

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОДАЖУ КВИТКІВ
НА ЗАХОДИ В СФЕРІ КУЛЬТУРИ
SOFTWARE FOR SELLING THE TICKETS FOR CULTURAL EVENTS

Виконав: студент(ка) 4 курсу, групи КНТ-218сп
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

Чорний Д.В.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 122 Комп'ютерні науки
 (код і найменування)
 Освітня програма (спеціалізація) Комп'ютерні науки
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Чорного Дениса Вікторовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення продажу квитків на заходи в сфері культури

Software for Selling the Tickets for Cultural Events

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) травень 2021 року

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

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

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Андрєєв М.О., асистент		

7. Дата видачі завдання “9” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис)

Чорний Д.В.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Скрупський С.Ю.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 82 с., 27 рис., 17 табл., 3 дод., 14 джерел.

КУЛЬТУРНА ПОДІЯ, КОНЦЕРТ, КВИТКОВИЙ ОПЕРАТОР, ЗАМОВЛЕННЯ КВИТКІВ, ВЕБЗАСТОСУНОК, DJANGO.

Об'єкт роботи – процес продажу квитків на заходи в сфері культури.

Предмет роботи – програмне забезпечення продажу квитків на заходи в сфері культури.

Мета роботи – розроблення програмного забезпечення продажу квитків на заходи в сфері культури для підтримки роботи квиткового оператора.

Матеріали, методи та технічні засоби: мова програмування Python, вебфреймворк Django, система керування базами даних MySQL.

Результати. Проведено аналіз предметної області з виділенням вимог та функціональних можливостей програмного забезпечення продажу квитків. Виконано проєктування програмного забезпечення з моделюванням варіантів використання системи та визначенням структури даних. Реалізовано вебзастосунок.

Висновки. Реалізована програма призначена для організації платформи для операторів з продажу, організаторів подій, розповсюджувачів квитків на ці події та відвідувачів подій. Організатори подій надають контент для внесення даних у програму. Розповсюджувачі квитків можуть реалізовувати безпосередньо свій бізнес через програму. Відвідувачі подій можуть оформляти та керувати замовленнями на покупку квитків.

Галузь використання – підтримка роботи операторів з продажу квитків на заходи в сфері культури, підтримка продажу квитків для окремих організаторів подій.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 27 figures, 17 tables, 3 appendixes, 14 sources.

CULTURAL EVENT, CONCERT, TICKET OPERATOR, TICKET ORDER, WEB APPLICATION, DJANGO.

The object of work is the process of selling tickets for cultural events.

The subject of work is software for the sale of tickets for events in the field of culture.

The purpose of the work is to develop software for ticket sales for cultural events to support the work of the ticket operator.

Materials, methods and technical means: Python programming language, Django web framework, MySQL database management system.

Results. The analysis of the subject area with the elicitation of requirements and functionality of the software for ticket sales. Software design with modeling of system usage options and data structure definition has been performed. Web application was implemented.

Conclusions. The implemented program is designed to organize a platform for sales operators. event organizers, distributors of tickets for these events and event visitors. Event organizers provide content for entering data into the program. Ticket distributors can perform their business directly through the program. Event visitors can make and manage ticket orders.

Field of usage is support for ticket sales operators for cultural events, support for ticket sales for individual event organizers.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Проблема створення програмного забезпечення продажу квитків на заходи в сфері культури.....	9
1.2 Огляд програмних аналогів.....	11
1.3 Висновки за розділом 1	17
2 Розробка архітектури програми.....	18
2.1 Вибір технологій розробки	18
2.2 Вимоги до програмного забезпечення	20
2.3 Проектування структури даних	23
2.4 Висновки за розділом 2	33
3 Розробка програми	34
3.1 Основна логіка виконання програми	34
3.2 Опис реалізації програми	35
3.3 Висновки за розділом 3	40
4 Експлуатація та тестування програми	41
4.1 Призначення програми	41
4.2 Умови виконання програми	41
4.3 Робота з програмою	41
Висновки	53
Перелік джерел посилання	54
Додаток А Технічне завдання	56
Додаток Б Текст програми	61
Додаток В Слайди презентації.....	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

QR	– Quick Response;
SEO	– search engine optimization;
БД	– база даних.

ВСТУП

Сфера культури в даний момент переживає кризу свого розвитку, адже культурні заходи в Україні та світі майже припинилися в звичному вигляді через пандемію. Проте цей момент кризи має призвести до бурхливого розвитку після завершення епідемії і зняття заборон.

Чим більш економічно розвинутим є місто, тим більше культурних подій відбувається у ньому. Раніше кожен організатор самостійно продавав квитки на свій захід. Тепер дана відповідальність частіше за все перенесена на спеціальні сервіси – квиткових операторів. Саме вони мають в своєму розпорядженні квитки на різні події і частіше за все продають їх через інтернет або використовують відповідне програмне забезпечення для підтримки своєї діяльності.

В Україні існує ряд квиткових операторів, але постійно виникають нові. Частина з них займається продажем квитків у певних регіонах. Саме на підтримку діяльності таких компаній і направлено створення програмного забезпечення, що виконується в роботі. Окрім того розроблюване програмне забезпечення може підтримувати роботу окремих організаторів подій, які хочуть самостійного контролю за цим процесом. В Україні за історію її незалежності виникало немало історій, пов'язаних з непорозуміннями між квитковими операторами і організаторами заходів або артистами. Тому місце для створення нових систем такого типу для підтримки діяльності сторони, у якої така проблема виникає, постійно існує.

Дану роботу направлено на розроблення програмного забезпечення продажу квитків на заходи в сфері культури для підтримки роботи квиткового оператора, що є метою роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблема створення програмного забезпечення продажу квитків на заходи в сфері культури

З процесом придбання квитків на культурні заходи безпосередньо пов'язані сайти для пошуку подій та платформи з розповсюдження або продажу квитків.

Наявність системи продажу квитків відзначається наступними позитивними характеристиками:

- взаємодія з широкою аудиторією: залучення організаторів подій до платформи дозволяє не витрачати час безпосередньо на організацію заходів, потрібно концентруватися на створенні гнучкої екосистеми для взаємодії, відповідно планування подій, пошук артистів, генерація ідей для залучення нової аудиторії лежить не в межах відповідальності застосунку;

- платформа продажу квитків надає інструменти для реєстрації, просування подій, це означає, що компаніям, що займаються подіями, не доведеться наймати менеджера заходів, оскільки платформа може робити все сама, заощаджуючи до 24% маркетингового бюджету [1].

Системи продажу квитків на культурні заходи повинні відповідати наступним вимогам:

- заощадження часу: однією з основних причин, за якими клієнти вирішують робити бронювання через Інтернет, є економія часу та енергії, важливо, щоб реєстрація та керування обліковими записами були простими як для менеджерів, так і для аудиторії, це також означає, що вебсайт і сервери повинні бути швидкими з мінімальними простоями, помилки потрібно виправляти та усувати їх пріоритетно, щоб уникнути втрати потенційних клієнтів;

- проста сплата: інтернет-платежі стабільно збільшуються, зараз онлайн гаманці використовуються для здійснення страхових платежів, поповнення мобільних платежів, навіть для громадського транспорту,

платформа повинна мати необхідні функції для негайної обробки платежів або для відхилення запитів за потреби;

– у випадку, якщо повернення коштів потрібно підтвердити або відмовити у цьому, платформа повинна мати необхідні інструменти, щоб зробити це безперебійним процесом з мінімальною кількістю або відсутністю подальших дій, необхідних клієнту;

– функція картографування місць дозволяє людям вибрати та знати, в якій зоні вони бронюють місця, – ця функція також допомагає організаторам заходів планувати зони ефективніше;

– описи майбутніх подій, а також інструменти аудіо-візуального маркетингу допоможуть залучити більше клієнтів, фірмові квитки з унікальними Quick Response (QR)-кодами роблять процес купівлі квитків зручним;

– одним з найважливіших факторів успішної платформи є знання клієнта та забезпечення постійних клієнтів, це можливо лише в тому випадку, якщо вони мають необхідні інструменти для створення належної бази даних (БД), а всі взаємодії відстежуються та контролюються;

– скасування події: до такого сценарію також слід завжди бути готовим, повинен бути план, що захищає від відмов, де користувачі можуть негайно отримувати сповіщення про такі події [2].

Система продажу квитків на культурні заходи повинна забезпечувати виконання наступних функціональних можливостей:

а) організація заходів: організатори подій повинні легко реєструвати свої події та керувати ними, при цьому потрібно мати можливість встановити:

- 1) назву події;
- 2) місцезнаходження;
- 3) дату та час;
- 4) тип події;
- 5) категорію події;

б) якісний опис події, що зробить подію привабливою для потенційних клієнтів: щоб перевірити, чи правильно було створено та описано подію, потрібно надати функції редагування, попереднього перегляду та видалення, а також потрібно додати список усіх створених для належного управління ними;

б) управління квитками: хорошою особливістю для організаторів подій є створення власних квитків або електронних квитків з QR-кодами для зручності реєстрації, після створення та брендуння квитків організаторам подій потрібно буде отримати деякі деталі від клієнтів, використовуючи форму замовлення, яка включає:

1) назву події та подробиці;

2) інструкції щодо реєстрації: поля для вставки контактних даних покупця, таких як повне ім'я, електронна адреса та будь-які інші дані, які згодом можна використовувати в маркетингових цілях;

в) організаторам подій зручним буде інструмент для управління подіями: це може бути список усіх замовлень та гостей, можливість перевірити, чи розпродані квитки, та можливість перевірити, чи присутні всі гості;

г) пошук подій: люди можуть шукати конкретну подію і не хочуть витрачати свій час на прокручування довгого списку, тому потрібно розділяти події за типами (концерти, фільми та будь-які інші типи заходів, які проводяться), фільтри будуть гарною ідеєю для пошуку, щоб дозволити користувачам вибирати події за типом, виконавцем, ціною, місцем розташування, датою та часом [1].

1.2 Огляд програмних аналогів

Найпопулярнішим квитковим оператором у світі є Ticketmaster [3], тому першим розглянутим програмним аналогом є його вебсайт (рис. 1.1) [3].

The screenshot shows the Ticketmaster website interface. At the top, there's a navigation bar with flags for DE and EN, and links for Newsletter, Musicals, Gift Cards, VIP Tickets, and FAQ. Below this is a blue header with the Ticketmaster logo and categories: Music, Sports, Arts & Theatre, Family, and More. A search bar is present with the text 'Search by Artist, Event or Venue' and a 'Sign In/Register' link. The main content area is titled 'All Arts & Theatre' and shows '11643 Upcoming Events'. A date filter is set to '20.06.2021'. Under the 'Germany' section, there are 5076 events listed. The list includes details for specific events like 'Tom Gaebel & His Orchestra - Verlegt auf die Seebühne Bremen' and 'TANZ DER VAMPIRE - Das Musical in Stuttgart', with options to 'Find tickets' or 'More info'. A sidebar on the right contains an 'Advertisement' placeholder and a 'SUB CATEGORIES' section with links to 'All Arts & Theatre', 'Ballet & Dance', 'Cinema', 'Magic', and a 'See More' link.

Рисунок 1.1 – Перегляд списку подій в Ticketmaster

Вебзастосунок Ticketmaster надає можливість пошуку подій за кодом або містом, за категорією дат (всі дати, найближчі вихідні або заданий проміжок дат), за іменем артиста або назвою події. Після виконання пошуку користувачу надається список подій у випадку пошуку подій або сторінка артиста з доступом до переліку подій, які відповідають цьому артисту. За артистом також можна переглянути його біографію, відгуки. Користувач може обрати потрібну подію та перейти до купівлі квитків (рис. 1.2) [3], натиснувши на відповідну кнопку.

За кожною подією можна розгорнути інформацію, переглянувши назву артиста та місце проведення. Звідси також можна перейти до місця проведення події, за яким переглянути всі наявні події, тобто розклад подій, інформацію про залу, зокрема схему розташування місць у ній.

The screenshot displays the Ticketmaster interface for a concert by The Weeknd at Rogers Arena, Vancouver, BC, on Friday, January 14, 2022, at 7:00 PM. The main area shows a circular seating chart with various sections labeled (e.g., 319, 318, 317, 316, 315, 314, 313, 312, 311, 310, 309, 308, 307, 306, 305, 304, 303, 302, 301, 300, 299, 298, 297, 296, 295, 294, 293, 292, 291, 290, 289, 288, 287, 286, 285, 284, 283, 282, 281, 280, 279, 278, 277, 276, 275, 274, 273, 272, 271, 270, 269, 268, 267, 266, 265, 264, 263, 262, 261, 260, 259, 258, 257, 256, 255, 254, 253, 252, 251, 250, 249, 248, 247, 246, 245, 244, 243, 242, 241, 240, 239, 238, 237, 236, 235, 234, 233, 232, 231, 230, 229, 228, 227, 226, 225, 224, 223, 222, 221, 220, 219, 218, 217, 216, 215, 214, 213, 212, 211, 210, 209, 208, 207, 206, 205, 204, 203, 202, 201, 200, 199, 198, 197, 196, 195, 194, 193, 192, 191, 190, 189, 188, 187, 186, 185, 184, 183, 182, 181, 180, 179, 178, 177, 176, 175, 174, 173, 172, 171, 170, 169, 168, 167, 166, 165, 164, 163, 162, 161, 160, 159, 158, 157, 156, 155, 154, 153, 152, 151, 150, 149, 148, 147, 146, 145, 144, 143, 142, 141, 140, 139, 138, 137, 136, 135, 134, 133, 132, 131, 130, 129, 128, 127, 126, 125, 124, 123, 122, 121, 120, 119, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0). The chart is divided into sections: SOUTH, EAST, NORTH, and MIX. A legend indicates the status of seats: Available (blue dot), Verified Resale (pink dot), Locked Offer (blue dot with lock), VIP (orange star), Accessibility (blue dot with wheelchair), and Unavailable (grey dot). The sidebar on the right shows the number of tickets (2), the price range (CA \$70 to CA \$392+), and a list of available seats with their prices and details. The list includes VIP Packages, Sec 319, Row 10, Sec 313, Row 11, Sec 313, Row 12, Sec 313, Row 14, and Sec 318, Row 11.

Рисунок 1.2 – Вибір місця в Ticketmaster

Під час вибору місця подій використовується два підхода:

- вибір місця за категорією;
- вибір конкретного місця за сіткою залу.

У випадку, представленому на рис. 1.1, користувачу відображається спочатку загальна схема з вибором категорії місця, а далі, після вибору категорії, розташування місць у цій зоні. Таким чином користувач може обрати конкретне місце. Доступний також вибір з переліку зі всіх доступних варіантів. Окрім того можна задати проміжок вартості, в який має входити квиток. Далі можна обрати потрібну кількість місць (зокрема якщо продаж продається за категорією квитків). Оформлення замовлення можливе тільки після авторизації в сервісі.

Усі події розділені на наступні категорії:

- концерти;

- спортивні заходи;
- мистецтво та театр;
- сімейні заходи.

Один з найвідоміших квиткових операторів в Україні – concert.ua [4]
(рис. 1.3) [4].

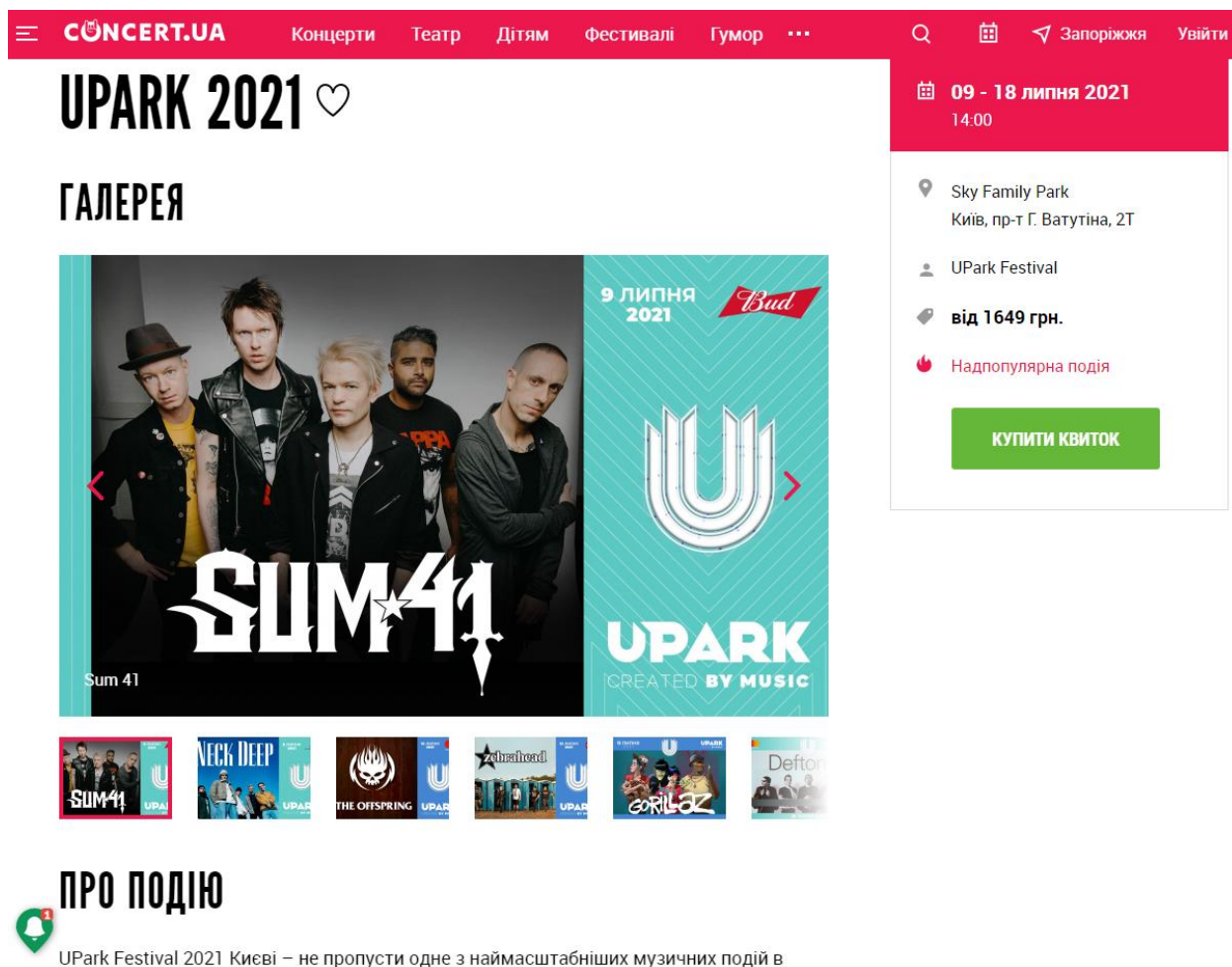


Рисунок 1.3 – Перегляд інформації про подію на вебсайті concert.ua

Усі події розділені на велику кількість категорій:

- концерти;
- театр;
- дитячі заходи;
- фестивалі;
- гумористичні події;
- танці;

- вечірки;
- екскурсії;
- семінари і тренінги;
- цирк.

Про кожну подію надається інформація, можна переглядати також інформацію за місцями проведення, звідки отримувати перелік доступних подій.

Надається можливість шукати події за календарем, але календар обмежений категоріями: сьогодні, завтра, на цьому тижні, в ці вихідні та у цьому місяці.

Під час замовлення надається можливість вибору місця за категорією або конкретна схема з визначенням позиції (рис. 1.4) [4].

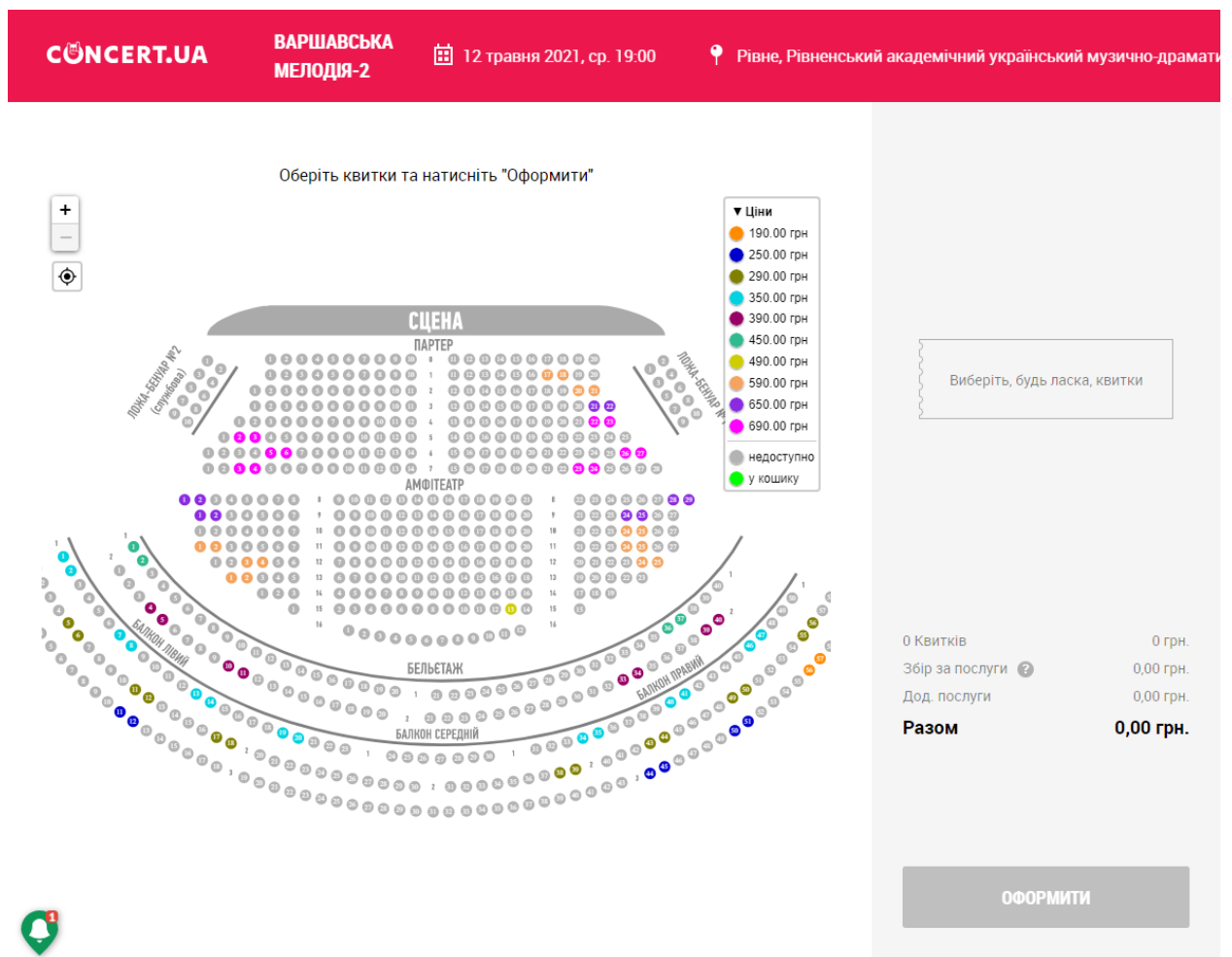


Рисунок 1.4 – Вибір місця при замовленні квитків у concert.ua

Ще одним оператором в Україні є kasa.in.ua [5] (рис. 1.5) [5].

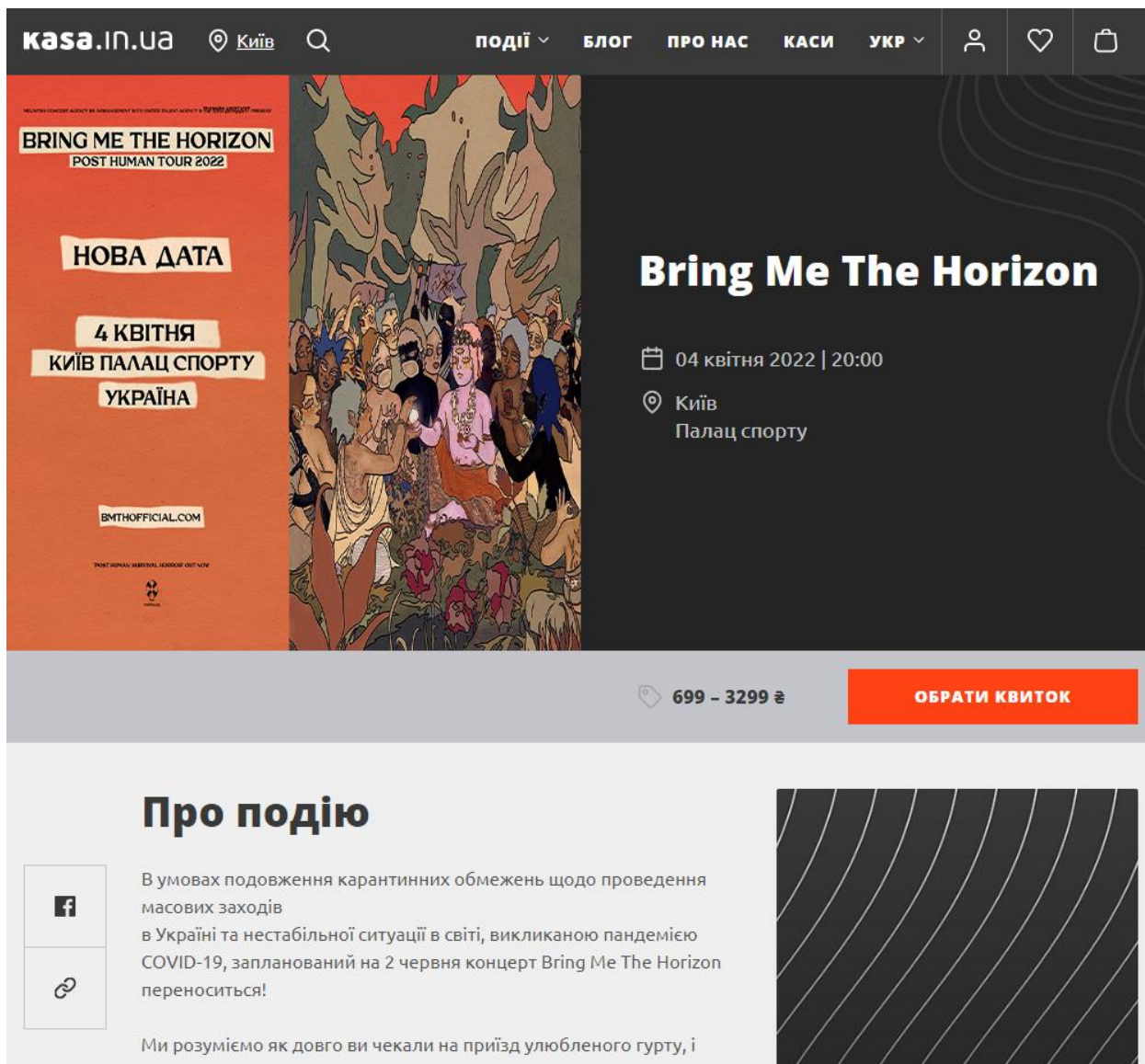


Рисунок 1.5 – Перегляд інформації про подію на вебсайті kasa.in.ua

Вебсайт сервісу розділяє всі події на категорії:

- концерти;
- фестивалі;
- театр;
- усі інші.

Вебзастосунок дозволяє переглядати інформацію про місце проведення, при цьому на місце проведення не відображається перелік подій, тільки загальна інформація.

Пошук квитків реалізується за артистом, місцем проведення та назвою події. Можна також обрати місто проведення заходу, після чого буде відображено перелік подій. У розширених налаштуваннях є можливість пошуку за ширшою кількістю категорій, за датами проведення, а також категоріями цих дат.

1.3 Висновки за розділом 1

Виконано аналіз проблеми створення програмного забезпечення продажу квитків на заходи в сфері культури, визначено позитивні характеристики таких систем відносно роботи над заходами в сфері культури, визначено вимоги до програми та розглянуто основні функціональні можливості таких систем.

Розглянуто основні функціональні можливості, підходи до категоризації подій програмних аналогів, що включають Ticketmaster, concert.ua, kasa.in.ua.

У ході подальшого виконання роботи необхідно розв'язати наступні завдання:

- визначення вимог до програми;
- проєктування програми;
- реалізація програми.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір технологій розробки

Для розробки програмного забезпечення використано мову програмування Python [6] через необхідність розробки вебзастосунку, власний досвід програмування саме цією мовою, а також через її гнучкі можливості для різних галузей розроблення програмного забезпечення.

Для Python існує два популярних вебфреймворка: Django [7] та Flask [8]. Їх було порівняно і представлено результати в табл. 2.1.

Для розробки було використано вебфреймворк Django через його переваги:

- докладна і якісна документація: якщо порівнювати Django з іншими технологіями з відкритим кодом, Django пропонує найкращу документацію на ринку, при чому підтримуються різні мови, не обов'язково потрібне довершене володіння англійською мовою;

- вебфреймворк Python: Python є однією з основних причин, чому Django привертає увагу серед усіх наявних на ринку вебфреймворків, Python може вирішити проблеми абсолютно різних напрямків, мова проста у використанні;

- search engine optimization (SEO)-оптимізованість: SEO – це оптимізація пошукової системи, що за назвою означає додавання вебсайту до пошукової системи таким чином, що він відображається у найкращих результатах, це потребує застосування особливостей використання алгоритмів пошукових систем, Django підтримує вебсайт за допомогою URL-адрес, а не IP-адрес на сервері, що полегшує для SEO-інженерів додавання вебсайту на сервер, тоді як веброзробник не повинен перетворювати URL-адресу в якийсь числовий код;

- висока масштабованість: масштабованість означає, на якому рівні може впроваджуватися технологія, використанню технології для масштабних

проектів сприяє висока відлагодженість коду самого Django, який є результатом досвіду великої кількості розробників;

- пропонує високий рівень безпеки: Django надзвичайно безпечний, за замовчуванням охоплює лазівки, які колись залишались відкритими для розробки бекенда;

- забезпечує швидку розробку: швидка розробка означає, що не знадобляться експертні знання, щоб створити повністю функціональний вебсайт, не потрібно створювати окремі файли сервера для проєктування бази даних (БД) та підключення, Django займається цією роботою та багатьма іншими завданнями, що можуть охоплювати половину часу проєкту і вони важливі для майже будь-якого вебсайту [9]-[10].

Таблиця 2.1 – Порівняння фреймворків розробки вебзастосунків мовою програмування Python

Показник	Django	Flask
Мова програмування	Python	Python
Вбудоване об'єктно-реляційне відображення	+	—
Вбудована адміністративна панель	+	—
Інтеграція форм	+	—
Вбудовані засоби безпеки	+	+—
Створення динамічних вебсторінок	+	+

2.2 Вимоги до програмного забезпечення

Основними вимогами до програмного забезпечення було визначено:

- можливість визначення схеми залу за мапою для продажу квитків за категоріями;
- можливість визначення сіткової схеми залу з завданням конкретних місць;
- підтримка роботи неавторизованого користувача;
- продаж квитків на декілька подій в одному замовленні;
- гнучке визначення категорій вартості квитків при продажі квитків на конкретні місця;
- можливість повного редагування замовлення з обмеженим доступом;
- можливість пошуку замовлень за номером телефону замовника.

Задля підтримки роботи квиткового оператора необхідно реалізувати роботу зі сторони квиткового оператора та зі сторони звичайних користувачів, які мають виконувати замовлення на покупку квитків. Звичайні користувачі будуть представлятися в ролі гостей та відвідувачів. Гостями є неавторизовані користувачі, а відвідувачами – авторизовані. Зі сторони квиткового оператора будуть працювати одразу дві групи користувачів – це адміністратори та менеджери. Адміністратори мають право вносити інформацію загального рівня в програму, тобто керувати даними, що зберігаються в БД. Менеджери мають доступ тільки для замовлень гостей: їх задача полягає в тому, щоб зафіксувати сам факт продажу квитків, якщо продаж відбувається через них.

Для забезпечення роботи всіх трьох ролей користувачів з програмою необхідно реалізувати наступні функціональні можливості:

- створення культурної події;
- внесення схеми залу на основі зон карти з продажем місць за категорією;

- внесення схеми залу з продажем квитків за місцем;
- визначення категорій вартості квитків на культурну подію;
- зонування зали за категоріями;
- пошук культурних подій у заданому місті;
- пошук культурних подій на вказану дату;
- пошук культурних подій за артистом;
- пошук культурних подій за назвою;
- оформлення замовлення на придбання квитків;
- перегляд власних замовлень;
- перегляд інформації про замовлення за номером;
- перегляд інформації про замовлення за номером телефону;
- визначення сплати за замовлення;
- перегляд культурних подій за категорією;
- перегляд культурних подій за місцем проведення (за залом);
- редагування схеми залу;
- редагування вартості квитків;
- редагування інформації про подію;
- визначення доступних місць на подію;
- автоматичне скасування замовлення у разі несплати;
- перегляд інформації про культурну подію;
- реєстрація акаунта;
- авторизація акаунта;
- перегляд інформації про залу;
- перегляд інформації про артиста;
- перегляд інформації про організатора;
- перегляд інформації про офіс для продажу квитків;
- оформлення запиту на повернення коштів;
- скасування подій;
- внесення і редагування даних про артиста;
- внесення і редагування даних про зали;

- внесення і редагування даних про організаторів;
- внесення і редагування даних про офіси з продажу квитків.

Сценарії використання програми користувачами з різними ролями представлено на рис. 2.1.

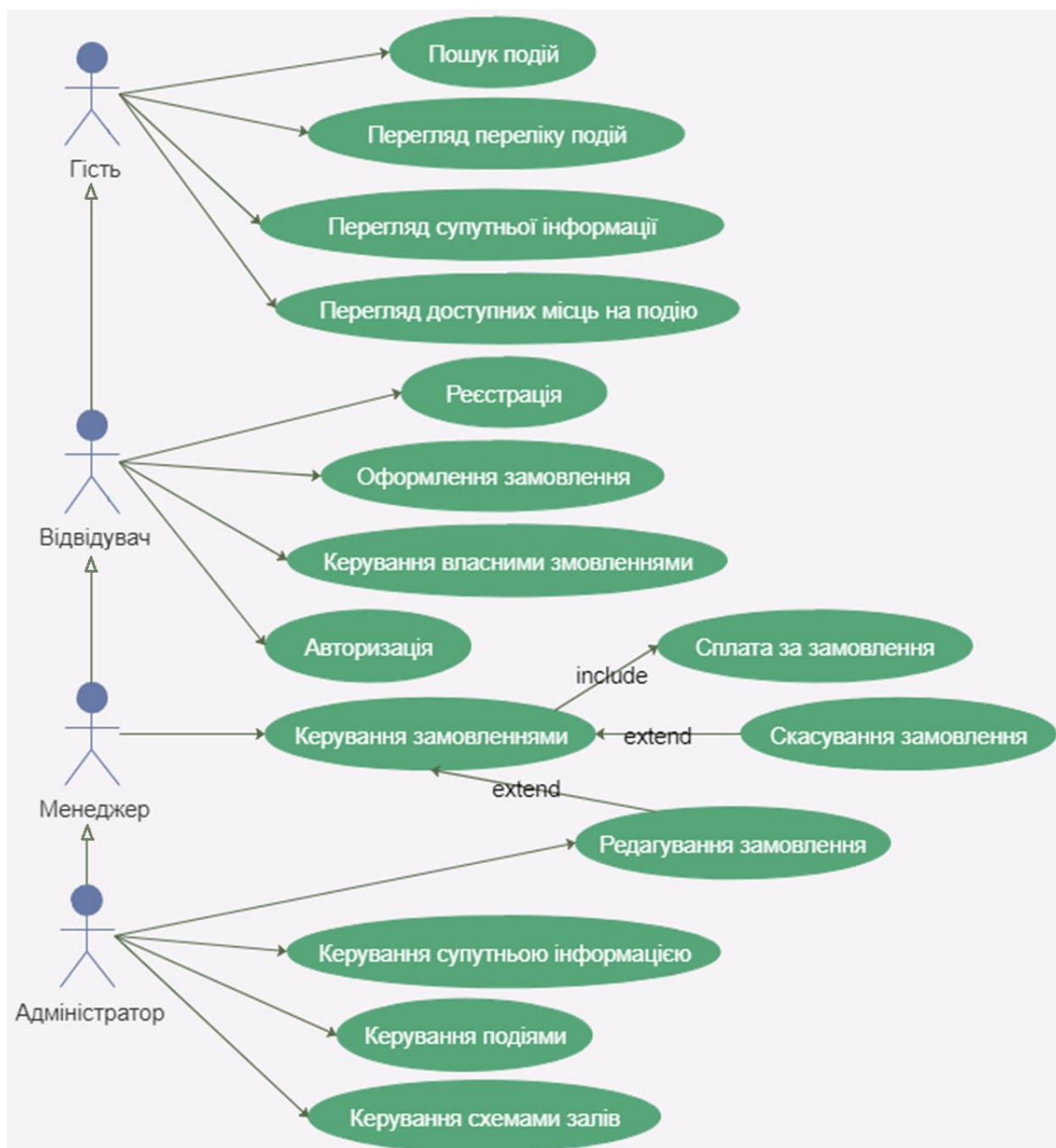


Рисунок 2.1 – Діаграма прецедентів

Гість може тільки передивлятися інформацію про події та дотичну до цього інформацію, а також переглядати інформацію про доступні місця на

обрану подію. При цьому замовити квиток на таке місце він не зможе, допоки не стане гостем, тобто не авторизується.

Відвідувачі можуть виконувати дії гостя, а також оформлювати замовлення, переглядати власні замовлення, скасовувати їх за потреби та оформляти запит на повернення коштів.

Менеджер може використовувати програму щодо всіх сценаріїв, які належать до гостя, підтримуючи таким чином його консультування або надання послуги через менеджера, а також керувати замовленнями, включаючи їх пошук, внесення даних про сплату. Але при цьому менеджер не має можливості стандартного редагування даних замовлення, тільки обмеженого.

Стандартне редагування доступне тільки адміністратору. Тобто у випадку будь-якої некоректності змінити вручну замовлення може тільки адміністратор. Менеджер може тільки підтвердити або відхилити замовлення.

Адміністратору доступний весь спектр функцій. Він може виконати будь-яку дію, передбачену попереднім переліком.

2.3 Проєктування структури даних

У даному підрозділі реалізовано проєктування структури даних, адже всі функціональні можливості пов'язані з роботою з даними.

Розглянемо таблиці, які було виділено під час створення БД. Системою керування БД було використано MySQL [11]: вона відзначається якісною зв'язкою з Django, є безкоштовною, до того ж вирізняється надійністю роботи.

Структура таблиці Organizer, що визначає організатора культурних подій, представлена в табл. 2.2.

Структура таблиці Artist, що визначає артиста або театральну труп, представлена в табл. 2.3.

Таблиця 2.2 – Структура таблиці Organizer

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Title	varchar(100)	–	–	Назва
3	Site	varchar(255)	–	–	Вебсайт
4	Phone	varchar(13)	–	–	Телефон
5	Address	varchar(255)	–	–	Адреса

Таблиця 2.3 – Структура таблиці Artist

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Title	varchar(100)	–	–	Ім'я
3	About	text	–	–	Опис
4	Site	varchar(255)	–	–	Вебсайт
5	Spotify	varchar(255)	–	–	Сторінка на Spotify
6	Deezer	varchar(255)	–	–	Сторінка на Deezer
7	Youtube-Music	varchar(255)	–	–	Сторінка на Youtube Music
8	Youtube	varchar(255)	–	–	Сторінка Youtube
9	Photo	varchar(255)	–	–	Фотографія

Структура таблиці Hall, що визначає місце проведення події (зал: театр, концертна зала, спортивний комплекс), представлена в табл. 2.4.

Таблиця Event, що визначає культурну подію, представлена в табл. 2.5.

Таблиця 2.4 – Структура таблиці Hall

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Title	varchar(100)	–	–	Назва
3	Site	varchar(255)	–	–	Вебсайт
4	Phone	varchar(13)	–	–	Телефон
5	Address	varchar(255)	–	–	Адреса
6	Photo	varchar(255)	–	–	Фотографія
7	About	text	–	–	Опис
8	City	varchar(30)	–	–	Місто

Таблиця 2.5 – Структура таблиці Event

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	ArtistId	int	–	+	Артист
3	HallId	int	–	+	Зала
4	OrganizerId	int	–	+	Організатор
5	Type	bool	–	–	Тип
6	When	datetime	–	–	Дата і час
7	About	text	–	–	Опис
8	CategoryId	int	–	+	Категорія
9	Photo	varchar(255)	–	–	Фотографія
10	Title	varchar(100)	–	–	Назва
11	CreatorId	int	–	+	Автор

Тип події визначає тип продажу квитків на подію, що може бути за категоріями або за окремими місцями. За категоріями визначаються категорії квитків і кількість квитків у цій категорії. За місцями визначаються конкретні місця. Creatorid є посиланням на запис у таблиці Django.

Структура таблиці Scheme, що визначає схему розташування місць в залі, представлена в табл. 2.6.

Таблиця 2.6 – Структура таблиці Scheme

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Title	varchar(100)	–	–	Назва схеми
3	HallId	text	–	+	Зала
4	Type	bool	–	–	Тип
5	Picture	varchar(255)	–	–	Зображення схеми
6	Wplaces	int	–	–	Кількість місць у ряді
7	Hplaces	int	–	–	Кількість рядів

У даній таблиці об'єднано варіанти схем за використання зображення і виділення зон (продаж квитків за категоріями), і за варіанту продажу конкретних місць (без використання зображення залу).

Для другого варіанту визначається розмітка залу, що включає максимальну кількість місць в ширину в залі та максимальну кількість рядів. Далі за цим набором чарунок визначається, в яких з них є місце, а в яких – немає.

Структура таблиці SchemeEvent, що визначає місця на події, що

розповсюджуються за принципом загальних категорій, представлена в табл. 2.7.

Таблиця 2.7 – Структура таблиці SchemeEvent

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Scheme-CategoryId	int	–	+	Категорія квитків у схемі
3	EventId	int	–	+	Подія
4	PriceId	int	–	+	Категорія вартості
5	Quantity	int	–	–	Кількість місць
6	Free	int	–	–	Кількість вільних місць

Структура таблиці PlaceEvent, що визначає місця на культурну подію за номерами, представлена в табл. 2.8.

Таблиця 2.8 – Структура таблиці PlaceEvent

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	EventId	int	–	+	Подія
3	PriceId	int	–	+	Категорія вартості
4	PlaceId	int	–	+	Місце у залі

Структура таблиці SchemeCategory, що визначає зони зображення схеми залу, представлена в табл. 2.9.

Таблиця 2.9 – Структура таблиці SchemeCategory

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	SchemeID	int	–	+	Схема залу
3	Coordinates	varchar(255)	–	–	Координати категорії на схемі

Поле Coordinates визначає координати вершин багатокутника, який на зображенні схеми залу визначає категорію квитків. Це необхідне для реалізації роботи з зображеннями через мапи.

Структура таблиці Price, що визначає вартість категорій квитків, представлена в табл. 2.10.

Таблиця 2.10 – Структура таблиці Price

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	CategoryPrice	int	–	–	Категорія вартості
3	Price	float	–	–	Вартість

Структура таблиці Place, що визначає конкретне місце у схемі залу з продажем місць за номерами, представлена в табл. 2.11.

Таблиця 2.11 – Структура таблиці Place

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	SchemeId	int	–	+	Схема залу
3	Wposition	int	–	–	Номер місця у ряді
4	Hposition	int	–	–	Номер ряду
5	PlaceTypeId	int	–	+	Тип місця
6	Type	bool	–	–	Є місце чи немає
7	Row	int	–	–	Ряд
8	Inrow	int	–	–	Місце

Структура таблиці PlaceType, що визначає тип місць (партер, балкон або більш специфічні типи – наприклад, номер готелю для вертикального концерту), представлена в табл. 2.12.

Таблиця 2.12 – Структура таблиці PlaceType

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Title	varchar(100)	–	–	Назва

Структура таблиці Ticket, що визначає квиток на культурну подію, який може визначатися або конкретним місцем або належати до категорії місць, представлена в табл. 2.13.

Таблиця 2.13 – Структура таблиці Ticket

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	PlaceEventId	int	–	+	Місце (з номером)
3	SchemeEventId	int	–	+	Місце (без номера)
4	Price	float	–	–	Вартість
5	Type	bool	–	–	Стан
6	OrderId	int	–	+	Замовлення

Структура таблиці Guest, що визначає відвідувача культурної події і містить про нього повну інформацію (обов'язково вказати номер телефону для можливої перевірки), представлена в табл. 2.14.

Таблиця 2.14 – Структура таблиці Guest

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	FirstName	varchar(100)	–	–	Ім'я
3	LastName	varchar(100)	–	–	Прізвище
4	Phone	varchar(13)	–	–	Телефон
5	Email	varchar(100)	–	–	Електронна пошта
6	UserId	int	–	+	Користувач Django

Структура таблиці Order визначає замовлення (табл. 2.15).

Структура таблиці Office, що визначає офіс, де можна забрати

фізичний квиток, представлена в табл. 2.16.

Таблиця 2.15 – Структура таблиці Order

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Price	float	–	–	Вартість
3	When	datetime	–	–	Дата формування
4	GuestId	int	–	+	Гість
5	State	tinyint	–	–	Стан
6	Type	bool	–	–	Друкований або електронний квиток
7	OfficeId	int	–	+	Офіс, де буде отримано квиток
8	Message	varchar(255)	–	–	Повідомлення від гостя
9	Code	varchar(20)	–	–	Код

Таблиця 2.16 – Структура таблиці Office

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Title	varchar(100)	–	–	Назва
3	Address	varchar(255)	–	–	Адреса
4	Phone	varchar(13)	–	–	Телефон

Структуру бази даних представлено на рис. 2.2.

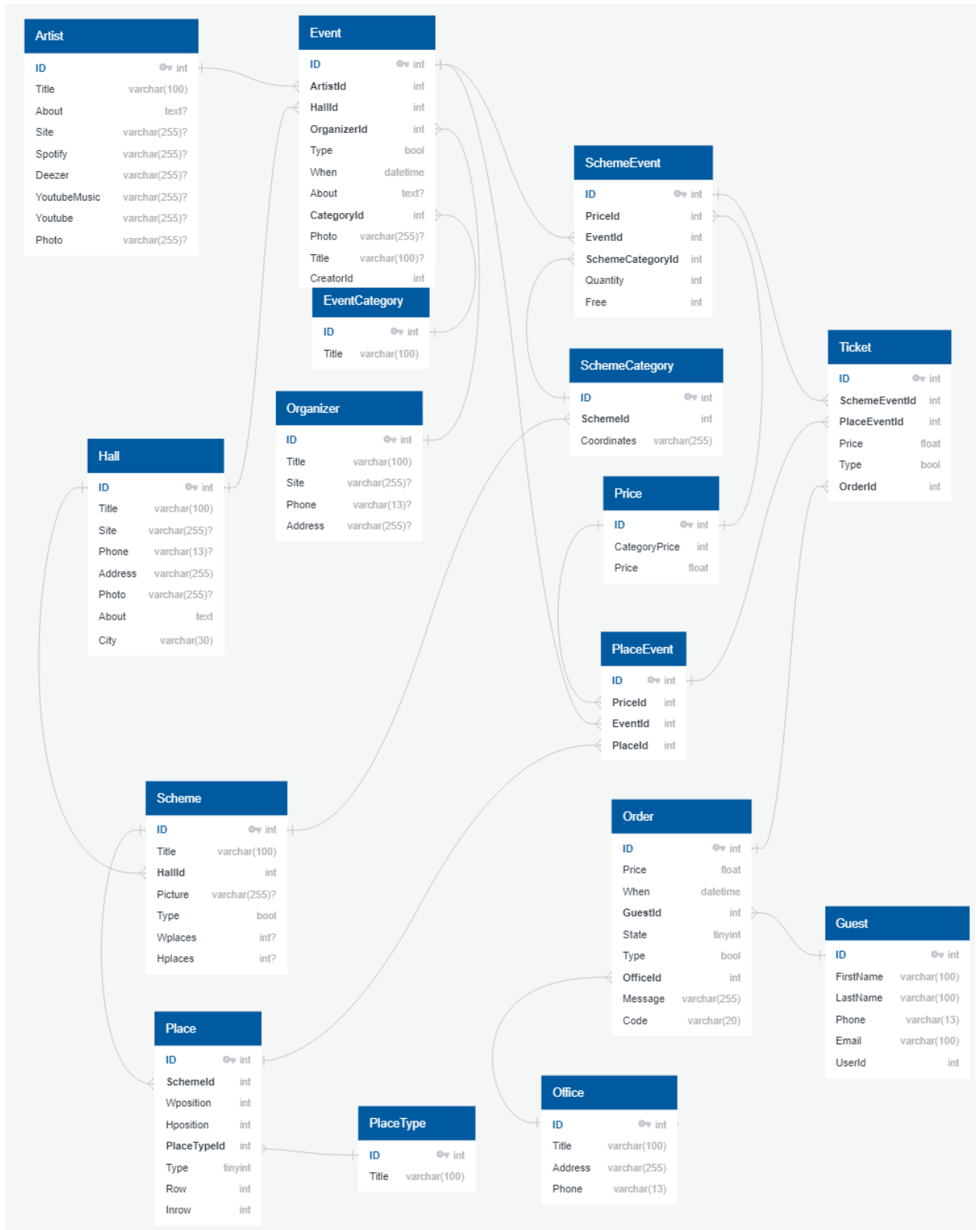


Рисунок 2.2 – Структура БД

Структура таблиці EventCategory, що визначає категорію культурної

події (музичний концерт певного напрямку, театральна вистава, цирк) представлена в табл. 2.17.

Таблиця 2.17 – Структура таблиці EventCategory

Номер поля	Назва поля	Тип поля	Первинний ключ	Зовнішній ключ	Опис
1	ID	int	+	–	Номер
2	Title	varchar(100)	–	–	Назва

2.4 Висновки за розділом 2

У другому розділі проаналізовано вебфреймворки розробки мовою Python, обрано вебфреймворк Django для реалізації, виділено вимоги до програми, виділено необхідні для реалізації функціональні можливості, визначено ролі користувачів, виконано моделювання сценаріїв роботи користувачів з програмою, реалізовано проектування структури даних на основі системи керування БД MySQL.

3 РОЗРОБКА ПРОГРАМИ

3.1 Основна логіка виконання програми

Логіка виконання програми побудована на взаємодії з користувачем, який обирає потрібні йому функціональні можливості для виконання в потрібний момент. При цьому головною сутністю предметної області є замовлення, оскільки проблема, пов'язана з продажем квитків. Відповідно саме на прикладі стану замовлення і можна показати, яким чином реалізовується логіка виконання програми.

Замовлення з'являється в той момент, коли його створює користувач системи. У цей момент замовлення можна вважати таким, що знаходиться у стані «створене».

Після цього до моменту сплати користувачем або визначення факту сплати менеджером замовлення може знаходитися або в стані «створене», або перейти до стану «скасоване». Якщо замовлення скасоване, то на цьому його життєвий цикл завершується.

Якщо замовлення сплачене, то у будь-який момент користувач може запитати повернення коштів. Якщо запит підтверджено, то замовлення переходить нову до стану «скасоване», завершуючи своє життя в системі.

Якщо запит не підтверджено або користувач не звертався з таким запитом, то змінити стан замовлення може також скасування організатором події, у такому випадку замовлення також перейде до стану «скасоване».

Якщо всіх цих подій не відбулося, то зі стану «сплачене» замовлення переходить до стану «виконане» у той момент, коли настає дата і час початку події. Цей стан також призводить до завершення життєвого циклу замовлення в системі.

Відповідну даному опису діаграму станів замовлення приведено на рис. 3.1.

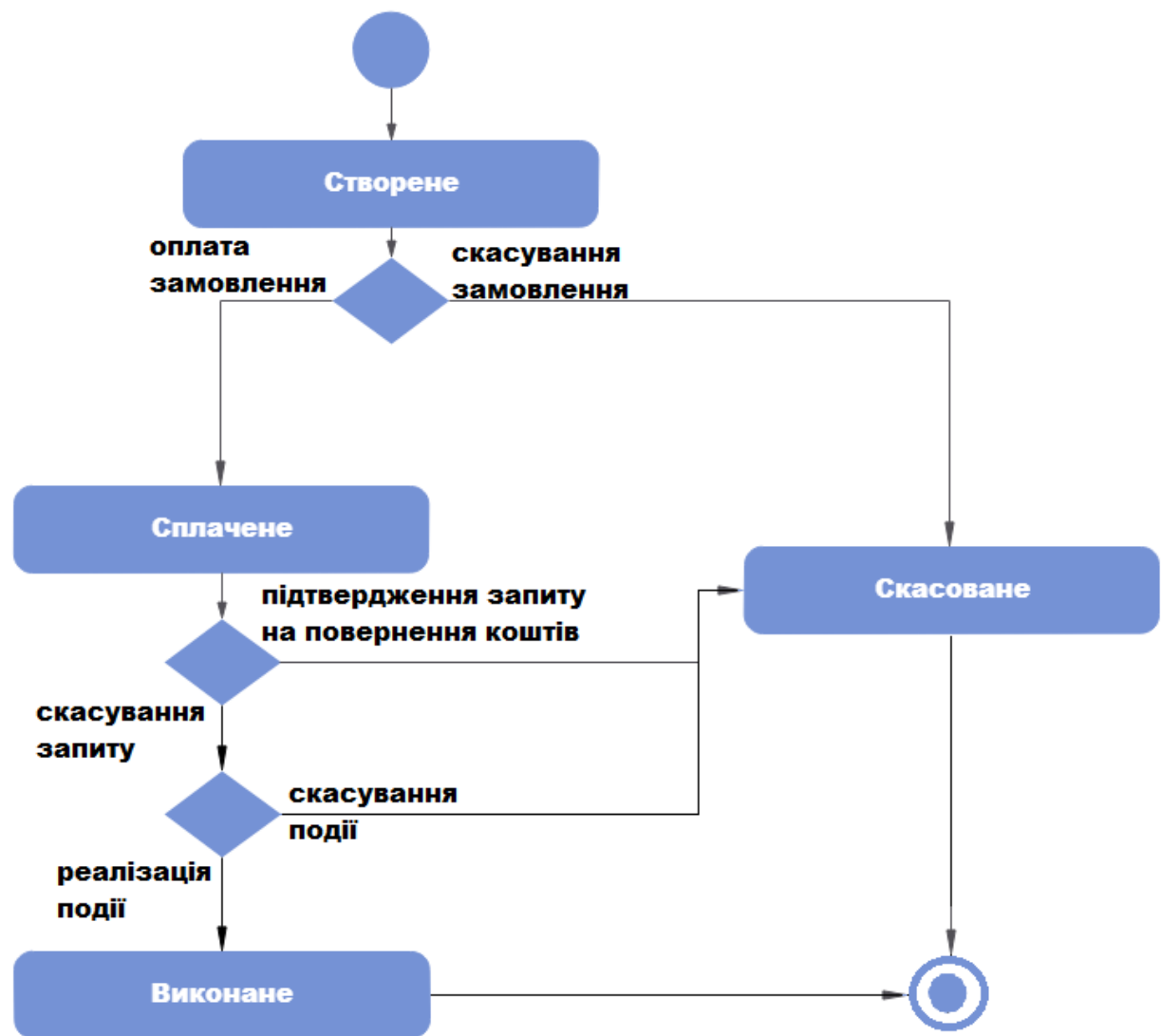


Рисунок 3.1 – Діаграма станів замовлення

3.2 Опис реалізації програми

Для реалізації програмного забезпечення першочергово було реалізовано роботу з даними. Для цього було створено класи моделей, що відповідають таблицям БД:

- Artist – модель артиста (виконавця або театральної трупи);
- Event – модель культурної події;
- Price – модель вартості квитків;
- PlaceEvent – модель розподілу місць у схемі зали;
- SchemeCategory – модель зонування схеми зали за категоріями;
- SchemeEvent – модель розподілу місць у залі за категоріями;

- Organizer – модель організатора;
- Place – модель місця у залі;
- Guest – модель гостя, що створено на основі моделі User системи аутентифікації Django з внесенням особистих даних;
- Office – модель пункту продажу квитків;
- PlaceType – модель типу місця в залі;
- Hall – модель зали;
- Scheme – модель схеми зали;
- EventCategory – модель категорії події;
- Order – модель замовлення;
- Ticket – модель квитка.

Для реалізації функціональних особливостей, описаних у розділі 2, було створено набір шаблонів сторінок, кожна з яких формує відповідне відображення для користувача, зокрема:

- `artistevents.html` – шаблон подій артиста;
- `artists.html` – шаблон списку артистів у системі;
- `categeventscheme.html` – шаблон створення схеми зали за категоріями;
- `categoryevents.html` – шаблон перегляду подій за категорією;
- `categoryprices.html` – шаблон встановлення вартості квитків на подію за категоріальної схеми зали;
- `halls.html` – шаблон перегляду залів у системі;
- `hall.html` – шаблон перегляду повних даних зали;
- `halledit.html` – шаблон редагування зали;
- `hallevts.html` – шаблон подій за залом;
- `numberorders.html` – шаблон пошуку замовлення за номером;
- `offices.html` – шаблон перегляду місць продажу квитків;
- `office.html` – шаблон повної інформації про місце продажу квитків;
- `officeedit.html` – шаблон редагування даних офісу;
- `orders.html` – шаблон перегляду замовлень за подіями організатора;

- `organizerevents.html` – шаблон списку подій організатора;
- `othercategoryevents.html` – шаблон перегляду переліку подій за іншими категоріями;
- `payment.html` – шаблон сплати за замовлення;
- `placeeventscheme.html` – шаблон створення схеми зали з продажем квитків за місцями;
- `placeprices.html` – шаблон встановлення вартості квитків на подію за продажу квитків на місця;
- `phoneorders.html` – шаблон пошуку замовлень за номером телефону;
- `returnorders.html` – шаблон керування запитами на повернення коштів;
- `searchevents.html` – шаблон пошуку подій;
- `visitororders.html` – шаблон керування власними замовленнями.

Шаблони сторінок створено на основі використання вільного HTML-шаблону [12].

Підготовлення даних, що заповнюють шаблон вебсторінки і формують відповідну вебсторінку, у Django реалізується через представлення, серед яких зокрема було розроблено:

- `ArtistDetail` – представлення повної інформації про артиста;
- `ArtistEventsList` – представлення подій за артистом;
- `EventCreate` – представлення створення події;
- `EventDetail` – представлення повної інформації про подію;
- `EventsCategoryList` – представлення подій за категорією;
- `EventsList` – представлення подій;
- `HallEventsList` – представлення подій за залом;
- `OrderCreate` – представлення створення замовлення;
- `OrderDetail` – представлення повної інформації про замовлення;
- `OrdersList` – представлення замовлень за подіями, які створив авторизований користувач;
- `OrganizerEventsList` – представлення подій за організатором;

- PhoneOrdersList – представлення замовлень за номером телефону;
- ReturnOrdersList – представлення запитів на повернення коштів;
- TicketDetail – представлення повної інформації про квиток;
- VisitorOrdersList – представлення замовлень за відвідувачем.

Усі представлення було об'єднано у відповідні групи, представлені відповідними пакетами, зокрема:

- artists.py – представлення роботи з набором артистів;
- event.py – представлення роботи над конкретною подією (створення, редагування, деталізація);
- events.py – представлення роботи з подіями (списками);
- halls.py – представлення роботи над залами та окремими їх одиницями;
- order.py – представлення роботи з замовленнями (створення, редагування, деталізація);
- orders.py – представлення роботи з замовленнями (списками);
- prices.py – представлення визначення вартості квитків на подію;
- schemes.py – представлення роботи зі схемами залів.

Загальну структуру застосунку з приведенням основних структурних одиниць представлено на схемі структури програми (рис. 3.2).

Представлення формується шляхом звернення до контексту моделей даних та отримання списків відповідних даних з їхнім подальшим фільтруванням.

Для визначення списку подій, з якими може працювати авторизований користувач (адміністратор), реалізується фільтрування подій за полем creator, перевіряючи його на відповідність обліковому запису користувача, інформація про який міститься в параметрі запиту self.request.user:

```
events = Event.objects.filter(creator = self.request.user)
```

Упорядкування об'єктів за потрібною характеристикою реалізується наступним чином (у даному випадку замовлення відображаються за датою оформлення, починаючи від останньої події і завершуючи найбільш ранньою):

```
orders = orders.order_by("-when")
```

Для реалізації створення QR-коду квитка було використано бібліотеку [13]. QR-код формується на основі рядка даних, що включає номер телефону замовника (відвідувача), номер події, присвоєний замовленню (не окремому квитку) у БД, та спеціальний код, що формується окремо для кожного замовлення і включає 20 випадкових символів. Ці 20 символів зберігаються в БД для подальшої перевірки, що може виконуватися під час перевірки на вході в залу.

Для реалізації сплати за замовлення засобами платіжної системи Приватбанку LiqPay було використано бібліотеку [14].

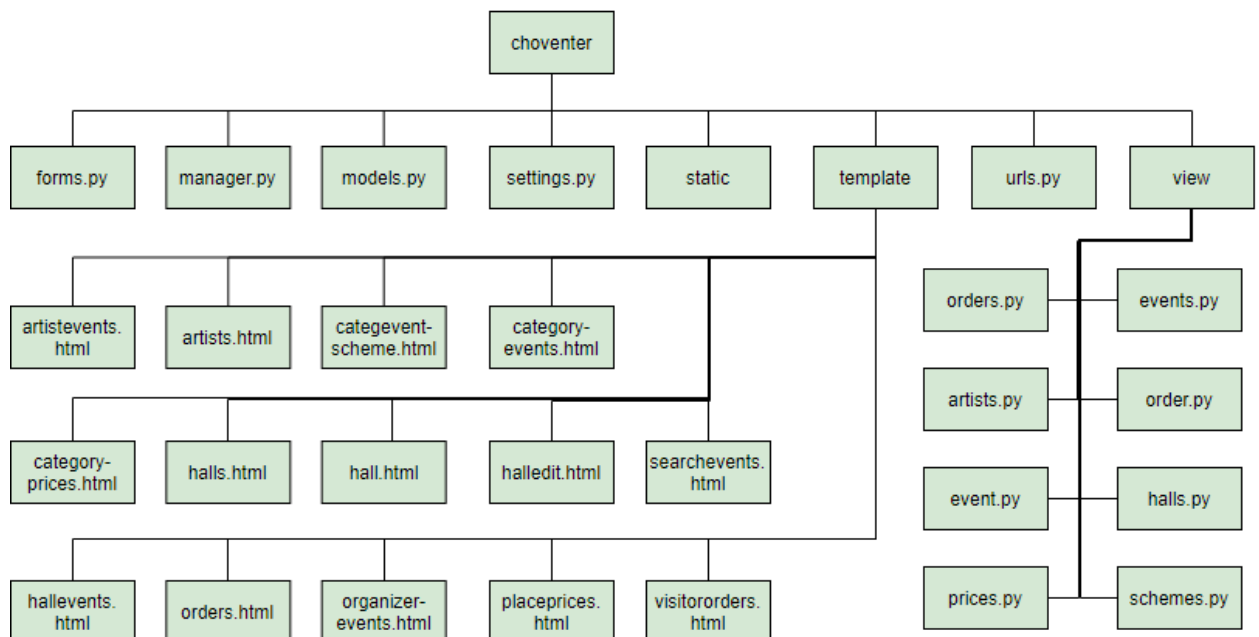


Рисунок 3.2 – Схема структури програми

3.3 Висновки за розділом 3

Розглянуто логіку виконання програми на основі визначення станів замовлення з представленням відповідної діаграми. Описано реалізацію програми з виділенням основних розроблених класів, використаних бібліотек та логіки реалізації.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для організації платформи для організаторів подій, розповсюджувачів квитків на ці події та відвідувачів подій. Організатори подій надають контент для внесення даних у програму. Розповсюджувачі квитків можуть реалізовувати безпосередньо свій бізнес через програму. Відвідувачі подій можуть задовольняти власні потреби в культурних подіях, оформлюючи та керуючи замовленнями на покупку квитків.

4.2 Умови виконання програми

Для виконання програми потрібно на серверній стороні забезпечити наступні умови:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

4.3 Робота з програмою

Користувач в залежності від своєї ролі в програмі отримує відповідне меню. Тобто набір пунктів у ньому різниться і відповідно визначає функціональні можливості, що доступні користувачу.

Гість має доступ до таких пунктів меню:

- «Події», що надає можливість переглядати доступні події;
- «Зали», що надає можливість переглянути всі зали або зали у заданому місті;

- «Артисти», що надає можливість подивитися на перелік артистів, з якими працює сервіс;
- «Офіси», що надає можливість подивитися на перелік офісів продажу квитків, де можна забрати фізичні квитки;
- «Організатори», що надає можливість переглянути інформацію про організаторів, з якими працює сервіс;
- «Увійти», що надає можливість авторизуватися або зареєструватися за необхідності.

Пункт меню «Події» (рис. 4.1) надає доступ до широкого спектра засобів перегляду списку подій на сторінці.

Першим засобом є перегляд подій за категоріями. Список категорій відображається зверху сторінки. Список категорій включає «Усі» для відображення всіх подій незалежно від їх категорії, «Музика» для відображення тільки музичних концертів, «Театр» для відображення театральних подій, «Діти» для відображення дитячих заходів, «Інші» для відображення усіх інших категорій подій.

Другим засобом є набір засобів пошуку подій. Форма пошуку дозволяє ввести назву артиста, події або зали в текстовому полі: пошук буде виконуватися у всіх цих трьох характеристиках одночасно на співпадіння частини назви з введеним значенням. Форма також містить список для вибору міста, в якому відбувається подія, та поле для завдання календарної дати, що дозволяє визначити дату події. Після того, як всі потрібні значення було задано, при цьому будь-які з них можуть залишитися порожніми, тоді усі події за категорією буде відображено нижче, можна натиснути на кнопку «Знайти» і отримати список всіх подій, що підпадають під критерії пошуку.

Кожна подія відображається рядком таблиці, ліворуч в якому демонструється дата події, окрім того рядок містить назву артиста, місто, де відбувається подія, і через двокрапку до нього назву події, якщо вона була визначена. Окрім того в рядку задається місце проведення події (зала), час початку події. Для перегляду деталізованих даних про подію потрібно

натиснути на кнопку «Про подію». З цієї ж сторінки можна буде перейти до придбання квитків.

Усі події, які відображаються, мають час реалізації не пізніше за поточний.

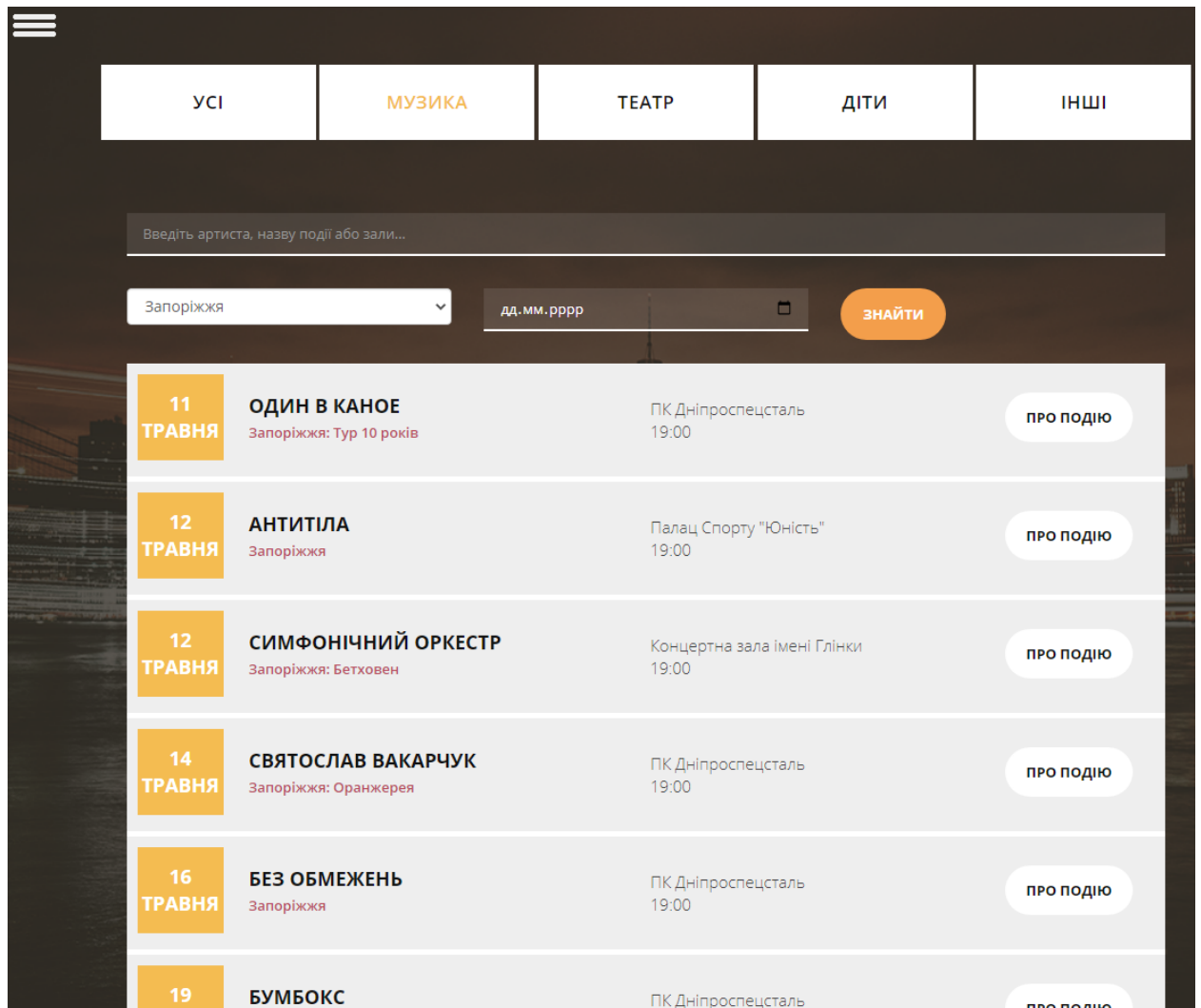


Рисунок 4.1 – Відображення списку подій за заданим містом

Якщо натиснути на ім'я артиста, то відкриється сторінка з інформацією про артиста та набором подій, в яких він бере участь найближчим часом. Також аналогічно можна перейти на сторінку залу.

Відвідувачу після авторизації додатково доступні наступні пункти меню:

– «Замовлення», що дозволяє переглянути власні замовлення на купівлю квитків;

– «Поточне», що дозволяє перейти до поточного, ще не підтвердженого замовлення.

Пункт «Замовлення» дозволяє переглядати всі власні замовлення, а також керувати ними. Звідси також доступний електронний квиток, якщо він був раніше оформлений (рис. 4.2).

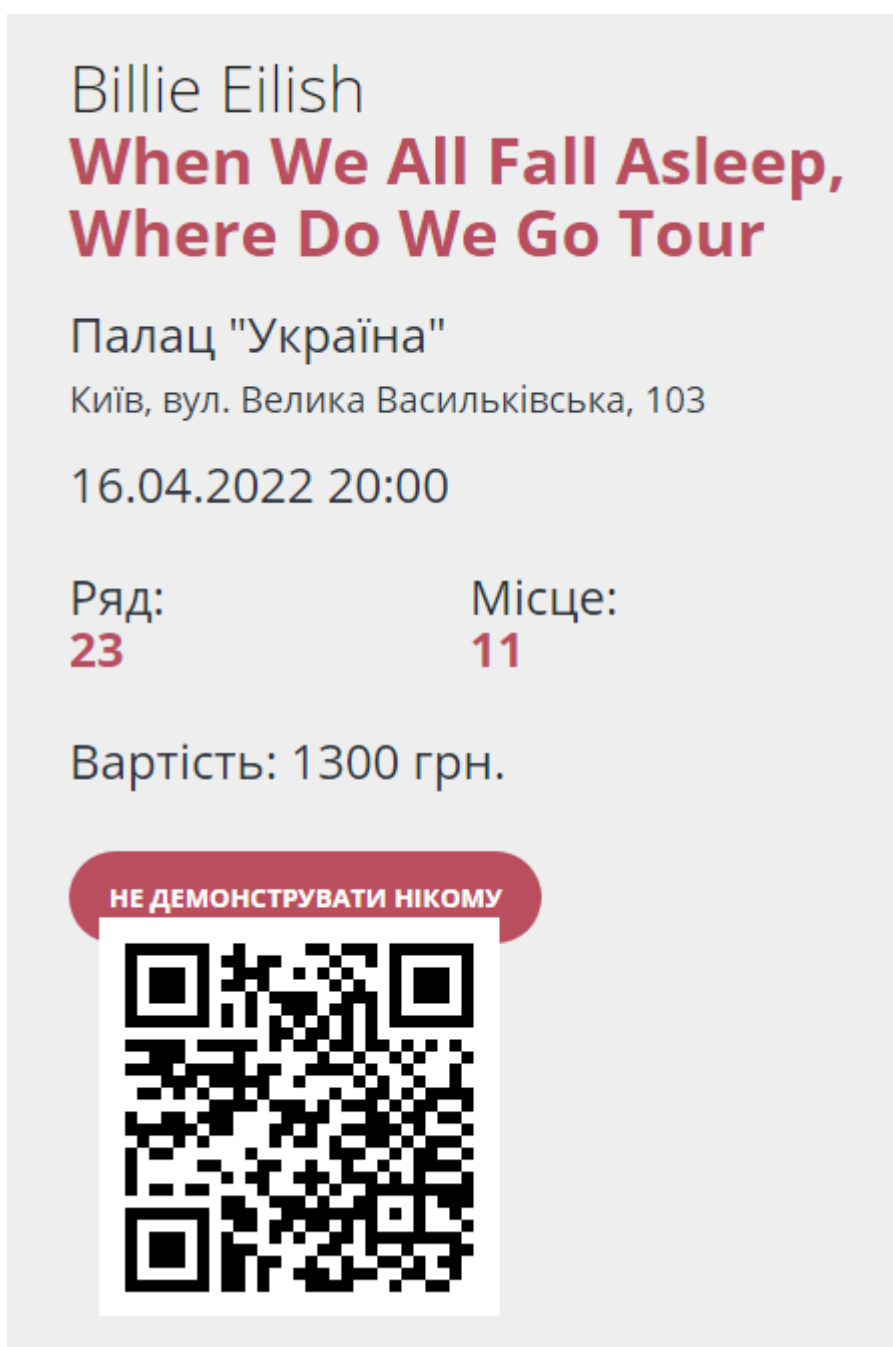


Рисунок 4.2 – Сформований програмою електронний квиток

Якщо в замовленні час на підтвердження замовлення ще не вийшов, то можна скасувати замовлення. Окрім того, якщо замовлення вже сплачено, то можна оформити запит на повернення коштів, вказавши причину повернення.

Електронний квиток містить:

- ім'я артиста вгорі;
- назву події нижче;
- інформацію про місце проведення події, в тому числі повну адресу місця проведення;
- час і дату початку події;
- інформацію про категорію квитка або ряд і місце, на яке було придбано квиток;
- вартість квитка;
- QR-код, який містить інформацію, яка не повинна розголошуватися відвідувачем нікому, і призначений для контролю доступу на захід.

Пункт «Поточне» дозволяє переглянути склад поточного замовлення та підтвердити його за необхідності або скасувати (рис. 4.3). У нього додаються всі квитки, які користувач, обравши відповідне місце, в кінці процесу додав до замовлення. Користувач може або підтвердити замовлення і отримати 3 дні на його сплату (тоді менеджер повинен підтвердити сплату), або сплатити онлайн і отримати в результаті електронний квиток.

Менеджеру додатково до пунктів меню гостя надається доступ до таких пунктів меню:

- «Номер замовлення», що надає можливість пошуку замовлення за номером;
- «Номер телефону», що надає можливість пошуку замовлення за номером телефону замовника.

За результатами пошуку аналогічно списку подій відображається список замовлень, що відповідають критеріям пошуку. Для кожного замовлення менеджер може позначити внесення сплати за допомогою кнопки

«Сплачено» або скасувати за допомогою кнопки «Скасувати». Якщо замовлення не було сплачено вчасно, воно буде автоматично скасоване.

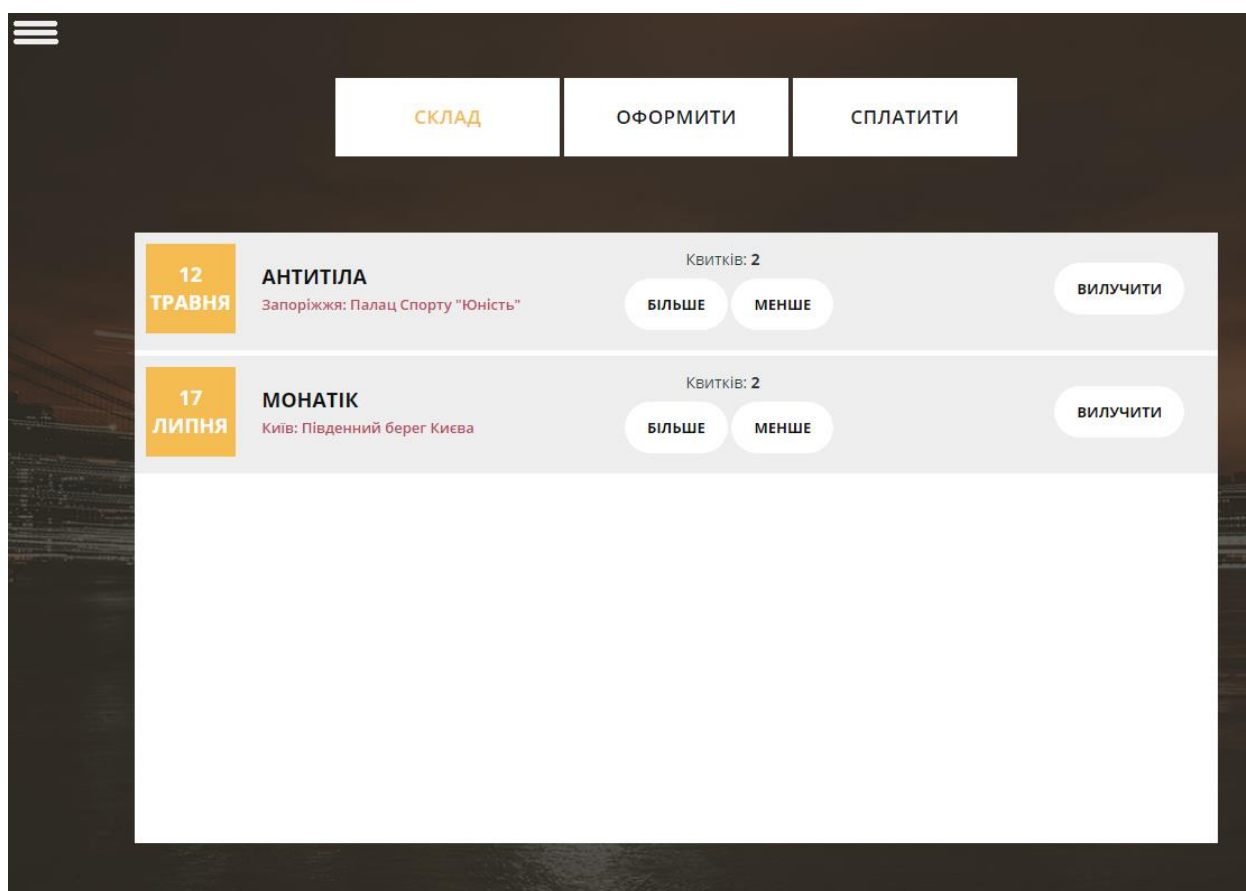


Рисунок 4.3 – Поточне замовлення

Адміністратору системи додатково до пунктів меню гостя надається доступ до таких пунктів меню:

- «Мої події», що надає можливість переглянути список всіх власних подій: фактично призводить до активізації категорії «Мої», що відображається адміністратору на відміну від усіх інших користувачів;

- «Замовлення», що надає можливість перегляду всіх замовлень, виконаних за подіями, які були створені даним користувачем;

- «Запити на повернення», що надає можливість перегляду списку всіх запитів користувачів на повернення коштів за квитки: адміністратор може або підтвердити запит, тоді кошти повернуться на рахунок даного користувача, а квитки будуть скасовані, або скасувати запит, тоді нічого не

зміняться, кошти повернуті не будуть, при цьому адміністратор приймає своє рішення в залежності від примітки до запиту, а також може скористатися даними про номер телефону або електронну пошту користувача;

– «Вихід», що надає можливість вийти з акаунта.

Сторінка «Мої події» (рис. 4.4) є фактично списком подій, що відображається на сторінці подій і відповідає тим подіям, які були створені авторизованим на даний момент користувачем.

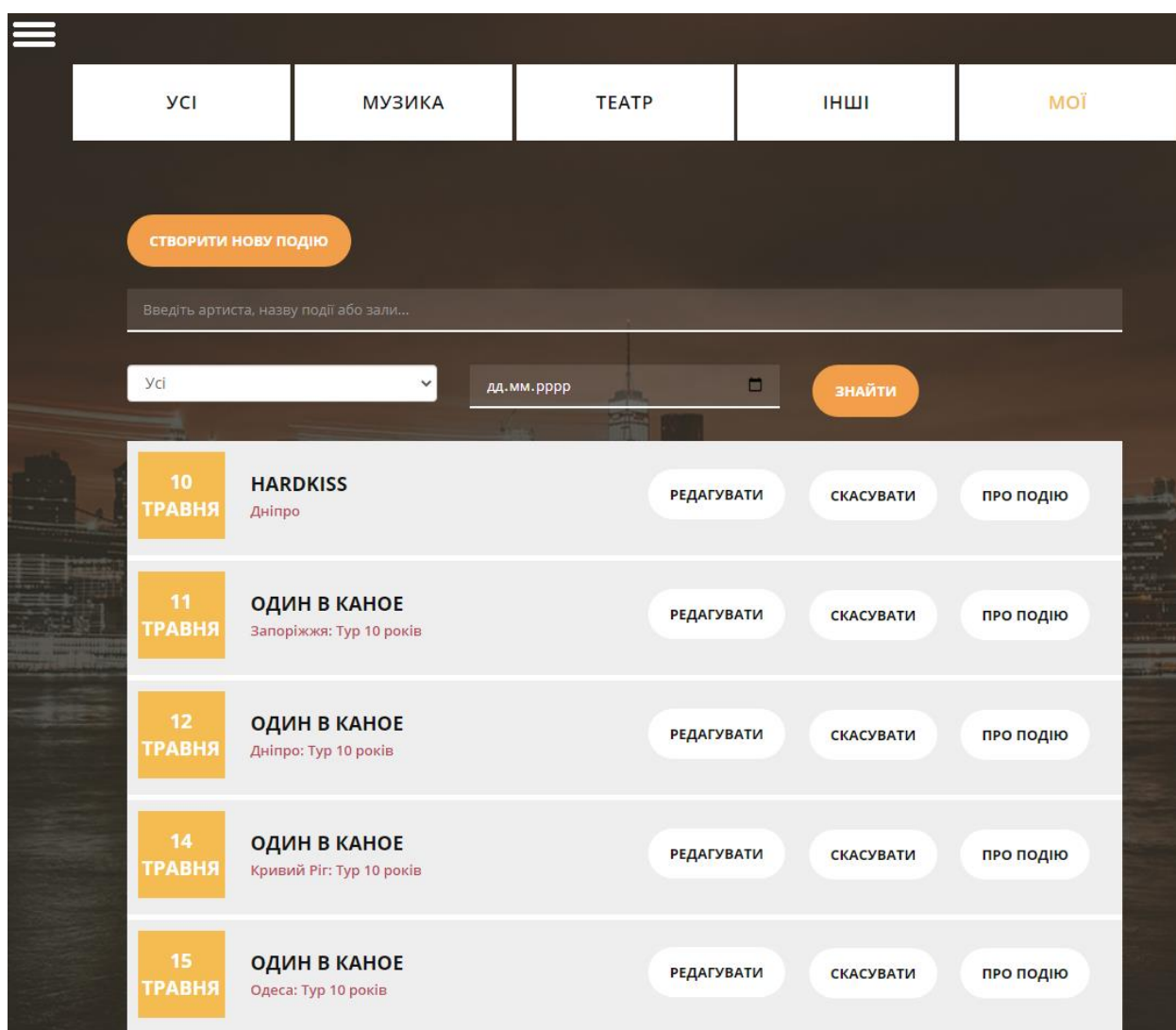


Рисунок 4.4 – Керування власними подіями

На відміну від інших користувачів адміністратору при цьому відображається для кожної події кнопка «Редагувати» для внесення змін в дані події, що зокрема включає і доступ до опису події, і змінювання вартості

квитків. Тобто за потреби в будь-який момент можна змінити вартість квитків. Тому в дані квитка зберігається інформація про його вартість, адже вартість місця може змінюватись.

Також адміністратору відображається кнопка «Скасувати», яка дозволяє повністю скасувати подію. Тоді у списках замовлень відповідних користувачів, які оформили квитки на дану подію, буде відображено відповідне повідомлення про скасування події. Ця функціональність важлива зокрема в поточний момент, коли через незалежні обставини велика частина запланованих концертів скасовується.

Якщо адміністратор відкриє сторінку залів через відповідний пункт меню, то йому, на відміну від гостя або відвідувача, буде доступною також можливість створення нових залів, редагування інформації про них та створення і редагування схем залів. Список відображається в такому випадку аналогічно підходу для подій, розглянутого вище.

Під час створення схеми залу (рис. 4.5) адміністратор (схема створюється саме для конкретного залу, що був обраний на попередньому кроці, як описано вище), спочатку задає на окремій сторінці кількість рядів та місць, тобто позицій за горизонталлю та вертикаллю, яку буде охоплювати схема залу, а після підтвердження цих дій адміністратору відкривається наступна сторінка. Тут йому відображається інформація про зал, що відображається у верхній частині сторінки, і схема залу, кількість квадратів у якій відповідає заданій кількості на попередньому кроці.

Кожна позиція на схемі позначена зеленим або сірим квадратом. Зелений відповідає тому, що в даній позиції буде продаватися місце. Сірий квадрат відповідає порожній позиції (тобто стільця в цьому місці немає).

При натисканні на позицію крісла зображення перетворюється з сірого на зелений або з зеленого на сірий. Тобто для того щоб активувати місце для продажу або навпаки деактивувати, потрібно натиснути на нього. При наведенні курсора на крісло, відображається підказка з номером ряду і місця та категорія місця.

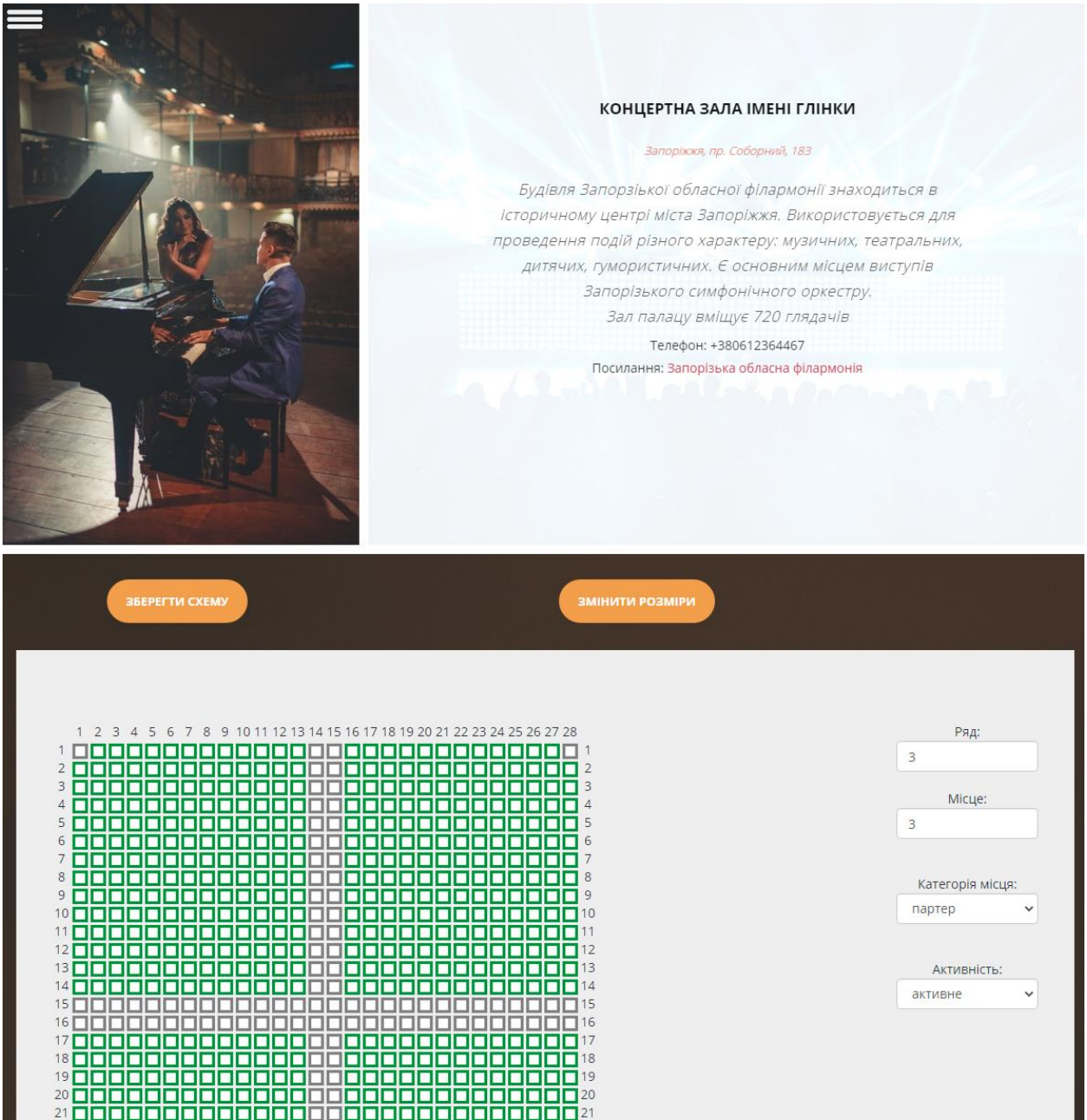


Рисунок 4.5 – Визначення схеми залу

Категорію місця можна обрати в правій частині. Тут відображається номер ряду та номер поточного місця. Для того щоб змінити їх, потрібно натиснути на відповідне зображення крісла, а тоді з випадваючого списку обрати категорію місця (партер, балкон і таке інше) та задати за необхідності активність (активне або неактивне), тобто буде продаватися це місце чи ні (дублює натиснення на зображення). Оскільки для вибору категорії місця доводиться все одно натискати на крісла, то спочатку всі місця у розмітці позначені сірими.

Зі сторінки події адміністратору доступний шляхом натискання кнопки «Вартість» перехід на сторінку визначення категорій вартості квитків

Під час створення події відображається зверху меню з кнопок, що проводять всім процесом створення події, кожна з яких стає активною на наступному етапі:

- «Подія» відповідає за створення події;
- «Зала» відповідає за вибір зали, в якій буде проходити подія;
- «Квитки» дозволяє створити квитки і визначити їх вартість;
- «Готово» дозволяє завершити роботу після внесення всіх даних про вартість квитків.

На сторінці «Квитки» (рис. 4.6) адміністратору відображається загальна інформація про подію, що включає назву артиста, назву події, назву зали та дату і час початку події.

При цьому відображається схема зали, яку було обрано раніше, створюючи залу в системі. Позиції, для яких місця на схемі не було визначено, не відображаються. Ряди або стовпці, в яких немає жодного місця, не рахуються в порядковий номер ряду або місця, тобто в квитку використовується система не позицій сітки, а конкретних рядів та місць.

Адміністратору потрібно з випадваючого переліку обрати категорію вартості квитків. Кожна категорія на схемі позначається кольором. Всього передбачено максимально 20 категорій вартості, при цьому двадцята категорія визначає недоступні місця, тобто ті, продаж яких не здійснюється. Вони позначаються сірим квадратом.

У текстовому полі «Вартість» після вибору категорії потрібно вказати вартість квитків.

Нижче розташовано список місць, що належать до цієї категорії. Зі списку можна обрати одну або декілька позицій одночасно і натиснувши на кнопку «Вилучити», прибрати місця з даної категорії.

Для додавання місця до категорії потрібно натиснути на нього на схемі і автоматично заповняться поля «Ряд» і «Місце». Значення сюди можна

також ввести вручну. Після визначення ряду і місця потрібно натиснути на кнопку «Додати» для внесення в таблицю місць.

Бумбокс
Десять років самотності
Концертна зала імені Глінки
26.06.2021 19:00

ЗБЕРЕГТИ

Категорія вартості:
2 (червона)

Вартість:
700

Місця:
Ряд: 1, Місце: 2
Ряд: 1, Місце: 3
Ряд: 1, Місце: 4
Ряд: 1, Місце: 5

ВИЛУЧИТИ

Ряд:
Місце:

ДОДАТИ

Рисунок 4.6 – Визначення категорій вартості квитків на конкретну подію

Після того, як визначення категорії завершено, потрібно натиснути на кнопку «Зберегти», після чого можна працювати над наступною категорією. Після збереження, на схемі відображаються задані місця потрібним кольором.

За замовчуванням всі місця належать до зеленого кольору і першої категорії.

Після завершення внесення всіх категорій потрібно натиснути на кнопку «Готово» зверху, дані будуть збережені повністю.

Після виконання даного пункту гості та відвідувачі, а також і менеджери зможуть працювати з подією, замовляти квитки. При перегляді доступних квитків за подією користувач також може обрати категорію квитків, що дозволить йому мати список з доступними квитками цієї вартості.

ВИСНОВКИ

У роботі розглянуто проблему розроблення програмного забезпечення продажу квитків на заходи в сфері культури. Для цього було виконано аналіз проблеми, визначено позитивні характеристики таких систем відносно роботи над заходами в сфері культури, визначено вимоги до програми та розглянуто основні функціональні можливості таких систем.

Проаналізовано фреймворки розробки вебзастосунків мовою Python, в результаті чого обрано вебфреймворк Django для реалізації. Виділено вимоги до програми, виділено необхідні для реалізації функціональні можливості. В якості ролей користувачів визначено неавторизованого користувача (гостя), відвідувача, менеджера та адміністратора. Виконано моделювання сценаріїв роботи користувачів з програмою. Виконано проєктування структури даних на основі системи керування базами даних MySQL з побудовою схеми бази даних.

Виконано програму реалізацію вебзастосунку. Приведено розроблені класи з описом принципів їх реалізації і приведенням відповідної діаграми класів. Розроблений вебзастосунок дозволяє створювати події, визначати схеми для продажу квитків на них за категоріальним принципом або на конкретні місця, використовуючи для цього універсальне рішення у вигляді сітки, оформляти замовлення на покупку квитків, формувати електронні квитки, керувати замовленнями.

Реалізована програма призначена для організації платформи для операторів з продажу, організаторів подій, розповсюджувачів квитків на ці події та відвідувачів подій. Організатори подій надають контент для внесення даних у програму. Розповсюджувачі квитків можуть реалізовувати безпосередньо свій бізнес через програму. Відвідувачі подій можуть оформляти та керувати замовленнями на покупку квитків.

Усі завдання дипломної кваліфікаційної роботи повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Osadchiy, V. How to Create an All-in-One Web Platform for Ticket Distribution [Electronic resource] / V. Osadchiy. – Access mode : <https://yalantis.com/blog/develop-ticket-distribution-platform/>
2. How to Create Web Platforms for Ticket Distribution [Electronic resource]. – Access mode : <https://www.app-scoop.com/blog/how-to-create-web-platforms-for-ticket-distribution>
3. Ticketmaster [Electronic resource] : Buy Verified Tickets for Concerts, Sports, Theatre and Events. – Access mode : <https://www.ticketmaster.com/>
4. Concert.ua [Електрон. ресурс]. – Режим доступу : <https://concert.ua/>
5. Kasa.in.ua [Електрон. ресурс]. – Режим доступу : <https://kasa.in.ua/>
6. Romano, F. Learn Web Development with Python [Text] : Get hands-on with Python Programming and Django web development / Fabrizio Romano, Gaston C. Hillar, Arun Ravindran. – Birmingham : Packt Publishing, 2018. – 796 p.
7. Web Development with Django [Text] : Learn to build modern web applications with a Python-based framework / Ben Shaw, Saurabh Badhwar, Andrew Bird, Bharath Chandra, Chris Guest. – Birmingham : Packt Publishing, 2021. – 826 p.
8. Grinberg, M. Flask Web Development [Text] : Developing Web Applications with Python / M. Grinberg. – Massachusetts : O'Reilly Media, 2014. – 258 p.
9. Building Websites with Django [Text] : Build and deploy professional websites with Python programming and the Django framework / Awanish Ranjan. – Mumbai : BPB Publications, 2021. – 242 p.
10. 8 Unique Features of Django that Makes it Better Framework [Electronic resource]. – Access mode : <https://data-flair.training/blogs/django->

features/

11. Ульман, Дж. Системы баз данных. Полный курс [Текст] / Джеффри Ульман, Дженнифер Уидом, Гектор Гарсия. – М. : Вильямс, 2016. – 1088 с.

12. Avalon Template [Electronic resource]. – Access mode : <https://templatemo.com/tm-513-avalon>

13. Pure python QR Code generator [Electronic resource]. – Access mode : <https://github.com/lincolnloop/python-qrcode>

14. LiqPay python sdk [Electronic resource]. – Access mode : <https://github.com/liqpay/sdk-python>

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення має надавати зручний доступ для купівлі квитків, для цього має підтримуватися доступ з різних пристроїв, надаватися зручний інтерфейс пошуку подій і замовлення квитків, а також інтерфейс керування цими подіями.

A.1 Підстави для розробки

Підставою для виконуваної розробки в межах дипломної кваліфікаційної роботи бакалавра є завдання, що було визначене за темою «Програмне забезпечення продажу квитків на заходи в сфері культури» і затверджене у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма, яка розробляється в роботі, призначена для організації платформи для організаторів подій, розповсюджувачів квитків на ці події та відвідувачів подій. Організатори подій надають контент для внесення даних у програму. Розповсюджувачі квитків можуть реалізовувати безпосередньо свій бізнес через програму. Відвідувачі подій можуть задовольняти власні потреби в культурних подіях, оформлюючи та керуючи замовленнями на покупку квитків.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати виконання наступних функціональних характеристик:

- створення культурної події;
- внесення схеми залу на основі зон карти з продажем місць за категорією;
- внесення схеми залу з продажем квитків за місцем;
- визначення категорій вартості квитків на культурну подію;
- зонування зали за категоріями;
- пошук культурних подій у заданому місті;
- пошук культурних подій на вказану дату;
- пошук культурних подій за артистом;
- пошук культурних подій за назвою;
- оформлення замовлення на придбання квитків;
- перегляд власних замовлень;
- перегляд інформації про замовлення за номером;
- перегляд інформації про замовлення за номером телефону;
- визначення сплати за замовлення;
- перегляд культурних подій за категорією;
- перегляд культурних подій за місцем проведення (за залом);
- редагування схеми залу;
- редагування вартості квитків;
- редагування інформації про подію;
- визначення доступних місць на подію;
- автоматичне скасування замовлення у разі несплати;
- перегляд інформації про культурну подію;
- реєстрація акаунта;
- авторизація акаунта;
- перегляд інформації про залу;
- перегляд інформації про артиста;
- перегляд інформації про організатора;
- перегляд інформації про офіс для продажу квитків;
- оформлення запиту на повернення коштів;

- скасування подій;
- внесення і редагування даних про артиста;
- внесення і редагування даних про зали;
- внесення і редагування даних про організаторів;
- внесення і редагування даних про офіси з продажу квитків.

A.3.2 Вимоги до надійності

Для забезпечення надійності має виконуватися валідація даних, що було введено користувачем в поля форм.

A.3.3 Умови експлуатації

Експлуатація програмного забезпечення має виконуватися на основі клієнт-серверної моделі. Для цього потрібно виділити окремий комп'ютер або скористатися виділеним сервером, який буде забезпечувати підтримку реалізації вебсервера. Вебсервер має бути доступним для запитів користувачів постійно.

A.3.4 Вимоги до складу та параметрів технічних засобів

Склад та параметри технічних засобів вебсервера було визначено для реалізації можливості доступу 300 користувачів одночасно. Відповідно вимоги можуть бути знижені, якщо потрібне забезпечення меншої кількості запитів, а для більшої кількості одночасних запитів вимоги навпаки мають бути збільшені. Мінімальні системні вимоги до вебсервера включають:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

А.3.5 Вимоги до транспортування та збереження

Безпосередньо програма має транспортуватися та зберігатися через репозиторій вебсервера. Збереження на окремих фізичних носіях не передбачається.

А.3.6 Вимоги до програмної документації

Основою розробки даного програмного проєкту є технічне завдання на нього.

Опис прийнятих рішень виконується в пояснювальній записці. У розділі «Експлуатація та тестування програми» має бути представлено опис процесу роботи з програмою. Інших програмних документів не передбачається.

А.4 Порядок контролю та приймання

Календарний план завдання на дипломну кваліфікаційну роботу визначає набір етапів, за якими має забезпечуватися виконання, а також результати, які мають бути отримані за виконання відповідного етапу. Саме на основі даних результатів має забезпечуватися приймання роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла views/events.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from .models import Event
from .models import Artist
from .models import Hall
from django.views.generic import ListView
from datetime import date
from django.shortcuts import get_object_or_404

class EventsList (ListView):
    model = Event
    template_name = '/template/searchevents.html'
    paginate_by = 10

    def get_queryset(self):
        self.reqtext = self.request.GET.get('request')
        self.reqdate = self.request.GET.get('date')
        self.reqcity = self.request.GET.get('city')
        result=[]
        if self.reqtext:
            artists = Artist.objects.get(title__icontains=self.reqtext)
            events = Event.objects.get(title__icontains=self.reqtext)
            halls = Hall.objects.get(title__icontains=self.reqtext)
            for art in artists:
                evs = events.filter(artist = art)
                for ev in evs:
                    if not self.reqdate and not self.reqcity:
                        result.append(ev)
                    elif not self.reqdate:
                        if self.reqcity == ev.hall.city:
                            result.append(ev)
                    elif not self.reqcity:
                        if ev.when.date() == self.reqdate:
                            result.append(ev)
            for ev in events:
                if not self.reqdate and not self.reqcity:
                    result.append(ev)
                elif not self.reqdate:
                    if self.reqcity == ev.hall.city:

```

```

        result.append(ev)
    elif not self.reqcity:
        if ev.when.date() == self.reqdate:
            result.append(ev)
    for h in halls:
        evs = events.filter(hall = h)
        for ev in evs:
            if not reqdate and not reqcity:
                result.append(ev)
            elif not self.reqdate:
                if self.reqcity == ev.hall.city:
                    result.append(ev)
            elif not self.reqcity:
                if ev.when.date() == self.reqdate:
                    result.append(ev)

    result = result.filter(when__gte=datetime.date.today())
    result = result.order_by(when)
    return result

    def get_context_data(self, **kwargs):
        infototemplate = super(EventsList,
self).get_context_data(**kwargs)
        infototemplate['request'] = self.reqtext
        infototemplate['date'] = self.reqdate
        infototemplate['city'] = self.reqcity
        return infototemplate

class EventsCategoryList (ListView):
    model = Event
    template_name = '/template/categoryevents.html'
    paginate_by = 10

    def get_queryset(self):
        self.eventcategory = self.request.GET.get('category')
        self.reqtext = self.request.GET.get('request')
        self.reqdate = self.request.GET.get('date')
        self.reqcity = self.request.GET.get('city')
        result=[]
        if self.reqtext:
            artists = Artist.objects.get(title__icontains=self.reqtext)
            events = Event.objects.get(title__icontains=self.reqtext)

```

```

        events = events.filter(category__title =
self.eventcategory)
        halls =
Hall.objects.get(title__icontains=self.reqtext)
        for art in artists:
            evs = events.filter(artist = art)
            evs = evs.filter(category__title =
self.eventcategory)
            for ev in evs:
                if not reqdate and not self.reqcity:
                    result.append(ev)
                elif not self.reqdate:
                    if self.reqcity == ev.hall.city:
                        result.append(ev)
                elif not self.reqcity:
                    if ev.when.date() == self.reqdate:
                        result.append(ev)
        for ev in events:
            if not self.reqdate and not self.reqcity:
                result.append(ev)
            elif not self.reqdate:
                if self.reqcity == ev.hall.city:
                    result.append(ev)
            elif not self.reqcity:
                if ev.when.date() == self.reqdate:
                    result.append(ev)
        for h in halls:
            evs = events.filter(hall = h)
            evs = evs.filter(category__title =
self.eventcategory)
            for ev in evs:
                if not self.reqdate and not
self.reqcity:
                    result.append(ev)
                elif not self.reqdate:
                    if self.reqcity == ev.hall.city:
                        result.append(ev)
                elif not self.reqcity:
                    if ev.when.date() == self.reqdate:
                        result.append(ev)
        result =
result.filter(when__gte=datetime.date.today())
        result = result.order_by("when")
        return result

def get_context_data(self, **kwargs):

```

```

        infototemplate = super(EventsList,
self).get_context_data(**kwargs)
        infototemplate['request'] = self.reqtext
        infototemplate['date'] = self.reqdate
        infototemplate['city'] = self.reqcity
        infototemplate['category'] = self.eventcategory
    return infototemplate

class OrganizerEventsList (LoginRequiredMixin, ListView):
    model = Event
    template_name = '/template/organizerevents.html'
    paginate_by = 10

    def get_queryset(self):
        self.reqtext = self.request.GET.get('request')
        self.reqdate = self.request.GET.get('date')
        self.reqcity = self.request.GET.get('city')
        result=[]
        if reqtext:
            artists = Artist.objects.get(title__icontains=self.reqtext)
            events = Event.objects.get(title__icontains=self.reqtext)
            events = events.filter(creator = self.request.user)
            halls = Hall.objects.get(title__icontains=self.reqtext)
            for art in artists:
                evs = events.filter(artist = art)
                evs = evs.filter(creator = self.request.user)
                for ev in evs:
                    if not self.reqdate and not self.reqcity:
                        result.append(ev)
                    elif not self.reqdate:
                        if self.reqcity == ev.hall.city:
                            result.append(ev)
                    elif not self.reqcity:
                        if ev.when.date() == self.reqdate:
                            result.append(ev)
            for ev in events:
                if not self.reqdate and not self.reqcity:
                    result.append(ev)
                elif not self.reqdate:

```

```

        if self.reqcity == ev.hall.city:
            result.append(ev)
        elif not self.reqcity:
            if ev.when.date() == self.reqdate:
                result.append(ev)
    for h in halls:
        evs = events.filter(hall = h)
        evs = evs.filter(creator =
self.request.user)
        for ev in evs:
            if not reqdate and not reqcity:
                result.append(ev)
            elif not self.reqdate:
                if self.reqcity == ev.hall.city:
                    result.append(ev)
            elif not self.reqcity:
                if ev.when.date() == self.reqdate:
                    result.append(ev)
    result =
result.filter(when__gte=datetime.date.today())
    result = result.order_by("when")
    return result

    def get_context_data(self, **kwargs):
        infotemplate = super(EventsList,
self).get_context_data(**kwargs)
        infotemplate['request'] = self.reqtext
        infotemplate['date'] = self.reqdate
        infotemplate['city'] = self.reqcity
        return infotemplate

class EventsOtherCategoryList (ListView):
    model = Event
    template_name = '/template/ocategoryevents.html'
    paginate_by = 10

    def get_queryset(self):
        self.reqtext = self.request.GET.get('request')
        self.reqdate = self.request.GET.get('date')
        self.reqcity = self.request.GET.get('city')
        result=[]
        if self.reqtext:
            artists =
Artist.objects.get(title__icontains=self.reqtext)

```

```

        events = events.exclude(category__id = 1)
        events = events.exclude(category__id = 2)
        events = events.exclude(category__id = 3)
        halls = halls
    Event.objects.get(title__icontains=self.reqtext)
    for art in artists:
        evs = events.filter(artist = art)
        evs = evs.exclude(category__id = 1)
        evs = evs.exclude(category__id = 2)
        evs = evs.exclude(category__id = 3)
        for ev in evs:
            if not reqdate and not self.reqcity:
                result.append(ev)
            elif not self.reqdate:
                if self.reqcity == ev.hall.city:
                    result.append(ev)
            elif not self.reqcity:
                if ev.when.date() == self.reqdate:
                    result.append(ev)
    for ev in events:
        if not self.reqdate and not self.reqcity:
            result.append(ev)
        elif not self.reqdate:
            if self.reqcity == ev.hall.city:
                result.append(ev)
        elif not self.reqcity:
            if ev.when.date() == self.reqdate:
                result.append(ev)
    for h in halls:
        evs = events.filter(hall = h)
        evs = evs.exclude(category__id = 1)
        evs = evs.exclude(category__id = 2)
        evs = evs.exclude(category__id = 3)
        for ev in evs:
            if not self.reqdate and not
self.reqcity:
                result.append(ev)
            elif not self.reqdate:
                if self.reqcity == ev.hall.city:
                    result.append(ev)
            elif not self.reqcity:
                if ev.when.date() == self.reqdate:
                    result.append(ev)

```

```

        result
result.filter(when__gte=datetime.date.today())
        result = result.order_by("when")
        return result

    def get_context_data(self, **kwargs):
        infotemplate = super(EventsList,
self).get_context_data(**kwargs)
        infotemplate['request'] = self.reqtext
        infotemplate['date'] = self.reqdate
        infotemplate['city'] = self.reqcity
        return infotemplate

class ArtistEventsList (LoginRequiredMixin, ListView):
    model = Event
    template_name = '/template/artistevents.html'
    paginate_by = 10

    def get_queryset(self):
        self.artistid = self.request.GET.get('artistid')
        art = get_object_or_404(Artist, pk=artistid)
        events = Event.objects.get(artist=art)
        events
events.filter(when__gte=datetime.date.today())
        events = events.order_by("when")
        return events

    def get_context_data(self, **kwargs):
        infotemplate = super(EventsList,
self).get_context_data(**kwargs)
        infotemplate['artistid'] = self.artistid
        return infotemplate

class HallEventsList (LoginRequiredMixin, ListView):
    model = Event
    template_name = 'hallevents.html'
    paginate_by = 10

    def get_queryset(self):
        self.halltid = self.request.GET.get('hallid')
        h = get_object_or_404(Hall, pk=artistid)
        events = Event.objects.get(hall=h)
        events
events.filter(when__gte=datetime.date.today())
        events = events.order_by("when")

```

```

        return events

    def get_context_data(self, **kwargs):
        infototemplate = super(EventsList,
self).get_context_data(**kwargs)
        infototemplate['hallid'] = self.hallid
        return infototemplate

```

Б.2 Текст файла views/orders.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from .models import Order
from .models import Event
from .models import Guest
from django.views.generic import ListView

class OrdersList (LoginRequiredMixin, ListView):
    model = Order
    template_name = '/template/orders.html'
    paginate_by = 10

    def get_queryset(self):
        orders = []
        events = Event.objects.filter(creator =
self.request.user)
        for ev in events:
            tickets=[]
            ts = tickets.filter(schemeevent__id == ev.id)
            tickets.append(ts)
            ts = tickets.filter(placeevent__id == ev.id)
            tickets.append(ts)
            for t in tickets:
                if t.order not in orders:
                    orders.append(t.order)
            orders = orders.order_by("-when")
        return orders

class VisitorOrdersList (LoginRequiredMixin, ListView):
    model = Order
    template_name = '/template/visitororders.html'
    paginate_by = 10

    def get_queryset(self):

```

```

        orders = super(VisitorOrdersList,
self).get_queryset()
        visitor = self.request.user
        orders = Order.objects.filter(guest__user =
visitor)

        orders = orders.order_by("-when")
        return orders

```

```

class PhoneOrdersList (LoginRequiredMixin, ListView):
    model = Order
    template_name = '/template/phoneorders.html'
    paginate_by = 10

    def get_queryset(self):
        orders = super(PhoneOrdersList,
self).get_queryset()
        self.phone = self.request.GET.get('phone')
        guests = Guest.objects.filter(phone = self.phone)
        resultorders = []
        for g in guests:
            r = orders.filter(guest = g)
            resultorders.append(r)
        resultorders = resultorders.order_by("-when")
        return resultorders

    def get_context_data(self, **kwargs):
        infototemplate = super(EventsList,
self).get_context_data(**kwargs)
        infototemplate['phone'] = self.phone

```

```

class NumberOrdersList (LoginRequiredMixin, ListView):
    model = Order
    template_name = '/template/numberorders.html'

    def get_queryset(self):
        orders = super(NumberOrdersList,
self).get_queryset()
        self.number = self.request.GET.get('number')
        orders = orders.filter(pk = self.number)
        return resultorders

    def get_context_data(self, **kwargs):
        infototemplate = super(EventsList,
self).get_context_data(**kwargs)

```

```

infototemplate['number'] = self.number

class ReturnOrdersList (LoginRequiredMixin, ListView):
    model = Order
    template_name = '/template/returnorders.html'
    paginate_by = 10

    def get_queryset(self):
        orders = []
        events = Event.objects.filter(creator =
self.request.user)
        for ev in events:
            tickets=[]
            ts = tickets.filter(schemeevent__id == ev.id)
            tickets.append(ts)
            ts = tickets.filter(placeevent__id == ev.id)
            tickets.append(ts)
            for t in tickets:
                if t.order not in orders:
                    o = t.order
                    if o.state == 0:
                        orders.append(o)
        orders = orders.order_by("-when")
        return orders

```

Б.3 Текст файла models.py

```

from django.db import models

class Artist (models.Model):
    title = models.CharField(max_length=100, unique=True)
    about = models.TextField()
    site = models.CharField(max_length=255, unique=True)
    spotify = models.CharField(max_length=255, unique=True)
    deezer = models.CharField(max_length=255, unique=True)
    youtubemusic = models.CharField(max_length=255,
unique=True)
    youtube = models.CharField(max_length=255, unique=True)
    photo = models.CharField(max_length=255)

    def __str__(self):
        return self.title

```

```

class Meta:
    ordering = ['-title']

class Event (models.Model):
    title = models.CharField(max_length=255)
    artist = ForeignKey (Artist, on_delete=models.CASCADE)
    hall = ForeignKey (Hall, on_delete=models.CASCADE)
    organizer = ForeignKey (Organizer,
on_delete=models.CASCADE)
    type = models.BooleanField()
    when = models.DateTimeField()
    about = models.TextField()
    category = ForeignKey (Category,
on_delete=models.CASCADE)
    photo = models.CharField(max_length=255)
    creator = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
    )

    def __str__(self):
        return self.title

class Meta:
    ordering = ['-when']

class Price (models.Model):
    categoryprice = models.IntegerField ()
    price = models.FloatField ()

class Meta:
    ordering = ['-categoryprice']

class PlaceEvent (models.Model):
    price = ForeignKey (Price, on_delete=models.CASCADE)
    event = ForeignKey (Event, on_delete=models.CASCADE)
    place = ForeignKey (Place, on_delete=models.CASCADE)

class SchemeCategory (models.Model):
    scheme = ForeignKey (Scheme, on_delete=models.CASCADE)
    coordinates = models.CharField(max_length=255)

```

```

class SchemeEvent (models.Model):
    price = ForeignKey (Price, on_delete=models.CASCADE)
    event = ForeignKey (Event, on_delete=models.CASCADE)
    schemecategory      =      ForeignKey      (SchemeCategory,
on_delete=models.CASCADE)
    quantity = models.IntegerField()
    free = models.IntegerField()

class Organizer (models.Model):
    title = models.CharField(max_length=100, unique=True)
    site = models.CharField(max_length=255)
    phone = models.CharField(max_length=13)
    address = models.CharField(max_length=255)

    class Meta:
        ordering = ['title']

class Place (models.Model):
    scheme = ForeignKey (Scheme, on_delete=models.CASCADE)
    wposition = models.IntegerField ()
    hposition = models.IntegerField()
    placetype      =      ForeignKey      (PlaceType,
on_delete=models.CASCADE)
    type = models.IntegerField()
    row = models.IntegerField()
    inrow = models.IntegerField()

class Guest (models.Model):
    firstname = models.CharField(max_length=100)
    lastname = models.CharField(max_length=100)
    phone = models.CharField(max_length=13)
    email = models.CharField(max_length=100)
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
    )

    class Meta:
        ordering = ['-lastname']

class Office (models.Model):
    title = models.CharField(max_length=100)

```

```

address = models.CharField(max_length=255)
phone = models.CharField(max_length=13)

class Meta:
    ordering = ['-title']

class PlaceType (models.Model):
    title = models.CharField(max_length=100)

    class Meta:
        ordering = ['-title']

class Hall (models.Model):
    title = models.CharField(max_length=100, unique=True)
    site = models.CharField(max_length=255)
    phone = models.CharField(max_length=13)
    address = models.CharField(max_length=255)
    photo = models.CharField(max_length=255)
    about = models.TextField()
    city = models.CharField(max_length=30)

class Scheme (models.Model):
    title = models.CharField(max_length=100, unique=True)
    hall = ForeignKey (Hall, on_delete=models.CASCADE)
    picture = models.CharField(max_length=255)
    type = models.BooleanField()
    wplaces = models.IntegerField()
    hplaces = models.IntegerField()

class EventCategory (models.Model):
    title = models.CharField(max_length=100)

    class Meta:
        ordering = ['-title']

class Order (models.Model):
    price = models.FloatField()
    when = models.DateTimeField()
    guest = ForeignKey (Guest, on_delete=models.CASCADE)
    type = models.BooleanField()
    office = ForeignKey (Office, on_delete=models.CASCADE)

```

```
message = models.CharField(max_length=255)
code = models.CharField(max_length=20)

class Ticket (models.Model):
    schemeevent = ForeignKey (SchemeEvent,
on_delete=models.CASCADE)
    placeevent = ForeignKey (PlaceEvent,
on_delete=models.CASCADE)
    price = models.FloatField()
    type = models.BooleanField()
    order = ForeignKey (Order, on_delete=models.CASCADE)
```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
кафедра програмних засобів

Дипломна кваліфікаційна робота

**Програмне забезпечення продажу
квитків на заходи в сфері культури**

Виконав:

студент групи КНТ-218сп

Д. В. Чорний

Керівник:

к.т.н, доцент

С. Ю. Скрупський

2021

Рисунок В.1 – Слайд № 1

2

Об'єкт роботи – процес продажу квитків на заходи в сфері культури.

Предмет роботи – програмне забезпечення продажу квитків на заходи в сфері культури.

Мета роботи – розроблення програмного забезпечення продажу квитків на заходи в сфері культури для підтримки роботи квиткового оператора.

Рисунок В.2 – Слайд № 2

Постановка завдань

3

Завдання роботи:

- визначення вимог до програми;
- проектування програми;
- реалізація програми.

Вимоги до програмного забезпечення:

- можливість визначення схеми залу за мапою для продажу квитків за категоріями;
- можливість визначення сіткової схеми залу з завданням конкретних місць;
- підтримка роботи неавторизованого користувача;
- продаж квитків на декілька подій в одному замовленні;
- гнучке визначення категорій вартості квитків при продажі квитків на конкретні місця;
- можливість повного редагування замовлення з обмеженим доступом;
- можливість пошуку замовлень за номером телефону замовника.

Рисунок В.3 – Слайд № 3

Вибір вебфреймворку

4

Показник	Django	Flask
Мова програмування	Python	Python
Вбудоване об'єктно-реляційне відображення	+	–
Вбудована адміністративна панель	+	–
Інтеграція форм	+	–
Вбудовані засоби безпеки	+	+-
Створення динамічних вебсторінок	+	+

Рисунок В.4 – Слайд № 4



Рисунок В.5 – Слайд № 5

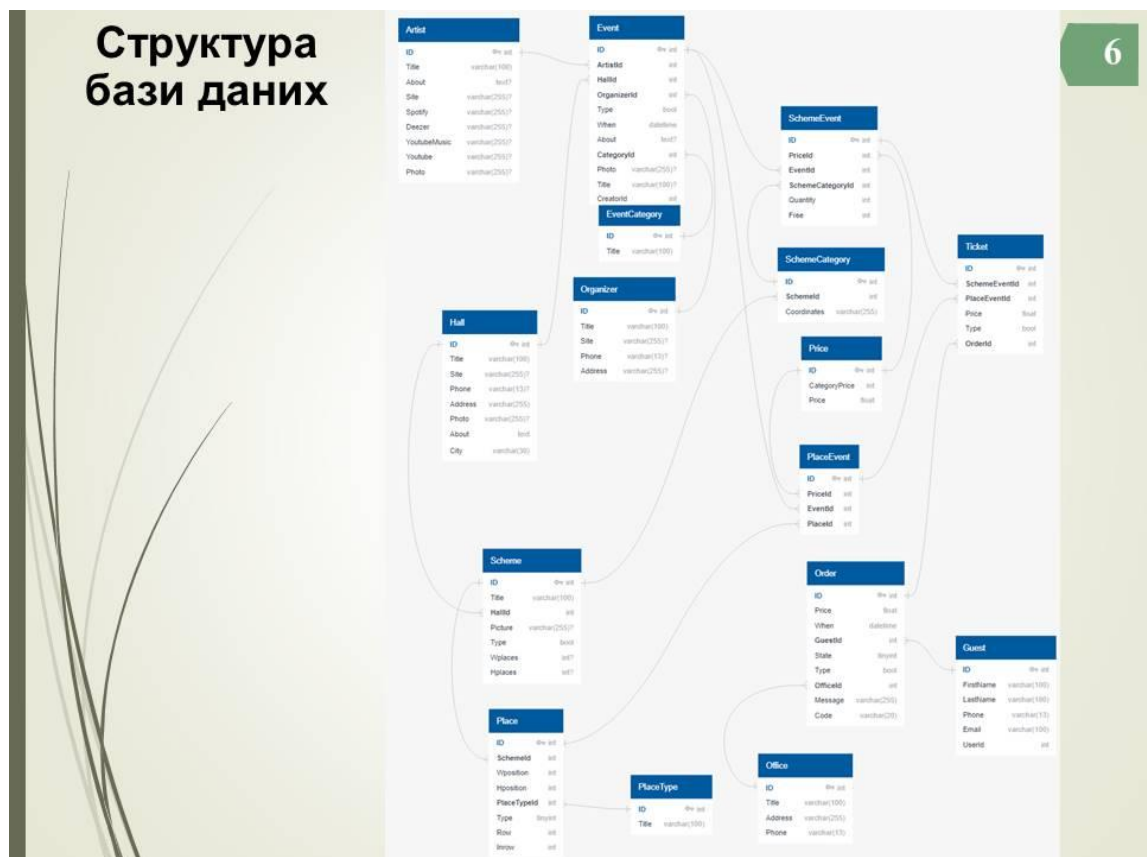


Рисунок В.6 – Слайд № 6

Схема структури програми

7

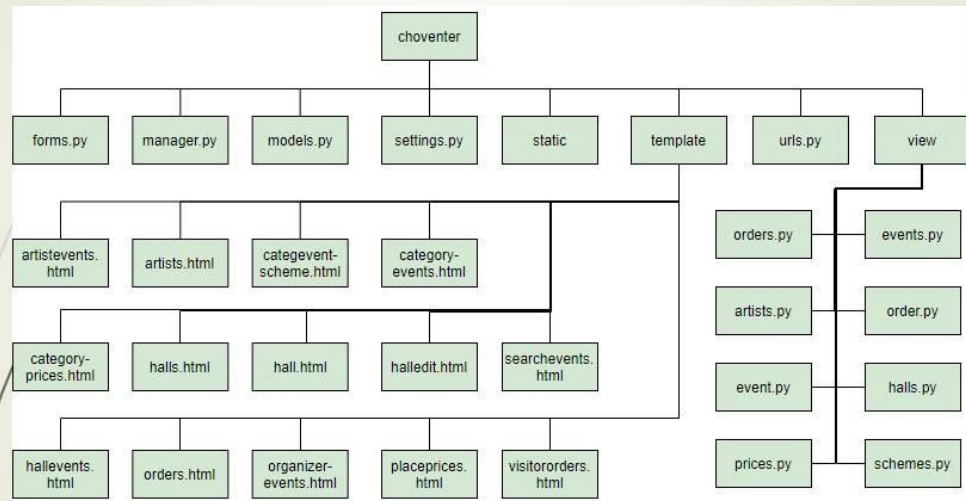


Рисунок В.7 – Слайд № 7

Діаграма станів замовлення

8

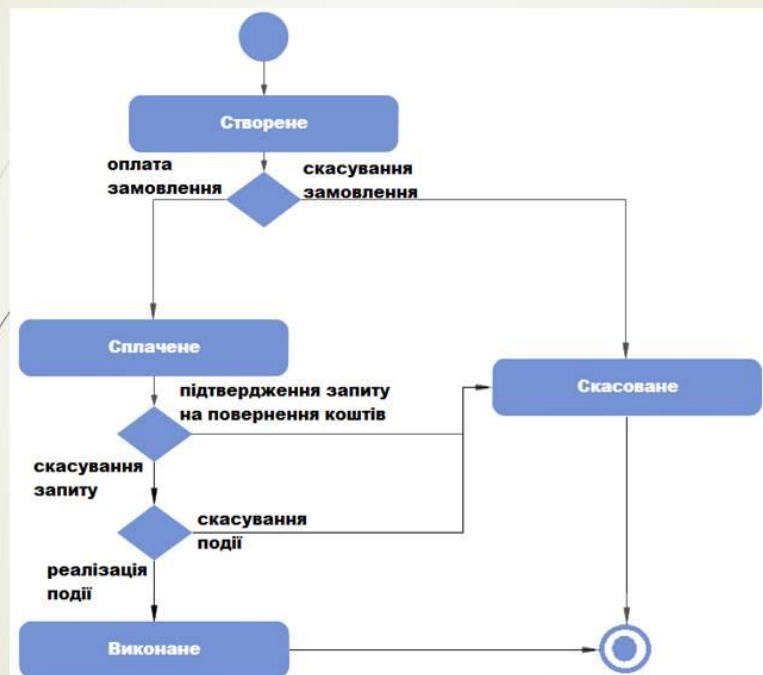


Рисунок В.8 – Слайд № 8

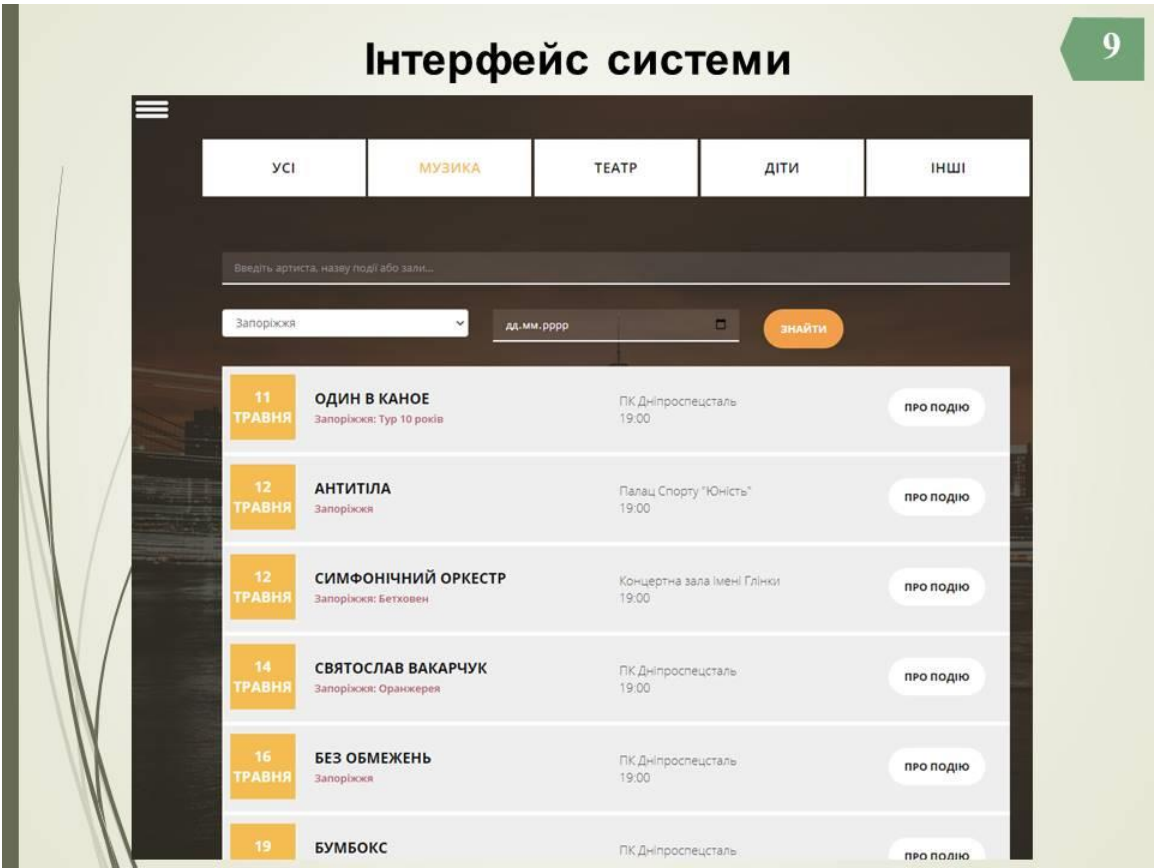


Рисунок В.9 – Слайд № 9

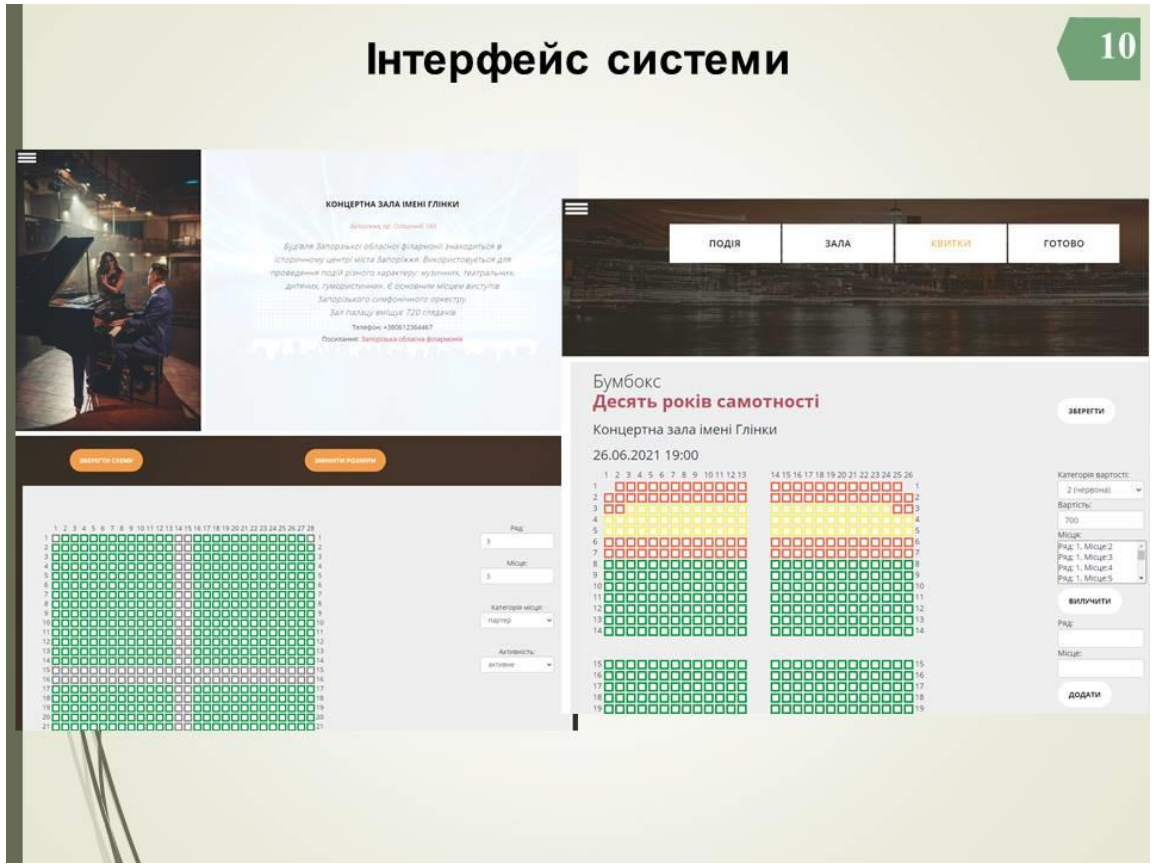
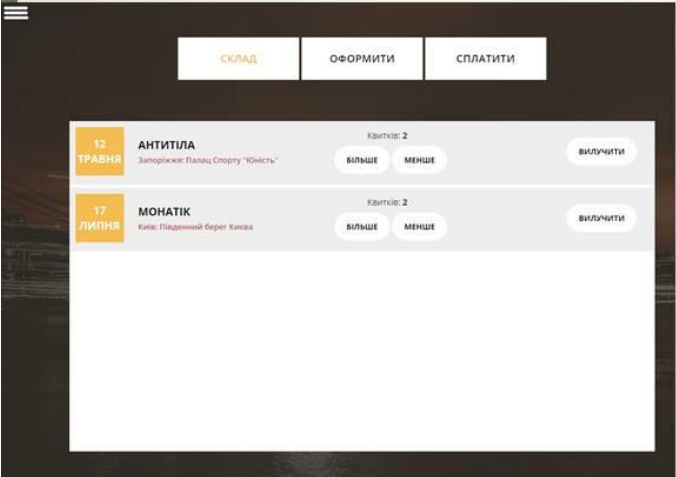


Рисунок В.10 – Слайд № 10

Інтерфейс системи

11



The screenshot shows a web application interface for ticket sales. At the top, there are three buttons: "СКЛАД", "ОФОРМИТИ", and "СПЛАТИТИ". Below these, there is a list of events. The first event is "АНТИТИЛА" on May 12th at the Zaporizhzhia Sports Palace, with 2 tickets available. The second event is "МОНАТІК" on July 17th at the Kyiv Pridnipro Riverbank, also with 2 tickets available. To the right of the event list, there is a detailed view for the "АНТИТИЛА" event, showing the artist "Billie Eilish", the album "When We All Fall Asleep, Where Do We Go Tour", the venue "Палац 'Україна'", the address "Київ, вул. Велика Васильківська, 103", the date and time "16.04.2022 20:00", the row "Ряд: 23", the seat "Місце: 11", and the price "Вартість: 1300 грн.". A QR code is displayed at the bottom right, with a warning "НЕ ДЕМОНСТРУВАТИ НІКОМУ" (Do not show to anyone).

Рисунок В.11 – Слайд № 11

Висновки

12

У роботі розглянуто проблему розроблення програмного забезпечення продажу квитків на заходи в сфері культури. Для цього було виконано аналіз проблеми, визначено позитивні характеристики таких систем відносно роботи над заходами в сфері культури, визначено вимоги до програми та розглянуто основні функціональні можливості таких систем.

Проаналізовано фреймворки розробки вебзастосунків мовою Python, в результаті чого обрано вебфреймворк Django для реалізації. Виділено вимоги до програми, виділено необхідні для реалізації функціональні можливості. В якості ролей користувачів визначено неавторизованого користувача, відвідувача, менеджера та адміністратора. Виконано моделювання сценаріїв роботи користувачів з програмою. Виконано проектування структури даних на основі системи керування базами даних MySQL з побудовою схеми бази даних.

Реалізована програма призначена для організації платформи для операторів з продажу, організаторів подій, розповсюджувачів квитків на ці події та відвідувачів подій. Організатори подій надають контент для внесення даних у програму. Розповсюджувачі квитків можуть реалізовувати безпосередньо свій бізнес через програму. Відвідувачі подій можуть оформляти та керувати замовленнями на покупку квитків.

Усі завдання дипломної кваліфікаційної роботи повністю виконано.

Рисунок В.12 – Слайд № 12