МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПОДІЙНО-ОРІЄНТОВАНОГО ІНТЕРНЕТ-МАГАЗИНУ З ВИКОРИСТАННЯМ PHP ФРЕЙМВОРКУ LARAVEL»**

Виконав: студент 2 курсу, групи 8.1218

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

Д.В. Гнап
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, доцент, к.ф.-м.н.

_____ Лісняк А.О.
(підпис)

« 29 » травня 2019 р

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гнапу Денису Володимировичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка подійно-орієнтованого інтернет-магазину з використанням PHP фрейморку Laravel

керівник роботи Мухін Віталій Вікторович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи 26 грудня 2019 року

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі
2. Основні теоретичні відомості

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

Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 29.05.2019

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	05.09.2019	
2.	Збір вихідних даних.	06.09.2019	
3.	Обробка методичних та теоретичних	21.09.2019	
	джерел.		
4.	Розробка першого та другого розділу.	07.10.2019	
5.	Розробка третього розділу.	21.11.2019	
6.	Оформлення та нормоконтроль	27.12.2019	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	15.01.2020	

Студент _____ Д.В. Гнап
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка подійно-орієнтованого інтернет-магазину з використанням PHP фрейморку Laravel»: 55 сторінки, 34 рисунки, 16 джерел.

BACKEND, BOOTSTRAP, CMS, COMPOSER, CSS, FRONTEND, HTML, HTTP, JAVASCRIPT, LARAVEL, MVC, PHP, WEB.

Об’єкт дослідження – можливості використання мови програмування PHP та фреймворку Laravel для розробки інтернет магазину.

Мета роботи: розробка подійно-орієнтованого інтернет-магазину з використанням мови програмування PHP та фреймворку Laravel.

Методи дослідження – аналітичний, практичний.

У роботі досліджуються можливості мови програмування PHP і фреймворка Laravel для розробки інтернет-магазину. Розглядається процес проектування і розробки системи з використанням веб технологій фронтенда і бекенда.

В роботі представлений процес проектування і розробки подійно-орієнтованого інтернет-магазину білетів.

Master’s Qualifying Thesis «Development of Event-Oriented Online Stores Using PHP Laravel Framework»: 56 pages, 34 images, 16 sources.

BACKEND, BOOTSTRAP, CMS, COMPOSER, CSS, FRONTEND, HTML, HTTP, JAVASCRIPT, LARAVEL, MVC, PHP, WEB.

The object of the study is the ability to use the PHP programming language and Laravel framework to develop an online store.

The aim of the study is development of an event-oriented online store using PHP programming language and Laravel framework.

The methods of research are analytical and practical.

The project explores the possibilities of PHP programming language and Laravel framework for developing an online store. The process of designing and developing the system using web front and backend technologies is considered.

The process of designing and developing an event-oriented online ticket store is presented in the project.

ЗМІСТ

ВСТУП

Обрана тема вважається актуальною на сьогоднішній день, тому що сьогодні мільйони людей щодня, не виходячи з дому, купують різні товари в електронних магазинах. У світі, а зокрема в Україні, величезними темпами зростає кількість користувачів Internet і, як наслідок, кількість «електронних» покупців і потенційних «електронних» покупців.

Електронні магазини істотно зменшують витрати виробника, заощадивши на утриманні звичайного магазину, розширюють ринки збуту, так само як і розширює можливість покупця – купувати будь-який товар в будь-який час в будь-якій країні. Це дає електронним магазинам велику перевагу перед звичайними магазинами. Цей момент є істотним при переході виробників з «звичайної» торгівлі на «електронну».

Висока якість продукції, вміння донести інформацію про продукт до споживача і ефективна система збуту, робить підприємство успішним на ринку. У багатьох компаніях зустрічаються проблеми збуту, які заважають ефективно працювати відділу продажів, і не зникають навіть з підбором професійних продавців. Вирішити їх можна лише шляхом автоматизації процесу продажів. У вузькому і технологічному сенсі, під електронним бізнесом раніше розумілося використання інформаційних технологій (в першу чергу пов'язаних з Інтернетом) для організації взаємодії підприємства із зовнішнім середовищем, включаючи постачальників, споживачів, партнерів тощо. При такому підході електронний бізнес виступає, перш за все, як досить складна прикладна інформаційна система. Більш широкий або концептуальний підхід розглядає електронний бізнес як спосіб підприємництва, що сприяє досягненню стратегічного успіху в нову інформаційну епоху. При такому розумінні електронний бізнес аж ніяк не зводиться до інформаційних технологій або активності в Інтернеті. Електронна комерція зачіпає всі аспекти бізнесу, включаючи стратегію,

процеси, організацію та технологію, і виводить його далеко за сформовані рамки.

Однією з найактуальніших галузей інтернет торгівлі, які розвиваються швидкими темпами, є продаж квитків. З недавнього часу з'явилася можливість купівлі квитків на майбутні заходи в певному місті, такі як: театр, кіно, концерт виконавця тощо. Кожна людина може знайти інформацію про свого улюбленого артиста або улюблений фільм в інтернеті і подивитися які найближчі події будуть проводитися в його місті. Інтернет магазин квитків - це можливість дізнатися інформацію про майбутні заходи і придбати квитки.

Оформлення замовлення проводиться таким чином: людина вибирає товар і оплачує його всіма доступними способами. Є невід’ємною частиною включення в список не тільки оплату банківською картою, але і за допомогою електронних грошей. Також є можливість повернення квитків в магазин якщо людина не може відвідати захід.

При розробці будь-якого програмного продукту, слід розглянути і проаналізувати вже існуючі сайти та системи для уникнення подібних помилок і недоліків.

Мета роботи полягає в розробці подійно-орієнтованого інтернет-магазину з використанням PHP фреймворку Laravel.

Для виконання даної мети в роботі поставлені наступні задачі:

– провести аналіз існуючих подійно-орієнтованих інтернет-магазинів;
– розробити інформаційну модель системи (у вигляді діаграм потоків даних, мовою UML, за допомогою блок-схем алгоритмів тощо);
– здійснити реалізацію системи у вигляді сайту подійно-орієнтованого магазину с використанням мови програмування PHP (фреймворку Laravel).

1 АНАЛІЗ ІСНУЮЧИХ ДЖЕРЕЛ

1.1 Аналіз існуючих подійно-орієнтованих інтернет-магазинів

На сьогоднішній день великою популярністю користуються онлайн магазини. Все частіше і частіше люди починають використовувати інтернет магазини для покупки тих чи інших товарів. Зокрема онлайн покупка квитків. Це має безліч позитивних сторін, таких як:

- різноманітність і зручність послуг, що надаються;
- величезна кількість заходів, які рекламуються в інтернеті;
- зручна навігація на сайтах-магазинах квитків;
- можливість замовлення квитка віддалено;
- можливість використання не тільки комп'ютерів, але і будь-яких гаджетів, таких як: телефон, планшет тощо.
- оплата проводиться картою або електронним гаманцем;
- можливість повернення квитка.

На рисунку (1.1) можна побачити тенденцію онлайн продажу за останні декілька років.

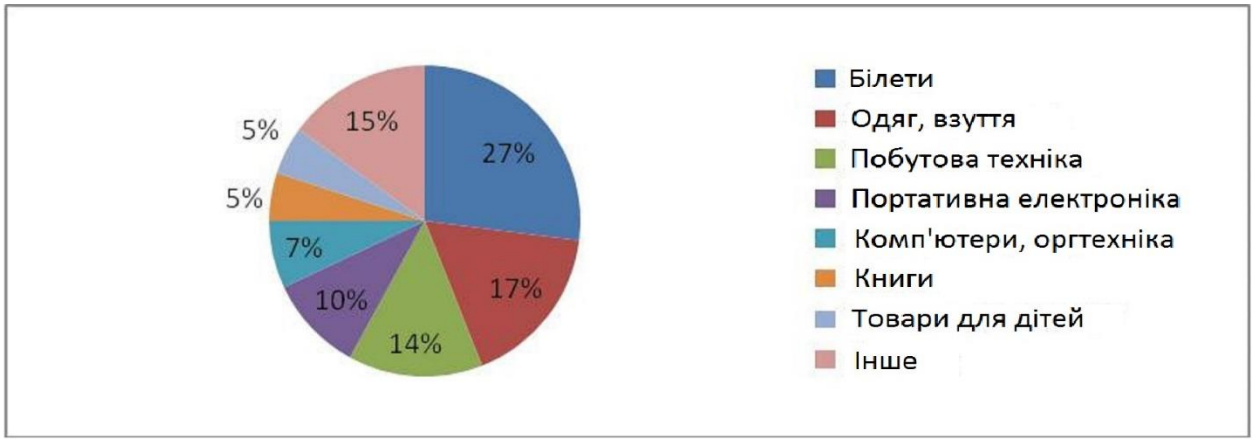

Рисунок 1.1 – Співвідношення онлайн продажів

Веб-сайт сьогодні є не тільки інформаційним засобом або візиткою, а повноцінним маркетинговим інструментом, що привертає нових клієнтів, що приносить прибуток. Існує величезна кількість довідково-інформаційних сайтів, що надають повну інформацію майже з будь-якого запиту. Набагато легше зайти на сайт і дізнатися все необхідне, ніж шукати інформацію в газетах і журналах.

Офіційної класифікації корпоративних інтернет-сайтів не існує. Тим не менше, більшість розробників групує проекти схожим чином. Ось найбільш поширена класифікація:

– сайт візитка;
– сайт вітрина;
– інтернет магазин;
– корпоративне представництво;
– промо-сайт;
– інформаційний проект;
– портал.

На сьогоднішній день існує величезна конкуренція на ринку, яка змушує розробників йти в ногу з часом. Визначення критеріїв ефективності є ключовим пунктом при побудові та супроводі сайту. Вибрані критерії є основним фактором, що визначає формування сайту. Однак в значній частині випадків на початку інтернет-проекту взагалі ніяких критеріїв не визначається.

Проводячи аналіз вимог, необхідних для розробки подійно-орієнтованих магазинів, можна виділити наступні [2,4,6]:

– зрозумілість і інформативність контенту;
– структура, тобто зручне розміщення інформації на сайті;
– гарне оформлення і дизайн;
– оновлення контенту;
– зручна назва і адреса в мережі Інтернет;
– швидкість завантаження сайту;

- продумана система зв'язку з відвідувачем;
- індексація сайту, його просування;
- реклама;
- інтеграція з соціальними мережами;
- впровадження платіжних систем в процес оплати.

Контент сайту – це вся інформація, що на ньому розміщена (тексти, відео, аудіо, зображення, фотографії). Від змісту сайту безпосередньо залежить його відвідуваність. Відвідувачі в першу чергу шукають корисну інформацію. Тому, тільки якісний і унікальний контент може гарантувати успіх.

Існують такі види контенту:

- текстове наповнення сайту(копірайтинг – унікальний авторський контент, який використовують для підвищення лояльності відвідувачів до товарів / послуг сайту);
- аудіонаповнення сайту це музика та аудіозаписи;
- відеонаповнення сайту це відео ролики, кліпи та скріншоти;
- зображення це – фотографії, анімація, картинки;
- заголовки, вони дуже важливі для просування сайту, вони повинні бути унікальними і інформативними, містити ключові фрази.

Структура відповідає за дві важливих складових успішності сайту. Від її правильності та логічної побудови залежить зручність користувача. Якщо структура розроблена неправильно, навігація незручна для пошуку споживачем необхідної категорії / підкатегорії, то він надовго не затримається на сайті, закривши вкладку з сайтом.

Структура сайту є найважливішим технічним інструментом з точки зору SEO. Неправильна побудова структури сайту значно ускладнює просування користувачів по сайту. Тому при розробці архітектури ресурсу, необхідно аналізувати розміщення кожного розділу і підрозділу, щоб все зробити правильно, задовольнивши потреби користувача і відповівши на вимоги пошукових роботів.

Перше враження від сайту користувачеві формує у нього бажання залишитися або ж залишити ресурс. Даний фактор є набагато більш важливим для утримання клієнта, ніж наповненість порталу. Пов'язано це з тим, що всього за кілька мілісекунд користувач встигає скласти чітке враження про веб-ресурсі і прийняти рішення, чи треба йому далі їм користуватися або ні.

Існує кілька найбільш важливих розділів, на яких загострюється увага відвідувача сайту. Серед них дослідники виділяють:

- логотип, який привертає погляд довше всіх інших розділів при першому перегляді сайту;
- головне меню навігації, вивчаючи яке користувач оцінює функціональність ресурсу;
- вікно пошуку, що змушує відвідувача затриматися на деякий час;
- головне зображення сторінки;
- текстовий контент;
- футер.

Всі ці компоненти складають дизайн сайту. Саме вони на 90% створюють перше враження про нього. Це було доведено багатьма дослідження, в тому числі і спеціальними тестами, які проводив Google для оцінки поведінкових реакцій користувачів мережі.

Всім відомо, що повільна робота сайту – це погано. Якщо сайт періодично пригальмовує, то у відвідувачів виникають серйозні проблеми при вирішенні своїх завдань. Але навіть, якщо ситуація з веб-ресурсом нормальна (сайт завантажується швидко на більшості пристроїв), то невеликі затримки у відображенні сайту стають причиною втрати аудиторії і падіння відсотка конверсії.

Велика частина інтернет-аудиторії для відвідування сайтів сьогодні використовує мобільні пристрої. А в пристроях канали доступу і внутрішні ресурси значно повільніше, ніж в персональних комп'ютерів.

Ще дуже важливий технічний аспект проблеми, який є третьою причиною поганої швидкості. Варто розуміти, що повільні сайти в основному

використовують більше ресурсних засобів на хостингу. А це додаткові витрати. Затримки при обробці даних серверами дозволяють зменшити пікові навантаження на сайт.

Форма зворотнього зв'язку є однією з найголовніших складових сучасного сайту. Причин цього чимало. В першу чергу, використання контактної форми в значній мірі полегшує відправку листів, завдяки спеціальним полям, відведеним для написання тексту. Подібний шаблон робить замовлення тієї чи іншої послуги більш зручним. Також плюсом є і те, що користувачам не потрібно вводити адресу електронної пошти власника сайту, так як в формі зворотного зв'язку цей момент вже передбачено. Більш того відвідувач навіть не дізнається вашої електронної адреси, що буде потенційної захистом від спаму.

Серед величезної кількості плюсів, які має форма зворотного зв'язку, важко було б розглядати недоліки, навіть якщо б вони були. Крім зручності для власника ресурсу і користувача, вона ще й істотно заощаджує час. Це відбувається завдяки тому, що людині, яка хоче поставити запитання, не потрібно витрачати дорогоцінні хвилини на запуск поштового клієнта, щоб відправити лист. Все що потрібно – це перейти на сторінку з формою зворотного зв'язку, ввести в призначених для цього полях ім'я, адреса електронної пошти, потрібний текст і кнопку відправки. Що стосується великих компаній, то тут при виборі адресата багато користувачів просто губляться, так як часто вказано безліч різних електронних скриньок. Більш того, при правильному використанні скриньки можна значною мірою підвищити конверсію.

Форма зворотнього зв'язку вже давно є найбільш прийнятним способом спілкування між власником ресурсу і користувачем.

В першу чергу, використання контактної форми в значній мірі полегшує відправку листів, завдяки спеціальним полях, відведених для написання тексту. Подібний шаблон робить замовлення тієї чи іншої послуги більш зручним. Також плюсом є і те, що користувачам не потрібно вводити адресу

електронної пошти власника сайту, так як в формі зворотного зв'язку цей момент вже передбачено.

Крім зручності для власника ресурсу і користувача, вона ще й істотно заощаджує час. Це відбувається завдяки тому, що людині, яка хоче поставити запитання, не потрібно витрачати дорогоцінні хвилини на запуск поштового клієнта, щоб відправити лист. Все що потрібно – це перейти на сторінку з формою зворотного зв'язку, ввести в призначених для цього полях ім'я, адреса електронної пошти, потрібний текст і кнопку відправки.

На сьогоднішній день дуже поширені соціальні мережі, тому важливим аспектом є інтеграція сайту з соціальними мережами. Іконки соціальних мереж, провідні на «представництво» вашого сайту в соціальних мережах (Фейсбук, Твіттер, Гугл тощо). Також Кнопки соціальних мереж, що дозволяють користувачеві легко «поділитися» зі своїми друзями посиланням на цікавий або корисний контент.

Адаптивність сайту – це можливість сайту правильно відображатися на різних пристроях з різними характеристиками.

Сьогодні сайти проглядаються на персональних комп'ютерах, планшетах, смартфонах. Кожен пристрій зі своїми характеристиками – швидкістю роботи, дозволом екрану, частотою. Якщо сайт не адаптован, то на комп'ютері він відобразиться правильно, а на телефоні може "поїхати" – блоки накладуться один на одного, шрифт може стати нечитабельним.

Технології розвиваються стрімкими темпами, виробники техніки не відстають від них. Смартфони, планшети, телевізори, інші електронні пристрої, що підключаються до мережі інтернет, випускаються на будь-який смак і бюджет, варіантів розміру екрану безліч. При цьому споживання контенту збільшується в геометричній прогресії. У боротьбі за клієнта власники сайтів використовують всі засоби і методи. Зокрема, на етапі розробки закладають можливість коректного відображення свого веб-ресурсу на різних пристроях.

Принципи адаптивності:

– потоковість;
– відносність одиниць вимірювання;
– використання контрольних точок;
– мінімальні і максимальні значення;
– вкладеність об'єктів;
– правильні шрифти;
– дотримання розмірів макетів;
– медіа-запити.

В Україні існує деяка кількість інтернет магазинів, які продають квитки. Виділимо найпопулярніші з них, які з'являються на першій сторінці в пошуку після запиту:

– PARTER (https://parter.ua/);
– tickethunt (https://tickethunt.net/);
– CARABAS (https://karabas.com/);
– CONCERT.UA (https://concert.ua/).

Назва	Зрозумілість контенту	Зручна структура	Адаптивність	Зручна назва та адреса в мережі	Швидкість сайту	Реклама	Наяіність платіжних систем
parter	✓	✓	✓	✓	Ні	✓	✓
tickethunt	✓	Ні	✓	✓	✓	Ні	✓
carabas	✓	✓	✓	✓	✓	✓	✓
concert	✓	✓	Ні	✓	Ні	Ні	✓

Рисунок 1.2 – Порівняння існуючих систем

Таким чином, можна зробити висновок, що існуючі системи, а саме інтернет-магазини квитків не досконалі (див. рис. 1.2). У деяких магазинах відсутня зручна структура сайту, а в деяких не реалізована адаптивність для різних пристроїв.

1.2 Аналіз інструментів для реалізації

Для реалізації інтернет магазину була вибрана мова програмування PHP. Це сучасна мова програмування, яка користується великим попитом на ринку і є однією з лідируючих мов на даний момент.

PHP (Personal Home Page Tools) – скриптова мова загального призначення, інтенсивно застосовується для розробки веб-додатків. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов, що застосовуються для створення динамічних веб-сайтів[4].

PHP входить в число мов з динамічною типізацією. Це означає, що тип даних визначається не при оголошенні змінної, а при присвоєнні значення. Також це мова, що інтерпретується. Написані програми інтерпретуються в момент звернення за допомогою спеціальних програм. Інтерпретуються язики не залежать від платформи, але поступаються компільовані мов в швидкості виконання.

На PHP написані популярні системи управління контентом (CMS), наприклад: WordPress, Joomla !, Drupal. На одному тільки WordPress'e працює близько третини від усіх сайтів в інтернеті. Це підтверджує актуальність і популярність PHP. Цією мовою написані і популярні фреймворки для створення сайтів, наприклад, Laravel, Yii2, Symfony.

Мова програмування PHP швидко розвивається та має безліч фреймворків. PHP-фреймворки мають величезні можливості і унікальні екосистеми, які підійдуть під величезну кількість завдань. Вони здатні створювати краще сформовані, безпечні і зрозумілі додатки та вебсайти за більш короткі проміжки часу.

Перерахуємо найпоширеніші фреймворки:

- Laravel;
- Symfony;
- Zend framework;
- Yii2 framework;

– Phalcon;
– CakePHP.

Кожен фреймворк призначений для вирішення конкретного типу завдань, але найпопулярнішим фреймворком для веб розробки є Laravel(див. рис. 1.3).

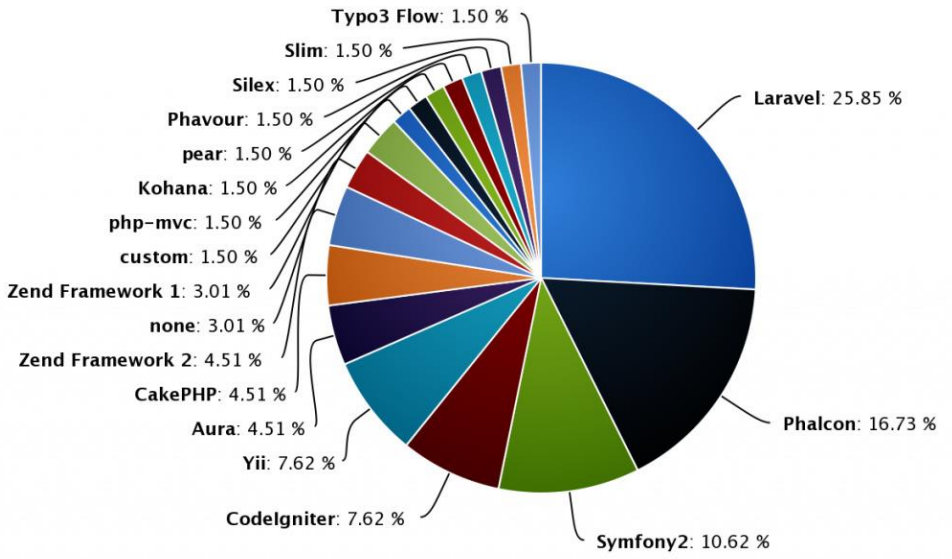

Рисунок 1.3 – популярність PHP фреймворків

Головною метою Laravel є створення багатих функціоналом веб-додатків. Laravel має багате ядро, яке виконує велику кількість звичайних завдань. Цей фреймворк володіє таким функціоналом як RESTful-роутинг, кешування, управління користувачами і аутентифікація. Завдяки всьому цьому Laravel прискорює процес розробки.

Перерахуємо деякі з плюсів Laravel:

– швидке і функціональне ядро;
– проста маршрутизація;
– вбудований шаблонизатор Blade;
– дуже гнучкі можливості для написання REST API;
– інтегрована система модульного тестування;
– великий обсяг документації.

Для зберігання даних інтернет магазину була обрана MySQL база даних. Це одна з лідируючих СУБД для малих і середніх веб проектів.

MySQL – вільна реляційна система управління базами даних. У кожній версії даного продукту виходить його безкоштовна редакція, і з досить хорошими технічними характеристиками.

Особливості СУБД MySQL:

– найчастіше використовується в якості віддаленого сервера;
– включає в себе велику кількість типів таблиць;
– поставляється із спеціальним типом EXAMPLE, що показує принцип створення нових таблиць;
– високий ступінь масштабованості за рахунок підтримки більшості популярних платформ;
– відкритий код – завдяки цьому дана СУБД постійно вдосконалюється і модернізується;
– максимальний розмір файлу таблиці бази даних обмежується лише можливостями операційної системи.

1.3 Технічне завдання

Визначимо головні задачі, які необхідні для забезпечення успіху проекту, а саме створення подійно-орієнтованого інтернет-магазину.

Складемо алгоритм роботи над майбутнім проектом:

– визначення майбутньої аудиторії користувачів;
– список конкретних вимог до системи, її функціонал;
– опис плану створення унікального дизайну;
– реалізація серверної частини проекту;
– реалізація клієнтської частини проекту;
– забезпечення взаємодії з базою даних;
– тестування системи.

Інтернет магазин це система, яку використовують люди для покупки тих чи інших товарів, в конкретному випадку це квитки на заходи. Складемо список можливостей користувача, які потрібно реалізувати в системі:

- реєстрація або авторизація в системі;
- можливість перегляду контенту;
- вибір бажаних товарів;
- оформлення замовлення через кошик;
- можливість редагування кошика;
- оплата замовлення через декілька можливих платіжних систем.

При розробці системи використовується архітектура MVC та її строгі вимоги, а саме взаємодія між трьома складовими: уявленнями, моделями і контролерами.

Для реалізації візуальній складовій виділені наступні вимоги:

- реалізувати навігаційну панель на сайті;
- розмістити інформаційний слайдер з зображеннями;
- наявність панелі класифікації товарів для зручності;
- реалізація блоків особливих квитків (анонсів), які повинні містити в собі: назву заходу, дату проведення, опис, вартість квитка;
- наявність зручного футера, який містить в собі: інтеграцію з соціальними мережами, зворотний зв'язок, інформацію про розробників.

ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Постановка задачі

Для початку процесу розробки сайту потрібно приділити час на планування архітектури майбутнього проекту і провести роботу над створенням алгоритму роботи над проектом.

Створення сайту – досить великий проект. Навіть якщо ресурс невеликих розмірів він може зіграти істотну роль у розвитку бізнесу, а тому навіть для односторінкового сайту потрібен великий обсяг робіт і ретельне опрацювання всіх деталей.

Виділимо основні етапи розробки:

- визначення цілей і задач web-сайту;
- вивчення цільової аудиторії;
- створення технічного завдання (ТЗ);
- прототипування;
- створення макетів дизайну;
- верстка;
- програмування;
- базове наповнення;
- тестування;
- здача готового проекту.

Були перераховані основні етапи розробки веб-ресурсу. Але цей алгоритм може змінюватися в залежності від підходів до розробки.

Розглянемо детальніше майбутній вигляд сайту, а саме його головну сторінку (див. рис. 2.1).

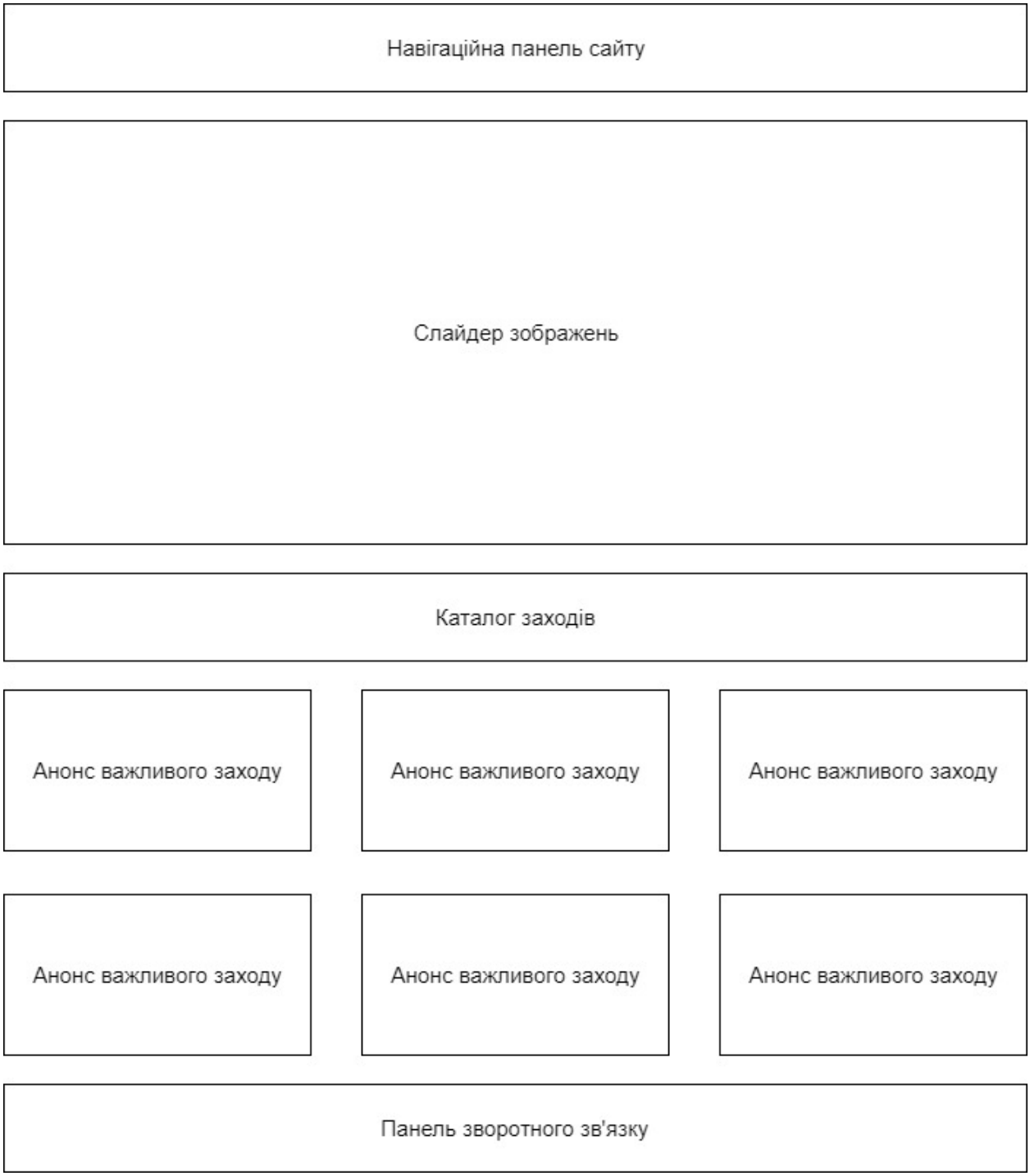

Рисунок 2.1 – Зовнішній вигляд сайту

Вгорі сайту розташовується навігаційна панель, з посиланнями на головну сторінку і інформацію про сайт. Також на верхній панелі користувач може знайти кнопки реєстрації та авторизації. Ця панель є одним з головних елементів сучасного сайту за сучасними стандартами верстки.

Нижче розташовується слайдер з інформаційними зображеннями. Слайдер – це спеціальний елемент веб-дизайну, який представляє собою блок певної ширини найчастіше в шапці веб-сторінки. Головна перевага в умовах, що змінюються в ручному або автоматичному режимі елементах – картинок, текстів і посилань [7].

Однією з найважливіших причин, по якій на сайті використовуються слайдери – це поведінка сучасних користувачів: зараз практично всі вважають за краще графічний вигляд інформації текстовій. Інтернет-користувачі хочуть отримувати максимум корисного контенту за мінімум часу. Якщо сайт цього не може забезпечити, то велика ймовірність того, що відвідувач піде на інший ресурс.

Нижче слайдера на головній сторінці ми зможемо знайти каталог типів заходів, де можна буде вибрати тип заходу: кіно, театр, цирк, для дітей тощо. Це зроблено для зручності пошуку по сайту. Наприклад, якщо користувач хоче купити квиток на певний захід, то йому просто слід вибрати вид і він отримає перелік усіх заходів.

Під каталогом ми можемо побачити анонси актуальних на даний момент подій. Вони виділені в спеціальні блоки, де розташовано назву заходу, його опис і зображення. Це дуже зручно, коли всі найактуальніші заходи розташовані на головній сторінці.

Внизу сторінки розташована панель зворотного зв'язку. Тут користувач може знайти різноманітні контакти для зв'язку з адміністрацією сайту.

2.2 Особливості шаблону MVC та систем на його основі

Не зважаючи на використання надсучасної мікросервісної архітектури тільки у майбутній перспективі, технології, що реально застосовані у проекті, також є досить актуальними, зважаючи хоча би на частоту використання саме Laravel при розробці сучасних веб-додатків.

Так даний фреймворк базується на використанні шаблону проектування MVC, який розшифровується як Model-View-Controller, тобто Модель-Вид-Контроллер.

Згідно з цим підходом, увесь програмний код (тобто весь додаток) слід розбити на три змістовних компоненти (див. рис. 2.2), відповідно:

– модель (інкапсулює бізнес-логіку додатку);
– вид або Представлення (керує зовнішнім виглядом додатку);
– контролер (реалізує зв’язок додатку із користувачем).

Модель є головною складовою програмного забезпечення, оскільки містить логіку (бізнес-логіку) виконання програми. Тут містяться усі дані, що мають сенс для заданої прикладної предметної області, і усі методи, що їх обробляють. Оскільки використання паттернів проектування саме по собі за умовчанням уже передбачає використання об’єктно-орієнтованої технології програмування (хоча насправді цілком можливо будувати програмне забезпечення за шаблоном MVC і за структурною технологією програмування, але, однак, це абсолютно не прийнято), то можна сказати, що у Моделі містяться усі *класи*, що реалізують бізнес-логіку додатку.

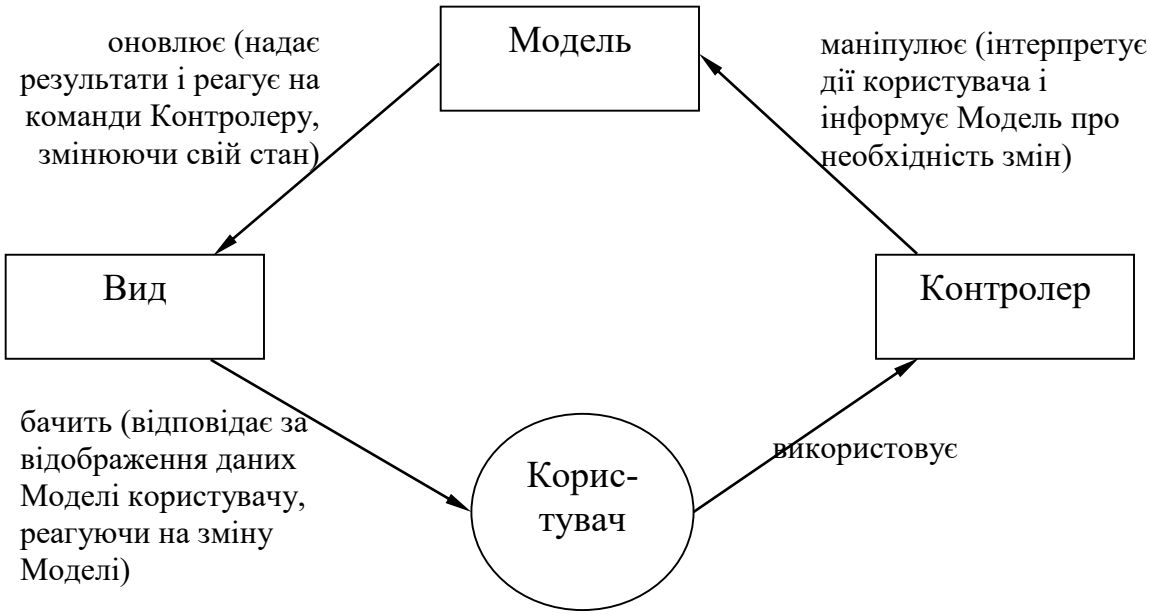

Рисунок 2.2 – Архітектура шаблону проектування MVC

Вид або Представлення є ще одним (із трьох) компонентів системи, який забезпечує зовнішній вигляд інформації, яка міститься та надається Моделлю, для користувача (сюди можна включати різноманітні відрисовки, розподіл структурованої інформації по елементам управління, одним словом різні способи надання інформації для людини). Рівень Представлення у великій мірі еквівалентний поняттю інтерфейса користувача системи, а саме визначає, як буде виглядати додаток у різних його станах.

Останнім компонентом системи є Контролер, який командує завантаженням методів Моделі та Виду, тобто здійснює загальне управління процесом роботи програмного забезпечення. Тут містяться методи-обробники усіх дій користувача у його користувацькому інтерфейсі. Усі три компоненти реалізуються окремими класами (або трьома наборами класів, якщо додаток дуже складний і кожна із трьох компонент підлягає подальшій декомпозиції, або хоча б деякі з них).

Відмітимо, що використання шаблону MVC є основним принципом роботи у фреймворку Laravel, тому приймаємо його у якості базового в даному дослідженні.

2.3 Use case моделювання

Діаграма варіантів використання, або діаграма прецедентів (англ. Use case diagram) в UML – діаграма, що відображає відносини між акторами і прецедентами і є складовою частиною моделі прецедентів, що дозволяє описати систему на концептуальному рівні [11].

Прецедент – можливість модельованої системи (частина її функціональності), завдяки якій користувач може отримати конкретний, вимірний і потрібний йому результат. Прецедент відповідає окремому сервісу системи, визначає один з варіантів її використання і описує типовий спосіб

взаємодії користувача з системою. Варіанти використання зазвичай застосовуються для специфікації зовнішніх вимог до системи.

Основне призначення діаграми – опис функціональності і поведінки, що дозволяє замовнику, кінцевому користувачеві і розробнику спільно обговорювати проектовану або існуючу систему.

При моделюванні системи за допомогою діаграми прецедентів потрібно:

– чітко відокремити систему від її оточення;
– визначити дійових осіб (акторів), їх взаємодія з системою і очікувану функціональність системи;
– визначити в глосарії предметної області поняття, які стосуються детального опису функціональності системи (тобто прецедентів).

Розглянемо вимоги системи інтернет магазину:

– користувач використовує веб сторінку інтернет магазину для перегляду інформації про заходи та покупки квитків на них;
– користувач може вибрати кілька квитків, після чого вони потрапляють в корзину;
– користувач переглядає кошик і оформлення замовлення на сайті;
– відбувається процес оплати замовлення.

В системі даного інтернет магазину ми розглянемо двох акторів, а саме зареєстрованого або незареєстрованого користувача (див. рис. 2.2).

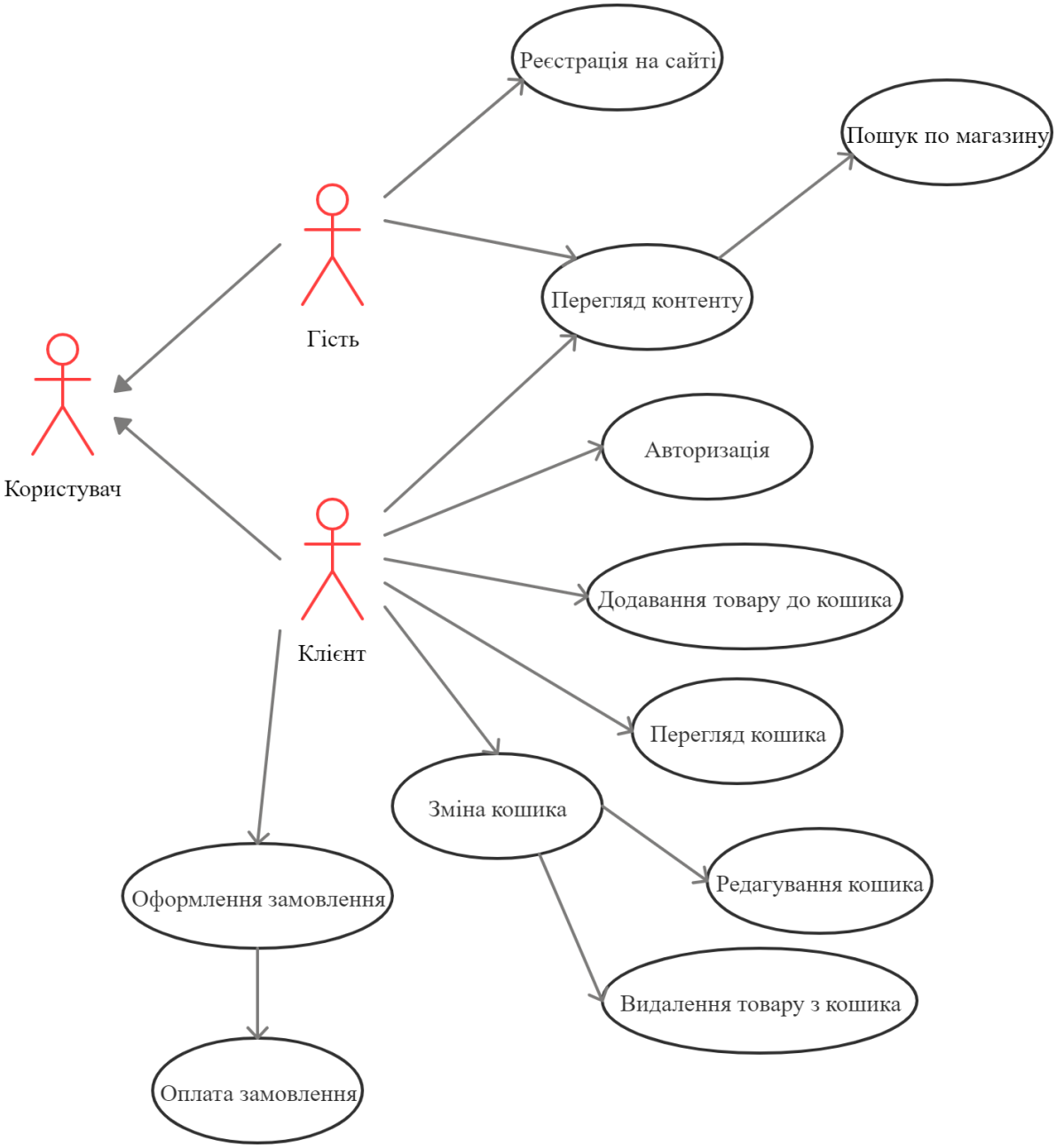

Рисунок 2.2 – Діаграма прецедентів

На діаграмі прецедентів ми можемо бачити двох акторів. Незареєстрований користувач може тільки переглядати контент сайту або створити собі обліковий запис. Зареєстрований користувач може використовувати весь функціонал сайту, а саме: авторизуватися, переглядати контент, здійснювати покупку і оплату.

2.4 Моделювання діяльності

При моделюванні поведінки проектованої або аналізованої системи виникає необхідність не тільки представити процес зміни її станів, але і деталізувати особливості алгоритмічної і логічної реалізації виконуваних системою операцій.

Алгоритмічні і логічні операції, що вимагають виконання в певній послідовності є однією з найголовніших властивостей системи. Важливо підкреслити ту обставину, що зі збільшенням складності системи суворе дотримання послідовності виконуваних операцій набуває все більш важливе значення.

Для моделювання процесу виконання операцій в мові UML використовуються так звані діаграми діяльності. Застосовувана в них графічна нотація багато в чому схожа з нотацію діаграми станів, оскільки на діаграмах діяльності також присутні позначення станів і переходів. Відмінність полягає в семантиці станів, які використовуються для уявлення не діяльностей, а дій, і у відсутності на переходах сигнатури подій. Кожний стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан спрацьовує тільки при завершенні цієї операції в попередньому стані. Графічно діаграма діяльності видається у формі графа діяльності, вершинами якого є стани дії, а дугами – переходи від одного стану дії до іншого.

Стан дії (action state) є спеціальним випадком стану з деякою вхідною дією і принаймні одним виходом зі стану переходом. Цей перехід неявно передбачає, що вхідна дія вже завершилася. Стан дії не може мати внутрішніх переходів, оскільки він є елементарним. Звичайне використання стану дії полягає в моделюванні одного кроку виконання алгоритму (процедури) або потоку управління.

Розглянемо діаграму активності інтернет магазину (див. рис. 2.3).

На діаграмі є одна точка входу і одна точка виходу. Користувач при купівлі квитків робить послідовність лінійних дій.

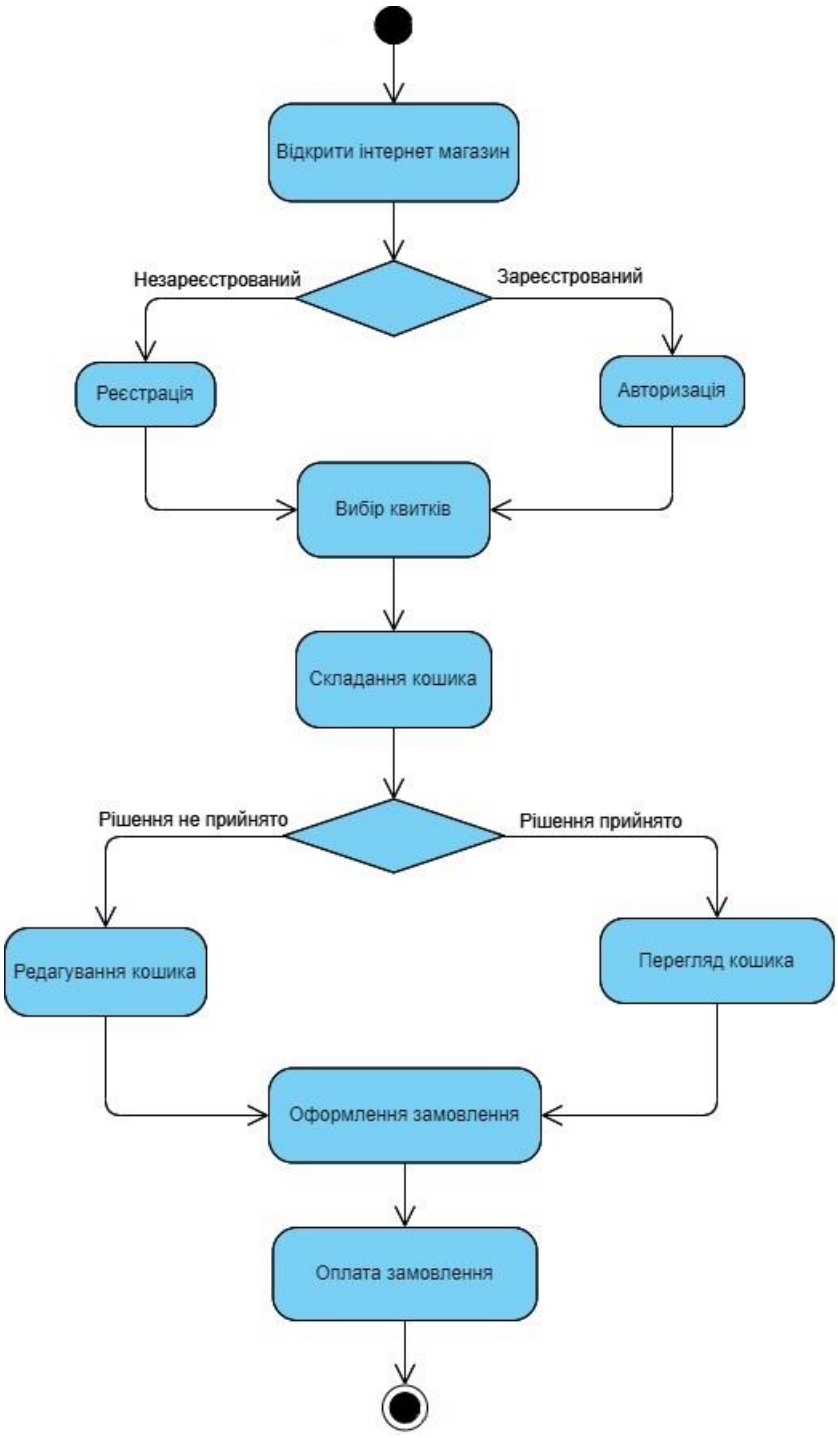

Рисунок 2.3 – Діаграма активності

Алгоритм купівлі квитків згідно діаграми активності:

– відкрити головну сторінку магазину;

– авторизуватися, якщо немає облікового запису, то зареєструватися;
– вибрати квитки для покупки;
– перевірити кошик із замовленням;
– оформити замовлення;
– розплатитися.

У загальному випадку дії на діаграмі діяльності виконуються над тими чи іншими об'єктами. Ці об'єкти або ініціюють виконання дій, або визначають деякий результат цих дій. При цьому дії специфікують виклики, які передаються від одного об'єкта графа діяльності до іншого. Оскільки в такому ракурсі об'єкти відіграють певну роль в розумінні процесу діяльності, іноді виникає необхідність явно вказати їх на діаграмі діяльності.

На початкових етапах проектування, коли деталі реалізації діяльностей в проектованій системі невідомі, побудова діаграми діяльності починають з виділення під-діяльностей, які в сукупності утворюють діяльність підсистем. В подальшому, в міру розробки діаграм класів і станів, ці під-діяльності уточнюються у вигляді окремих вкладених діаграм діяльності компонентів підсистем, якими виступають класи і об'єкти.

2.5 Моделювання класів

Діаграма класів займає центральне місце в проектуванні об'єктно-орієнтованої системи. Нотація класів використовується на різних етапах проектування і будується з різним ступенем деталізації. Мова UML застосовується не тільки для проектування, але і з метою документування, а також ескізування проекту. На діаграмі класів за допомогою спеціальних символів зображуються типи даних програми і відносини між ними, хоча в деяких випадках можуть використовуватися і деякі інші елементи - пакети і навіть екземпляри класів.

Розглянемо діаграму класів інтернет магазину (див.рис. 2.4).

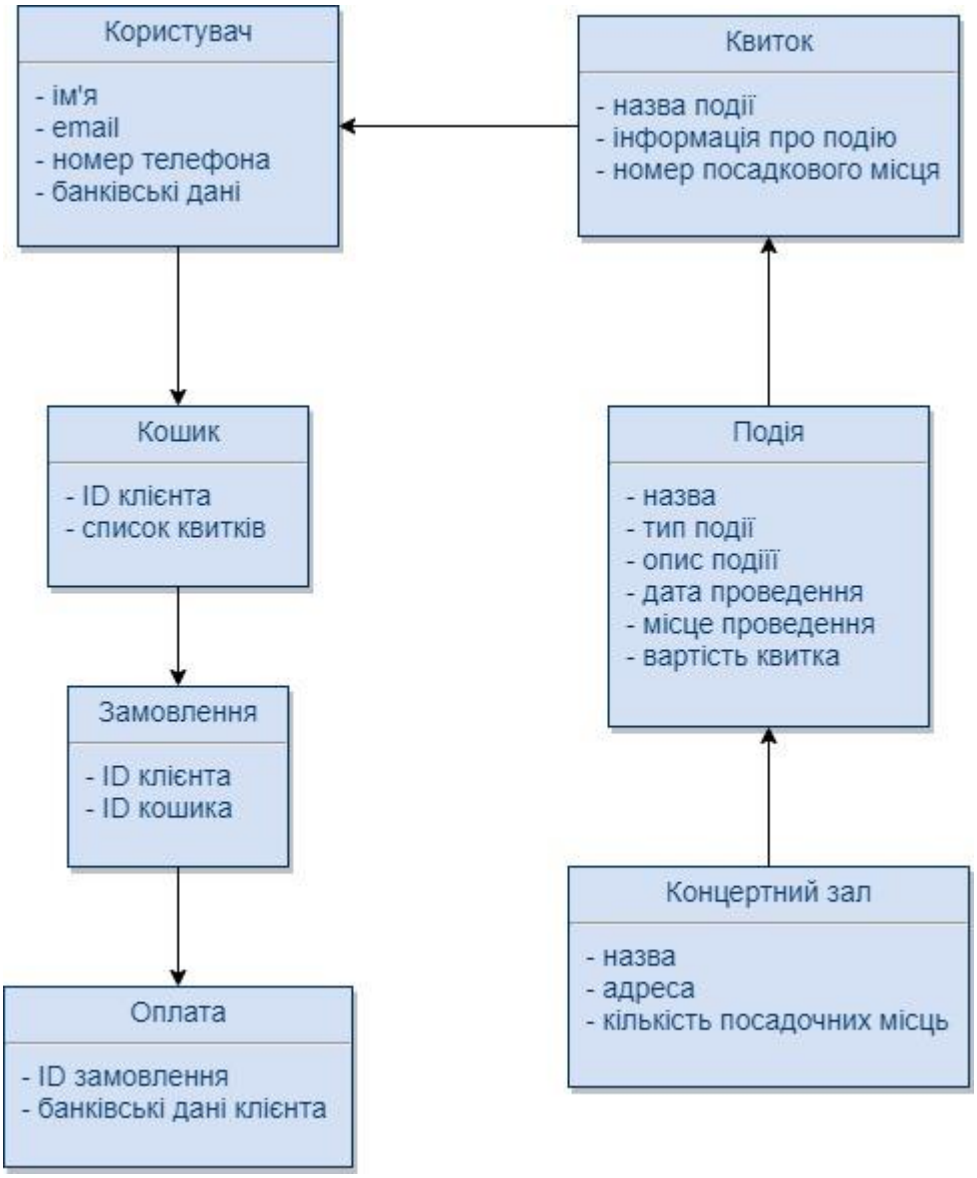

Рисунок 2.4 – Діаграма класів

На діаграмі класів ми можемо бачити залежність класів між собою. Так квиток залежить від конкретного заходу, який в свою чергу залежить від місця проведення цього заходу.

2.6 Моделювання взаємодій

Моделювання взаємодій (interaction modeling) охоплює питання взаємодії між об'єктами, необхідними для виконання прецеденту. Моделі взаємодії використовуються на більш розвинених стадіях аналізу вимог, коли стає відомою модель класів, так що посилання на об'єкти спираються на модель класів.

Взаємодія (interaction) являє собою набір повідомлень, якими обмінюються об'єкти відповідно до встановлених між ними зв'язків (останні можуть бути постійними або тимчасовими). Діаграма послідовностей представляється двовимірним графом. Об'єкти розташовуються по горизонталі. Послідовності повідомлень розташовуються зверху вниз по вертикалі. Кожна вертикальна лінія називається лінією життя (lifeline) об'єкта.

Стрілки являють кожне повідомлення, яке направляється від об'єкта відправника до операції (методу) об'єкта одержувача. Для кожного повідомлення, як мінімум, вказується його ім'я. Крім того, для повідомлення можуть бути вказані фактичні аргументи повідомлення і інша інформація, що управляє. Фактичні аргументи відповідають формальним аргументам методу об'єкта-одержувача.

Фактичні аргументи можуть бути вхідними аргументами (передаються від відправника до одержувача) або вихідними аргументами (передаються від одержувача назад до відправника). Вхідні аргументи можуть бути позначені ключовим словом in (якщо ключове слово відсутнє, то передбачається, що аргумент – вхідний). Вихідні аргументи позначаються ключовим словом out. Допускаються також аргументи типу inout ("вхідні-вихідні"), однак, для об'єктно-орієнтованого підходу вони не характерні.

Взаємодії можна моделювати на різних рівнях абстрагування. На найвищому рівні взаємодія системи з зовнішніми дійовими особами описується варіантами використання. Кожен варіант використання описує

елемент функціональності, що надається системою її користувачам. Варіанти використання корисні для подання в моделі неформальних вимог.

Розглянемо діаграму взаємодій інтернет магазину квитків (див. рис. 2.5).

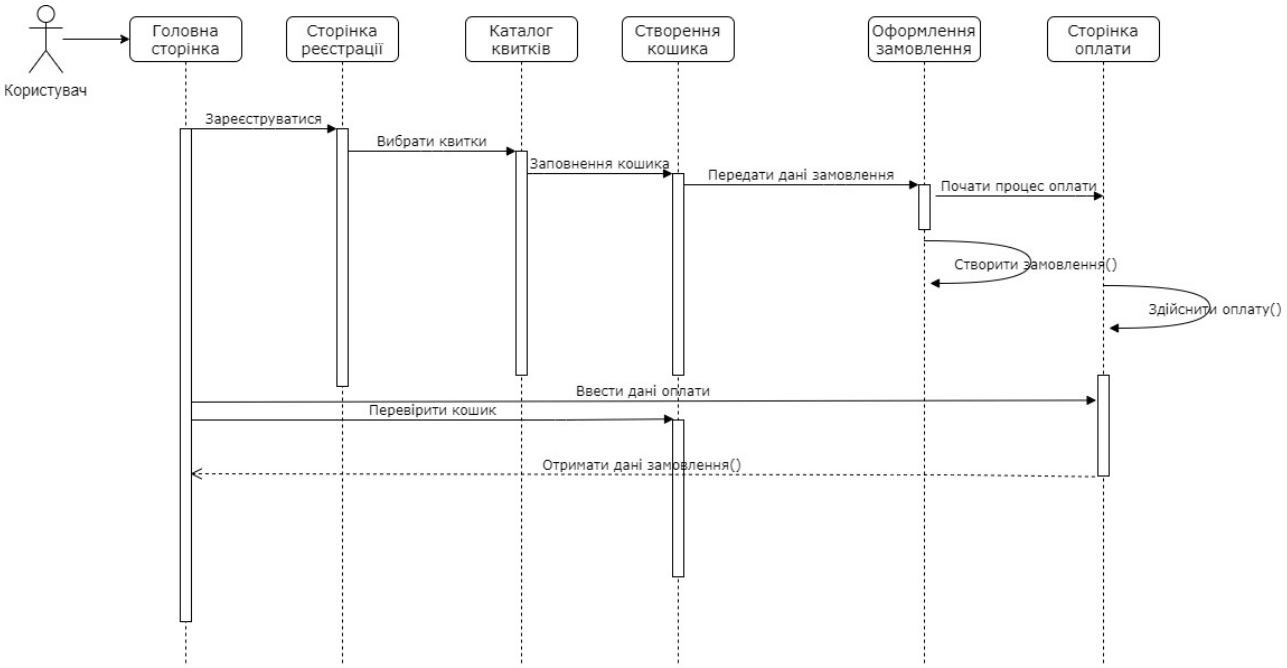

Рисунок 2.5 – Діаграма взаємодій

На даній діаграмі ми можемо бачити алгоритм взаємодії між різними етапами процесу купівлі квитків.

2.7 Моделювання станів

Головне призначення діаграми станів – описати можливі послідовності станів і переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу. Найчастіше діаграми станів використовуються для опису поведінки окремих екземплярів класів (об'єктів), але вони також можуть бути застосовані для специфікації функціональності інших компонентів моделей, таких як варіанти використання, актори, підсистеми, операції і методи.

Діаграма станів по суті є графом спеціального виду, який представляє певний автомат. Вершинами цього графа є стану і деякі інші типи елементів автомата, які зображуються відповідними графічними символами. Дуги графа служать для позначення переходів зі стану в стан. Для розуміння семантики конкретної діаграми станів необхідно представляти не тільки особливості поведінки модельованої сутності, а й знати загальні відомості з теорії автоматів [8].

Діаграма станів має такі властивості:

– фіксує динаміку зміни стану класів;
– описує поведінку об'єктів в рамках декількох прецедентів;
– стан об'єкта позначається поточним значенням атрибута об'єкта.

На рисунку 2.6 представлена діаграма станів інтернет магазину квитків.

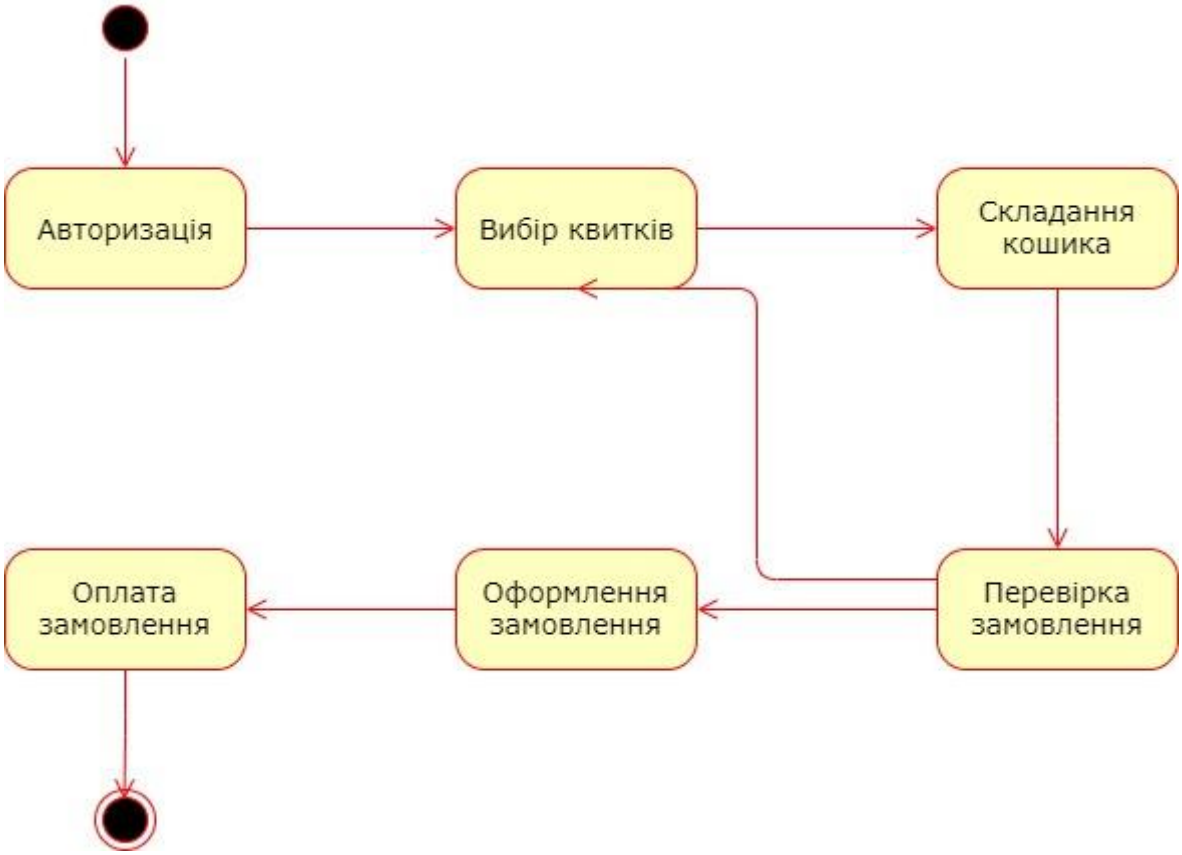

Рисунок 2.6 – Діаграма станів

3 ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ІНТЕРНЕТ-МАГАЗИНУ БІЛЕТІВ З ВИКОРИСТАННЯМ LARAVEL

3.1 Розгортання проекту

Для початку роботи над практичною частиною проекту, нам слід розгорнути веб-сервер і створити безпосередньо проект Laravel. Першим кроком є установка менеджера залежностей Composer. Для цього потрібно перейти на офіційний сайт Composer і встановити менеджер залежностей на комп'ютер. Після установки потрібно перевірити чи коректно встановлений композер на комп'ютер, ввівши в командному рядку команду composer (див. рис. 3.1).

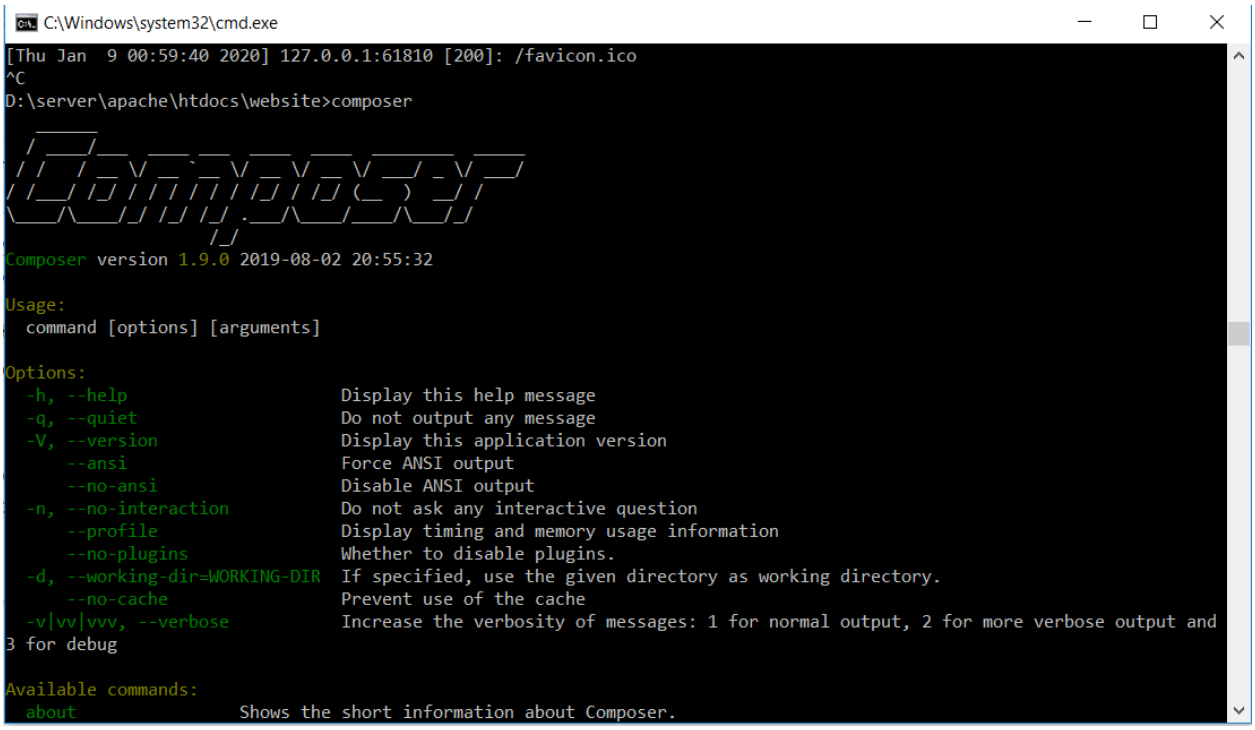

Рисунок 3.1 – Результат команди composer у cmd

Наступним кроком потрібно встановити Laravel глобально на робочий комп'ютер. Для цього потрібно ввести команду (див. рис. 3.2) в командному рядку, яка встановить Laravel глобально на комп'ютер через Composer.

```
composer global require "laravel/installer"
```

Рисунок 3.2 – Установка Laravel через командний рядок

Тепер можна створити новий проект Laravel і почати роботу з ним. Це дуже просто, знову використовується командний рядок і пакетний менеджер Composer (див. рис. 3.3).

```
composer create-project --prefer-dist laravel/laravel blog
```

Рисунок 3.3 – Створення нового проекту

Для того щоб запустити проект, потрібно використовувати команду командного рядка artisan (див. рис. 3.4).

```
php artisan serve
```

Рисунок 3.4 – Запуск проекту

Laravel самостійно створює новий проект зі наступний структурою документів (див. рис. 3.5).

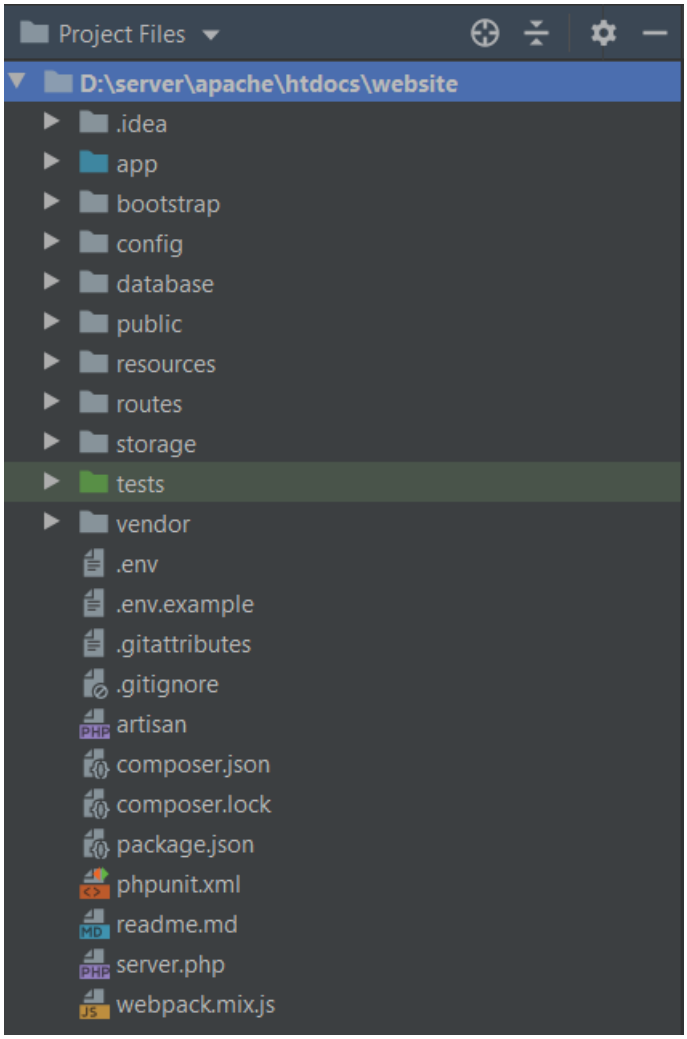

Рисунок 3.5 – Структура проекту

3.2 Система авторизації

Першим кроком практичної реалізації проекту є аутентифікація користувача на сайті. Артисан має вбудовану команду для цього (див. рис. 3.6).

```
php artisan make:auth
```

Рисунок 3.6 – Команда аутентифікації в артисан

Після застосування цієї команди в командному рядку, Laravel створює модель, контролери і 2 уявлення для реєстрації і авторизації (див. рис. 3.7).

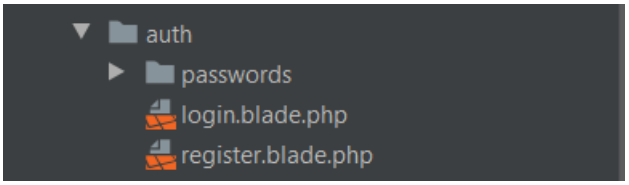

Рисунок 3.7 – Уявлення створені після команди make:auth

Також створюються 4 контролера, контролер для авторизації, для реєстрації, підтвердження пароля і контролер втрати пароля (див. рис. 3.8).

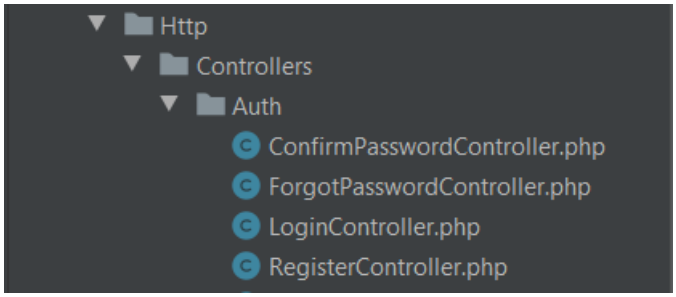

Рисунок 3.8 – Створені контролери

3.3 Створення міграцій

Для початку створимо міграції для таблиці користувачів (див. рис. 3.9).

```
public function up()
{
    Schema::create( table: 'users', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'name');
        $table->string( column: 'email')->unique();
        $table->timestamp( column: 'email_verified_at')->nullable();
        $table->string( column: 'password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Рисунок 3.9 – Міграції для таблиці Users

Наступним кроком створимо міграції для таблиці заходів (див. рис. 3.10).

```
public function up()
{
    Schema::create( table: 'events', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'name');
        $table->string( column: 'type');
        $table->text( column: 'description');
        $table->dateTime( column: 'date');
        $table->string( column: 'address');
        $table->string( column: 'img');
        $table->timestamps();
    });
}
```

Рисунок 3.10 – Міграції для таблиці Events

Також створимо міграції для таблиці квитків (див. рис. 3.11).

```
public function up()
{
    Schema::create( table: 'tickets', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->unsignedBigInteger( column: 'event_id');
        $table->float( column: 'price');
        $table->timestamps();
        $table->foreign( columns: 'event_id')
            ->references('id')
            ->on('events')
            ->onDelete('cascade')
            ->onUpdate('cascade');
    });
}
```

Рисунок 3.11 – Міграції для таблиці Tickets

Останнім кроком створимо міграції для кошика (див. рис. 3.12).

```php
public function up()
{
    Schema::create( table: 'cart', function (Blueprint $table) {
        $table->increments( column: 'id');
        $table->unsignedBigInteger( column: 'user_id');
        $table->unsignedBigInteger( column: 'ticket_id');
        $table->timestamps();
        $table->foreign( columns: 'user_id')
            ->references('id')
            ->on('users')
            ->onDelete('cascade')
            ->onUpdate('cascade');
        $table->foreign( columns: 'ticket_id')
            ->references('id')
            ->on('tickets')
            ->onDelete('cascade')
            ->onUpdate('cascade');
    });
}
```

Рисунок 3.12 – Міграції для таблиці Cart

3.4 Створення сторінок згідно макету

Для початку роботи відзначимо структуру уявлень, які будуть в проекті (див. рис. 3.13).

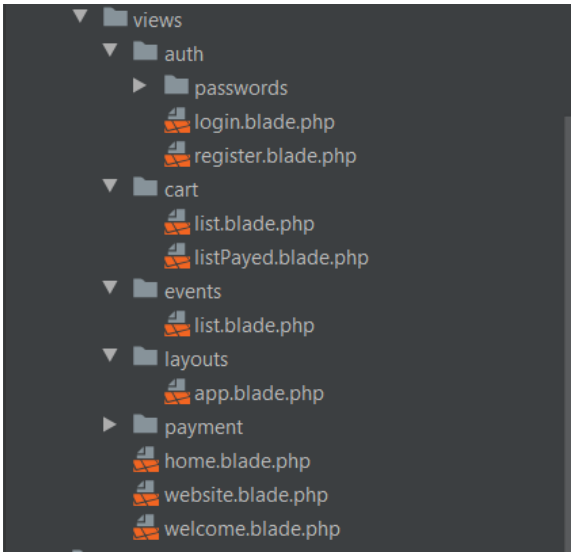

Рисунок 3.13 – Уявлення проекту

Створимо уявлення стартової сторінки, розмістимо на ній навігаційну панель і слайдер зображень (див. рис. А.1). Також в окремому уявленні створимо відображення контенту на головну сторінку сайту (див. рис. А.2). Має місце також приділити час футеру сайта (див. рис. А.3).

3.5 Приклади роботи інтернет-магазину

Користувач заходить на сайт і бачить стартову сторінку (див. рис. 3.14 та 3.15).

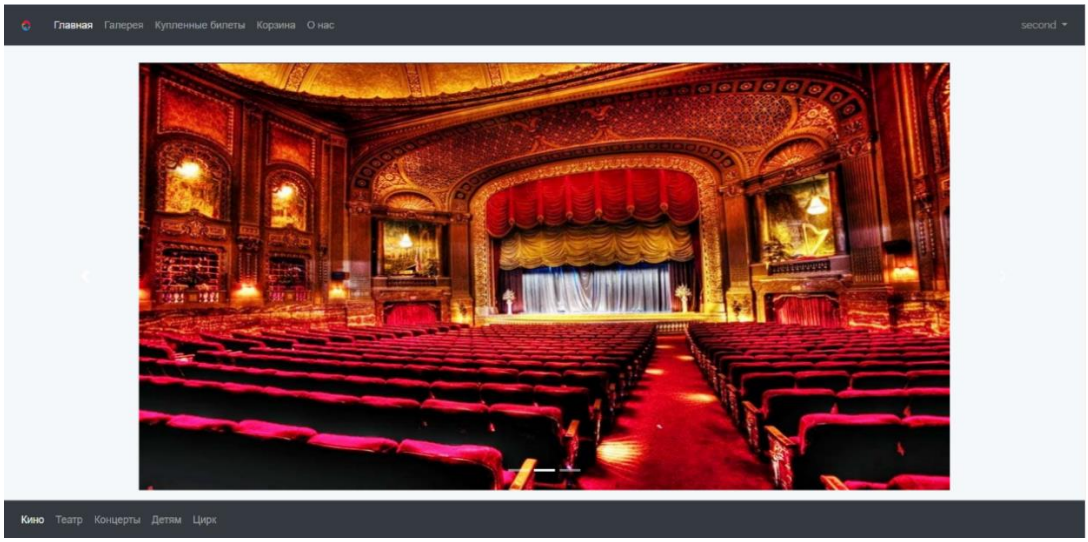

Рисунок 3.14 – Стартова сторінка сайту частина 1

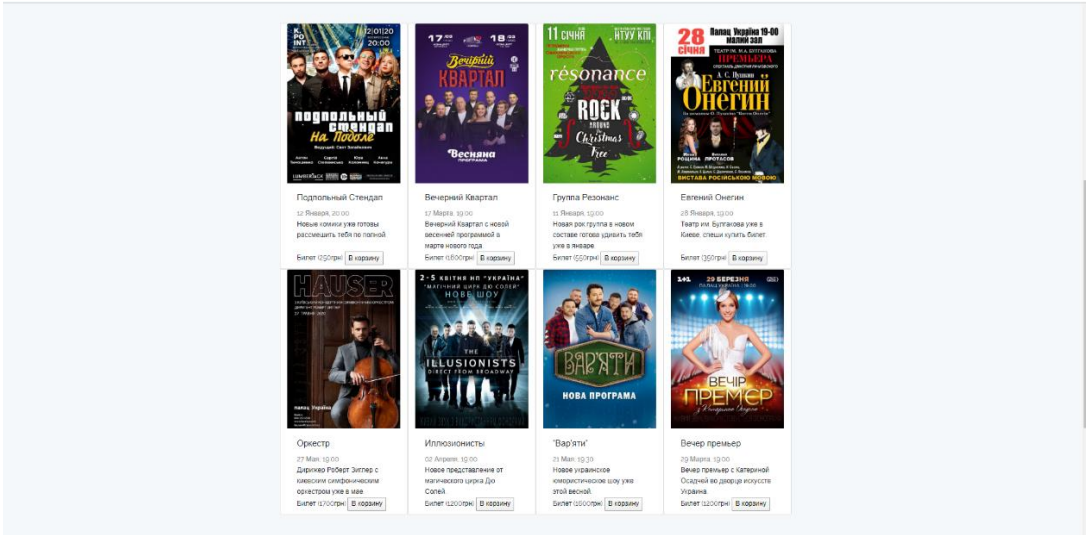

Рисунок 3.15 – Стартова сторінка сайту частина 2

Для оформлення замовлення на сайті користувач повинен зареєструватися або авторизуватися на сайті (див. рис. 3.16 та 3.17).

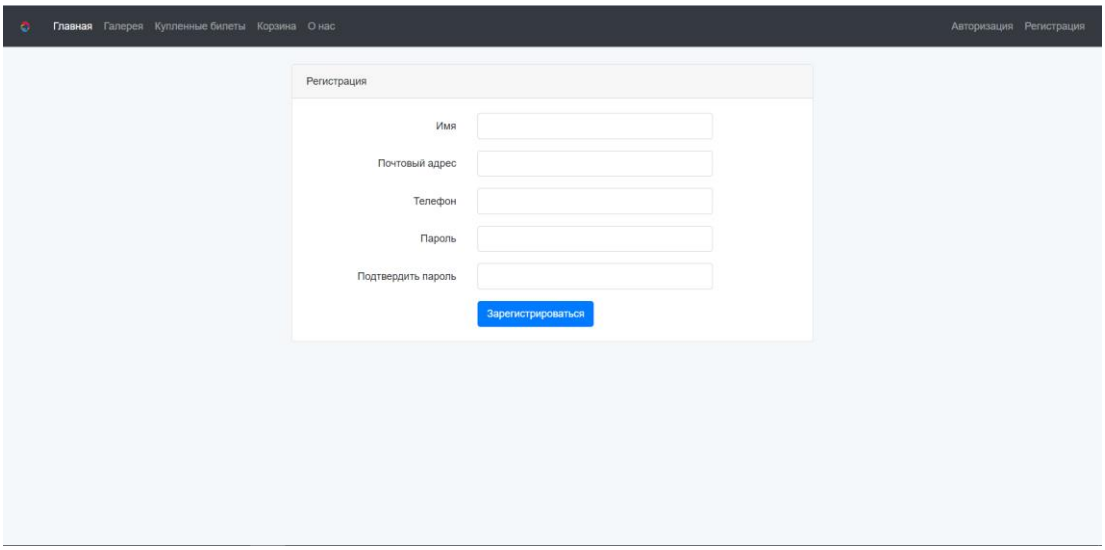

Рисунок 3.16 – Форма реєстрації

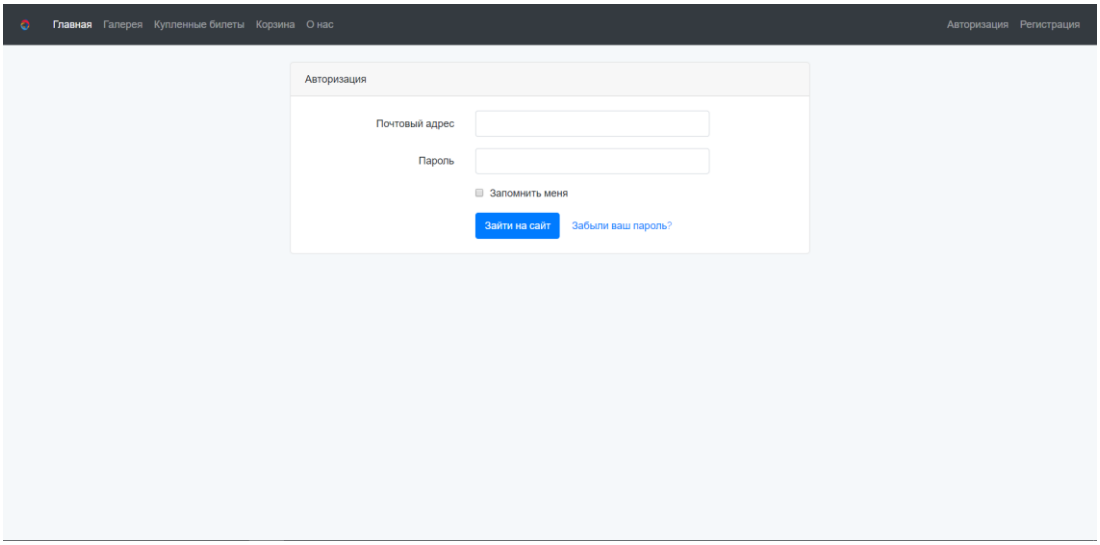

Рисунок 3.17 – Форма авторизації

Після авторизації в системі користувач може вибрати потрібні йому квитки і додати їх у кошик (див. рис. 3.18). Для того, щоб квиток потрапив до кошика потрібно натиснути кнопку "В кошик". Після цього слід перейти у

вкладку кошик на верхній навігаційній панелі, де можна буде побачити список усіх обраних квитків (див. рис. 3.19).

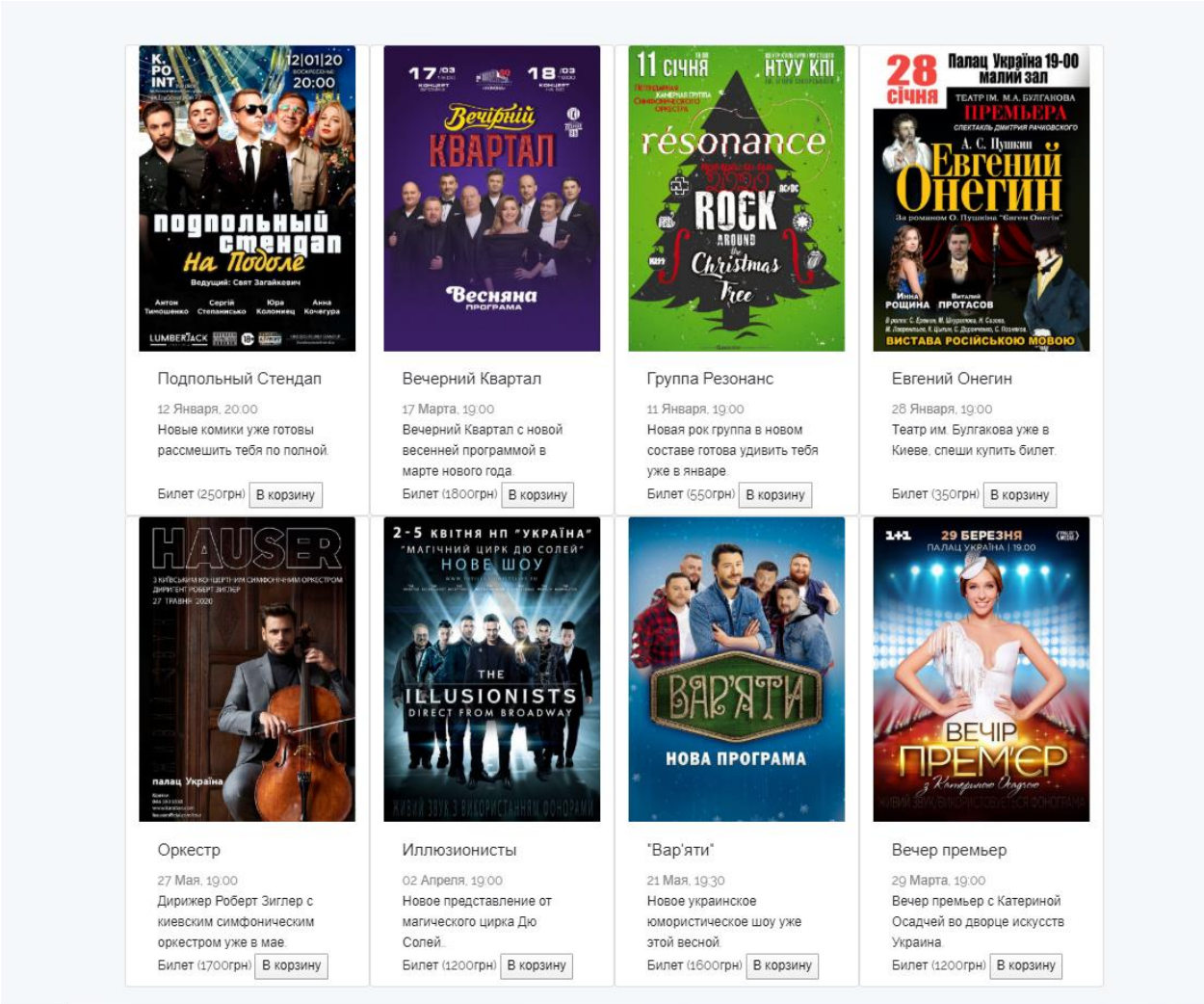

Рисунок 3.18 – Вибір квитків на сайті

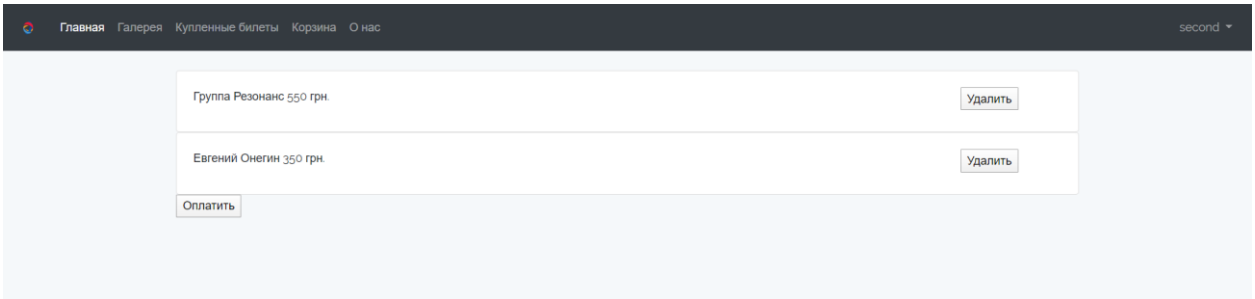

Рисунок 3.19 – Кошик з квитками

Користувач може видалити квитки якщо він передумав або перейти до оплати. Якщо користувач натискає кнопку оплатити, то він переходить у вкладку оплати (див. рис. 3.20).

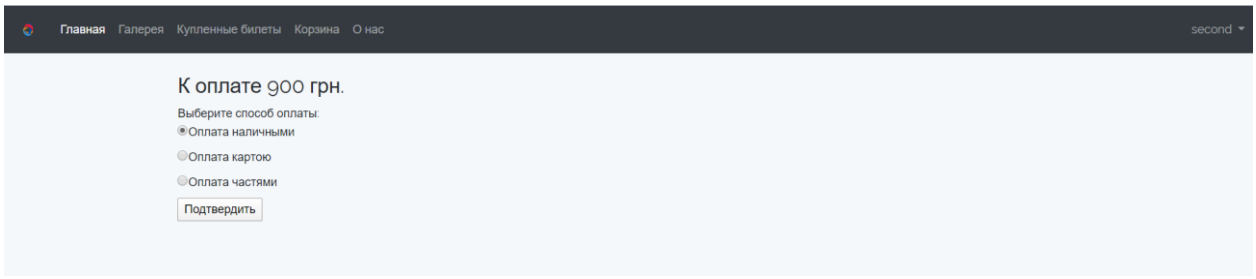

Рисунок 3.20 – Вкладка оплати замовлення

Після оплати замовлення будь-яким із запропонованих способів, користувач може подивитися історію куплених квитків у вкладці "Куплені квитки" на верхній навігаційній панелі (див. рис. 3.21).

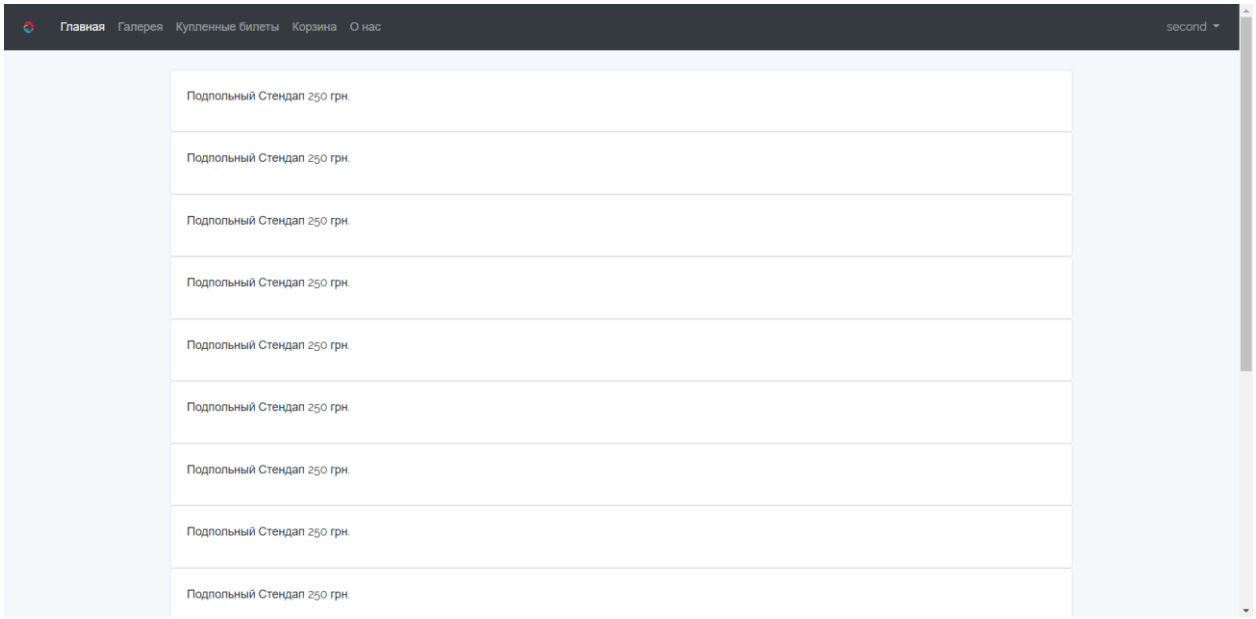

Рисунок 3.22 – Вкладку куплених квитків

Користувач також може використовувати інформацію, яка розташована внизу сайту в футері для інтеграції з соціальними мережами і зв'язку з керівництвом сайту (див. рис. 3.23).

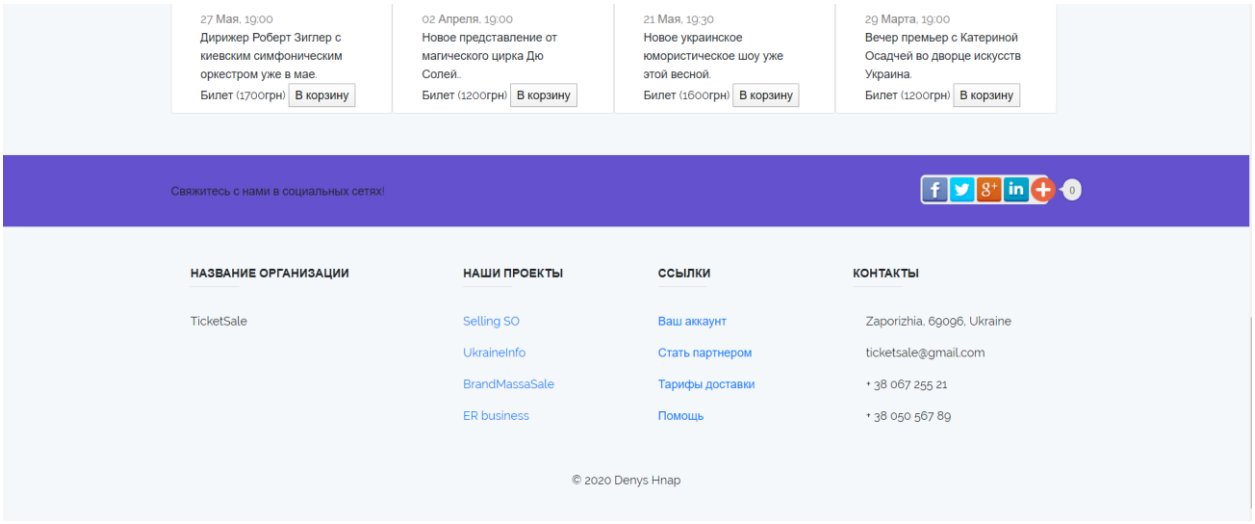

Рисунок 3.24 – Футер сайту

Сайт також є адаптивним для всіх пристроїв, які поширені на сьогоднішній день (див. рис.3.25). Тому можна використовувати будь-який гаджет для замовлення квитків.

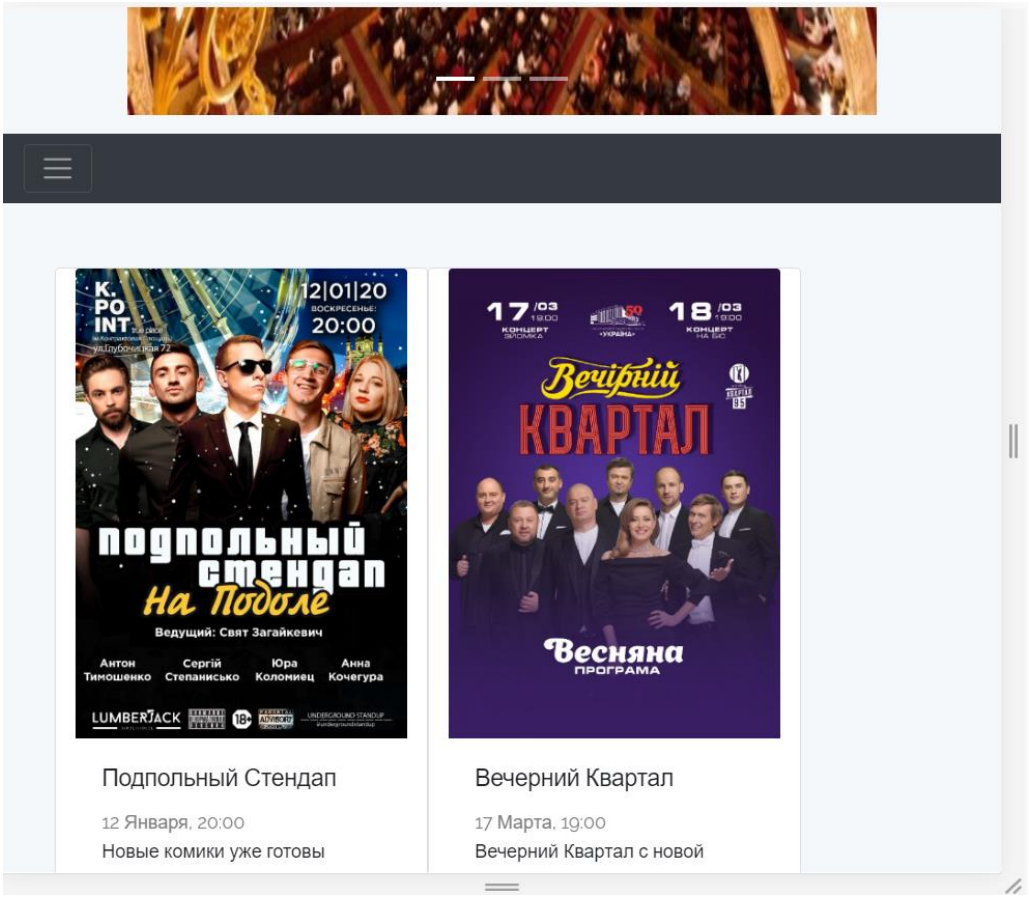

Рисунок 3.25 – Адаптивність сайту

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було виконано усі поставлені задачі, а саме:

- було проведено аналіз існуючих подійно-орієнтованих інтернет-магазинів;
- розроблено інформаційну модель системи(у вигляді діаграм потоків даних, мовою UML, за допомогою блок-схем алгоритмів тощо);
- здійснено реалізацію системи у вигляді сайту подійно-орієнтовного магазину з використанням мови програмування PHP(фреймворку Laravel).

Поставлена задача була виконана у повному обсязі, зі зручним інтерфейсом. Всі вимоги до реалізації та рекомендації по вдосконаленню проекту враховані в процесі реалізації. Було проведено функціональне тестування та тестування зручності користувача. Також була розроблена детальна інструкція користувача.

Програмний продукт може бути удосконалений шляхом додавання нових функцій, поліпшення інтерфейсу, розширення можливостей.

ПЕРЕЛІК ПОСИЛАНЬ

1. Laravel 5 Fundamentals URL: https://laracasts.com/series/laravel-5-fundamentals (дата звернення 16.10.2019).
2. Laravel по-русски URL: https://laravel.ru/docs/v5 (дата звернення 11.09.2019).
3. Laravel documentation URL: http://laravel.su/docs/5.4 (дата звернення 12.10.2019).
4. Котеров Д. PHP 7. В подлиннике, Санкт-Петербург: БХВ-Петербург, 2017. 1088 с.
5. Kelt Dockins, Kelt Dockins – Design Patterns in PHP and Laravel, London: Apress, 2017. 238 p.
6. Sanjib S. Beginning Laravel : Build Websites with Laravel 5.8, Berkley: aPress, 2018. 422 с.
7. Sanjib S. Laravel 5.7.*: Middleware, Authentication, Authorization Explained, 2018. 240 с.
8. Johnathon K. Laravel Companion / Koster Johnathon., 2018. 698 с.
9. Архітектура веб додатків URL: https://tproger.ru/translations/web-architecture-101/ (дата звернення 02.10.2019).
10. MVC архітектура URL: https://habr.com/ru/post/181772/ (дата звернення 16.10.2019).
11. Class в PHP URL: https://www.php.net/manual (дата звернення 06.10.2019).
12. ООП в PHP URL: https://cs.petrsu.ru/~kulakov/courses/php/ (дата звернення 02.10.2019).
13. Критерії оцінки веб-сайту URL: https://in-scale.ru/blog/kriterii-ocenki-sajta (дата звернення 17.09.2019).
14. Никопольский А. П. Javascript на примерах, СПб: Наука и Техника, 2017. 274 с.

15. Навчальний посібник «Методи тестування та оцінки якості програмного забезпечення із застосуванням Pairwise тестування» для студентів денної та заочної форми навчання. Полтава : ПолтНТУ 2016. 391 с.

16. Етапи розробки сайту URL: https://wezom.com.ua/blog/etapy-razrabotki-sajta (дата звернення 17.09.2019).

ДОДАТОК А

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a href="#" class="navbar-brand">
    <img src="https://autodoktor.com.ua/wp-content/uploads/Logo/Total-logo-earth.png" width="30" height="30"
         alt="logo">
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
          aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a href="/home" class="nav-link">Главная</a>
      </li>
      <li class="nav-item">
        <a href="#" class="nav-link">Галерея</a>
      </li>
      <li class="nav-item">
        <a href="/cart-payed" class="nav-link">Купленные билеты</a>
      </li>
      <li class="nav-item">
        <a href="/cart" class="nav-link">Корзина</a>
      </li>
      <li class="nav-item">
```

```
                <a href="#" class="nav-link">О нас</a>
            </li>
        </ul>
    </div>
    <ul class="navbar-nav ml-auto">
    @guest
        <div class="btn-group">
            <li class="nav-item">
                <a class="nav-link" href="{{ route('login') }}">Авторизация</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="{{ route('register') }}">Регистрация</a>
            </li>
        </div>
    @else
        <li class="nav-item dropdown">
            <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false" v-pre>
                {{ Auth::user()->name }} <span class="caret"></span>
            </a>
            <div class="dropdown-menu dropdown-menu-right" aria-
labelledby="navbarDropdown">
                <a class="dropdown-item" href="{{ route('logout') }}"
                  onclick="event.preventDefault();
                                document.getElementById('logout-form').submit();">
                   {{ __('Logout') }}
                </a>
                <form id="logout-form" action="{{ route('logout') }}" method="POST"
style="display: none;">
```

```
                @csrf
            </form>
        </div>
    </li>
  @endguest
  </ul>
</nav>
```

Рисунок А.1

```
<div class="container" id="content">
  <div class="row col-xl-12">
    @foreach($events as $event)
      <div class="card col-xl-3" style="width: 18rem;">
        <img class="card-img-top" src="{{ $event->img }}" alt="Card image
cap">
        <div class="card-body" style="position: relative">
          <h5 class="card-title">{{ $event->name }}</h5>
          <span style="color: gray">{{ $event->type }}</span>
          <p class="card-text">{{ $event->description }}.</p>
          @foreach($event->tickets as $index => $ticket)
            <form style="position: absolute;bottom:5px; top:80%;"
method="post"
action="/events/{{$event->getKey()}}/tickets/{{$ticket->getKey()}}/cart">
              <p>
                Билет ({{$ticket->price}}грн)
                <button type="submit">В корзину</button>
              </p>
              {{ csrf_field() }}
            </form>
          @endforeach
```

```
            </div>
          </div>
        @endforeach
    </div>
</div>
```

Рисунок А.2

```
<div id="footer">
    <footer class="page-footer font-small unique-color-dark">

        <div style="background-color: #6351ce;">
            <div class="container">

                <div class="row py-4 d-flex align-items-center">

                    <div class="col-md-6 col-lg-5 text-center text-md-left mb-4 mb-md-
0">
                        <h6 class="mb-0">Свяжитесь с нами в социальных сетях!</h6>
                    </div>

                    <div class="col-md-6 col-lg-7 text-center text-md-right">
                        <script type="text/javascript">(function () {
                            if (window.pluso) if (typeof window.pluso.start == "function")
return;
                            if (window.ifpluso == undefined) {
                                window.ifpluso = 1;
                                var d = document, s = d.createElement('script'), g =
'getElementsByTagName';
                                s.type = 'text/javascript';
                                s.charset = 'UTF-8';
```

```
                        s.async = true;
                        s.src = ('https:' == window.location.protocol ? 'https' : 'http')
+ '://share.pluso.ru/pluso-like.js';
                        var h = d[g]('body')[0];
                        h.appendChild(s);
                    }
                })();
            </script>
            <div class="pluso" data-background="#ebebeb"
                data-options="medium,square,line,horizontal,counter,theme=01"
                data-services="facebook,twitter,google,linkedin"></div>
        </div>

    </div>

</div>

<div class="container text-center text-md-left mt-5">

    <div class="row mt-3">

        <div class="col-md-3 col-lg-4 col-xl-3 mx-auto mb-4">

            <!-- Content -->
            <h6 class="text-uppercase font-weight-bold">Название
организации</h6>
            <hr class="deep-purple accent-2 mb-4 mt-0 d-inline-block mx-auto"
style="width: 60px;">
            <p>TicketSale</p>
```

```
            </div>

            <div class="col-md-2 col-lg-2 col-xl-2 mx-auto mb-4">

               <!-- Links -->
               <h6 class="text-uppercase font-weight-bold">Наши проекты</h6>
               <hr class="deep-purple accent-2 mb-4 mt-0 d-inline-block mx-auto"
style="width: 60px;">
               <p>
                  <a href="#!">Selling SO</a>
               </p>
               <p>
                  <a href="#!">UkraineInfo</a>
               </p>
               <p>
                  <a href="#!">BrandMassaSale</a>
               </p>
               <p>
                  <a href="#!">ER business</a>
               </p>

            </div>

            <div class="col-md-3 col-lg-2 col-xl-2 mx-auto mb-4">

               <!-- Links -->
               <h6 class="text-uppercase font-weight-bold">Ссылки</h6>
               <hr class="deep-purple accent-2 mb-4 mt-0 d-inline-block mx-auto"
style="width: 60px;">
```

```
            <p>
              <a href="#!">Ваш аккаунт</a>
            </p>
            <p>
              <a href="#!">Стать партнером</a>
            </p>
            <p>
              <a href="#!">Тарифы доставки</a>
            </p>
            <p>
              <a href="#!">Помощь</a>
            </p>

          </div>

          <div class="col-md-4 col-lg-3 col-xl-3 mx-auto mb-md-0 mb-4">

            <h6 class="text-uppercase font-weight-bold">Контакты</h6>
            <hr class="deep-purple accent-2 mb-4 mt-0 d-inline-block mx-auto"
style="width: 60px;">
            <p>
              <i class="fas fa-home mr-3"></i> Zaporizhia, 69096, Ukraine</p>
            <p>
              <i class="fas fa-envelope mr-3"></i> ticketsale@gmail.com</p>
            <p>
              <i class="fas fa-phone mr-3"></i> + 38 067 255 21</p>
            <p>
              <i class="fas fa-print mr-3"></i> + 38 050 567 89</p>
          </div>
```

```
            </div>
        </div>
        <div class="footer-copyright text-center py-3">© 2020 Denys Hnap</div>
        <!-- Copyright -->
    </footer>
    <!-- Footer -->
</div>
```

Рисунок А.3