

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Платформа для бронювання та придбання квитків»

Виконав студент групи ПЗп-20-1
Бойко Дмитро Євгенійович

Керівник роботи:
ЛЕВЧЕНКО Антон Юрійович

Консультант:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
МОРОЗОВ Андрій Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«02» вересня 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БОЙКО Дмитро Євгенійович**

Керівник роботи: **ЛЕВЧЕНКО Антон Юрійович**

Тема роботи: **«Платформа для бронювання та придбання квитків»**,
затверджена Наказом закладу вищої освіти від **«02» вересня 2024 р., №433/с.**

Вихідні дані для роботи: Методи та засоби розробки веб-сайту за визначеною предметною областю. Об'єктом дослідження є застосування технологій для реалізації онлайн-сервісу замовлення та бронювання квитків. Перелік сервісів для реалізації, розклад сеансів, вартість квитків, кількість місць у залі, перелік додаткових опцій замовлення, технології реалізації, вимоги до сумісності програмного продукту.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М.	02.09.2024р.	02.09.2024р.
2	Єфремов Ю.М.	02.09.2024р.	02.09.2024р.
3	Єфремов Ю.М.	02.09.2024р.	02.09.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу онлайн-платформи з бронювання та продажу квитків та пояснювальної записки.

Пояснювальна записка до випускної роботи містить 56 сторінок, 30 ілюстрацій, 6 таблиць, 4 додатки та 17 літературних джерел.

Метою роботи є розробка онлайн-платформи для бронювання та придбання квитків, який надає користувачам зручний та простий спосіб придбати квитки на різні події.

В роботі було визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Обґрунтовано використання архітектури додатку.

Проведено обґрунтування актуальності теми, аналіз методів та алгоритмів, а також досліджено можливості обраних технологій та їх актуальність. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об'єктної структури системи. Також наводяться результати тестування системи.

Ключові слова: аналіз даних, модель класифікації, онлайн-платформа, замовлення квитків, API, PHP, WEB-ДОДАТОК, САЙТ, СУБД, SPA, PHPMYADMIN, JS, SQL.

					ІПЗ.КР.Б – 121 – 24 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Д.С. Бойко			Платформа для бронювання та купівлі квитків	Літ.	Арк.	Аркуші	
Керівник		А.Ю. Левченко							
Консультант		Ю. М. Єфремов					3	70	
Рецензент		А.В. Морозов				Житомирська політехніка, група ІПЗп-20-1			
Зав. каф.		Т.В. Вакалюк							
					Пояснювальна записка				

ABSTRACT

The bachelor's final thesis consists of a software package for an online ticket booking and sales platform and an explanatory note.

The explanatory note to the thesis contains 56 pages, 30 illustrations, 6 tables, 4 appendices and 17 references.

The aim of the work is to develop an online ticket booking and purchase platform that provides users with a convenient and easy way to purchase tickets for various events.

The paper identifies the main tasks for the development of the system and analyses analogues of the developed system. The use of the application architecture is substantiated.

The relevance of the topic is substantiated, methods and algorithms are analysed, and the capabilities of the selected technologies and their relevance are investigated. Scenarios of the software system operation, the general structure of the software system, a description of the algorithms of interaction of individual modules of the system, a description of the object structure of the system are presented. The system testing results are also presented.

Keywords: data analysis, classification model, online platform, ticket ordering, API, PHP, WEB APP, SITE, DBMS, SPA, PHPMYADMIN, JS, SQL.

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПЛАТФОРМИ ДЛЯ БРОНЮВАННЯ ТА КУПІВЛІ КВИТКІВ “VOZBO”	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Вибір архітектури веб-орієнтованої системи до обраної теми	13
1.4 Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів.....	15
Висновки до першого розділу.....	16
РОЗДІЛ 2 ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ БРОНЮВАННЯ ТА КУПІВЛІ КВИТКІВ “VOZBO”	18
2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи.....	18
2.2 Розробка структури бази даних системи	28
2.3 Проектування та реалізація алгоритмів роботи системи	31
2.4 Реалізація онлайн-сервісу з бронювання та купівлі квитків “VOZBO”	37
Висновки до другого розділу:	40
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-ПЛАТФОРМИ БРОНЮВАННЯ ТА ПРОДАЖУ КВИТКІВ “VOZBO”	41
3.1 Порядок встановлення та налаштування параметрів системи	41
3.3 Тестування роботи програмного продукту	51
Висновки до третього розділу	52
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	54
ДОДАТКИ.....	56

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

REST – Representational state transfer

SPA – Single-Page Application

URI – Uniform Resource Identifier

БД – База даних

ІС – Інформаційна система

СУБД – Система управління базами даних

ВСТУП

Актуальність теми розробки платформи для бронювання та купівлі квитків обумовлена кількома ключовими факторами. Сьогодні спостерігається стрімке зростання популярності онлайн-сервісів, які спрощують вирішення багатьох повсякденних завдань. Бронювання квитків через інтернет стає дедалі зручнішим способом відвідування заходів, адже дозволяє заощаджувати час та уникати черг. Крім того, розвиток сучасних технологій, таких як мобільні додатки, штучний інтелект та великі дані, відкриває можливості для вдосконалення таких платформ, роблячи їх більш інтуїтивними та персоналізованими.

Для бізнесу, що займається організацією подій, такі платформи є важливим інструментом для ефективного управління продажами, аналізу попиту та взаємодії з клієнтами. У контексті глобальних тенденцій, таких як зростання попиту на безконтактні сервіси після пандемії, онлайн-платформи набувають ще більшої значущості, оскільки забезпечують безпечний та гнучкий доступ до послуг. Крім того, конкурентне середовище вимагає від компаній постійного вдосконалення своїх продуктів та послуг, що робить розробку інноваційної платформи важливою складовою їх успіху на ринку.

Таким чином, створення платформи для бронювання та купівлі квитків відповідає сучасним потребам користувачів і ринку, що робить цю тему дипломного проекту актуальною та перспективною.

Метою дослідження є розробка алгоритмів роботи та реалізація програмного комплексу веб-орієнтованої системи для бронювання та продажу квитків на різні події.

Досягнення зазначеної мети включає виконання таких завдань:

- аналіз проблемних питань потенційних користувачів та пошук оптимальних рішень для їх вирішення;
- визначення архітектури та загальної структури системи;
- створення інформаційного та математичного забезпечення системи;
- перевірка та тестування функціональності розробленої технології.

Об'єктом дослідження є процес побудови моделі для покращення та полегшення бронювання та замовлення квитків.

Предметом дослідження є використання веб-орієнтованих технологій для розробки онлайн-сервісу бронювання та купівлі квитків "BOZBO".

Методи дослідження У кваліфікаційній роботі було застосовано такі методи: спостереження та інтерв'ювання, візуальне програмування, математичне моделювання та веб-програмування.

Практичне значення роботи полягає в проектуванні та програмній реалізації онлайн-платформи для бронювання та купівлі квитків під назвою "BOZBO".

Розроблена інформаційна технологія сприяє підвищенню якості обслуговування артистів, продюсерів, організаторів та інших учасників процесу організації подій. Вона підвищує ефективність, знижує вплив людського фактора та допомагає уникнути помилок. Проведене тестування показало, що програмний продукт є функціональним і готовим до використання.

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПЛАТФОРМИ ДЛЯ БРОНЮВАННЯ ТА КУПІВЛІ КВИТКІВ “VOZBO”

1.1 Постановка задачі

Постановка задачі полягає в конкретному визначенні завдань і цілей проекту. У випадку онлайн-платформи для бронювання та купівлі квитків "Vozbo", постановка задачі включає наступні етапи:

1. Розробка інтерфейсу користувача: Створення гарного та інтуїтивно зрозумілого інтерфейсу, що дозволяє користувачам легко знаходити інформацію про вистави, концерти, спортивні події та здійснювати замовлення квитків.
2. Розробка функціональності замовлення квитків: Реалізація можливості вибору події, локації, перегляду відгуків та оплати квитків онлайн.
3. Розробка адміністративної панелі: Розробка функціоналу та інтерфейсу для адміністратора, що дозволяє керувати інформацією про події, локації. Включає можливості додавання та редагування, а також керування доступністю події та локацій на сайті.
4. Інтеграція з платіжними системами: Забезпечення можливості зручної, швидкої та безпечної оплати квитків онлайн.
5. Розробка користувацької системи управління: Впровадження механізмів реєстрації та авторизації користувачів, а також забезпечення захисту й конфіденційності їхніх персональних даних.
6. Тестування та оптимізація: Проведення тестування системи для перевірки її функціональності, стабільності та продуктивності. Усунення помилок і вдосконалення системи на основі отриманих результатів тестування. Постановка задачі детально описує обсяг необхідних робіт та визначає основні цілі, які потрібно досягти під час розробки онлайн-платформи бронювання та купівлі квитків на публічні події “Vozbo”

1.2 Аналіз аналогів програмного продукту

Аналіз аналогів програмного продукту на прикладі сайтів "Karabas" та "Concert.ua" може надати цінну інформацію про існуючі рішення у сфері онлайн-сервісів замовлення квитків для публічних подій. Давайте розглянемо деякі особливості цих сервісів та їхніх веб-орієнтованих систем.

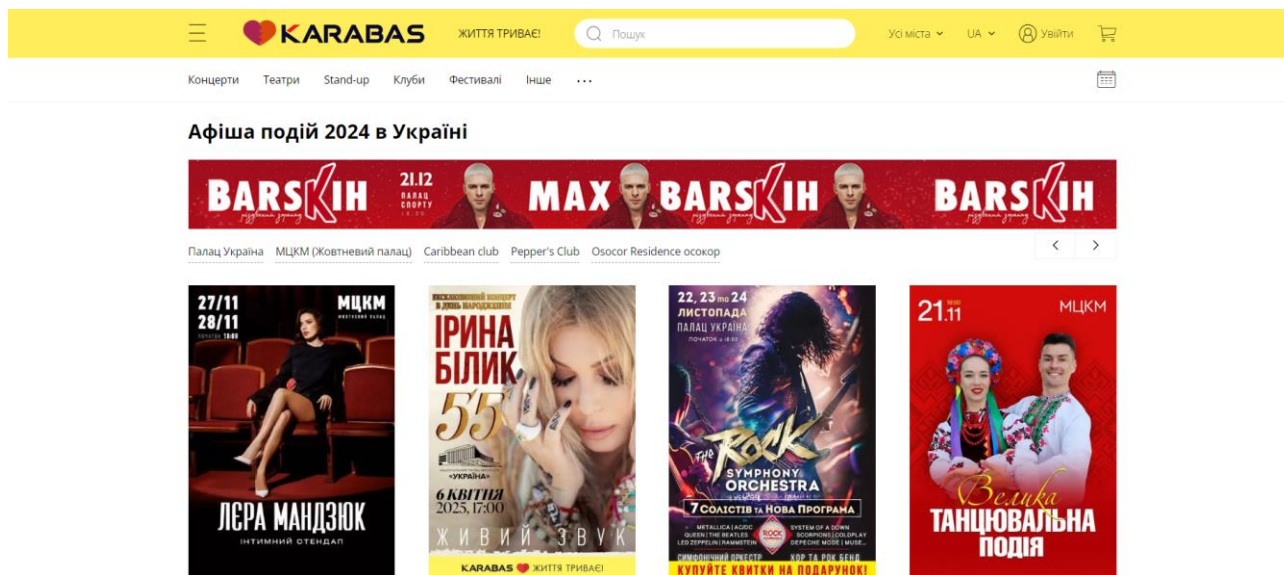


Рисунок 1.1 – Приклад онлайн сервісу “Karabas”

"Karabas" є одним з найпопулярніших в Україні веб-сервісів з продажу квитків на розважальні заходи: концерти, шоу, театральні постановки, циркові вистави, фестивалі, тощо... (рис. 1.1).

Основні особливості їхньої системи включають:

1. Широкий функціонал: Веб-сайт "Karabas" має досить розширений функціонал, де користувачі за допомогою різних фільтрів можуть швидко знайти потрібну подію, переглянути дату та замовити квитки.
2. Різноманітні послуги: Крім замовлення квитків, система "Karabas" також пропонує додаткові послуги, такі як вибір місць у залі, можливість обрати залу, тощо...

Потенційні мінуси систем замовлення квитків онлайн у веб-додатку "Karabas":

- 1. Застарілий інтерфейс, що може викликати в користувача складнощі із пошуком необхідного того чи іншого елементу функціоналу та не найкраще візуальне враження під час використання веб-сервісу.
- 2. Іноді можуть виникати проблеми з онлайн-платежами, такі як технічні помилки під час транзакцій. Відсутність технічної підтримки.

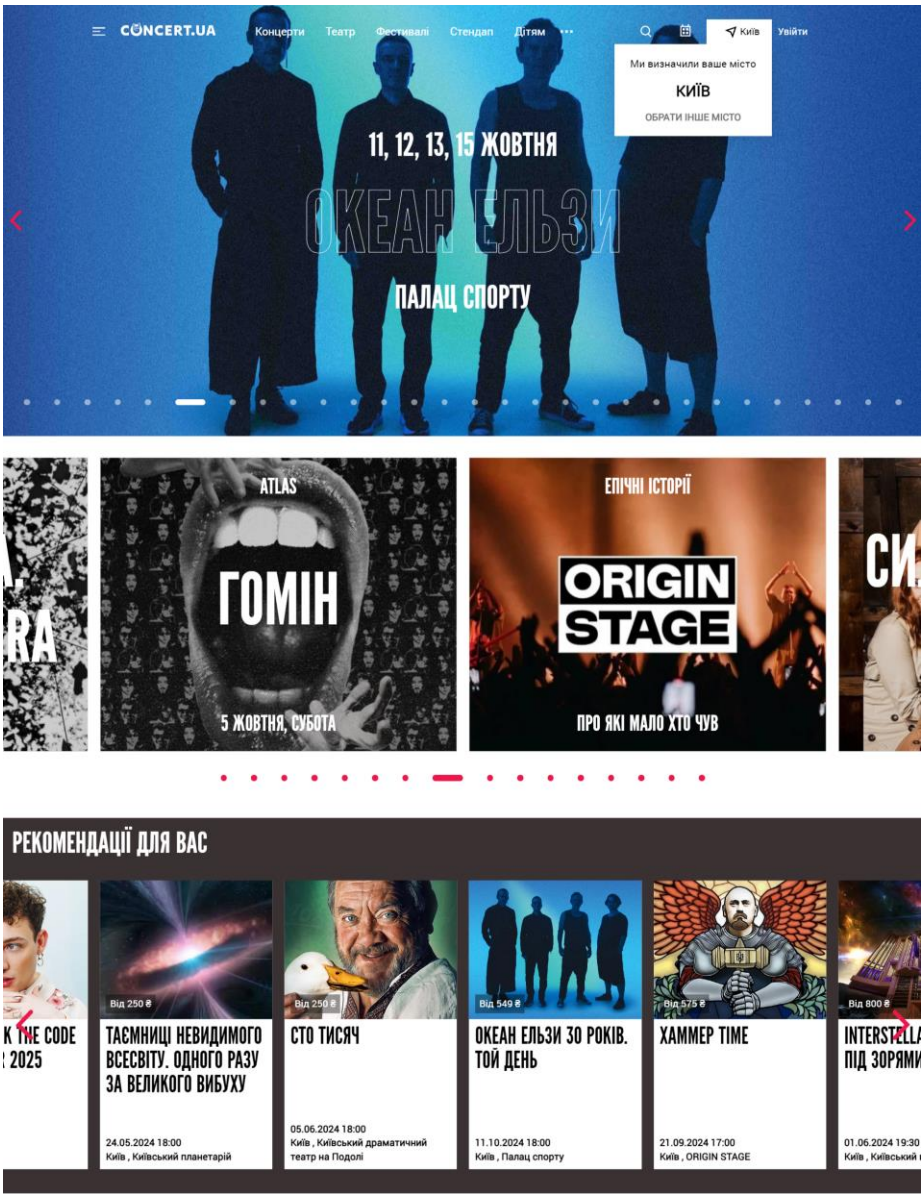


Рисунок 1.2 – Приклад онлайн сервісу “Concert.ua”

"Concert.ua", (рис. 1.2), також є популярним івент-оператором в Україні, який надає можливість замовляти квитки онлайн на різні події через свій веб-сайт. Особливості системи "Concert.ua" включають:

1. Інтерфейс. Сайт має приємний та інтуїтивно зрозумілий інтерфейс, який легко використовувати навіть новим користувачам.
2. Широкий вибір подій. На платформі представлений великий асортимент заходів: концерти, вистави, спортивні події, фестивалі тощо, що дозволяє користувачам знайти щось на свій смак.
3. Підтримка клієнтів. Concert.ua має службу підтримки, яка швидко й якісно реагує на запити користувачів сайту і допомагає у вирішенні будь яких проблем.

Потенційні мінуси систем замовлення квитків онлайн у кінотеатрах "Concert.ua":

1. Сервісний збір. Комісії за обробку квитків можуть збільшити загальну вартість покупки, що не завжди одразу помітно користувачам сервісу.
2. Обмежений вибір місць та подій у локальних місцях. Деякі події можуть мати обмежену кількість доступних місць або відсутність можливості вибору конкретного місця в залі. Не дивлячись на те, що Concert.ua покриває велику кількість публічних подій, в менших містах України вибір може бути обмеженим.

Узагалі, обидва онлайн-сервіси "Karabas" та "Concert.ua" надають зручні та функціональні системи для замовлення квитків на події, дозволяючи користувачам швидко знайти необхідний захід, придбати квитки, обрати місце, скористатися додатковими послугами та отримати персоналізоване обслуговування. Аналіз цих аналогів може бути корисним для врахування їхніх переваг і недоліків при створенні веб-системи замовлення квитків для нашого проєкту "BOZBO".

1.3 Вибір архітектури веб-орієнтованої системи до обраної теми

Вибір архітектури веб-орієнтованої системи для проєкту "Онлайн-сервіс бронювання та купівлі квитків "BOZBO"" є ключовим етапом розробки. Продукт повинен бути якісним, зрозумілим, доступним і простим у використанні. Основна мета архітектури — забезпечити структурованість, надійність, можливість масштабування та керування програмного рішення. Це допоможе розробникам краще розуміти взаємодію між компонентами системи та ефективно вирішувати складні завдання, пов'язані з розробкою та підтримкою ПЗ.

В (рис. 1.3) представлено порівняння архітектур Single-Page Application (SPA) та клієнт-серверної моделі. Архітектура SPA [18] є відмінним вибором для веб-системи замовлення квитків, оскільки вона забезпечує швидкий та зручний користувацький інтерфейс, дозволяючи здійснювати взаємодію без перезавантаження сторінок.

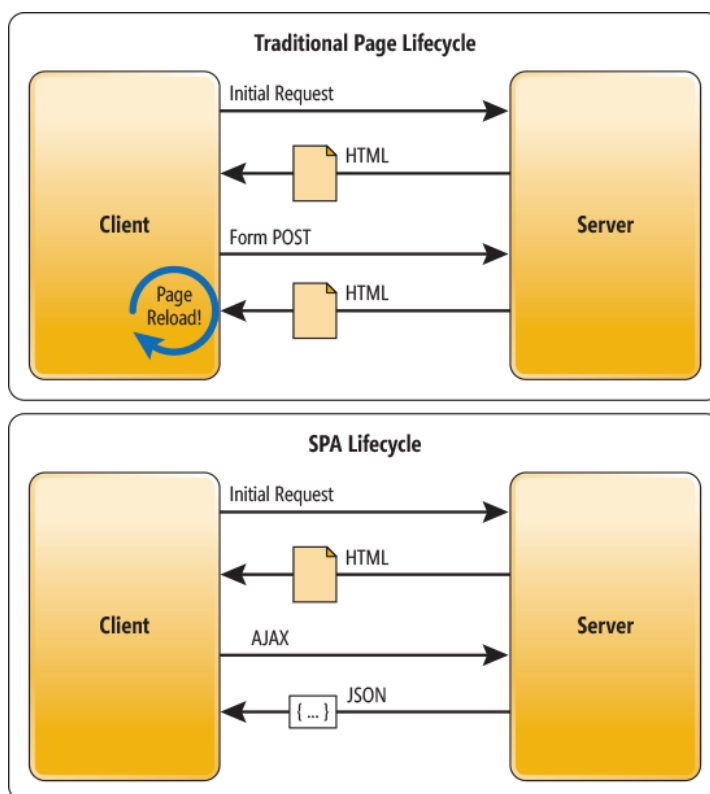


Рисунок 1.3 – Порівняння та загальний вигляд SPA та Клієнт-серверної архітектури

Основними компонентами такої архітектури є:

– Front-end (клієнтська частина): Цей додаток працює на стороні користувача та використовує HTML, CSS і JavaScript. SPA[18] завантажується один раз під час першого відкриття сторінки, після чого реагує на дії користувача та оновлює дані без необхідності повного перезавантаження сторінки, що робить взаємодію більш плавною.

– Back-end (серверна частина): Відповідає за обробку запитів від клієнтської частини, здійснює такі операції, як обробка замовлень, проведення оплат, робота з базою даних тощо. Після обробки запитів сервер повертає відповіді у форматі JSON або XML.

– API (Application Programming Interface): Набір функцій і протоколів, що забезпечує зв'язок між клієнтом і сервером. API дозволяє SPA[18] надсилати запити до сервера та отримувати відповідні дані.

– Дані: SPA використовує базу даних для зберігання інформації про фільми, розклад сеансів, наявність квитків тощо. Дані отримуються через API для відображення актуальної інформації користувачам.

Переваги архітектури SPA:

– Можливість створення багатофункціональних додатків без потреби у перезавантаженні сторінки.

– Швидка і плавна взаємодія завдяки відсутності повного перезавантаження сторінок.

– Висока масштабованість і зручність розробки, адже розподіл логіки між клієнтською та серверною частинами дозволяє паралельно розробляти фронтенд і бекенд.

– Зручна навігація та інтерфейс, що дозволяють користувачам швидко переходити між різними функціями.

1.4 Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів

Інструментальні засоби для реалізації програмного продукту включають набір програмних інструментів і технологій, які застосовуються для розробки, тестування та впровадження системи.

Для розробки було обрано редактор PhpStorm, оскільки компанія JetBrains надає студентам безкоштовну підписку.

Мови програмування: Для створення веб-системи використовувалися JavaScript і PHP. JavaScript[12] був основною мовою для фронтенд-частини, а PHP[9] використовувався для бекенду.

Бази даних: Інформація про фільми, сеанси, зали та інші дані зберігалася в реляційній базі даних MySQL[3].

OpenServer: Це зручне серверне середовище для роботи з PHP, MySQL та Apache, яке містить усі необхідні компоненти для запуску веб-додатків, такі як веб-сервер, база даних та PHP-інтерпретатор.

Системи керування версіями: Для контролю версій коду та спільної роботи над проєктом використовувалася система керування версіями Git.

Системні вимоги:

Платформа повинна задовільняти вимогам на комп'ютері, що знаходиться в наступній конфігурації:

- 4 ГБ оперативної пам'яті
- Процесор з тактовою частотою від 1 GHz
- Операційна система будь-яка, що підтримує використання інтернету
- Браузер
- HTML, JS, SQLITE3

Вимоги до СУБД та доступу до даних:

- У ядрі системи повинна бути представлена СУБД реляційного доступу.
- Фото повинні зберігатися на хмарному сховищі.

Вимоги до системного забезпечення:

- Операційна система: оновлені версії Windows, такі як Windows 10 або Windows 11, актуальні версії macOS, різні дистрибутиви Linux, такі як Ubuntu, Fedora, Debian тощо.
- Веб-браузери: останні версії популярних браузерів, таких як Google Chrome, Mozilla Firefox, Microsoft Edge та Safari.

Вимоги до серверного обладнання:

- обсяг оперативної пам'яті – 8 GB;
- обсяг жорсткого диску від 250 GB;
- процесор з частотою від 3GHz;
- node.js версії 18 і вище;

Висновки до першого розділу

Обґрунтовано актуальність теми, яка полягає в необхідності створення інформаційної системи для онлайн-замовлення квитків, що відповідає потребам сучасних користувачів, які цінують зручність та швидкість. Веб-система полегшить процес придбання квитків для користувачів і підвищить ефективність роботи кінотеатру.

Проведено аналіз аналогічних програмних продуктів на прикладі сервісів Karabas та Concert.ua, які пропонують замовлення квитків на різноманітні заходи. Наприклад, Concert.ua пропонує широкий вибір фільмів, але має обмежені додаткові послуги та може бути незручним при виборі місць. Cinema.ua має обмежений вибір фільмів і менш розвинену функціональність,

що створює можливість розробити конкурентоспроможний продукт з більшим функціоналом та ширшим вибором фільмів.

Виконано аналіз та обґрунтовано вибір архітектури веб-системи для цього проєкту. Архітектура SPA (Single-Page Application)[18] була обрана як оптимальний варіант завдяки її здатності забезпечити швидкий і зручний користувацький інтерфейс, відсутність потреби в перезавантаженні сторінок і легку масштабованість.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ БРОНЮВАННЯ ТА КУПІВЛІ КВИТКІВ “BOZBO”

2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи

Розглянемо основні вимоги, яким повинна відповідати система для забезпечення безперервного виконання покладених на неї функцій, що дозволить відповідати всім необхідним стандартам, специфікаціям та іншим нормативним документам. Описано умови, яких користувач має дотримуватися для роботи з даною інформаційною системою, а також основні бізнесові переваги цієї системи. Визначено можливості системи, її обмеження та показники якості.

Бізнес-вимоги:

1. Основні цілі: покращення обслуговування клієнтів та легкий доступ до інформації про проведення різноманітних публічних заходів.
2. Можливості: перегляд подій, коментування подій, пошук подій, придбання квитків, оплата квитків.
3. Представлення проекту: проект буде реалізовано у вигляді веб-додатку, що містить зручний інтерфейс та необхідну інформацію за даною тематикою.

Вимоги користувачів:

- Гість (неавторизований користувач)

- Можливість пошуку подій
- Можливість отримати детальну інформацію про подію
- Можливість отримати детальну інформацію про локацію на якій проходить подію
- Можливість переглянути події, які плануються на певній локації
- Можливість перегляду коментарів для певної події
- Можливість перегляду чату

- Можливість авторизуватись/zareєструватись
реєструватися як клієнт або організатор/адміністратор

- Клієнт (авторизований користувач)

- Можливість клієнтів купувати квитки на певну подію, за умови їх наявності
- Можливість клієнтів залишити коментар для певної події
- Можливість клієнтів залишити повідомлення у чаті

– Організатор

- Можливість додавати подію
- Можливість редагувати інформацію про подію
- Можливість видалити власнододаний кіносеанс
- Можливість додавати локацію
- Можливість редагувати інформацію про локацію
- Можливість видалити власнододану локацію

– Адміністратор

- Можливість редагувати будь-яку подію
- Можливість редагувати будь-яку локацію
- Можливість видаляти будь-яку подію
- Можливість видалити будь-яку локацію

Характеристика об'єкту комп'ютеризації:

Користувач буде мати можливість переглянути доступні події, опис до них, на якій локації вона буде відбуватись та купити квитки. Адміністратор через веб-інтерфейс буде мати можливість редагувати, видаляти, додавати дані, користуватися всіма функціями системи.

Функціональні вимоги:

1. Підключення сторонніх API для реалізації оплати квитка.
2. Підключення пакетів для отримання готових модулів.
3. Авторизація користувачів у системі: в системі повинна бути представлена можливість реєстрації користувача, присвоєння йому ролі (клієнт, організатор, адміністратор).
4. Можливість збереження інформації: система повинна зберігати інформацію та завантажені файли для подальшого відображення.
5. Можливість безпечно проводити фінансові операції: система повинна коректно обробляти всі запити, що пов'язані із фінансами.

Нефункціональні вимоги:

1. Сприйняття

- а. Час, необхідний для ознайомлення з інструментами платформи для звичайних користувачів, становить 30-40 хвилин, а для досвідчених - не більше 15 хвилин.
- б. Система повинна надавати відповідь на звичайні запитання не більше 3 секунд, а на складні - не більше 10 секунд.
- с. Інтерфейс платформи повинен бути інтуїтивно зрозумілим для користувачів та не вимагати додаткового навчання.

2. Надійність

- а. Доступність платформи повинна бути не менше 90%, тобто час, витрачений на обслуговування, не повинен перевищувати 10% від загального часу роботи.
- б. Середній час безперервної роботи становить 58 робочих днів.
- с. Максимальна кількість помилок та дефектів в роботі платформи - 1 на 1000 запитів користувача.

3. Продуктивність

а. Система повинна забезпечувати одночасну роботу принаймні 100 користувачів, які взаємодіють зі спільною базою даних.

4. Можливість експлуатації

а. Масштабування - система повинна мати можливість збільшувати потужність із збільшенням користувачів, не порушуючи роботу платформи.

б. Оновлення версій повинно відбуватися у час, коли навантаження на платформу є мінімальним, із зупинкою роботи сайту протягом кількох секунд.

Системні вимоги:

1. Вимоги до середовища виконання:

Система повинна задовольняти зазначені вимоги на комп'ютері в наступній мінімальній комплектації:

- підтримка браузера Microsoft Edge версії 12+ або
- підтримка браузера Opera версії 30+ або
- підтримка браузера Safari версії 9+
- підтримка браузера Google Chrome версії 45+ або
- підтримка браузера Mozilla Firefox версії 38+ або

2. Вимоги до СУБД та доступу до даних:

У ядрі системи повинна бути представлена СУБД реляційного доступу;

Бізнес правила:

Система повинна відповідати всім вимогам та стандартам, які використовуються при розробці сучасних веб-застосунків на базі архітектури SPA (Single-page application). Система повинна відповідати всім стандартам інтерфейса користувачів інтернету.

Антивимоги:

- система не використовує недійсну інформацію або ту, що може завдати шкоди психіці користувачів.
- система не допускає використання інформації, що суперечить нормам моралі та законодавству.

Бізнес-логіка:

Система пропонує послуги, а саме купівлю квитків на події для полегшення обслуговування клієнтів.

Підсумовуючи, побудуємо діаграму варіантів використання (Use Case Diagram). В (рис. 2.1) зображено побудовану діаграму варіантів використання.

Неавторизований користувач має можливість зареєструватись на сайті, або перед цим переглянути доступний контент, інформацію на сайті про найближчі подій та ознайомитись з детальною інформацією, локаціями, на яких ті будуть відбуватись.

Авторизований користувач має можливість обрати подію переглянути деталі про неї, переглянути коментарі до події та після чого придбати квитки. Також організатори після авторизації можуть додавати події та локації на сайт, редагувати та видаляти власні.

Адміністратор може переглядати інформацію про всі події, редагувати та видаляти їх, також локації на яких будуть відбуватись публічні події.

Ефективним інструментом для візуалізації послідовності дій та станів програмної системи є діаграма варіантів використання. Ця діаграма дозволяє уявно зобразити робочі процеси, логіку прийняття рішень та ролей, відображаючи суть та логіку певного процесу чи функціональної системи:

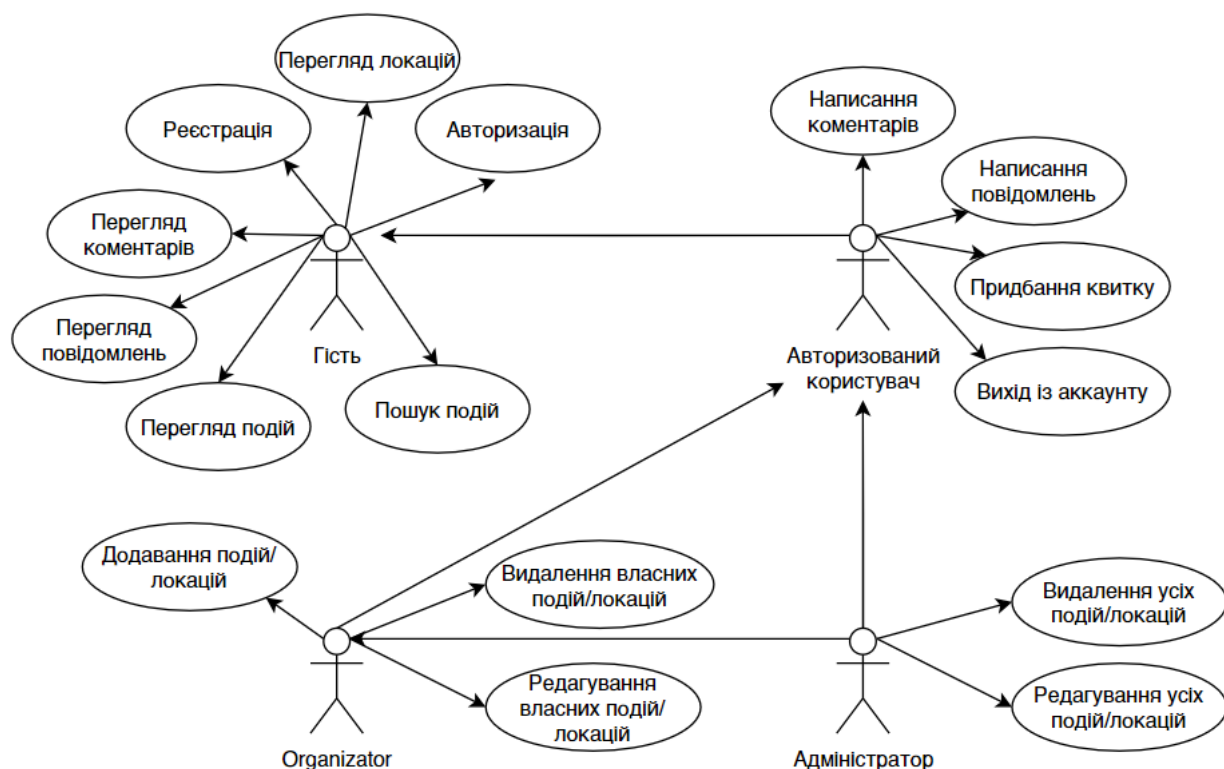


Рисунок 2.1 – Діаграма варіантів використання

Об'єктно-орієнтовану модель системи прийнято розглядати у вигляді діаграми класів (Class diagram). Для побудови даної діаграми спочатку потрібно розібратися, що вона собою представляє.

Діаграма класів – це тип діаграми статичної структури, який описує структуру системи, показуючи класи системи, їхні атрибути, операції (або методи) і зв'язки між об'єктами. В (рис. 2.2) зображено побудовану діаграму класів.

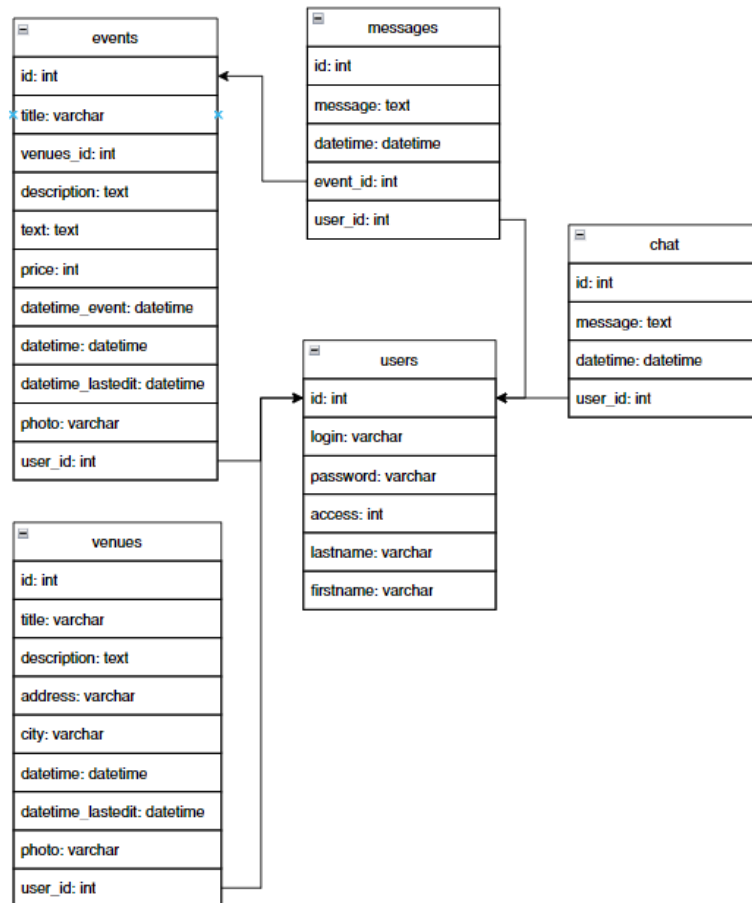


Рисунок 2.2 – Діаграма класів

Нижче наведено опис класів:

- Клас Users містить інформацію про зареєстрованого користувача та реалізовує методи, що відповідають за визначення адміністратора та роботу з паролями.
- Клас Events відповідає за публічні події (концерти, вистави, спортивні події, тощо...) та реалізовує методи, що необхідні для коректної роботи цього класу
- Клас Venues відповідає за локації, на яких відбуваються події
- Клас Chat відповідає за повідомлення в чаті сайту та реалізовує методи, що додають їх на сайт
- Клас Messages відповідає за коментарі користувачів під подіями та реалізовує методи, що додають їх на сайт

Також була розроблена діаграма, яка відображає послідовний процес додавання закладу/локації на веб-платформі.

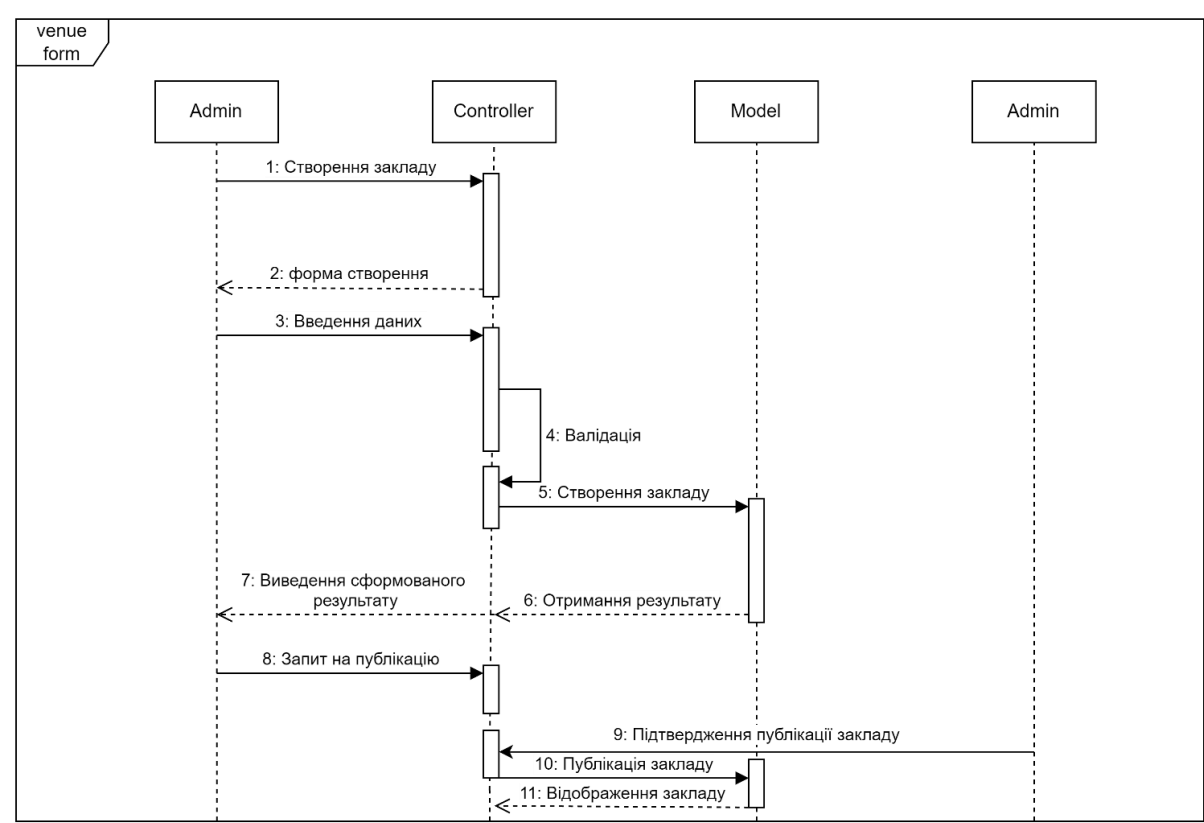
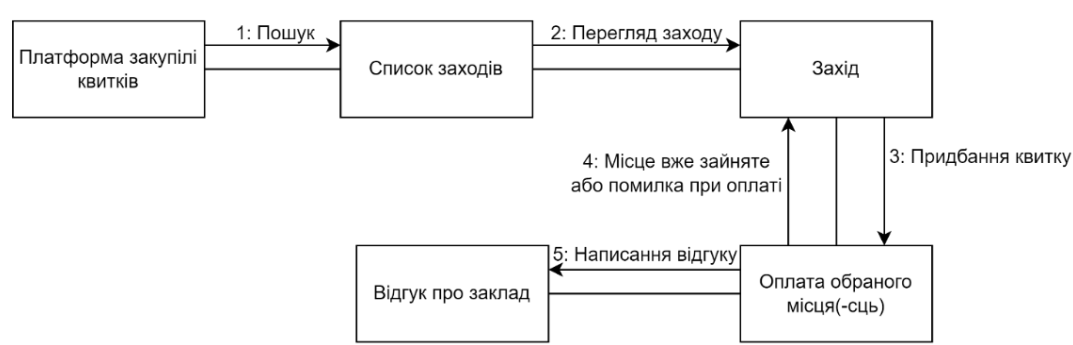


Рисунок 2.3 – Діаграма послідовності створення закладу(локації)

Також була розроблена діаграма, які відображає послідовний процес придбання квитку до закладу(кінотеатру) на веб-платформі.



Діаграма кооперації придбання квитка

Рисунок 2.4 – Діаграма кооперації придбання квитку до кінотеатру

При проектуванні важливо врахувати можливі труднощі, з якими програма може зіткнутися. Однією з таких труднощів є необхідність розширення або зміни логіки програми, що може вимагати значних зусиль та часу для вирішення. Для таких ситуацій були створені певні шаблони проектування, які надають рекомендації з розв'язання типових проблем.

За допомогою патернів проектування, технології змінюються надзвичайно швидко, вони використовуються для спрощення та розв'язання проблем. Використання цих патернів значно полегшує процес роботи з технологіями. Тому вже на етапі моделювання майбутньої системи детально продумується, які патерни будуть використовуватись.

Розглянувши три ролі, можна дійти висновку, що вони є доволі схожі й мають спільні особливості, тож було вирішено задіяти патерн проектування “Прототип” для створення ролей Адміністратор, Користувач(він ж й клієнт) й Адміністратор на базі спільної сутності Користувач.

Реалізація різних ролей користувачів за допомогою патерна «Прототип»:

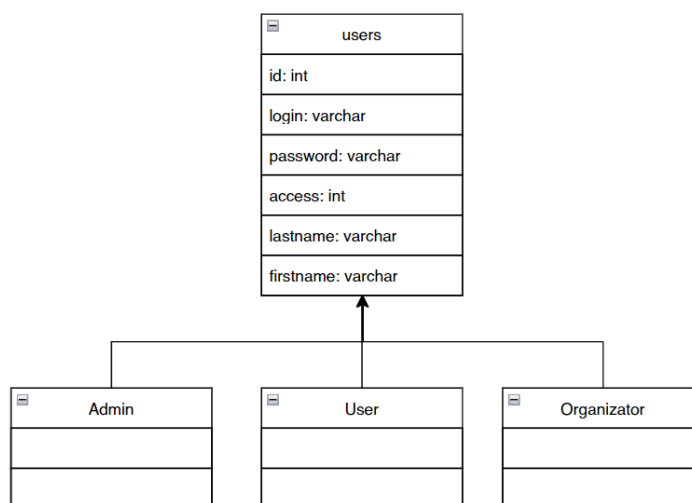


Рисунок 2.5 – Патерн “Прототип”

Для розробки системи також був застосований патерн "Unit of Work". Однак немає необхідності реалізовувати його вручну, оскільки більшість сучасних фреймворків вже мають цей патерн вбудованим. Патерн "Unit of Work" відстежує всі зміни в об'єктах і дозволяє виконати їх як єдину транзакцію.

Цей патерн, що використовується для управління транзакціями та змінами даних у реляційних базах, групує кілька операцій у межах однієї транзакції, забезпечуючи консистентність та атомарність змін. У межах "одиниці роботи" виконується додавання, видалення та оновлення даних, а зміни спершу зберігаються в буфері, щоб потім бути застосованими під час коміту транзакції до бази даних.

"Unit of Work" дозволяє зберігати цілісність даних, спрощує керування транзакціями та підвищує продуктивність, зменшуючи кількість звернень до бази даних. Сучасні фреймворки й бібліотеки часто мають вбудовану підтримку цього патерна, що полегшує його використання в розробці програмного забезпечення.

В (рис. 2.6) зображено вигляд патерну "Unit Of Work".

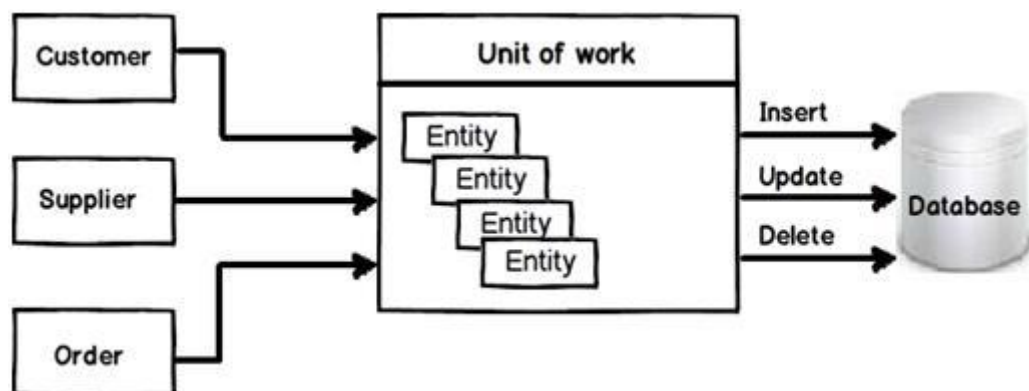


Рисунок 2.6 – Вигляд патерну "Unit Of Work"

2.2 Розробка структури бази даних системи

Для створення бази даних було використано MySQL[3]. Це відкрита система управління базами даних, побудована на реляційній моделі. MySQL є однією з найпопулярніших і широко використовуваних реляційних СУБД, пропонуючи безліч функцій для зберігання, організації та обробки даних. MySQL часто обирають для різних проєктів, таких як веб-додатки, системи управління контентом, електронна комерція тощо. Завдяки простоті використання, високій продуктивності та багатому функціоналу, цей інструмент є потужним рішенням для роботи з базами даних.

Для збереження даних про події, була створена таблиця “events”, (табл. 2.1). Вона містить інформацію про події з інформацією про оплату.

Таблиця 2.1

Структура таблиці “events”

Назва	Тип даних	PK	FK	Опис поля
id	int(11)	+	–	Id ордеру (максимальна кількість символів 11)
title	varchar(255)	–	–	Заголовок події (максимальна кількість символів 255)
venues_id	int(11)	–	–	Id локації (максимальна кількість символів 11)
description	text	–	–	Опис афіши
text	text	–	–	Текст афіши
price	int(11)	–	–	Ціна квитку (максимальна кількість символів 11)
datetime_event	datetime	–	–	Дата й час майб.події
datetime	datetime	–	–	Дата й час додавання
datetime_lastedit	datetime	–	+	Дата й час редагування

Продовження таблиці 2.1

photo	varchar(255)	–	–	Назва фото (максимальна кількість символів 255)
user_id	int(11)	–	–	Id автора (максимальна кількість символів 11)

Для збереження даних про користувача, була створена таблиця “users”, (табл. 2.2). Вона містить інформацію про користувача та данні для входу до особистого кабінету.

Таблиця 2.2

Структура таблиці “users”

Назва	Тип даних	PK	FK	Опис поля
id	int(11)	+	–	Id користувача (максимальна кількість символів 11)
login	varchar(255)	–	–	Ім'я користувача (максимальна кількість символів 255)
password	varchar(255)	–	–	Email користувача (максимальна кількість символів 255)
access	int(11)	–	–	Рівень доступу користувача (максимальна кількість символів 11)
lastname	varchar(255)	–	–	Прізвище користувача (максимальна кількість символів 255)
firstname	varchar(255)	–	–	Ім'я користувача (максимальна кількість символів 255)

“messages” – містить інформацію про всі коментарі до подій (табл. 2.3).

Таблиця 2.3

Структура таблиці “messages”

Назва	Тип даних	РК	FK	Опис поля
id	int(11)	+	—	Id коментаря (максимальна кількість символів 11)
message	text	—	—	Текст коментарю
datetime	datetime	—	—	Дата початку сеансу
event_id	int(11)	—	—	Id події (максимальна кількість символів 11)
user_id	int(11)	—	—	Id автора (максимальна кількість символів 11)

“chat ” – містить інформацію про всі повідомлення у чаті (табл. 2.4).

Таблиця 2.4

Структура таблиці “chat ”

Назва	Тип даних	РК	FK	Опис поля
id	int(11)	+	—	Id повідомлення (максимальна кількість символів 11)
message	text	—	—	Текст повідомлення
datetime	datetime	—	—	Дата й час відправлення
user_id	int(11)	—	—	Id автора (максимальна кількість символів 11)

“venues” – містить інформацію про всі локації (табл. 2.5).

Таблиця 2.5

Структура таблиці “venues”

Назва	Тип даних	PK	FK	Опис поля
id	int	+	–	Id локації
title	varchar(255)	–	–	Заголовок локації (максимальна кількість символів 255)
description	text	–	–	Опис локації
address	varchar(255)	–	–	Адрес локації (максимальна кількість символів 255)
city	varchar(255)	–	–	Місто локації (максимальна кількість символів 255)
datetime	datetime	–	–	Дата й час додавання
datetime_lastedit	datetime	–	+	Дата й час редагування
photo	varchar(255)	–	–	Назва фото (максимальна кількість символів 255)
user_id	int(11)	–	–	Id автора (максимальна кількість символів 11)

Переходимо до розробки та реалізації алгоритмів роботи системи замовлення квитків відповідно до запропонованих рішень, зокрема алгоритмів пошуку подій та локацій, оброблення замовлень квитків та платежів, забезпечення безпеки.

2.3 Проектування та реалізація алгоритмів роботи системи

Проектування та реалізація алгоритмів роботи системи замовлення квитків включає наступні аспекти:

1. Алгоритм пошуку розважальних подій. Система повинна мати ефективний алгоритм, який дозволяє користувачам швидко знаходити доступні

концерти, вистави, тощо... Цей алгоритм може базуватися на параметрах пошуку, як по назві.

2. Алгоритм обробки платежів. При здійсненні оплати за квитки система повинна мати безпечний алгоритм, який забезпечує захист фінансових даних користувачів та підтверджує успішну оплату. Цей алгоритм може використовувати платіжні шлюзи.

3. Алгоритм забезпечення безпеки. Система повинна мати алгоритми, що забезпечують захист особистих даних користувачів, безпеку транзакцій та запобігання шахрайству. Це може включати шифрування даних та автентифікацію користувачів.

Оскільки для нашої системи обрано архітектуру SPA[18], для її повної функціональності ми будемо використовувати RESTful API.

RESTful API(інтерфейс програмування додатків) — це інтерфейс, який базується на принципах архітектури REST і дозволяє створювати веб-сервіси для доступу до ресурсів і управління ними.

Основні риси RESTful API: – Застосування HTTP методів: RESTful API використовує методи HTTP, такі як GET, POST, PUT і DELETE для взаємодії з ресурсами. Наприклад, GET дозволяє отримати ресурс, POST — створити новий, PUT — оновити, а DELETE — видалити. – Унікальні URI для ресурсів:

Кожен ресурс ідентифікується за допомогою унікального URI (уніфікованого ідентифікатора ресурсу). – Стандартні формати даних: Для обміну даними через RESTful API використовуються формати, такі як JSON (JavaScript Object Notation) або XML (eXtensible Markup Language), причому JSON є найбільш поширеним. – Безстейтовість: RESTful API не зберігає інформації про попередні запити — кожен запит містить усю необхідну для його виконання інформацію. – Можливість кешування: RESTful API підтримує кешування, що підвищує ефективність системи, оскільки клієнти можуть зберігати та повторно використовувати отримані відповіді.

В рисунку (рис. 2.7) показано загальний алгоритм роботи системи:

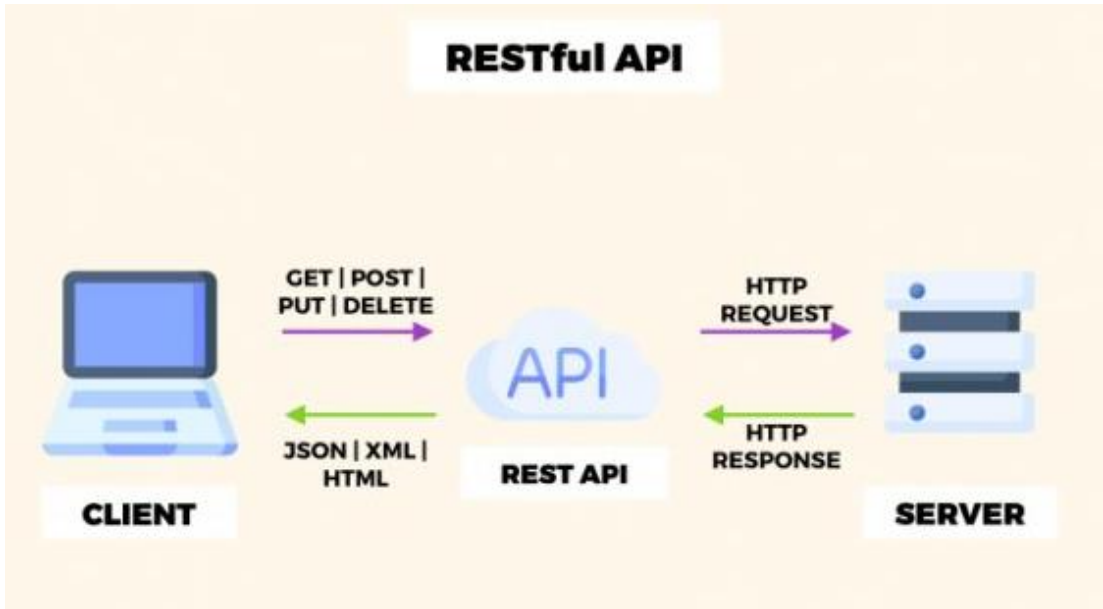


Рисунок 2.7 – Загальний алгоритм роботи системи

Під час згадки алгоритму роботи та можливих станів програмної системи відразу приходить на думку побудова діаграми діяльності (activity diagram).

Діаграма діяльності є важливою діаграмою поведінки для опису динамічних аспектів системи. Діаграма діяльності – це, вдосконалена версія блок-схеми, яка моделює потік від однієї діяльності до іншої. На рисунку (рис. 2.8) зображено діаграму діяльності:

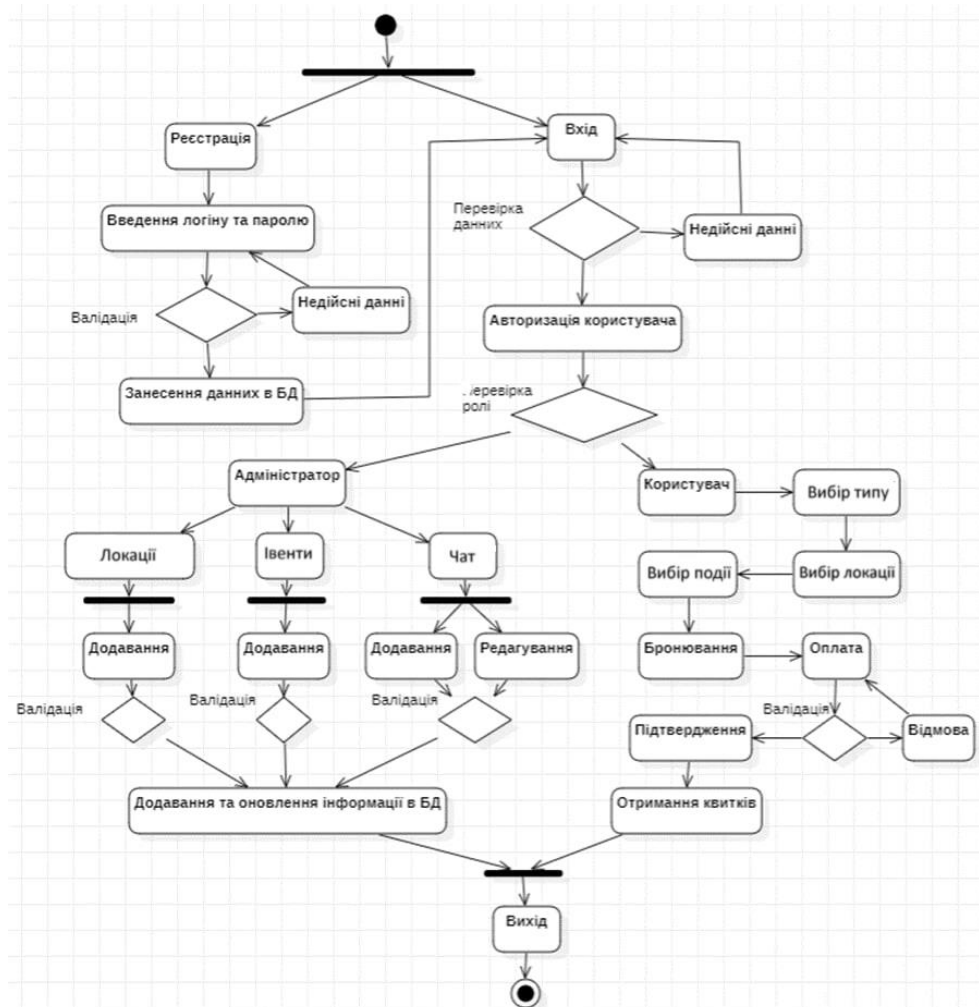


Рисунок 2.8 – Діаграма діяльності

Опис функціоналу (користувача):

1. Користувач (неавторизований) має можливість переглянути доступні локації та події.
2. Авторизація (Так – перехід до головної сторінки, ні – перехід до сторінки авторизації з повідомленням про помилку).
3. Перегляд доступних подій та інформації про них.
4. Замовлення квитків.
 - Вибір локації
 - Вибір події
 - Перегляд відгуків
 - Оплата.
5. Кінець роботи.

Опис функціоналу (адміністратора):

1. Адміністратор має можливість додати локацію.
2. Адміністратор має можливість додати події.
3. Відредагувати вже існуючі події.
4. Відредагувати вже існуючі локації.
5. Видалити вже існуючі події.
6. Видалити вже існуючі локації.
7. Додати подію вже до існуючої локації.
8. Кінець роботи.

На рисунку (рис. 2.9) зображено діаграму компонентів.

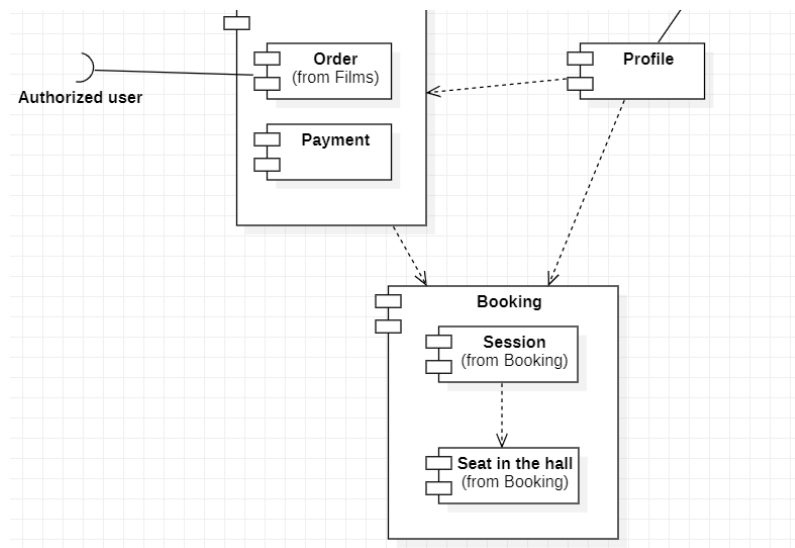


Рисунок 2.9 – Діаграма компонентів

Аналіз діаграми

1. Користувач має бути авторизованим для повноцінного використання функціоналу системи в повному образі.
2. Система складається з одного компонента та його підкомпонента.
3. Підкомпонент Booking відповідає за здійснення оплати для отримання квитків на обраний сеанс.
4. Підсистема Events відповідає за обраний фільм та його інформацію.
5. Компонент Profile Відповідає за профіль користувача та відображає всі квитки з особистою інформацією клієнта.

Розглянемо алгоритм оплати fondy.ua. Для більшої наочності було побудовано діаграму. На рисунку (рис. 2.10) зображено діаграму послідовності fondy.ua.

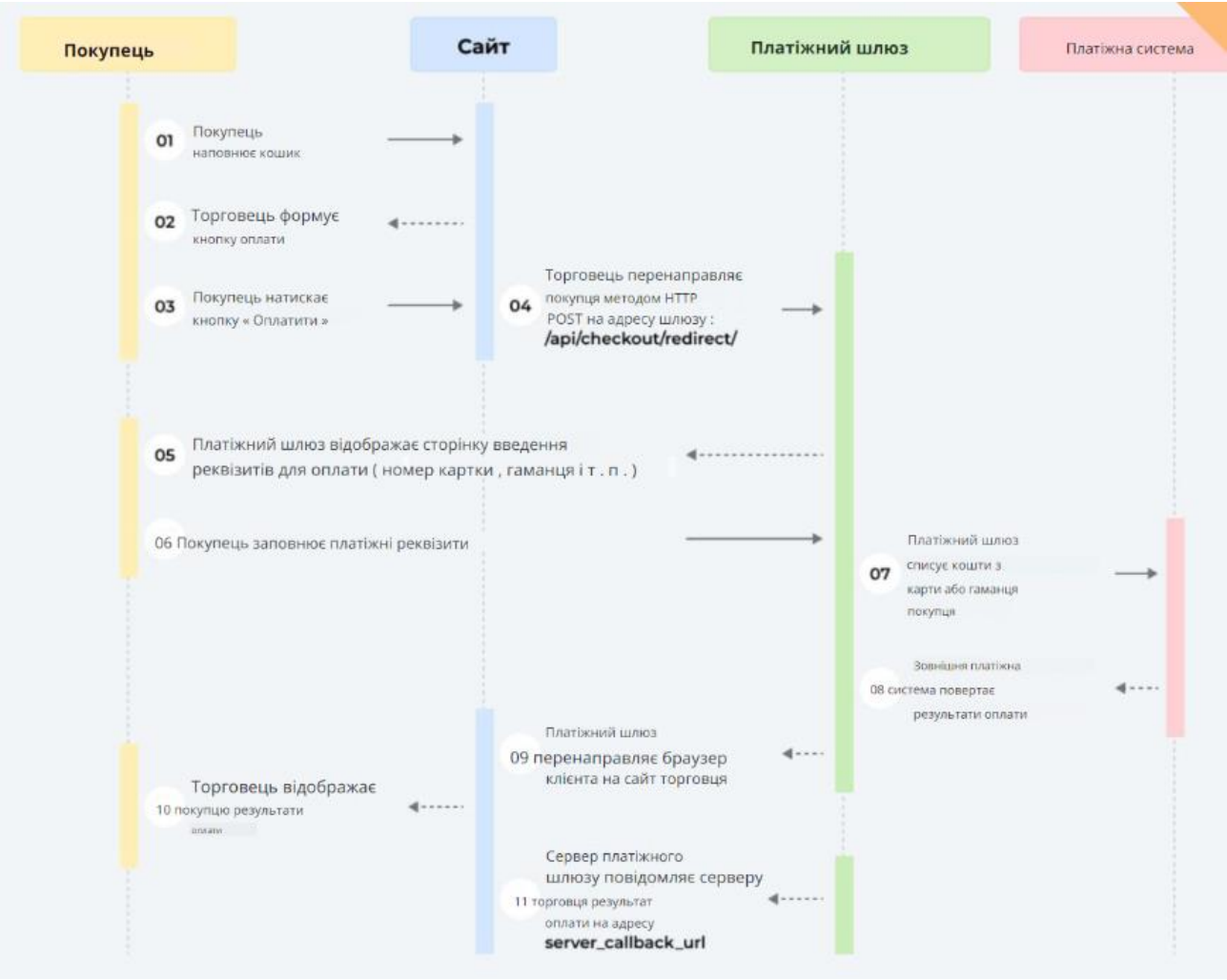


Рисунок 2.10 – Діаграма послідовності fondy.ua

Після того, як авторизований користувач обирає локацію та подію, він може натиснути кнопку для переходу до оплати. Після успішного завершення транзакції статус його замовлення оновлюється в базі даних, що підтверджує покупку. Після цього користувач може завантажити PDF-файл з квитком у своєму особистому кабінеті на веб-сайті. Також доступна можливість переглянути раніше придбані квитки. Функція перегляду дозволяє користувачу контролювати свої попередні покупки, забезпечуючи зручний доступ до інформації про придбані квитки. Квиток містить важливі деталі, такі як назва події, дата, час та інші важливі відомості, що полегшують користувачу доступ до потрібної події.

2.4 Реалізація онлайн-сервісу з бронювання та купівлі квитків “BOZBO”

Для того щоб замовити квитки, користувач повинен пройти реєстрацію. Розглянемо процес її реалізації. Метод обробляє запит на створення нового користувача, додаючи його до бази даних з введеними даними. Пароль хешується для безпеки, після чого генерується токен доступу для автентифікації. Користувачу автоматично призначається роль "user", і у відповідь повертаються токен доступу та дані користувача, що підтверджує успішну реєстрацію.

На рисунку (рис. 2.11) зображено метод actionRegister, що відповідає за реєстрацію нового користувача.

```
function actionRegister()
{
    if ($this->isPost())
    {
        $result = $this->usersModel->AddUser($_POST);
        if ($result === true) {
            return $this->renderMessage('ok', 'Користувача успішно додано', null,
                [
                    'MainTitle' => 'Реєстрація на сайті',
                    'PageTitle' => '',
                ]);
        } else
        {
            $message = implode('<br> ', $result);
            return $this->render('register', null,
                [
                    'MainTitle' => 'Реєстрація на сайті',
                    'PageTitle' => '',
                    'MessageText' => $message,
                    'MessageClass' => 'danger',
                ]);
        }
    } else {
        return $this->render('register', null,
            [
                'MainTitle' => 'Реєстрація на сайті',
                'PageTitle' => '',
            ]);
    }
}
```

Рисунок 2.11 – Метод actionRegister, що відповідає за реєстрацію користувача

Після реєстрації, користувачу необхідно увійти на сайт. Метод перевіряє, чи не є користувач вже авторизованим. Якщо так, то виводить відповідне сповіщення про це, якщо ж ні, то продовжує процес авторизації, звіряє введений логін та пароль із тими, які є в базі даних, якщо ж один із них збігається, то користувач є авторизованим. Якщо ж ні, то виводиться повідомлення про те, що введено неправильний логін або пароль.

На рисунку (рис. 2.12) зображено метод actionLogin, що відповідає за авторизацію користувача.

```
function actionLogin()
{
    $title = 'Вхід на сайт';
    if(isset($_SESSION['user']))
        return $this->renderMessage('warning', 'Ви вже увійшли на сайт...', null,
            [
                'MainTitle' => $title,
                'PageTitle' => '',
            ]);
    if ($_this->isPost())
    {
        $user = $this->usersModel->AuthUser($_POST['login'], $_POST['password']);
        if (!empty($user))
        {
            $_SESSION['user'] = $user;
            return $this->renderMessage('ok', 'Ви успішно увійшли на сайт!', null,
                [
                    'MainTitle' => $title,
                    'PageTitle' => '',
                ]);
        } else {
            return $this->render('login', null,
                [
                    'MainTitle' => $title,
                    'PageTitle' => '',
                    'MessageText' => 'Неправильний логін або пароль :(',
                    'MessageClass' => 'danger',
                ]);
        }
    } else {
        return $this->render('login', null,
            [
                'MainTitle' => $title,
                'PageTitle' => '',
            ]);
    }
}
```

Рисунок 2.12 – Метод actionLogin, що відповідає за авторизацію користувача

Наступний метод обробляє запит на додавання нової події до системи з інформацією про локацію та подію. Від перевіряє чи користувач авторизований, в негативному випадку, виводить повідомлення про те, що доступ заборонено. Якщо ж користувач авторизований на сайті, то метод далі реалізовує додавання нової події, приймає данні із форми, та сформовує файл зображення, надаючи йому назву та відправляє запит до бази даних.

На рисунку (рис. 2.13) зображено метод actionAdd, що відповідає за додавання події.

```
public function actionAdd()
{
    if (empty($this->user))
    {
        $titleForbidden = 'Доступ заборонено';
        return $this->render('forbidden', null,
            [
                'MainTitle' => $titleForbidden,
                'PageTitle' => $titleForbidden,
            ]);
    }
    $title = 'Додавання';
    if($this->isPost())
    {
        $result = $this->venuesModel->AddVenues($_POST);
        if ($result['error'] === false) { /** фото **/
            $allowed_types = ['image/png', 'image/jpeg'];
            if (is_file($_FILES['file']['tmp_name']) && in_array($_FILES['file']['type'], $allowed_types)) {
                switch ($_FILES['file']['tmp_name']){
                    case 'image/png':
                        $extension = 'png';
                        break;
                    default :
                        $extension = 'jpg';
                }
                $name = $result['id'].'.'.uniqid().'.'.$extension;
                move_uploaded_file($_FILES['file']['tmp_name'], 'files/venues/'.$name);
                $this->venuesModel->ChangePhoto($result['id'], $name);
            }
            return $this->renderMessage('ok', 'Успішно додано', null,
                [
                    'PageTitle' => $title,
                    'MainTitle' => $title
                ]);
        } else
        {
            $message = implode('<br/>', $result['messages']);
            return $this->render('form', ['model' => $_POST], [
                'PageTitle' => $title,
                'MainTitle' => $title,
                'MessageText' => $message,
                'MessageClass' => 'danger'
            ]);
        }
    } else {
        return $this->render('form', ['model' => $_POST],
            [
                'MainTitle' => $title,
                'PageTitle' => $title
            ]);
    }
}
```

Рисунок 2.13 – Метод actionAdd, що відповідає за додавання події

Висновки до другого розділу:

1. Проведено аналіз можливих сценаріїв використання системи, визначено основні цілі її функціонування. Об'єктно-орієнтована архітектура системи розроблена з урахуванням потреб користувачів та ключових функцій сервісу.
2. Створено структуру бази даних, яка включає необхідні таблиці та зв'язки для зберігання даних про фільми, сеанси, квитки, користувачів та інші важливі відомості.
3. Розроблено та реалізовано алгоритми, що забезпечують функціонування системи, зокрема процеси пошуку сеансів, обробки замовлень квитків, бронювання місць та інші ключові операції.
4. Впроваджено онлайн-сервіс бронювання та купівлі квитків "BOZBO", де враховано всі попередні етапи, створено зручний і функціональний інтерфейс для користувачів.

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-ПЛАТФОРМИ БРОНЮВАННЯ ТА ПРОДАЖУ КВИТКІВ “VOZBO”

3.1 Порядок встановлення та налаштування параметрів системи

Для кращого розуміння розгортання системи використовується діаграма розгортання (Deployment diagram). Вона відображає фізичну архітектуру системи, показуючи розташування її компонентів і зв'язки між ними. Ця діаграма ілюструє, як програмні компоненти розміщуються на різних апаратних засобах, таких як сервери, комп'ютери або мобільні пристрої. Вона дає змогу зрозуміти взаємодію фізичних компонентів системи та їхнє розташування. Діаграма розгортання є важливим інструментом для планування та аналізу інфраструктури перед впровадженням системи.

На рисунку (рис. 3.1) зображено діаграму розгортання.

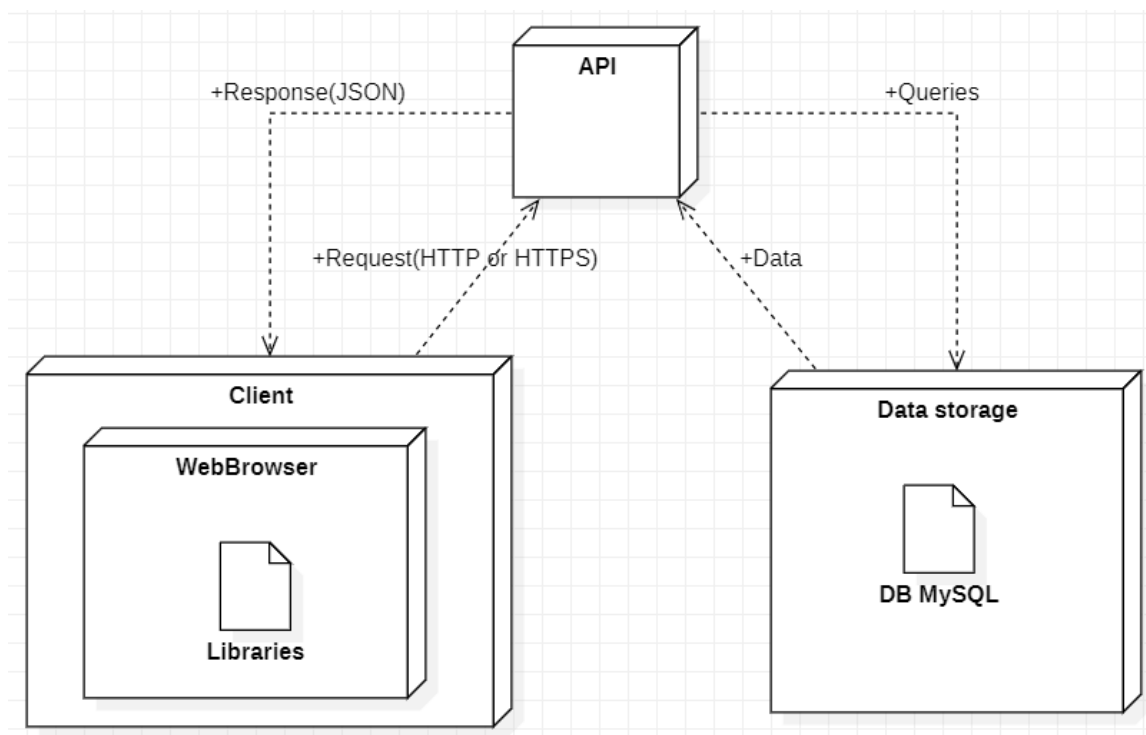


Рисунок 3.1 – Діаграма розгортання

Проаналізувавши діаграму, можемо зробити такі висновки:

- Бекенд взаємодіє з сервером для отримання необхідних ресурсів.
- При запуску додатка клієнт отримує початкову HTML-розмітку, а потім оновлює її за допомогою асинхронних запитів до API.

- Веб-застосунок має розподілену архітектуру: бекенд працює на порту 8000, а клієнтська частина — на порту 3000.
- Бекенд здійснює запити до сервера для доступу до необхідних ресурсів.

Ця архітектура розділяє бекенд і клієнтську частини, дозволяючи їх незалежно розгортати та масштабувати. Бекенд відповідає за доступ до ресурсів і виконання бізнес-логіки, тоді як клієнтська частина забезпечує візуалізацію та взаємодію з користувачем. Використання API дозволяє клієнту оновлювати дані асинхронно, підтримуючи динамічне оновлення сторінок і покращуючи користувацький досвід.

Структура інтерфейсу та порядок взаємодії з онлайн-сервісом

Коли користувач переходить на сторінку нашого застосунку, його зустрічає головна сторінка з останніми доданими подіями та іншими категоріями:

На рисунку (рис. 3.2) зображено головну сторінку.

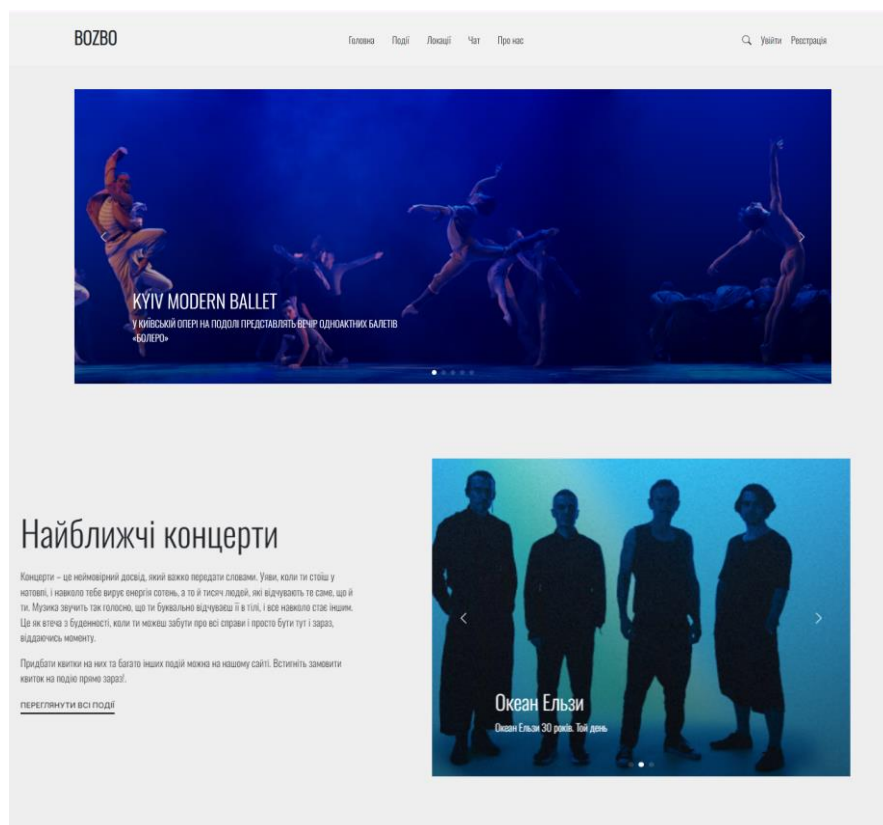


Рисунок 3.2 – Головна сторінка

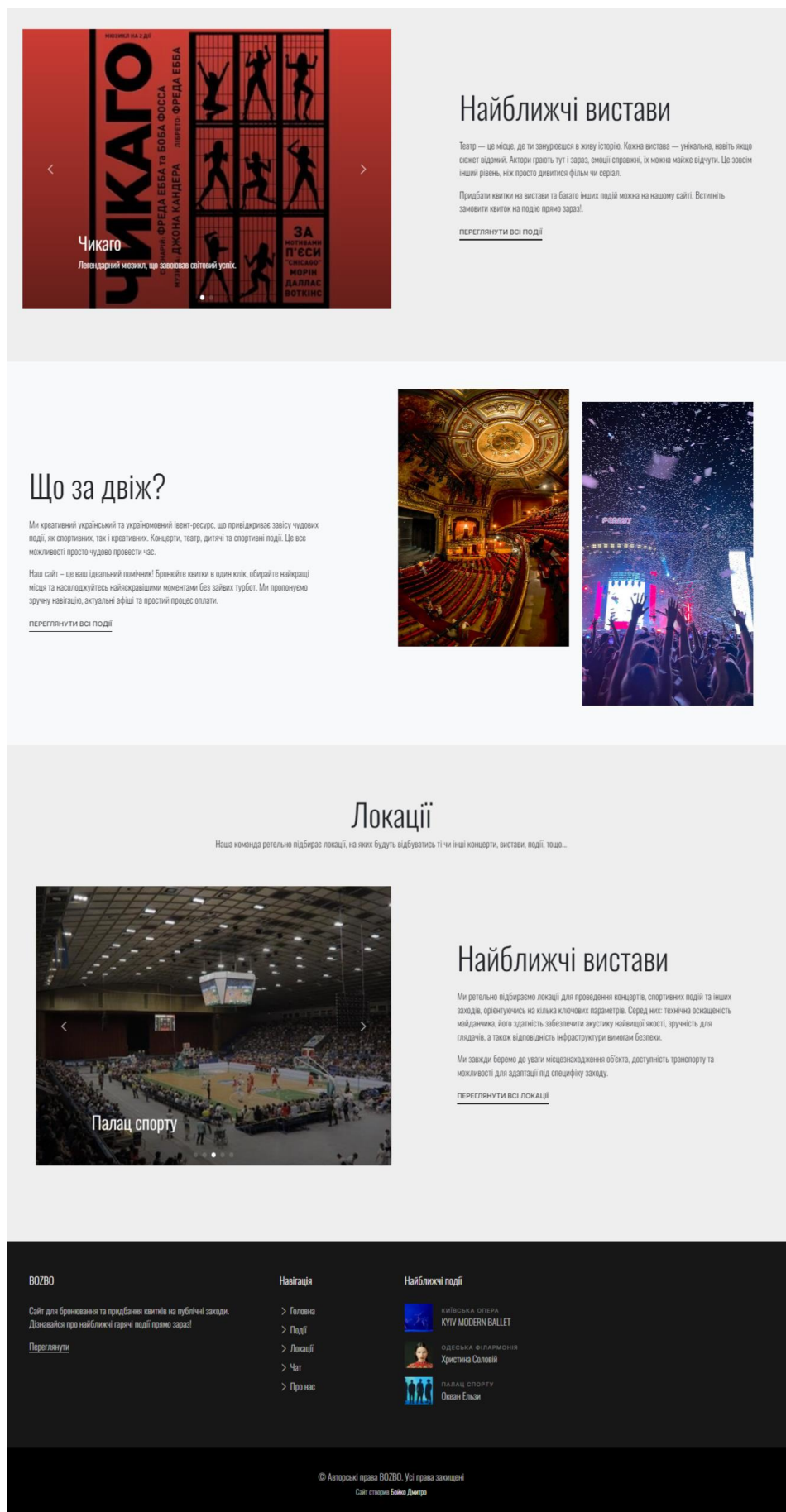


Рисунок 3.3 – Продовження головної сторінки

Користувач має можливість зробити авторизацію, зареєструватись або ознайомитись з усіма доступними подіями.

В рисунку (рис. 3.4) зображено вікно реєстрації.

The screenshot shows the BOZBO website's registration page. The header includes the BOZBO logo, navigation links (Головна, Події, Локації, Чат, Про нас), and a search icon with links to Увійти and Реєстрація. The main content area features a white registration form titled "Реєстрація". Below the title, it states "Аккаунт необхідний для того, щоб користуватись чатом, коментувати події, тощо..." and provides a link "Увійти можна тут". The form contains input fields for Ім'я*, Прізвище*, Звідки ви?, Номер телефону, Ел.пошта*, Пароль*, and Повторіть пароль*. A checkbox for "Я згоден з Правилами користування*" is present, followed by a "Створити аккаунт" button.

Рисунок 3.4 – Вікно реєстрації

Користувач може зробити авторизацію на сайті ввівши інформацію у відповідні поля. В рисунку (рис. 3.5) зображено вікно авторизації.

The screenshot shows the BOZBO website's login page. The header is identical to the registration page. The main content area features a white login form titled "Увійти". Below the title, it says "Створити аккаунт тут!". The form has input fields for "Ваша ел.пошта" and "Пароль", followed by a "Увійти" button.

Рисунок 3.5 – Вікно авторизації

Після авторизації, користувач може перейти на сторінку із всіма доступними подіями. В рисунку (рис. 3.6) зображено сторінку подій.

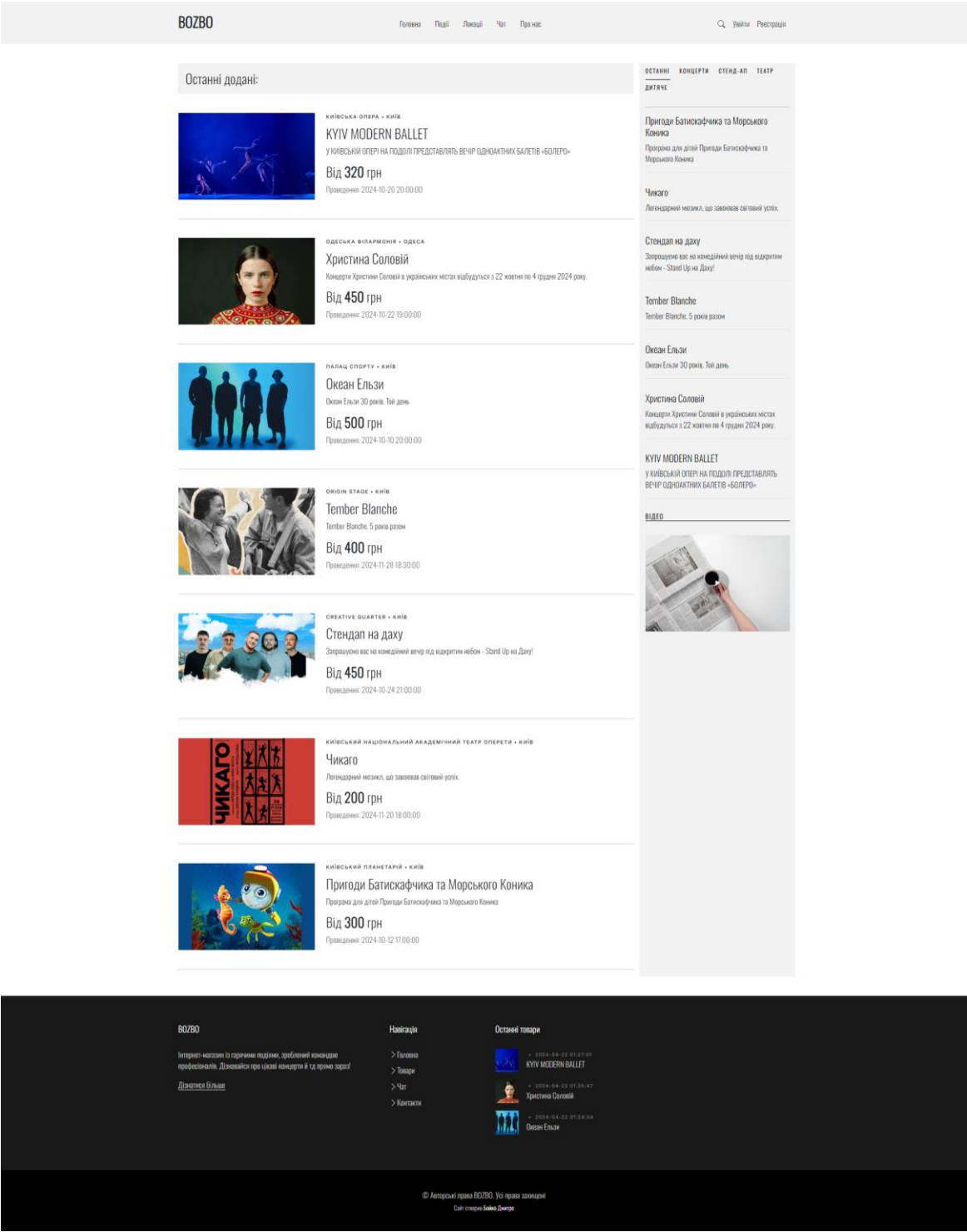


Рисунок 3.6 – Сторінка подій

Користувач може відкрити будь-яку подію й переглянути інформацію про неї. В рисунку (рис. 3.7) зображено сторінку події.

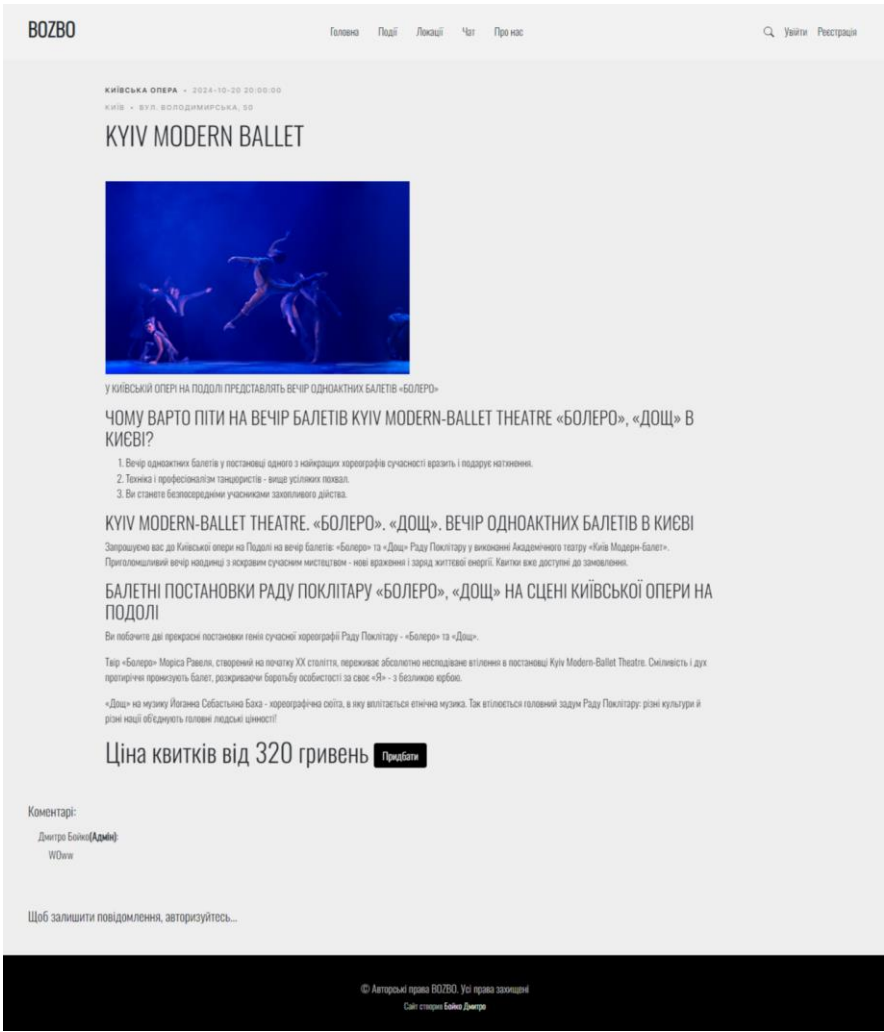


Рисунок 3.7 – Сторінка події

ПЕРЕГЛЯНУВШИ ДЕТАЛІ ПРО ПОДІЇ, КОРИСТУВАЧ МОЖЕ ПРИДБАТИ КВИТКИ, НАЖАВШИ НА КНОПКУ “ПРИДБАТИ”. В РИСУНКУ (РИС. 3.8) ЗОБРАЖЕНО ВІКНО КУПІВЛІ.

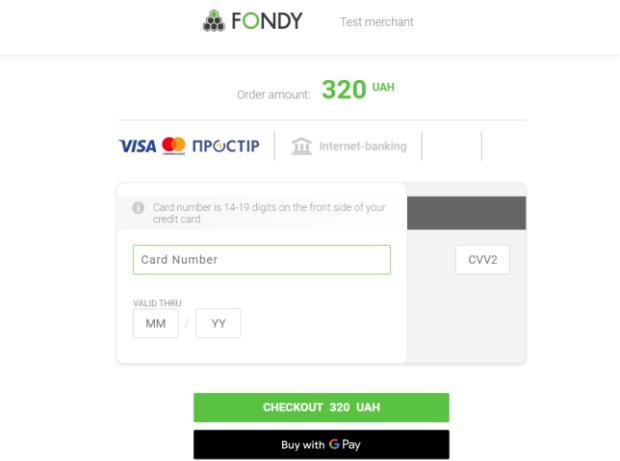


Рисунок 3.8 – Вікно купівлі

Також користувачу доступна сторінка, де він може переглянути всі існуючі локації, на яких можуть відбуватись події та деталі про неї. В рисунку (рис. 3.9) та (рис. 3.10) зображено сторінку із локаціями та сторінку локації.

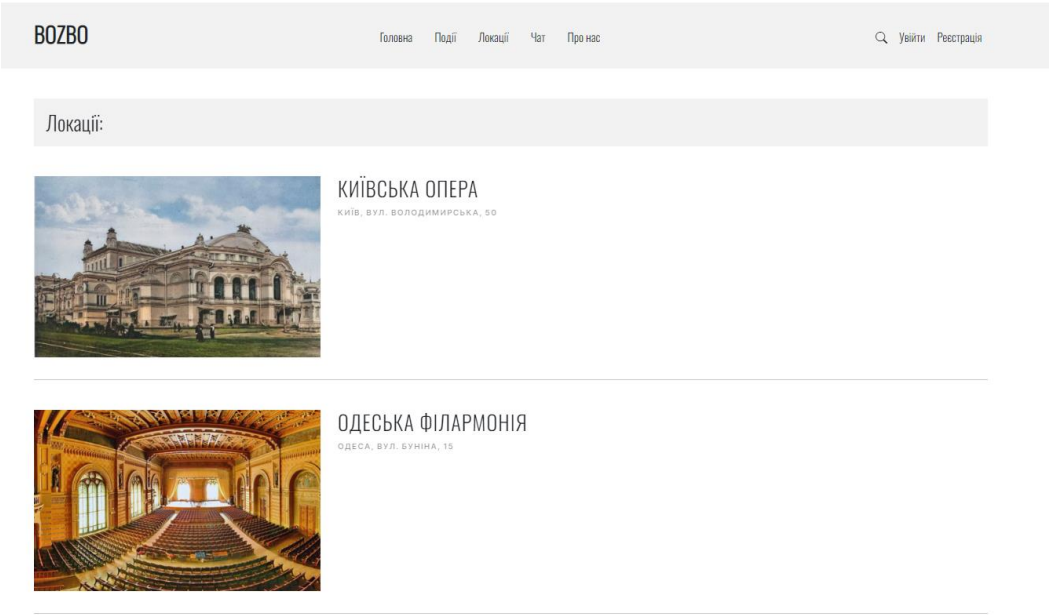


Рисунок 3.9 – Сторінка із локаціями

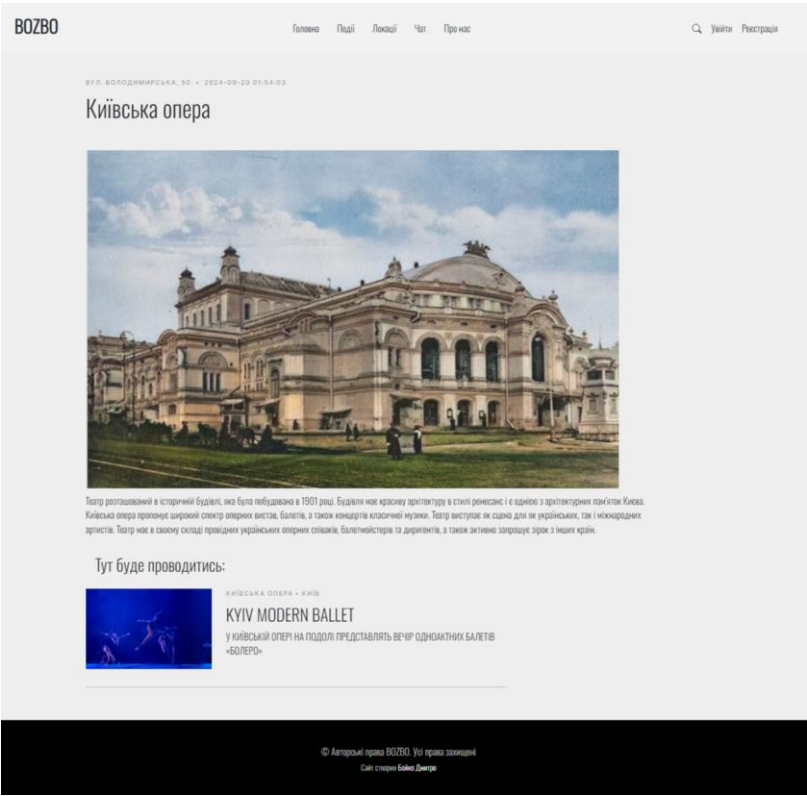


Рисунок 3.10 – Сторінка локації

Також користувач може відвідати загальний чат користувачів. В рисунку (рис. 3.11) зображено сторінку із чатом.

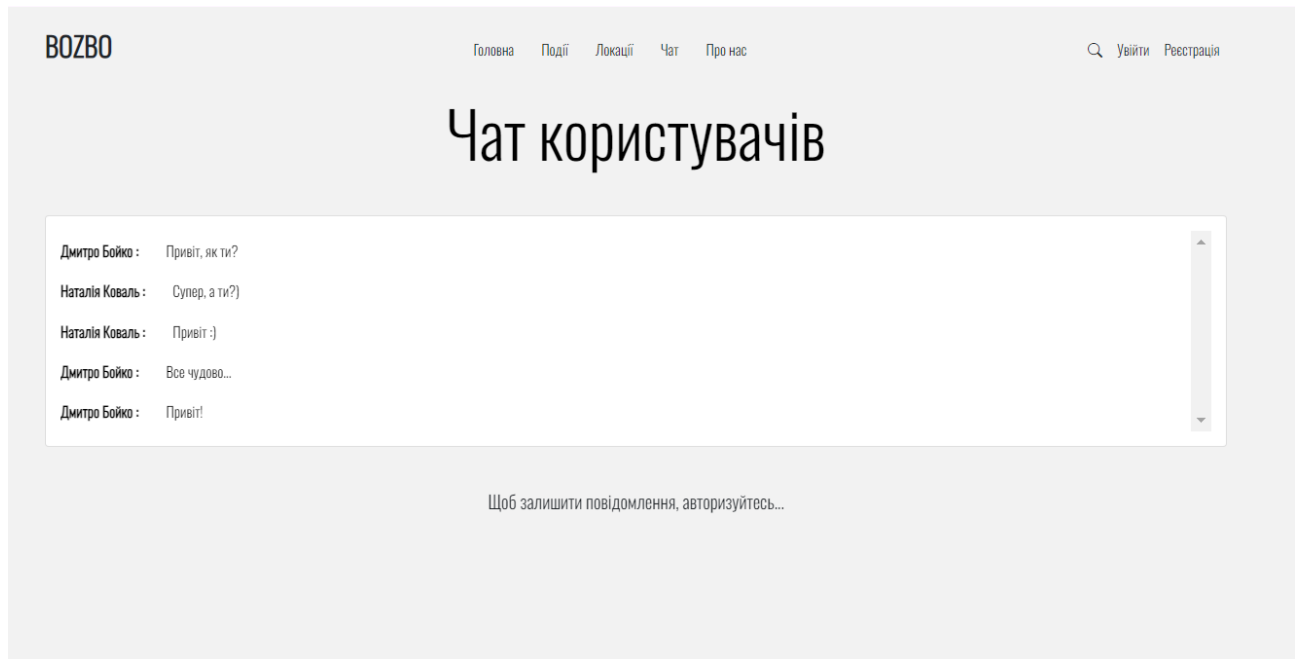


Рисунок 3.11 – Чат

Також на сайті реалізований функціонал пошуку. В рисунку (рис. 3.12) зображено сторінку локацію.



Рисунок 3.12 – Сторінка локації

Для того, щоб додати та відреагувати подію чи локацію, користувачу потрібно заповнити форму. В рисунку (рис. 3.13) зображено сторінку із формою.

Рисунок 3.13 – Сторінка локації

Також на сайті був обов'язково реалізований функціонал видалення певної події чи локації. В рисунку (рис. 3.14) зображено сторінку із формою.

Рисунок 3.14 – Сторінка видалення

На сайті реалізована сторінка із інформацією про творців сайту, його концепцією, ідеєю та метою, контактною формою для зв'язку із адміністрацією веб-платформи. В рисунку (рис. 3.15) зображено сторінку “Про нас”.



Рисунок 3.15 – Сторінка “Про нас”

3.3 Тестування роботи програмного продукту

Тестування продукту є ключовим етапом у розробці програмного забезпечення. Воно дозволяє перевірити коректність роботи функцій, виявити помилки та проблеми, а також гарантувати високу якість продукту перед його запуском.

Цей процес є безперервним і виконується на різних стадіях розробки, а також після виходу продукту. Тестування допомагає забезпечити надійність і якість продукту, мінімізувати ризики появи дефектів і покращити загальну задоволеність користувачів.

Таблиця 3.1

Test Case таблиця

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка авторизації з валідними даними	1. URL: http://bozbo/users/login 2. Введіть email 3. Введіть пароль 4. Натисніть “Увійти”	Email: admin@gmail.com Пароль: admin	Користувач повинен увійти до системи	Як і очікувалось	Pass
Перевірка авторизації з недійсними даними	1. URL: http://bozbo/users/login 2. Введіть email 3. Введіть пароль 4. Натисніть “Увійти”	Email: abc@gmail.com Пароль: blabla	Користувач не повинен увійти до системи	Як і очікувалось	Pass
Перевірка додавання події з валідними даними	1. Увійти до системи у ролі адміністратора 2. Перейти до вкладки “Додати подію” 3. Заповнити відповідні поля інформацією 4. Натисніть кнопку “Зберегти”	Title: Христина Соловій Venues: Одеська філармонія Description: Концерти Хрис... Text: Чому варт... Price: 320 Date: 2024-10-22 Time: 19:00	Повинна додатись подія до БД	Як і очікувалось	Pass

Продовження таблиці 3.1

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
Перевірка додавання локації	1. Увійти до системи у ролі адміністратора 2. Перейти до вкладки “Додати локацію” 3. Заповнити відповідні поля інформацією 4. Натиснути кнопку “Зберегти”	Title: Одеська філармонія Description: Будівля: філармонія розмі... Address: вул.Буніна 15 City: Одеса	Повиненна додаться локація до БД	Як і очікувалось	Pass

Мета тестування полягає в перевірці продукту з метою забезпечення його якості, виявлення помилок та проблем, а також підтвердження відповідності вимогам та очікуванням.

Висновки до третього розділу

1. Описано процес інсталяції та налаштування параметрів системи, необхідних для її коректного функціонування.
2. Розглянуто структуру інтерфейсу та порядок взаємодії з онлайн-сервісом. Вказано, що клієнтська частина є додатком, який використовує бібліотеки для створення інтерфейсу користувача та керування логікою оновлення даних і сторінок. Адміністратори мають доступ до адмін-панелі для виконання відповідних функцій.
3. Описано процес тестування продукту, яке є ключовим етапом розробки програмного забезпечення. Тестування дозволяє перевірити правильність функціонування, виявити помилки та гарантувати високу якість продукту перед його запуском. Це безперервний процес, що проводиться на різних стадіях розробки та після виходу продукту.
4. Показано кроки для коректного встановлення та налаштування системи.

ВИСНОВКИ

При виконанні бакалаврської випускної роботи на тему "Платформа для бронювання та купівлі квитків" було здійснено детальну розробку та впровадження веб-орієнтованої системи, що забезпечує зручний та ефективний процес замовлення квитків.

Проведений аналіз існуючих аналогів, таких як Cinema.ua та Karabas, виявив певні обмеження і недоліки цих систем. У запропонованій платформі було вирішено низку цих проблем, що дозволило поліпшити її функціональність та зручність для користувачів.

Актуальність теми підтверджена зростанням попиту на онлайн-сервіси та електронну комерцію, зокрема у сфері культурних заходів, таких як концерти, вистави та інші події. Створена інтуїтивно-зрозуміла та багатофункціональна системи для купівлі квитків значно покращує взаємодію користувачів з такими сервісами.

Процес проектування та впровадження системи включав розробку алгоритмів для пошуку подій і локацій, а також заходів безпеки для захисту даних і запобігання шахрайству. Реалізація цих елементів дозволила створити надійну і функціональну платформу, яка максимально відповідає потребам користувачів.

Таким чином, при виконанні бакалаврської роботи на тему "Платформа для бронювання та купівлі квитків 'BOZBO'" було створено ефективну веб-систему для зручного замовлення квитків, яка може стати основою для подальшого розвитку подібних рішень у сфері культурних та розважальних заходів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація JSON [Електронний ресурс] – Режим доступу до ресурсу: <http://json.com/>
2. Патерни проектування [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns/what-is-pattern>
3. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com/>
4. Онлайн-платформа Udemy [Електронний ресурс] – Режим доступу до ресурсу: <https://ua.udemy.com/>
5. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу: <https://evergreens.com.ua/ua/articles/uml-diagrams.html>
6. А.Зандстра, Ф.Матт. PHP Objects, Patterns, and Practice. [Електронний ресурс]. Режим доступу до ресурсу: <https://www.apress.com/la/book/9781430260325>
7. Завантажувати файли за допомогою хмарного сховища в Інтернеті. [Електронний ресурс] – 2022. Режим доступу до ресурсу: <https://firebase.google.com/docs/storage/web/upload-files>
8. Повна документація phpMyAdmin. [Електронний ресурс] - 2022. Режим доступу до ресурсу: <https://www.phpmyadmin.net/docs/>
9. Технічна документація мови програмування PHP. [Електронний ресурс]. Режим доступу до ресурсу: <https://www.php.net/manual/uk/index.php>
10. Технічна документація HTML. [Електронний ресурс]. Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>
11. Технічна документація CSS. [Електронний ресурс]. Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
12. Технічна документація JS. [Електронний ресурс]. Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

13. Yii2 Developer Documentation. [Електронний ресурс]. Режим доступу до ресурсу: <https://www.yiiframework.com/doc/guide/2.0/uk>
14. Windler C., Daubois A. Clean Code in PHP: Expert tips and best practices to write beautiful, human-friendly, and maintainable PHP. Birmingham : Packt Publishing, 2022. 264 p.
15. Nixon R. Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites (Learning PHP, MYSQL, Javascript, CSS & HTML5). O'Reilly Media, 2021. 823 p.
16. Murach J., Harris R. Murach's PHP and MySQL. Fresno : Mike Murach & Associates, 2022. 848 p.
17. Duckett J. PHP & MySQL: Server-side Web Development. Wiley, 2022. 672 p.
18. SPA в програмуванні. [Електронний ресурс]. Режим доступу до ресурсу: <https://foxminded.ua/spa-u-prohramuvanni/>

ДОДАТКИ

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	.
	.					56

Клас що відповідає за реєстрацію, авторизацію, валідацію користувача.

```
<?php

namespace controllers;
use core\Controller;

/**
 * Контроллер для модуля User
 */
class Users extends Controller
{
    protected $usersModel;
    function __construct()
    {
        $this->usersModel = new \models\Users();
    }

    function actionLogout()
    {
        $title = 'Вихід з аккаунту';
        unset($_SESSION['user']);
        return $this->renderMessage('warning', 'Ви вийшли з вашого аккаунту...', null,
            [
                'MainTitle' => $title,
                'PageTitle' => ",
            ]);
    }

    function actionLogin()
    {
        $title = 'Вхід на сайт';
        if(isset($_SESSION['user']))
            return $this->renderMessage('warning', 'Ви вже увійшли на сайт...', null,
                [
                    'MainTitle' => $title,
                    'PageTitle' => ",
                ]);
        if ($this->isPost())
        {
            $user = $this->usersModel->AuthUser($_POST['login'], $_POST['password']);
            if (!empty($user))
            {
                $_SESSION['user'] = $user;
                return $this->renderMessage('ok', 'Ви успішно увійшли на сайт!', null,
                    [
                        'MainTitle' => $title,
```

```

        'PageTitle' => ",
    });
} else {
    return $this->render('login', null,
        [
            'MainTitle' => $title,
            'PageTitle' => ",
            'MessageText' => 'Неправильний логін або пароль :(',
            'MessageClass' => 'danger',
        ]
    );
}
} else {
    return $this->render('login', null,
        [
            'MainTitle' => $title,
            'PageTitle' => ",
        ]
    );
}
}

function actionRegister()
{
    if ($this->isPost())
    {
        $result = $this->usersModel->AddUser($_POST);
        if ($result === true) {
            return $this->renderMessage('ok', 'Користувача успішно додано', null,
                [
                    'MainTitle' => 'Реєстрація на сайті',
                    'PageTitle' => ",
                ]
            );
        } else
        {
            $message = implode('<br> ', $result);
            return $this->render('register', null,
                [
                    'MainTitle' => 'Реєстрація на сайті',
                    'PageTitle' => ",
                    'MessageText' => $message,
                    'MessageClass' => 'danger',
                ]
            );
        }
    } else {
        return $this->render('register', null,
            [
                'MainTitle' => 'Реєстрація на сайті',
                'PageTitle' => ",
            ]
        );
    }
}
}

```

Клас що відповідає за додавання, редагування або видалення локації.

```
<?php

namespace controllers;
use core\Controller;

/**
 * Контроллер для модуля venues
 */
class Venues extends Controller
{
    protected $user;
    protected $venuesModel;
    protected $userModel;
    public function __construct()
    {
        $this->userModel = new \models\Users();
        $this->venuesModel = new \models\Venues();
        $this->user = $this->userModel->GetCurrentUser();
    }

    /**
     * Відображення початкової сторінки модуля
     */
    public function actionIndex()
    {
        global $Config;
        $title = 'Локації';
        $lastvenues = $this->venuesModel->GetLastVenues($Config['venuesCount']);
        return $this->render('index', ['lastvenues'=>$lastvenues], [
            'MainTitle' => $title,
            'PageTitle' => $title,
        ]);
    }

    /**
     * Перегляд
     */
    public function actionView()
    {
        $id = $_GET['id'];
        $venues = $this->venuesModel->GetVenuesById($id);
        $title = $venues['title'];
        return $this->render('view', ['model' => $venues], [
            'MainTitle' => $title,
            'PageTitle' => $title,
        ]);
    }
}
```

```

    });
}

/**
 * Додавання
 */
public function actionAdd()
{
    if (empty($this->user))
    {
        $titleForbidden = 'Доступ заборонено';
        return $this->render('forbidden', null,
            [
                'MainTitle' => $titleForbidden,
                'PageTitle' => $titleForbidden,
            ]
        );
    }
    $title = 'Додавання';
    if($this->isPost())
    {
        $result = $this->venuesModel->AddVenues($_POST);
        if ($result['error'] === false) { /** фото */
            $allowed_types = ['image/png', 'image/jpeg'];
            if (is_file($_FILES['file']['tmp_name']) && in_array($_FILES['file']['type'], $allowed_types)) {
                switch ($_FILES['file']['tmp_name']){
                    case 'image/png':
                        $extension = 'png';
                        break;
                    default :
                        $extension = 'jpg';
                }
                $name = $result['id'].'_'.uniqid().'.'.$extension;
                move_uploaded_file($_FILES['file']['tmp_name'], 'files/venues/'.$name);
                $this->venuesModel->ChangePhoto($result['id'], $name);
            }
            return $this->renderMessage('ok', 'Успішно додано', null,
                [
                    'PageTitle' => $title,
                    'MainTitle' => $title
                ]
            );
        } else
        {
            $message = implode('<br/>', $result['messages']);
            return $this->render('form', ['model' => $_POST], [
                'PageTitle' => $title,
                'MainTitle' => $title,
                'MessageText' => $message,
                'MessageClass' => 'danger'
            ]
        );
    }
} else {

```

```

return $this->render('form', ['model' => $_POST],
    [
        'MainTitle' => $title,
        'PageTitle' => $title
    ]);
}
}

/*
 * Редагування
 */
public function actionEdit()
{
    $id = $_GET['id'];
    $venues = $this->venuesModel->GetVenuesById($id);
    $titleForbidden = 'Доступ заборонено';
    if (empty($this->user) || $venues['user_id'] != $this->userModel->GetCurrentUser()['id'] && $this->userModel->GetCurrentUser()['access'] != 1)
    {
        return $this->render('forbidden', null,
            [
                'MainTitle' => $titleForbidden,
                'PageTitle' => $titleForbidden,
            ]);
    }
    $title = 'Редагування';
    if($this->isPost())
    {
        $result = $this->venuesModel->UpdateVenues($_POST, $id);
        if ($result === true) {
            $allowed_types = ['image/png', 'image/jpeg'];
            if (is_file($_FILES['file']['tmp_name']) && in_array($_FILES['file']['type'], $allowed_types)) {
                switch ($_FILES['file']['tmp_name']){
                    case 'image/png':
                        $extension = 'png';
                        break;
                    default :
                        $extension = 'jpg';
                }
                $name = $id.'_'.uniqid().'.'.$extension;
                move_uploaded_file($_FILES['file']['tmp_name'], 'files/venues/'.$name);
                $this->venuesModel->ChangePhoto($id, $name);
            }
        }
        return $this->renderMessage('ok', 'Редаговано', null,
            [
                'MainTitle' => $title,
                'PageTitle' => $title,
            ]);
    } else {
        $message = implode('<br/>', $result);
        return $this->render('form', ['model' => $venues],

```

```

        [
            'MainTitle' => $title,
            'PageTitle' => $title,
            'MessageText' => $message,
            'MessageClass' => 'ok',
        ]);
    }
} else {
    return $this->render('form', ['model' => $venues],
        [
            'MainTitle' => $title,
            'PageTitle' => $title,
        ]);
}
}

/**
 * Видалення
 */
public function actionDelete()
{
    $title = 'Видалення';
    $id = $_GET['id'];
    if(isset($_GET['confirm']) && $_GET['confirm'] == 'yes')
    {
        if($this->venuesModel->DeleteVenues($id))
        {
            header('Location: /venues/');
        } else {
            return $this->renderMessage('error', 'Помилка видалення', null,
                [
                    'MainTitle' => $title,
                    'PageTitle' => $title,
                ]);
        }
    } else {
        $venues = $this->venuesModel->GetVenuesById($id);
        return $this->render('delete', ['model' => $venues],
            [
                'MainTitle' => $title,
                'PageTitle' => $title,
            ]);
    }
}
}
}

```

Клас, що відповідає за додавання, виведення, перемикання та редагування події

```
<?php

namespace controllers;
use core\Controller;

/**
 * Контроллер для модуля events
 */
class Events extends Controller
{
    protected $user;
    protected $eventModel;
    protected $userModel;
    public function __construct()
    {
        $this->userModel = new \models\Users();
        $this->eventModel = new \models\Events();
        $this->user = $this->userModel->GetCurrentUser();
    }

    /**
     * Відображення початкової сторінки модуля
     */
    public function actionIndex()
    {
        global $Config;
        $title = 'Події';
        $lastevents = $this->eventModel->GetLastEvents($Config['eventsCount']);
        return $this->render('index', ['lastevents'=>$lastevents], [
            'MainTitle' => $title,
            'PageTitle' => $title,
        ]);
    }

    /**
     * Перегляд
     */
    public function actionView()
    {
        $id = $_GET['id'];
        $events = $this->eventModel->GetEventsById($id);
        $title = $events['title'];
        return $this->render('view', ['model' => $events], [
            'MainTitle' => $title,
            'PageTitle' => $title,
        ]
    }
}
```

```

    });
}

/**
 * Додавання
 */
public function actionAdd()
{
    if (empty($this->user))
    {
        $titleForbidden = 'Доступ заборонено';
        return $this->render('forbidden', null,
            [
                'MainTitle' => $titleForbidden,
                'PageTitle' => $titleForbidden,
            ]
        );
    }
    $title = 'Додавання';
    if($this->isPost())
    {
        $result = $this->eventModel->AddEvents($_POST);
        if ($result['error'] === false) { /** фото */
            $allowed_types = ['image/png', 'image/jpeg'];
            if (is_file($_FILES['file']['tmp_name']) && in_array($_FILES['file']['type'], $allowed_types)) {
                switch ($_FILES['file']['tmp_name']){
                    case 'image/png':
                        $extension = 'png';
                        break;
                    default :
                        $extension = 'jpg';
                }
                $name = $result['id'].'_'.uniqid().'.'.$extension;
                move_uploaded_file($_FILES['file']['tmp_name'], 'files/events/'.$name);
                $this->eventModel->ChangePhoto($result['id'], $name);
            }
            return $this->renderMessage('ok', 'Успішно додано', null,
                [
                    'PageTitle' => $title,
                    'MainTitle' => $title
                ]
            );
        } else
        {
            $message = implode('<br/>', $result['messages']);
            return $this->render('form', ['model' => $_POST], [
                'PageTitle' => $title,
                'MainTitle' => $title,
                'MessageText' => $message,
                'MessageClass' => 'danger'
            ]
        );
    }
} else {

```

```

return $this->render('form', ['model' => $_POST],
    [
        'MainTitle' => $title,
        'PageTitle' => $title
    ]);
}
}

/*
 * Редагування
 */
public function actionEdit()
{
    $id = $_GET['id'];
    $events = $this->eventModel->GetEventsById($id);
    $titleForbidden = 'Доступ заборонено';
    if (empty($this->user) || $events['user_id'] != $this->userModel->GetCurrentUser()['id'] && $this->userModel->GetCurrentUser()['access'] != 1)
    {
        return $this->render('forbidden', null,
            [
                'MainTitle' => $titleForbidden,
                'PageTitle' => $titleForbidden,
            ]);
    }
    $title = 'Редагування';
    if($this->isPost())
    {
        $result = $this->eventModel->UpdateEvents($_POST, $id);
        if ($result === true) {
            $allowed_types = ['image/png', 'image/jpeg'];
            if (is_file($_FILES['file']['tmp_name']) && in_array($_FILES['file']['type'], $allowed_types)) {
                switch ($_FILES['file']['tmp_name']){
                    case 'image/png':
                        $extension = 'png';
                        break;
                    default :
                        $extension = 'jpg';
                }
                $name = $id.'_'.uniqid().'.'.$extension;
                move_uploaded_file($_FILES['file']['tmp_name'], 'files/events/'.$name);
                $this->eventModel->ChangePhoto($id, $name);
            }
        }
        return $this->renderMessage('ok', 'Редаговано', null,
            [
                'MainTitle' => $title,
                'PageTitle' => $title,
            ]);
    } else {
        $message = implode('<br/>', $result);
        return $this->render('form', ['model' => $events],

```

```

        [
            'MainTitle' => $title,
            'PageTitle' => $title,
            'MessageText' => $message,
            'MessageClass' => 'ok',
        ]);
    }
} else {
    return $this->render('form', ['model' => $events],
        [
            'MainTitle' => $title,
            'PageTitle' => $title,
        ]);
}
}

public function actionDelete()
{
    $title = 'Видалення';
    $id = $_GET['id'];
    if(isset($_GET['confirm']) && $_GET['confirm'] == 'yes')
    {
        if($this->eventModel->DeleteEvents($id))
        {
            header('Location: /events/');
        } else {
            return $this->renderMessage('error', 'Помилка видалення', null,
                [
                    'MainTitle' => $title,
                    'PageTitle' => $title,
                ]);
        }
    } else {
        $events = $this->eventModel->GetEventsById($id);
        return $this->render('delete', ['model' => $events],
            [
                'MainTitle' => $title,
                'PageTitle' => $title,
            ]);
    }
}
}

```

Універсальний шаблон, який є на усіх сторінках сайту (хеадер, футер, тощо)

```

<?php
$userModel = new \models\Users();
$name = $userModel->GetCurrentUser();
?>

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title><?=$MainTitle ?> | BOZBO</title>
  <meta content="" name="description">
  <meta content="" name="keywords">

  <!-- Favicons -->
  <link href="assets/img/favicon.png" rel="icon">
  <link href="assets/img/apple-touch-icon.png" rel="apple-touch-icon">

  <!-- Google Fonts -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
href="https://fonts.googleapis.com/css2?family=EB+Garamond:wght@400;500&family=Inter:wght@400;500&family=Playfair+Display:ital,wght@0,400;0,700;1,400;1,700&display=swap" rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="assets/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="assets/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
  <link href="assets/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="assets/vendor/aos/aos.css" rel="stylesheet">

  <!-- Template Main CSS Files -->
  <link href="assets/css/variables.css" rel="stylesheet">
  <link href="assets/css/main.css" rel="stylesheet">

</head>

<body style="background-color: #eee;">
  <!-- ===== Header ===== -->
  <header id="header" class="header d-flex flex-column align-items-center fixed-top">
    <div class="container-fluid container-xl d-flex align-items-center justify-content-between m-4">

      <a href="/" class="logo d-flex align-items-center">

```

```

<!-- Uncomment the line below if you also wish to use an image logo -->
<!-- 
<h1>BOZBO</h1>
</a>

<nav id="navbar" class="navbar">
  <ul>
    <li><a href="/">Головна</a></li>
    <li><a href="/events">Події</a></li>
    <li><a href="/venues">Локації</a></li>
    <? if($name['id'] != 0) {?>
    <li class="dropdown"><a><span>Дії</span> <i class="bi bi-chevron-down dropdown-
indicator"></i></a>
      <ul>
        <li><a href="http://bozbo/events/add">Додати подію</a></li>
        <li><a href="http://bozbo/venues/add">Додати локацію</a></li>
      </ul>
    </li>
    <? } ;?>
    <li><a href="/chat">Чат</a></li>
    <li><a href="/contact">Про нас</a></li>
  </ul>
</nav><!-- .navbar -->

<div class="d-flex justify-content-center">
  <div class="position-relative">
    <a href="#" class="mx-2 js-search-open"><span class="bi-search"></span></a>

    <div class="search-form-wrap js-search-form-wrap">
      <form action="search" class="search-form d-flex flex-row" method="POST">
        <button name="search_btn" class="icon bi-search border-0"></button>
        <input type="search" name="search" placeholder="Пошук" class="form-control" required="">
        <button class="btn js-search-close"><span class="bi-x"></span></button>
      </form>
    </div><!-- End Search Form -->
  </div>

  <div class="d-flex justify-content-center">
    <i class="bi bi-list mobile-nav-toggle"></i>
    <nav id="navbar" class="navbar2">
      <ul>
        <? if (!$userModel->IsAuth()) : ?>
        <li><a href="/users/login">Увійти</a></li>
        <li><a href="/users/register">Рєєстрація</a></li>
        <? else : ?>
        <? if($name['access'] == 1) {?>
        <ul><li><a href="/users/logout">Вийти <b><?=$name['firstname'];?></b>★</a></li></ul>
        <? } else {?>
        <ul><li><a href="/users/logout">Вийти <b><?=$name['firstname'];?></b></a></li></ul>
        <? } ;?>
      </li>
    </ul>
  </div>

```

```

        <? endif; ?>
    </ul>
</nav>
</div>
</div>
</div>
<? if($MessageText != null){?>
<div class="container-fluid container-xl d-flex align-items-center justify-content-between">
<div class="mt-3 btn btn-outline-dark bg-dark text-white"><p class="m-2"><?=$MessageText?></p></div>
</div>
<?};?>
</header><!-- End Header -->

<!-- ===== Content ===== -->
<?=$PageContent ?>

<!-- ===== Footer ===== -->
<footer id="footer" class="footer">
<div class="footer-legal">
<div class="container">
<div class="row justify-content-between">
<div class="col-md-12 text-center mb-md-0">
<div class="copyright">
    © Авторські права <strong><span>BOZBO</span></strong>. Усі права захищені
</div>
<div class="credits">
    Сайт створив <a href="/">Бойко Дмитро</a>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>

</footer>

<a href="#" class="scroll-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

<!-- Vendor JS Files -->
<script src="assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
<script src="assets/vendor/swiper/swiper-bundle.min.js"></script>
<script src="assets/vendor/glightbox/js/glightbox.min.js"></script>
<script src="assets/vendor/aos/aos.js"></script>
<script src="assets/vendor/php-email-form/validate.js"></script>

<!-- Template Main JS File -->
<script src="assets/js/main.js"></script>

<script src="/alien/ckeditor/ckeditor.js"></script>
<script>

```

```
CKEDITOR.replace('text');
</script>

</body>
</html>
```