        <h3>Детальний звіт по квитках</h3>
        <div class="tickets-list">
            {% for ticket in sold_tickets %}
            <div class="ticket-item">
                <div class="ticket-info">
                    <strong>Квиток #{{ ticket.id }}</strong>
                    <span>Куплено: {{ ticket.purchase_date.strftime('%d.%m.%Y %H:%M') }}</span>
                    {% if ticket.sector %}
                    <span>Місце: Сектор {{ ticket.sector }}, Ряд {{ ticket.row }}, Місце {{ ticket.seat_number
                        }}</span>
                    {% endif %}
                </div>
                <div class="ticket-price">{{ event.price }} грн</div>
            </div>
            {% endfor %}
        </div>
    </div>
  </div>
</div>
```

```
<script>
    document.addEventListener('DOMContentLoaded', function () {
        document.querySelectorAll('.bar-fill').forEach(function (bar) {
            var width = bar.getAttribute('data-width');
            bar.style.width = width + '%';
        });
    });
</script>
{% endblock %}
```