МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ТУРИСТИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ LARAVEL»**

Виконав: студент 2 курсу, групи 8.1211-1іпз
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
О.О. Карасьов
(ініціали та прізвище)

Керівник старший викладач кафедри програмної інженерії, к.ф.-м.н. Чопорова О.В
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

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, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Карасьову Олексію Олександровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи (проєкту) Розробка туристичної інформаційної системи з використанням фреймворку Laravel

керівник роботи (проєкту) Чопорова Оксана Володимирівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від «04» травня 2022 року № 500-с

2. Строк подання студентом роботи

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. Проєктування інформаційної системи.
3. Реалізація вебдодатку.

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 04.05.2022

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.	10.07.2022	
2.	Обробка методичних та теоретичних	15.07.2022	
	джерел.		
3.	Вибір засобів реалізації.	20.07.2022	
4.	Розробка першого розділу.	01.08.2022	
5.	Проєктування вебдодатку.	09.08.2022	
6.	Створення макетів веб сайту.	15.08.2022	
7.	Розробка другого розділу.	19.08.2022	
8.	Розробка вебдодатку.	01.09.2022	
9.	Розробка третього розділу.	27.09.2022	
10.	Оформлення та нормоконтроль	01.12.2022	
	кваліфікаційної роботи.		
11.	Захист кваліфікаційної роботи.	15.12.2022	

Студент _____ О.О. Карасьов
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Чопорова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційної системи з використанням фреймворку Laravel»: 58 с., 36 рис., 5 табл., 10 джерел, 2 додатки.

ВЕБДОДАТОК, КНИГИ, САЙТ GOOGLE MAPS, TAILWIND UI LARAVEL, LARAVEL BREEZE, LARAVEL LIVEWIRE.

Об’єкт дослідження – процес розробки інформаційної системи з використанням фреймворку Laravel.

Мета роботи: дослідження створення вебдоданків на базі фреймворку Laravel та взаємодії серверної частини з компонентами фреймворку.

Метод дослідження – аналітичний, порівняльний.

Процес створення сайту завжди займав величезну кількість часу. Тому існують різні PHP-фреймворки, які значно полегшують і прискорюють розробку. Простіше кажучи – це готовий каркас або бібліотека коду. У кваліфікаційній роботі розглянуто сучасні методи створення вебдодатків на базі фреймворку Laravel, а також огляд і порівняння актуальних фреймворків. Були визначені недоліки і переваги PHP-фреймворку Laravel. Для реалізації практичної частини роботи було спроєктовано та розроблено інформаційна система на базі даних зі створенням каталогу туристичних місць та зображенням їх на google maps.

SUMMARY

Master's qualifying paper «Development of the Tourist Information System using the Laravel Framework»: 58 pages, 36 figures, 5 tables, 10 references, 2 supplements.

WEB APP, SITE, BOOKS, GOOGLE MAPS, TAILWIND UI, LARAVEL, LARAVEL BREEZE, LARAVEL LIVEWIRE.

The object of research is the process of developing an information system using the Laravel framework.

The aim of the study is the creation of web applications based on the Laravel framework and the interaction of the server part with the components of the framework.

The methods of research are analytical, comparative.

The process of creating a site has always taken a huge amount of time. Therefore, there are various PHP frameworks that greatly facilitate and accelerate development. Simply put, it's a ready-made framework or code library. The qualification work considers modern methods of creating web applications based on the Laravel framework, as well as a review and comparison of current frameworks. The disadvantages and advantages of the PHP framework Laravel have been identified. To implement the practical part of the work, an information system was designed and developed based on a database with the creation of a catalog of tourist places and their image on google maps.

ЗМІСТ

Завдання на кваліфікаційну роботу……..….2
Реферат……4
Summary……...5
Вступ 8
1 Робота з вебфреймворком Laravel 9
1.1 Аналіз та опис сутностей 9
1.2 Фреймворк Laravel 10
1.3 Фреймворк Tailwind UI 13
1.4 Висновки до розділу 1 15
2 Проєктування туристичної інформаційної системи 16
2.1 Вимоги до проєкту 16
2.2 Додатки, використані при розробці проєкту 17
2.3 Бібліотеки та фреймворки 18
2.4 Функціональне проєктування 21
2.5 Концептуальне проєктування 24
2.6 Логічне проєктування 25
2.7 Висновки до розділу 2 30
3 Розробка туристичної інформаційної системи 32
3.1 Початкові налаштування проєкту 32
3.2 Створення БД 36
3.3 Розробка дизайну проєкту 39
3.4 Middleware для аутентифікації 43
3.5 HTTP-контролери 44
3.5.1 Контролер підтвердження 45
3.5.2 Контролер входу 46
3.5.3 Реєстратор контролера 46
3.5.4 Контролер перевірки електронної пошти 47

3.5.5 Інші контролери туристичної інформаційної системи 48
Перелік посилань 50
Додаток А 52
Додаток Б 55

ВСТУП

Майбутні інженери-програмістами, менеджери програмних проєктів в індустрії розробки програмних систем, замовного програмування і програмного аутсорсингу для фінансового, виробничого, телекомунікаційного секторів економіки, освіти, охорони здоров'я, індустрії розваг, підприємств торгівлі та оборонної промисловості – усі вони можуть застосовувати будь-які безкоштовні вебфреймворки для розробки з використанням архітектурної моделі MVC.

Величезний проміжок часу в сфері програмування, а саме в сфері веброзробки, PHP був найулюбленішою мовою програмування в світі. І це зовсім не випадково. PHP-розробка ведеться швидко, тому в результаті проєкту відрізняється високим рівнем безпеки і її легко підтримувати.

Для допомоги веброзробникам в справі створення їх чудових проєктів було створено величезну кількість різних PHP-фреймворків.

Метою дослідження є ознайомлення з вебфреймворком Laravel та набуття вмінь створення вебсторінок, тобто сторінок в Інтернеті.

Робота складається із 3 розділів: огляд сучасного становища проблеми; робота з вебфреймворком Laravel; розробка туристичної інформаційної системи.

Інформаційна система надасть користувачам можливість перегляду всіх туристичних місць, а також авторизуватись або зареєструватись на сайті. Так само туристична інформаційна система має бути простим в управлінні, для того щоб адміністратор сайту зміг швидко і зручно додавати туристичні місця на сайт для користувачів.

1 РОБОТА З ВЕБФРЕЙМВОРКОМ LARAVEL

1.1 Аналіз та опис сутностей

Розробка сайтів на PHP багатьом не подобається, оскільки вважається що PHP – це погана мова, що не має ні гарного пакетного менеджера, ні оптимальної структури для написання сайтів.

На тему PHP можна міркувати довго, тому краще надати статистику, наведену всесвітньо відомим сервісом w3techs, яку можна побачити на рисунку 1.1.

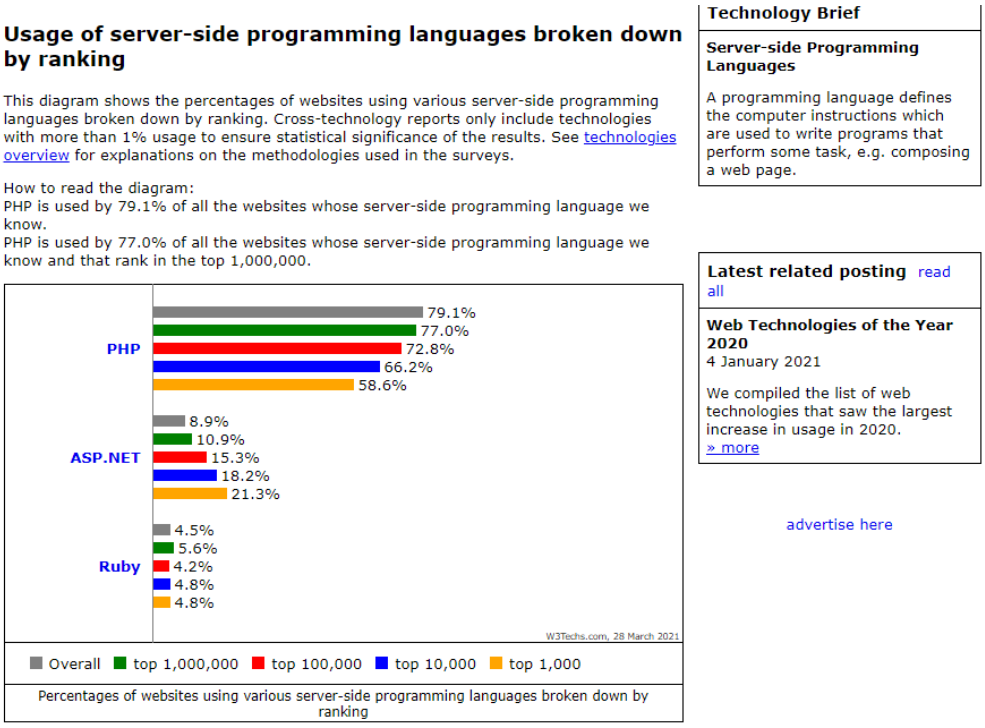

Рисунок 1.1 – Статистика з w3techs

Як ми бачимо, близько 79 % сайтів усього світу, що зараз розроблено або розробляються, використовують мову PHP. У той же час, коли говорять про розробку сайтів на PHP, то в більшості випадків згадують фреймворк Laravel. Laravel має неймовірну популярність і завоював безліч нагород через

свою простоту, функціональність, а також добре описану документацію. Laravel робить все те ж саме, що і PHP, але робить це «добре».

Код, прописаний на Laravel, має загальну зрозумілу структуру. У проєкті за замовчуванням використовується схема MVC (View, Model, Controller), що розподіляє всі файли на категорії, а також за рахунок Laravel ви отримуєте дуже зручну структуру для розробки і управління базою даних.

Зрозуміло, що можна писати сайти і без використання фреймворків, але з їх використанням можна отримати: по-перше, безліч готових рішень для ваших проєктів, а, по-друге, весь ваш код буде коректно структурований, і інші програмісти спокійно зможуть в ньому розібратися.

1.2 Фреймворк Laravel

У цьому підрозділі метою буде не досконально вивчити фреймворк Laravel, а лише познайомиться з його можливостями, подивитися, наскільки зручний він в роботі і наскільки просто в ньому буде створювати додатки.

Для початку потрібно переглянути останню доступну документацію Laravel, яка є на офіційному сайті, яку в майбутньому буде використано у цій роботі. Так само можна подивитися вимоги до сервера, програмного забезпечення і, зокрема, версії PHP [1].

Перед установкою Laravel необхідно всередину нашого проєкту встановити пакетний менеджер, який називається Composer. Після цього установка Laravel розбита на два етапи. Перший етап полягає в тому, що необхідно встановити Laravel глобально, на комп'ютер. Другий етап – установка Laravel локально, тобто установка Laravel всередину будь-якого конкретного проєкту.

Після успішної установки Laravel, як показано на рисунку 1.2, потрібно звернутися до команди локального сервера і запустити його. Після цього

можна буде подивитися на створений сайт за URL адресою, яку буде показано в терміналі.

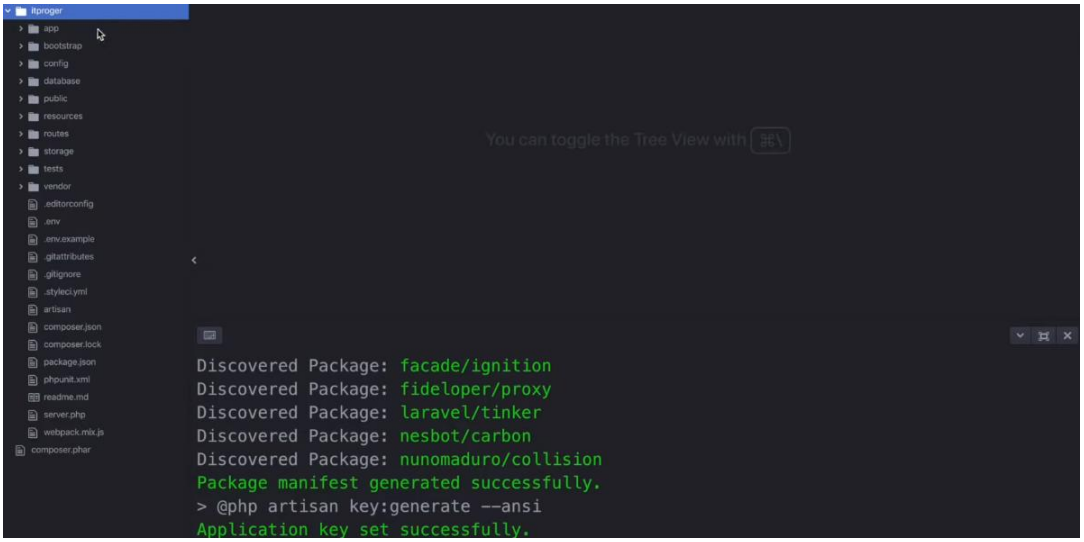

Рисунок 1.2 – Встановлення Laravel

Після переходу за URL адресою в браузері відобразиться наш сайт. На даний момент буде відображатися шаблонний проєкт Laravel, який створюється автоматично при створенні порожнього Laravel проєкту, як показано на рисунку 1.3.

Рисунок 1.3 – Шаблон проєкту Laravel

Далі можна повністю зануритися саме в розробку вебдодатку з використанням Laravel, а також створити html-шаблони, які потрібні для того, щоб виводити Front-end складову.

Наприклад, веброзробники досить часто при створенні своїх сайтів стикаються з проблемою – з громіздким кодом, який дублюється для кожної окремої сторінки, навіть якщо на ній змінюються лише декілька слів або форм. Laravel надає можливість вбудовування можливих html-файлів в інші html-файли, тобто, по суті, таке собі успадкування.

Відповідно, можна створити окремий файл, який буде головним, всередину якого можна вбудовувати усі додаткові html-теги. Після створення основного шаблона, ми прописуємо будь-який код простого оформлення сторінки, наприклад: «Моя сторінка» і замість вмісту тега body, потрібно вказати директиву, яка скаже про те, що тут буде відображатися код в залежності від певного шаблону. Далі прописуємо: «@yield ('content')» в тезі body, в даному випадку 'content', це ім'я блоку коду, називати його можна, як завгодно. Далі, в створеному файлі, який є додатковим, можна залишити тільки те, що буде вбудовуватися всередину самого тега body і прописуємо таку директиву як: «@extends ()», і сказати, від якого файли ми все успадковуємо. Потім ми прописуємо ще одну директиву, це «@section ()», тут ми говоримо в яку секцію шаблону ми будемо вбудовувати шматок коду, наприклад «<h1> Сторінка контактів« </ h1> »і далі, закриваємо цю секцію:« @endsection ».

Після завершення всіх попередніх кроків, зайшовши на головну сторінку і подивившись на код, натиснувши правою кнопкою миші «Перегляд коду», можна помітити, що для головної сторінки у нас була додана вся стандартна html-структура і єдине, що у нас відрізняється на головній сторінці, так це вміст тегу body, як показано на рисунку 1.4.

Рисунок 1.4 – Створення нових сторінок

Тепер, при переході на різні сторінки, наприклад «about», яку можна створити раніше в проєкті, буде відображатися різний вміст коду, наприклад, в тезі Title.

1.3 Фреймворк Tailwind UI

TailwindCSS – це CSS-бібліотека, яка спрощує стилізацію HTML, тим самим шляхом, як це робить Bootstrap – додаючи величезну кількість різноманітних класів. Але, на відміну від Bootstrap, який додає вже готові до вживання компоненти, такі як кнопки, алерти та навбари, класи TailwindCSS націлені на конкретну властивість. TailwindCSS не має заздалегідь написаної кнопки, її потрібно зробити самостійно [2].

Здається, це невелика різниця між TailWindCSS та всіма іншими потоками CSS, на чолі, з яких Bootstrap [6], а саме класи як властивості, а не класів як компоненти, змінюються на все. І якщо зовні TailWindCSS схожий на них, він по суті, він щось зовсім інше – новий підхід до написання CSS.

Якщо писати CSS в inline-стилі, для однакових або майже однакових кнопок кожен раз доведеться писати майже один і той самий код. Цю проблему можна вирішити у TailwindCSS, як показано на рисунку 1.5.

```
<button class="px-4 py-2 font-bold text-white bg-blue-500 rounded">Отправить</button>
```

Рисунок 1.5 – Тест коду з Tailwind

Якщо ви використовуєте будь-яку передню рамку, цієї проблеми взагалі не буде. Не потрібна директива @Apply. Ви можете просто зробити окремий компонент для будь -якого повторюваного коду. Легко та зручно, як показано на рисунку 1.6.

```
<div x-data="cards()" class="space-y-12">
  <template x-for="card in cards">
    <div>
      <img class="rounded" :src="card.img" :alt="card.imgAlt" />
      <div class="mt-2">
        <div x-text="card.eyebrow" class="text-xs font-bold text-gray-600 uppercase"></div>
        <div class="font-bold leading-snug text-gray-700">
          <a x-text="card.title" :href="card.url" class="hover:underline"></a>
        </div>
      </div>
    </div>
  </template>
</div>

<script>
  // ...
</script>
```

Рисунок 1.6 – Тест коду з Tailwind

То яке ж місце займає TailwindCSS в історії CSS? Він зручніший за ванільний CSS і не створює проблем із вкладеністю та колізією імен класів. Він добре працює з усіма препроцесорами.

TailwindCSS розширює роботу з БЕМ (Блок-Елемент-Модифікатор) [3]. Не потрібно робити модифікатор для кожної нагоди, просто зробіть базовий стиль і перезапишіть зверху, що потрібно. Якщо модифікований блок використовується не в одному місці, а в багатьох, можна зробити модифікатор. TailwindCSS у цьому не обмежується.

1.4 Висновки до розділу 1

Розробка туристичної системи буде виконуватися на базі PHP, з використанням таких популярних фреймворків як Laravel, Tailwind UI. Вони всі є основою моєї розробки вебдодатку для інформаційної туристичної системи, тому що вони дуже прості у використанні, а також одні з найбільш популярних і функціональних. Завдяки цим інструментам не потрібно зайвий раз використовувати безліч непотрібних доповнень до PHP, а також написання коду відбуватиметься у спеціально виділених вікнах функцій під ці фреймворки.

2 ПРОЄКТУВАННЯ ТУРИСТИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Вимоги до проєкту

Для дослідження предметної області з використанням Laravel було обрану тему туристична система. Подібна тема дозволяє в повній мірі розглянути роботу самого фреймворка Laravel, а також побачити всі залежності, які можуть виникнути при моделюванні проєкту.

Основною метою розробки додатку є отримання практичних навичок з роботою фреймворку, а також створення інформаційної системи для візуалізації даних.

Основні функціональні вимоги до проєкту:

1) можливість авторизуватися та зареєструватися в системі;
2) можливість створювати, видаляти та редагувати туристичні місця;
3) можливість створювати, видаляти та редагувати фільтри для місць;
4) можливість переглядати перелік туристичних місць по фільтрам;
5) можливість редагування користувачем особистого профілю;
6) можливість показу маршруту по туристичним місцям користувачам.

До нефункціональних вимог можливо віднести:

1) простий для розуміння користувача інтерфейс;
2) швидкодія.

У адміністратора сайту є можливість переглядати профілі користувачів, які зареєструвалися на сайті. Тому користувачі можуть авторизуватися повторно на цьому сайті.

Доступні функціональні можливості для різних типів користувачів можливо представити у вигляді діаграми прецедентів:

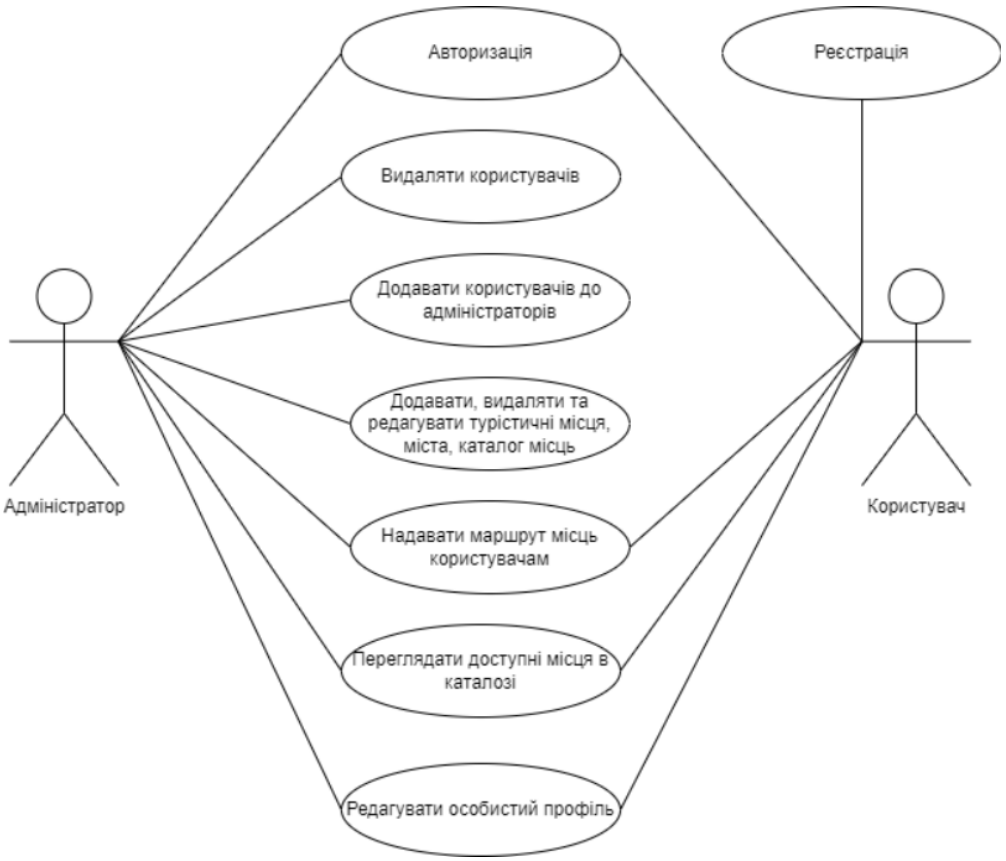

Рисунок 2.1 – Діаграма прецедентів для туристичної інформаційної системи

Як можна побачити на рисунку 2.1, адміністратор сайту має право до усіх доступних можливостей сайту на відміну від користувачів, які мають право тільки на декілька можливостей сайту.

2.2 Додатки, використані при розробці проєкту

Для реалізації проєкту було використано наступні додатки: PhpStorm, Node.js, Tailwind UI. Розглянемо використані додатки далі детальніше.

PhpStorm – це інтегроване середовище розробки на PHP з інтелектуальним редактором, яке глибоко розуміє код, підтримує PHP 7.3, 5.6, 5.5, 5.4 і 5.3 для сучасних і класичних проєктів, забезпечує краще в індустрії автодоповнення коду, рефакторинг, запобігання помилок нальоту і підтримує змішування мов.

Програмне забезпечення JetBrains PhpStorm є спеціалізованим засобом web-розробки, орієнтованим на web-додатки та інші види програм, які можна створювати за допомогою мови PHP і з використанням HTML, JavaScript і CSS. Рішення PhpStorm здійснює розгортання і синхронізацію проєктів через протокол FTP. Середа PhpStorm пропонує функції автоматичного завершення мовних конструкцій PHP в коді, інспектування коду, різні алгоритми рефакторинга і швидку навігацію по коду.

Node.js – це серверна платформа, що працює на двигуні Google Chrome – V8, який вміє компілювати JavaScript код в машинний код [5]. Ця система виконує JavaScript окремо від вашого браузера. Node.js можна встановити на сервер і виконувати на ньому код, віддаючи результат виконання користувачам. На ньому ж можна робити окремі додатки, використовуючи додаткові фреймворки. Node.js використовує подієво-орієнтовану модель і введення та виведення архітектуру, що робить його легким і ефективним.

Tailwind UI – Найбільша бібліотека інтерфейсу Tailwind [6]. Tailwind UI Kit дає розробникам можливість швидко відстежувати робочий процес за допомогою 1300 компонентів і 30 шаблонів. Адаптивний дизайн, який забезпечує послідовний досвід роботи з точками перерв. Ретельно перевірені компоненти, що відповідають найвищим стандартам якості. Компоненти, сумісні з різними браузерами, узгоджені в Інтернеті. Створений на основі ефекту зрізу бордюрів, доступність WCAG забезпечує справді інклюзивний досвід

2.3 Бібліотеки та фреймворки

Guzzle – це HTTP-клієнт PHP, який дозволяє легко надсилати HTTP-запити та легко інтегрувати його з вебслужбами. Простий інтерфейс для створення рядків запитів, запитів POST, потокової передачі великих завантажень, потокової передачі великих завантажень, використання файлів

cookie HTTP, завантаження даних JSON тощо. Можна надсилати як синхронні, так і асинхронні запити за допомогою одного інтерфейсу. Використовує інтерфейси PSR-7 для запитів, відповідей і потоків. Це дозволяє вам використовувати інші сумісні бібліотеки PSR-7 із Guzzle. Абстрагує основний транспорт HTTP, дозволяючи писати агностичний код середовища та транспорту; тобто відсутність жорсткої залежності від cURL, потоків PHP, сокетів або неблокуючих циклів подій. Система проміжного програмного забезпечення дозволяє доповнювати та компонувати поведінку клієнта.

```php
$client = new GuzzleHttp\Client();
$res = $client->request('GET', 'https://api.github.com/user', [
    'auth' => ['user', 'pass']
]);
echo $res->getStatusCode();
// "200"
echo $res->getHeader('content-type')[0];
// 'application/json; charset=utf8'
echo $res->getBody();
// {"type":"User"...'

// Send an asynchronous request.
$request = new \GuzzleHttp\Psr7\Request('GET', 'http://httpbin.org');
$promise = $client->sendAsync($request)->then(function ($response) {
    echo 'I completed! ' . $response->getBody();
});
$promise->wait();
```

Рисунок 2.2 – Приклад роботи з бібліотекою Guzzle

Fruitcake Laravel-cors – Пакет дозволяє надсилати заголовки Cross-Origin Resource Sharing із конфігурацією проміжного програмного забезпечення Laravel.

Laravel Breeze – забезпечує мінімальну та просту відправну точку для створення програми Laravel з автентифікацією. У стилі Tailwind Breeze публікує контролери автентифікації та представлення для вашої програми,

які можна легко налаштувати відповідно до потреб вашої програми. Laravel Breeze працює на базі Blade і Tailwind. Якщо ви шукаєте більш надійний стартовий набір Laravel, який включає двофакторну автентифікацію, підтримку Livewire / Inertia тощо, ознайомтеся з Laravel Jetstream [4].

Laravel – це безкоштовний вебфреймворк мовою програмування загального призначення PHP з відкритим кодом, призначений для розробки з використанням архітектурної моделі Model View Controller, призначений для розробки вебдодатків будь-якого рівня складності.

Laravel Sanctum – надає легку систему автентифікації для SPA (односторінкових програм), мобільних програм і простих API на основі маркерів. Sanctum дозволяє кожному користувачеві вашої програми генерувати кілька маркерів API для свого облікового запису. Цим маркерам можуть бути надані здібності/області, які визначають, які дії дозволено виконувати маркерам.

Laravel Tinker – дозволяє вам взаємодіяти з усією програмою Laravel у командному рядку, включаючи моделі Eloquent, завдання, події тощо. Tinker використовує список дозволів, щоб визначити, які команди Artisan дозволено виконувати в його оболонці. За замовчуванням ви можете запускати команди clear-compiled, down, env, inspire, migrate, optimize і up. Якщо ви хочете дозволити більше команд, ви можете додати їх до масиву команд у вашому конфігураційному файлі tinker.php.

```php
<?php

namespace App\Console\Commands;

use App\Models\User;
use App\Support\DripEmailer;
use Illuminate\Console\Command;

class SendEmails extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'mail:send {user}';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Send a marketing email to a user';

    /**
     * Execute the console command.
     *
     * @param  \App\Support\DripEmailer  $drip
     * @return mixed
     */
    public function handle(DripEmailer $drip)
    {
        $drip->send(User::find($this->argument('user')));
    }
}
```

Рисунок 2.3 – Приклад роботи з Laravel Tinker

2.4 Функціональне проєктування

Проєктування нових видів і зразків машин, пристроїв, апаратів представляє складний і тривалий процес, що включає розробку початкових даних, креслень, технічної документації, необхідних для виготовлення

дослідних зразків і наступного виробництва і експлуатації об'єктів проєктування.

В процесі проєктування виникає необхідність створення опису, необхідного для побудови ще неіснуючого об'єкту. Тому можна скористатися методологією IDEF0, суттю якої є побудова ієрархічної системи діаграм. Тобто кожна діаграма поодинці буде описом частини системи.

Перед побудовою моделі необхідно визначитися, яка модель системи буде побудована. А також визначення позиції, з точки зору якої будується модель. Точку зору найкраще уявляти собі як позицію людини або об'єкта, в яке треба встати, щоб побачити систему в дії.

Нижче надано діаграму методології IDEF0 першого рівня (див. рис. 2.4).

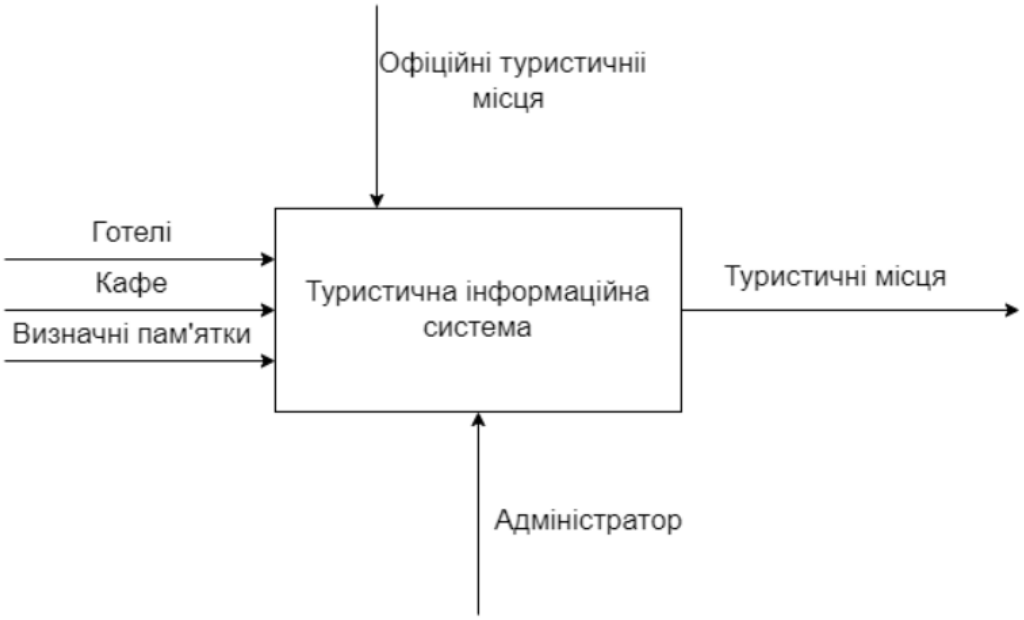

Рис. 2.4 – Контекстна діаграма IDEF0

Після побудови контекстна діаграма деталізується за допомогою діаграми декомпозиції першого рівня. На цій діаграмі відображаються функції системи, які повинні бути реалізовані в рамках основної функції. Діаграма декомпозиції першого рівня для даної задачі приведена на рисунку 2.5.

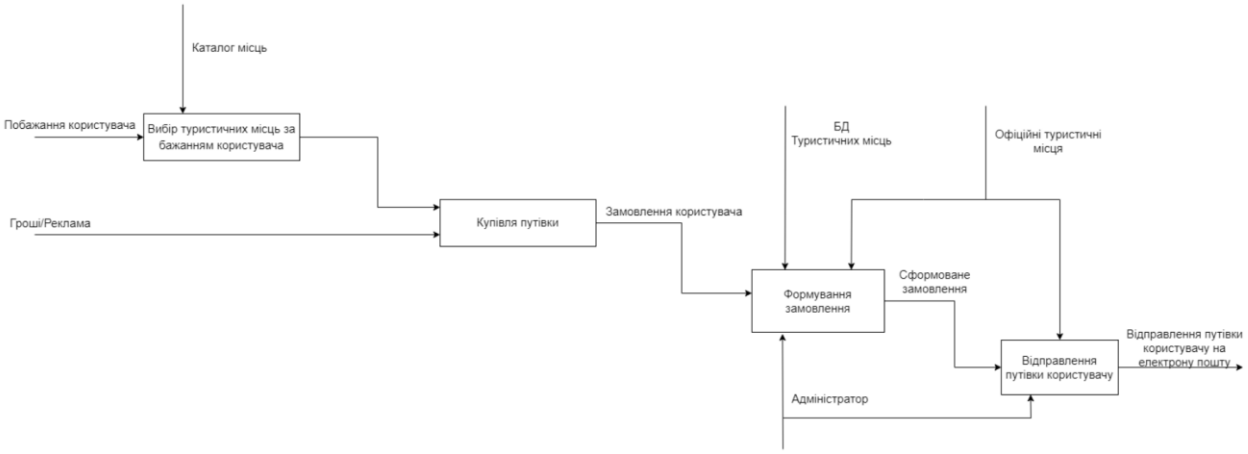

Рисунок 2.5 – Діаграма декомпозиції туристичної системи

Система працює за класичною схемою клієнт-серверного додатку. Користувацька частина системи отримує інформацію та дані з запитів відправлених на сервер, який в свою чергу оброблює їх та за необхідністю звертається до даних що зберігаються у БД.

Рисунок 2.6 – Діаграма розгортання туристичної інформаційної системи

Для більшого розуміння взаємодії компонентів системи необхідно розглянути окремі функції системи та роботу частин додатку всередині них. Наприклад можливо розглянути функцію аторизації:

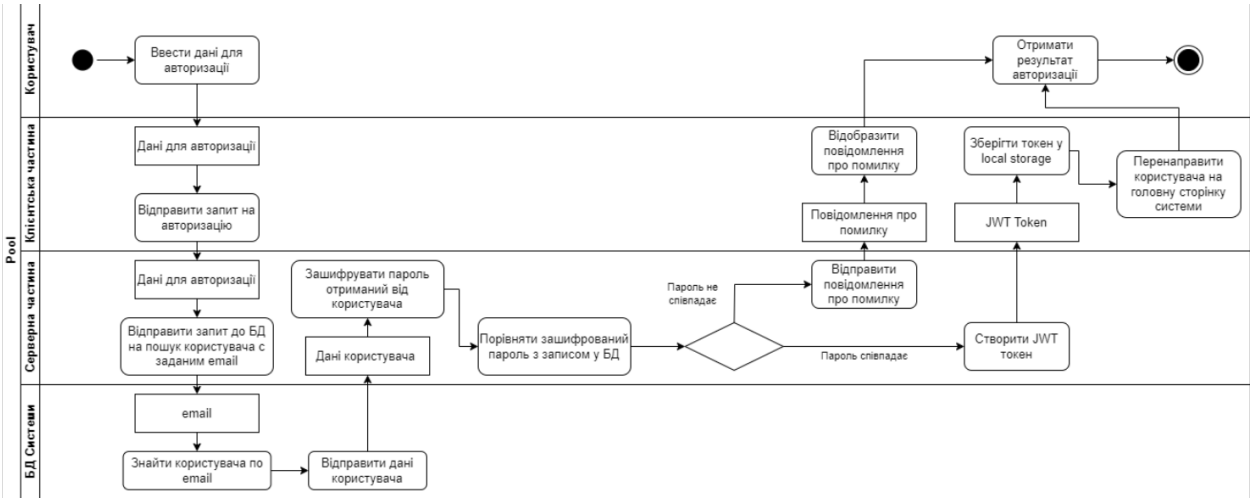

Рисунок 2.7 – Діаграма діяльності для функції авторизації

Як можна побачити з діаграми (див. рис 2.7), сервер виконує роль контролера аутентифікації та доступу до ресурсів системи. А клієнтська частина надає користувачу інтерфейс взаємодії з сервером та відображає наявні дані.

2.5 Концептуальне проєктування

Концептуальне проєктування можна показати у вигляді ER діаграми. Діаграми «сутність-зв'язок» допомагають розібратися у відносинах між різними елементами системи, такими як товари, покупці і номера замовлень. Перш ніж взятися за складання бази даних, ЕР-діаграма дозволяє спочатку вибудувати її концепцію від загальної структури до конкретних способів взаємодії між складовими. Тому, при проєктуванні бази даних з метою розробки програмного забезпечення, ER-діаграми допоможуть наочно уявити структуру потрібної бази даних, а також виявити і усунути її недоліки вже на

початковому етапі роботи.

Нижче буде наведено реалізовану концептуальну ER-діаграму (див. рис. 2.8).

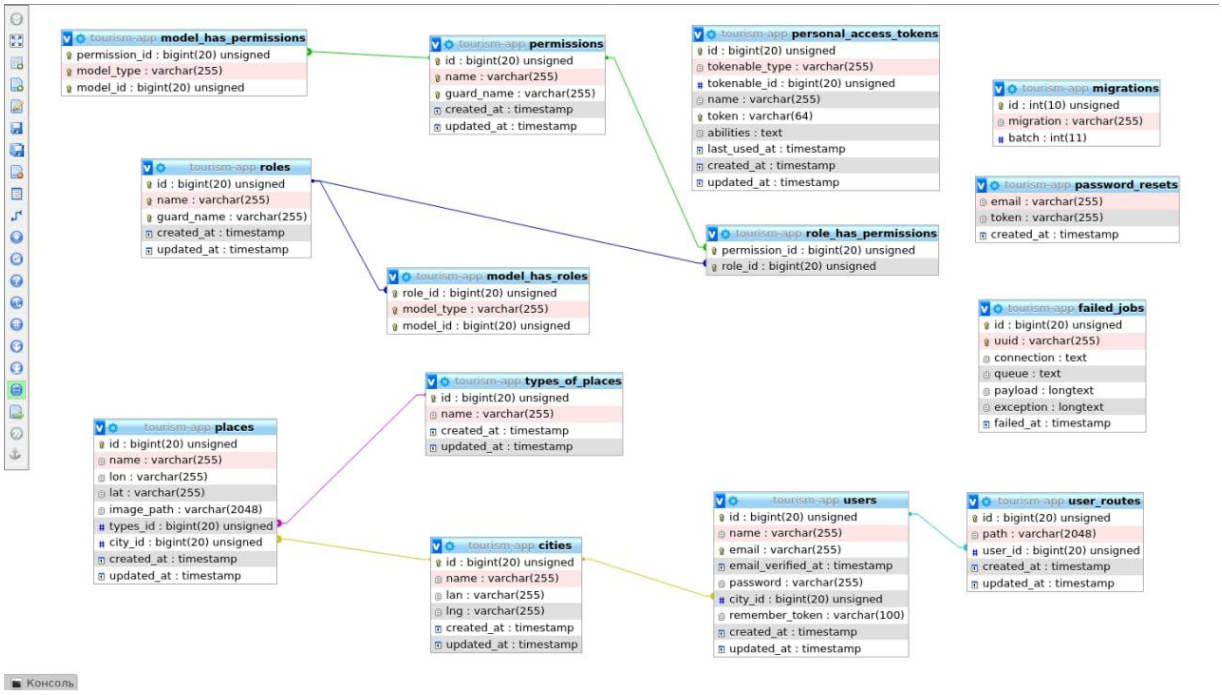

Рисунок 2.8 – ER-діаграма

2.6 Логічне проєктування

Основна думка логічного проєктування полягає у створення логічної моделі даних та перетворенні концептуальної моделі даних у логічну модель даної системи з урахуванням обраного типу СКБД.

Таблиця type_of_places (див. табл. 2.1) зберігає типи місць, який присвоюється самому місцю. Тобто завдяки цій таблиці адміністратор може створювати типи місць, щоб в майбутньому користувачі змогли побачити їх при виборі потрібних їм туристичних місць. Перший атрибут id – первинний ключ, на кожній таблиці він буде однаковий, тому його характеристики будуть перелічені лише один раз.

Таблиця 2.1 – Структура таблиці «types_of_places»

Назва	**Тип даних**	**Властивості**	**Зв'язок**
id	bigint(20)	Auto_increment	–
name	Varchar(255)	–	–
created_at	TimeStamps	nullable	–
updated_at	TimeStamps	nullable	–

Атрибут name зберігає назву типу місць та є символьним типом даних з обмеженням на кількість символів 255.

Атрибут created_at – час, коли було створено цей тип місць, та updated_at – час, коли тип місць був оновлений при редагуванні місць.

Таблиця places (див. табл. 2.2) є однією з головних таблиць всього логічного проектування. Вона зберігає туристичні місця, які є одним із найголовніших атрибутів. Завдяки цій таблиці, адміністратор може створювати місця, як із таблицею types_of_places.

Таблиця 2.2 – Структура таблиці «places»

Назва	**Тип даних**	**Властивості**	**Зв'язок**
id	bigint(20)	Auto_increment	–
name	Varchar(255)	–	–
lon	Varchar(255)	–	–
lat	Varchar(255)	–	–
image:path	Varchar(2048)	–	–
types_id	bigint(20)	–	Від types_of_places (id)
city_id	bigint(20)	–	Від cities (id)
created_at	TimeStamps	nullable	–
updated_at	TimeStamps	nullable	–

Атрибут Lon зберігає свою поточну довжину, наприклад, коли натискається кнопка, вона повертає поточну довжину на основі вибраного користувачем місцезнаходження.

Атрибут Lat зберігає свою поточну широту, наприклад, коли натискається кнопка, вона повертає поточну широту на основі вибраного користувачем місцезнаходження.

Атрибут name зберігає назву типу місць та є символьним типом даних з обмеженням на кількість символів 255.

Атрибут image:path зберігає шлях, до якого потрібно отримати доступ до зображення з моєї папки зображень під публічним шляхом.

Атрибут types_id зберігає первинний ключ типів місць, які користувач може переглядати у каталозі, та які адміністратор може добавляти та видаляти із системи.

Атрибут city_id зберігає первинний ключ місць, які користувач може переглядати у каталозі, та які адміністратор може добавляти та видаляти із системи.

Атрибут created_at – час, коли було створено цей тип місць, та updated_at – час, коли тип місць був оновлений при редагуванні місць.

Таблиця cities (див. табл. 2.3) займає вагому частину логічного проектування. Вона зберігає туристичні міста, які є одним із найголовніших атрибутів. Завдяки цій таблиці, адміністратор може створювати міста, як із таблицею places та types_of_places.

Таблиця 2.3 – Структура таблиці «cities»

Назва	**Тип даних**	**Властивості**	**Зв’язок**
id	bigint(20)	Auto_increment	–
name	Varchar(255)	–	–
lan	Varchar(255)	–	–
lat	Varchar(255)	–	–

Продовження таблиці 2.3

Назва	**Тип даних**	**Властивості**	**Зв'язок**
created_at	TimeStamps	nullable	–
updated_at	TimeStamps	nullable	–

Атрибут name зберігає назву типу місць та є символьним типом даних з обмеженням на кількість символів 255.

Атрибут Lan зберігає свою поточну довжину, наприклад, коли натискається кнопка, вона повертає поточну довжину на основі вибраного користувачем місцезнаходження.

Атрибут Lat зберігає свою поточну широту, наприклад, коли натискається кнопка, вона повертає поточну широту на основі вибраного користувачем місцезнаходження.

Атрибут created_at – час, коли було створено цей тип місць, та updated_at – час, коли тип місць був оновлений при редагуванні місць.

Таблиця users (див. табл. 2.4) зберігає самих користувачів, які повинні при вході на сайт зареєструватися чи авторизуватися, якщо вони вже були зареєстровані. Тому що через цю таблицю, адміністратор може створювати місця, як із таблицею typei_ of_places та цей атрибут присутній у самих місць.

Таблиця 2.4 – Структура таблиці «users»

Назва	**Тип даних**	**Властивості**	**Зв'язок**
id	Bigint(20)	Auto_increment	–
name	Varchar(255)	–	–
email	Varchar(255)	–	–
email_verified_at	TimeStamps	nullable	–
password	Varchar(255)	–	–
city_id	Bigint(20)	–	Від cities (id)

Продовження таблиці 2.4

Назва	**Тип даних**	**Властивості**	**Зв'язок**
remember_token	Varchar(100)	–	–
created_at	TimeStamps	nullable	–
updated_at	TimeStamps	nullable	–

Атрибут name зберігає назву типу місць та є символьним типом даних з обмеженням на кількість символів 255.

Атрибут email зберігає email користувачів, який вони вводять при реєстрації на сайті. Завдяки саме цьому атрибуту можна привласнювати туристичні місця юзерам, тобто місця будуть відіслані саме по цьому email користувачам. Атрибут email є символьним типом даних з обмеженням на кількість символів 255.

Атрибут email_verified_at – час, коли був сотворений цей email. Тип даних created_at та updated_at є TimeStamps.

Атрибут password зберігає пароль користувачів, який вони вводять при реєстрації на сайті. Завдяки саме цьому атрибуту користувачі авторизуються на сайті системи. Атрибут password є символьним типом даних з обмеженням на кількість символів 255.

Атрибут remember_token зберігає пароль користувачів у зашифрованому виді, який вони вводять при реєстрації на сайті. Атрибут remember_token є символьним типом даних з обмеженням на кількість символів 100.

Атрибут created_at – час, коли був сотворений цей користувач, та updated_at – час, коли користувач був оновлений при редагуванні особистого профілю користуваів. Тип даних created_at та updated_at є TimeStamps.

Таблиця user_routes (див. табл. 2.5) зберігає маршрути, які користувач може створювати на сайті. Ці маршрути будуть відображатися в системі на мапі після усіх вибраних місць всім зареєстрованим користувачам.

Таблиця 2.5 – Структура таблиці «user_routes»

Назва	**Тип даних**	**Властивості**	**Зв'язок**
id	bigint(20)	Auto_increment	–
path	Varchar(2048)	–	–
user_id	bigint(20)	–	Від users (id)
created_at	TimeStamps	nullable	–
updated_at	TimeStamps	nullable	–

Атрибут path може зберігати маршрут користувача та передавати його самим користувачам. Атрибут path є символьним типом даних з обмеженням на кількість символів 2048.

Атрибут user_id зберігає первинний ключ користувача, який він отримує при реєстрації на сайті системи.

Атрибут created_at – час, коли була сотворена ця книга, та updated_at – час, коли адміністратор оновив дані книжок при їх редагуванні. Тип даних created_at та updated_at є TimeStamps.

2.7 Висновки до розділу 2

У другому розділі були розглянуті вимоги до проєкту, а також доступні функціональні можливості для різних типів користувачів були представлені як діаграми прецедентів. Проєкт був реалізований у середовищі PhpStorm із використанням серверної платформи Node.js, а також CC найбільшою бібліотекою інтерфейсу. Також були використані такі фреймворки як Guzzle, Fruitcake Laravel-cors, Laravel, Laravel Breeze, Laravel Sanctum, Laravel Tinker.

Завдяки функціональному проектуванню наочно спроектовано модель інформаційної туристичної системи з використанням таких методології як IDEF0, а також діаграм декомпозиції, розгортання та діяльності.

Концептуальне проектування допомагає розібратися у відносинах між

різними елементами системи на основі збудованої ER-діаграми.

І останнє, логічне проєктування допомагає детальніше описати моделі даних концептуального проектування в логічну модель даної системи за допомогою побудованих таблиць і системи.

3 РОЗРОБКА ТУРИСТИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Початкові налаштування проєкту

З початку роботи над проєктом необхідно визначитися, які необхідно завантажити пакети для розробки, та як потрібно налаштувати зв’язок між серверною та клієнтською частиною додатку.

Через те, що у роботі з проєктом використовується Node.js, потрібна ініціалізація нового пакету за допомогою утиліти npm init. Вона запитує у розробника інформацію пов’язану з проєктом.

Як результат, директорія з проєктом повинна поповнитися новим файлом package.json що буде містити усі залежності проєкту та інформацію пов’язану з ним.

Далі потрібно завантажити пакети, що необхідні для розробки. Це виконується командою npm install <package-name>, або її скорочена форма npm i <package-name>. Після встановлення пакетів, вони повинні додатися до папки node_modules.

Після усіх махінацій з Node.js вміст файлу package.json з усіма необхідними інструментами повинен виглядати як на рисунку 3.1.

```
{  "private": true,
   "scripts":
    { "dev": "npm run development",
     "development": "cross-env NODE_ENV=development
node_modules/webpack/bin/webpack.js --progress --
config=node_modules/laravel-mix/setup/webpack.config.js",
     "watch": "npm run development -- --watch",
     "watch-poll": "npm run watch -- --watch-poll",
     "hot": "cross-env NODE_ENV=development
node_modules/webpack-dev-server/bin/webpack-dev-server.js --inline --hot --
disable-host-check --config=node_modules/laravel-mix/setup/webpack.config.js",
     "prod": "npm run production",
     "production": "cross-env NODE_ENV=production
node_modules/webpack/bin/webpack.js --no-progress --
config=node_modules/laravel-mix/setup/webpack.config.js"},
     "devDependencies":
    {"axios": "^0.19",
     "bootstrap": "^4.0.0",
     "cross-env": "^7.0",
     "jquery": "^3.2",
     "laravel-mix": "^5.0.1",
     "lodash": "^4.17.19",
     "popper.js": "^1.12",
     "resolve-url-loader": "^3.1.0",
     "sass": "^1.15.2",
     "sass-loader": "^8.0.0",
     "vue-template-compiler": "^2.6.12"    }}
```

Рисунок 3.1 – Текст коду package.json

Наступним кроком є встановлення Composer у наш проєкт. Оскільки ми працюємо з Laravel, то цей пакетний менеджер для PHP забезпечує стандартний формат для управління залежностями у програмному забезпеченні та необхідними бібліотеками.

Це не має бути дуже складно, тому що для встановлення Composer просто потрібно перейти на їх офіціальний сайт та скопіювати у консоль код, як показано на рисунку 3.2.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') === '756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d37e9a59d27d63f46af1d4d07ee0f76181c7d3') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Рисунок 3.2 – Текст коду Composer

Composer підключений і тепер через нього можна встановити фреймворк, який використовується у цьому проєкту – Laravel.

Для цього прописуємо команду по встановленні Laravel глобально на комп'ютер (див. рис. 3.3).

```
php composer .phar global require laravel / installer
```

Рисунок 3.3 – Текст коду Composer

Після усіх махінацій з Laravel вміст файлу composer.json з усіма необхідними інструментами повинен виглядати так, як на рисунках 3.4, 3.5.

```
{    "name": "laravel/laravel",
    "type": "project",
    "description": "The Laravel Framework.",
    "keywords": [
        "framework",
        "laravel"    ],
    "license": "MIT",
    "require": {
        "php": "^7.2.5|^8.0",
        "fideloper/proxy": "^4.4",
        "fruitcake/laravel-cors": "^2.0",
        "guzzlehttp/guzzle": "^6.3.1|^7.0.1",
        "laravel/framework": "^7.29",
        "laravel/tinker": "^2.5",
        "laravel/ui": "^2.4"    },
    "require-dev": {
        "facade/ignition": "^2.0",
        "fakerphp/faker": "^1.9.1",
        "mockery/mockery": "^1.3.1",
        "nunomaduro/collision": "^4.3",
        "phpunit/phpunit": "^8.5.8|^9.3.3"    },
    "config": {
        "optimize-autoloader": true,
        "preferred-install": "dist",
        "sort-packages": true    },
    "extra": {
        "laravel": {
            "dont-discover": []        }
    },
```

Рисунок 3.4 – Текст коду composer.json

```
    "autoload": {
        "psr-4": {
            "App\\": "app/"        },
        "classmap": [
            "database/seeds",
            "database/factories"        ]    },
    "autoload-dev": {
        "psr-4": {
            "Tests\\": "tests/"        }    },
    "minimum-stability": "dev",
    "prefer-stable": true,
    "scripts": {
        "post-autoload-dump": [
            "Illuminate\\Foundation\\ComposerScripts::postAutoloadDump",
            "@php artisan package:discover --ansi"        ],
        "post-root-package-install": [
            "@php -r \"file_exists('.env') || copy('.env.example', '.env');\""        ],
        "post-create-project-cmd": [
            "@php artisan key:generate --ansi"        ]    }}
```

Рисунок 3.5 – Текст коду composer.json

Далі створюється проєкт на основі вже встановленого Laravel. Після завантаження та додавання всіх файлів ми маємо шаблонний порожній проєкт.

3.2 Створення БД

Windows Subsystem for Linux (WSL) – підсистема ОС Windows 10 та Windows 11, що дозволяє розробникам, тестувальникам запускати нативні

програми Linux, писати скрипти, виконувати команди безпосередньо з Windows. В оновленій Windows з'явилася 2-я версія WSL, в якій використовується повноцінне ядро Linux з можливістю запуску додатків і контейнерів Docker, реалізована висока швидкість завантаження, невеликий обсяг ресурсів, що споживаються, управління у фоновому режимі, оновлення ядра. Таким чином, ви зможете запускати ELF64 програми, які можуть отримувати доступ до файлової системи Windows без використання сторонніх порто.

Образ ядра Linux в Windows 11 є легкою віртуальною машиною, для запуску якої не потрібно ставити повноцінну роль Hyper-V. Системні виклики Linux транслюються на льоту у виклики Windows без використання емулятора.

Ви можете увімкнути компоненти WSL у Windows 10 за допомогою PowerShell (див. рис. 3.6).

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
Enable-WindowsOptionalFeature -Online -FeatureName VirtualMachinePlatform
```

Рисунок 3.6 – Текст коду WSL у PowerShell

Тепер потрібно виконати оновлення WSL до версії 2. Для цього потрібно зайти на сайт Microsoft, завантажити файл wsl_update_x64.msi, встановити його. Після завершення побачите картинку, як показано на рисунку 3.7.

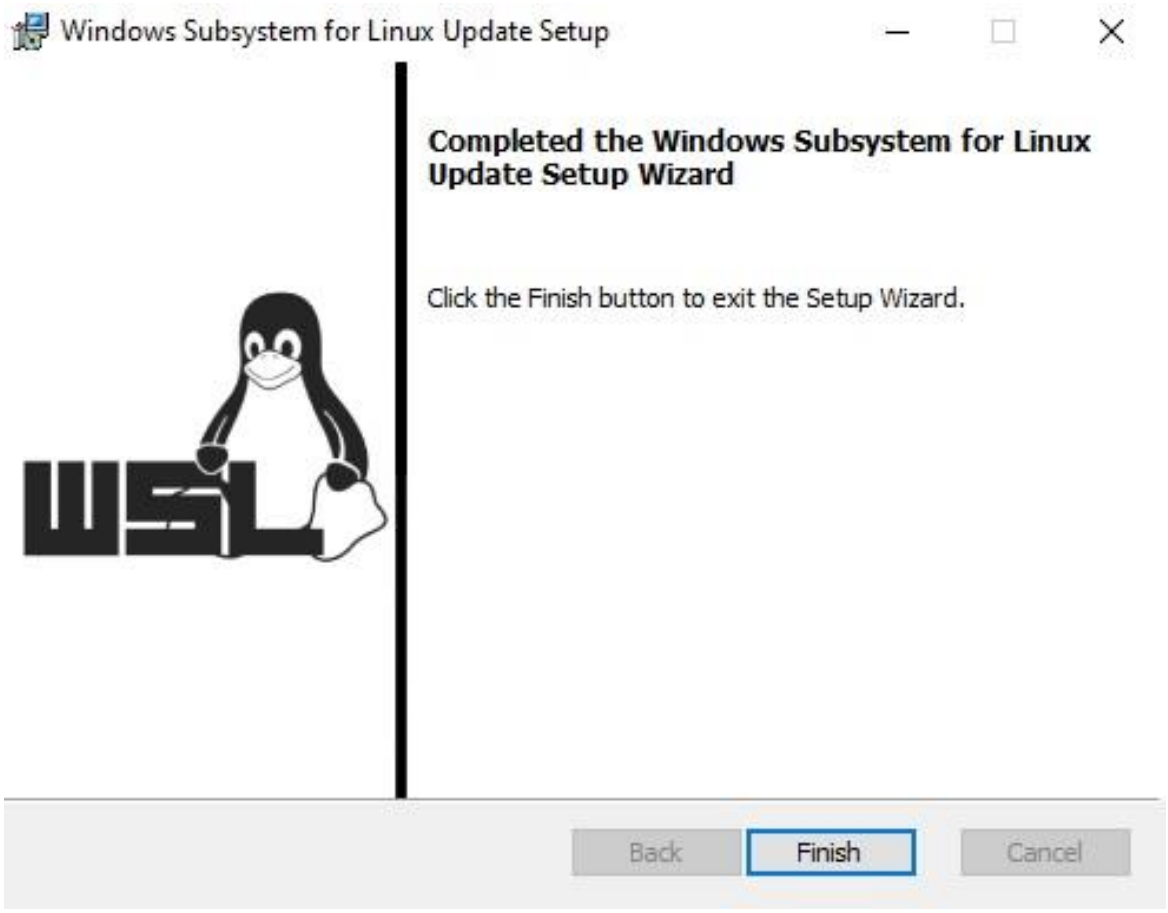

Рисунок 3.7 – Завершення встановлення wsl файлу

Далі вам необхідно відкрити Microsoft Store, у пошуку ввести слово «Linux». У списку виберіть потрібний дистрибутив. Доступні Ubuntu, Debian, Kali Linux, Linux Cheatsheet, SUSE Linux Enterprise Server15, OpenSUSE Leap 15-1, Pengwin Enterprise, Fedora Remix for WSL або інші.

Якщо у вас вимкнено Windows Store або ви хочете встановити дистрибутив WSL в Core редакції Windows Server, ви можете завантажити дистрибутив Ubuntu за допомогою PowerShell командлета Invoke-WebRequest (див. рис. 3.8).

```
Invoke-WebRequest https://aka.ms/wslubuntu2004 -OutFile ubuntu-2004.zip –UseBasicParsing.
```

Рисунок 3.8 – Текст коду дистрибутиву Ubuntu у PowerShell

3.3 Розробка дизайну проєкту

Перед початком роботи над макетами дизайну необхідно визначити, які сторінки буде містити клієнтська частина додатку. Тому на рисунку 3.9 показана карта сторінок туристичної інформаційної системи.

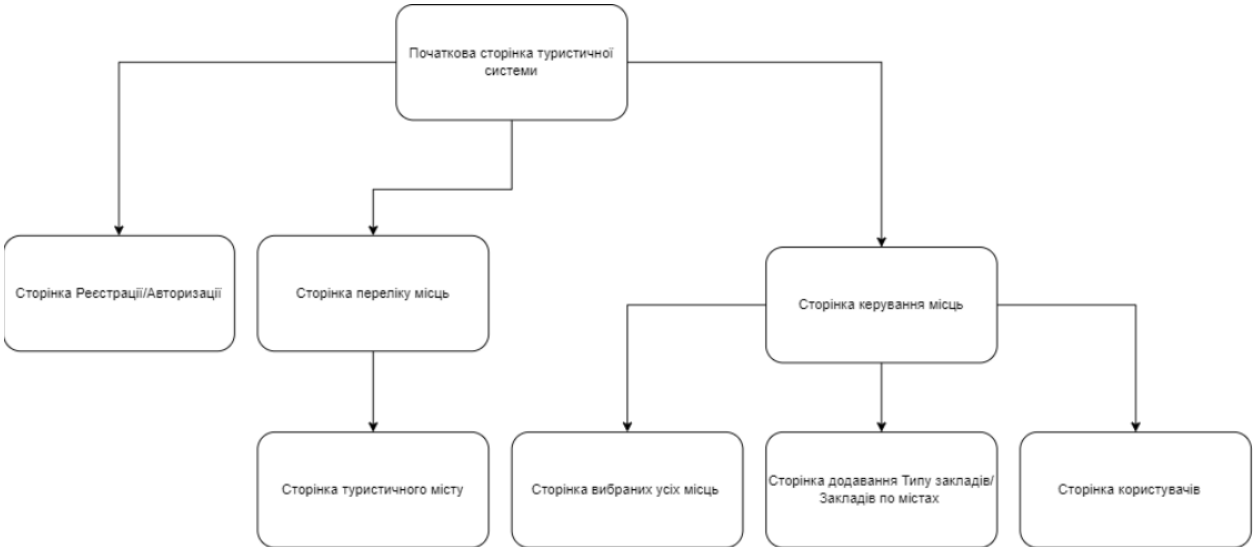

Рисунок 3.9 – Карта сторінок Туристичної системи

Наступним кроком є створення макетів сторінок з використанням готових шаблонів Tailwind UI. Першим кроком було створено головну сторінку туристичної системи, на якій має бути кнопка з реєстрацією та авторизацією, кнопка з переходом на сторінку особистого кабінету, кнопка за переходом на сторінку до переліку туристичних місць.

На початковій сторінці Туристичної системи відображається весь каталог існуючих туристичних місць, доданих адміністратором, які може бачити користувач, а також переглядати їх, як показано на рисунку 3.10.

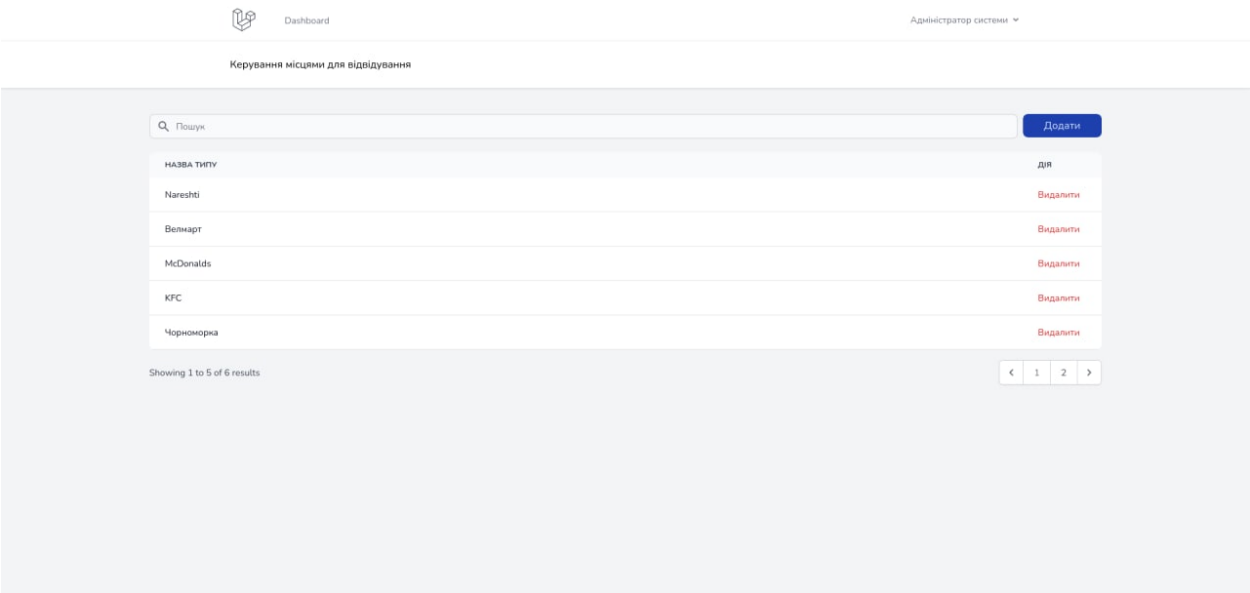

Рисунок 3.10 – Початкова сторінка Туристичної системи

Сторінка адміністратора/користувача, де вони можуть керувати своїми туристичними місцями, як показано на рисунку 3.11.

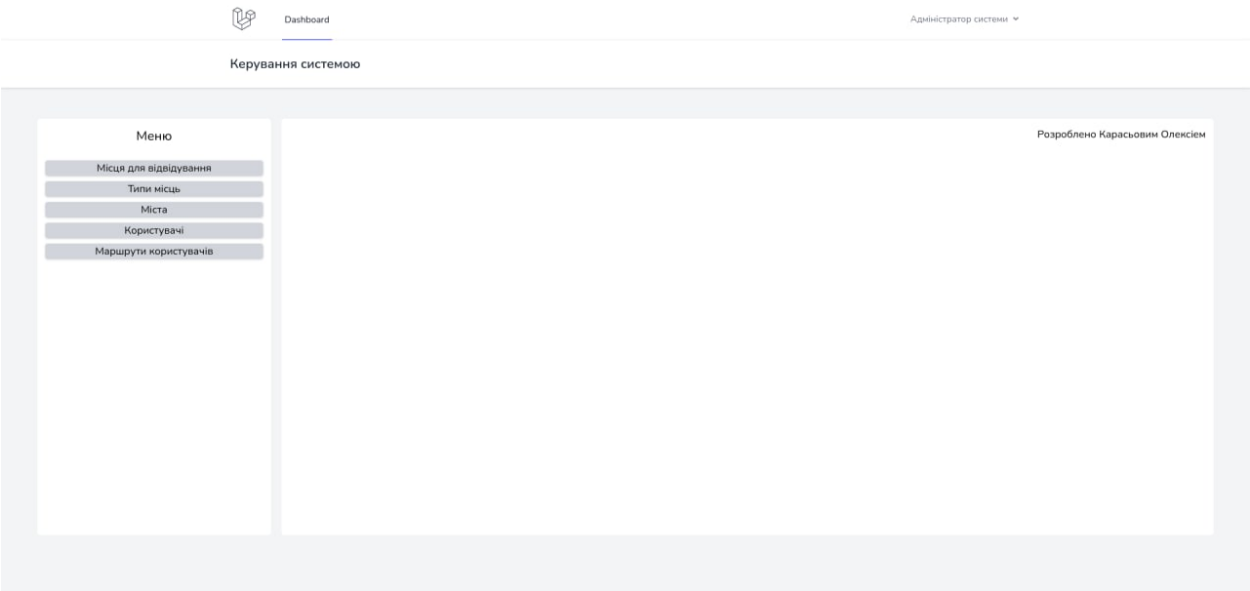

Рисунок 3.11 – Сторінка керування місць

На рисунку 3.12 показана сторінка адміністратора, де він може додавати тип місця для відвідування.

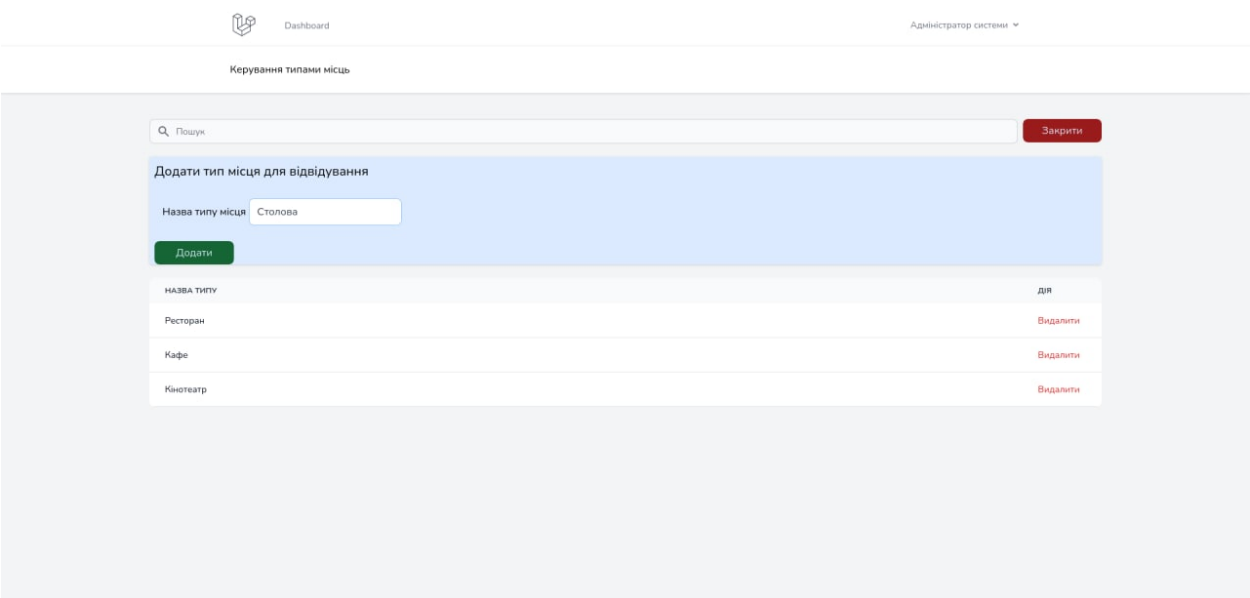

Рисунок 3.12 – Сторінка додавання типу місць

На рисунку 3.13 показана сторінка адміністратора, де він може додавати місто карті у якому знаходиться туристичний об’єкт для відвідування.

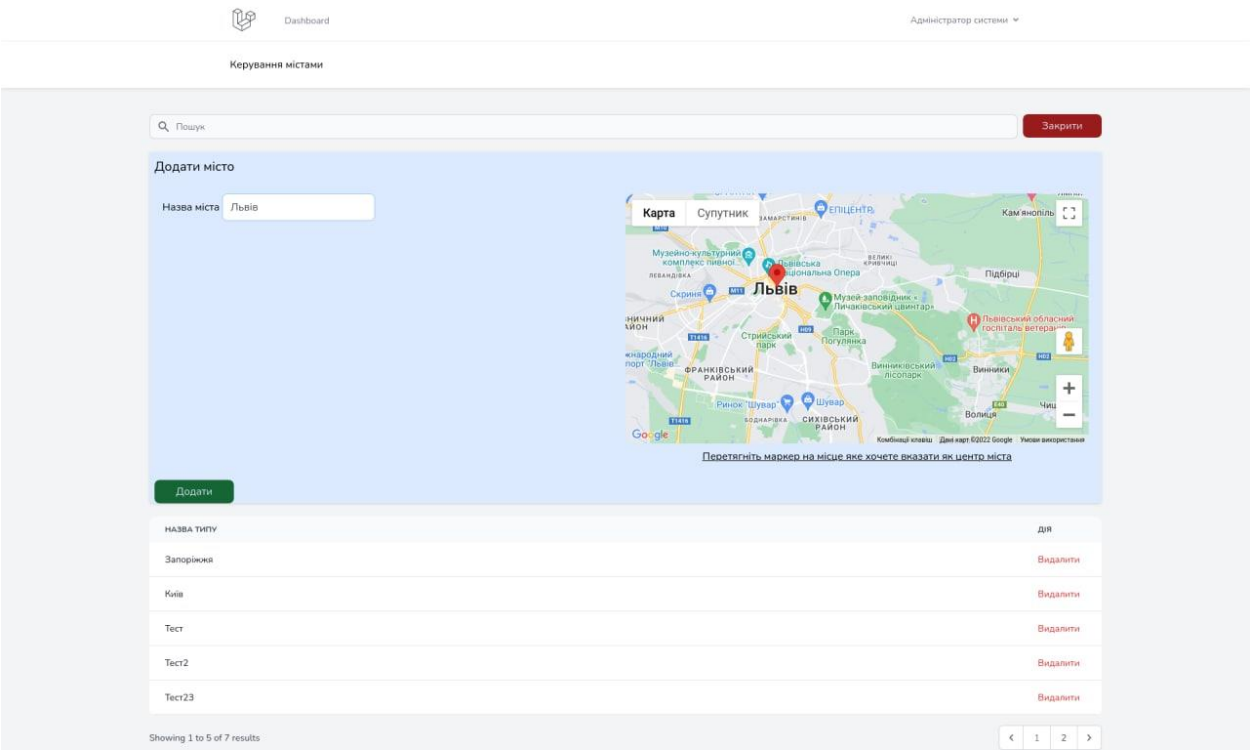

Рисунок 3.13 – Сторінка додавання місту туристичного об'єкта

На рисунку 3.14 показана сторінка адміністратора, де він може додавати місто та точку на Google карті у якому знаходиться туристичний об’єкт для відвідування.

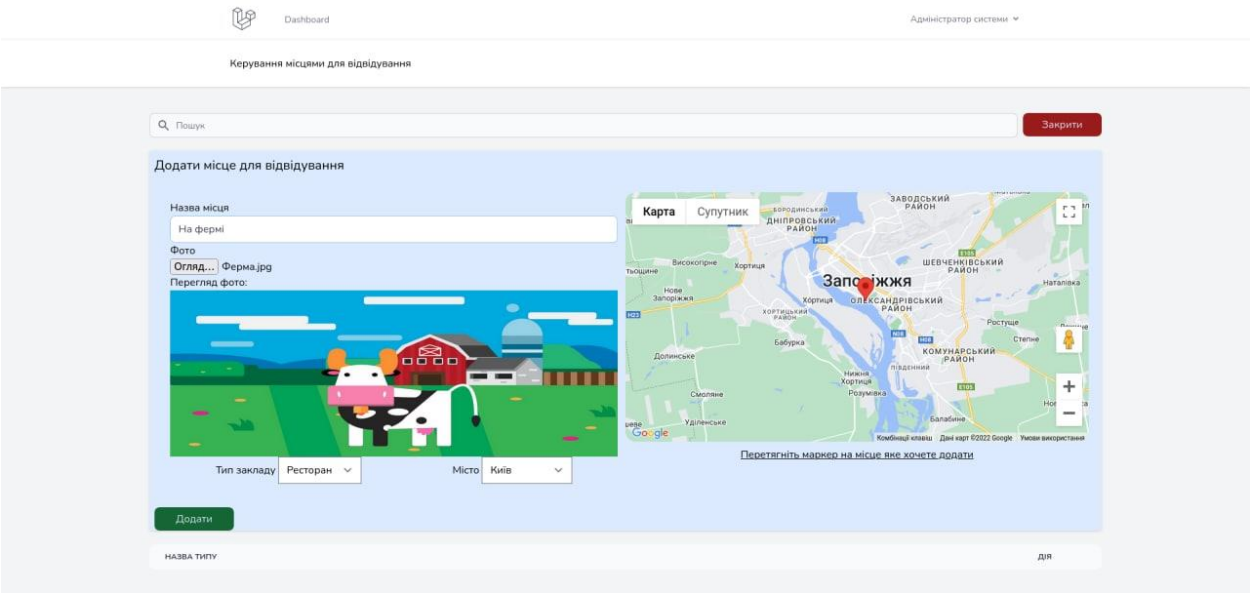

Рисунок 3.14 – Сторінка додавання місця відвідування

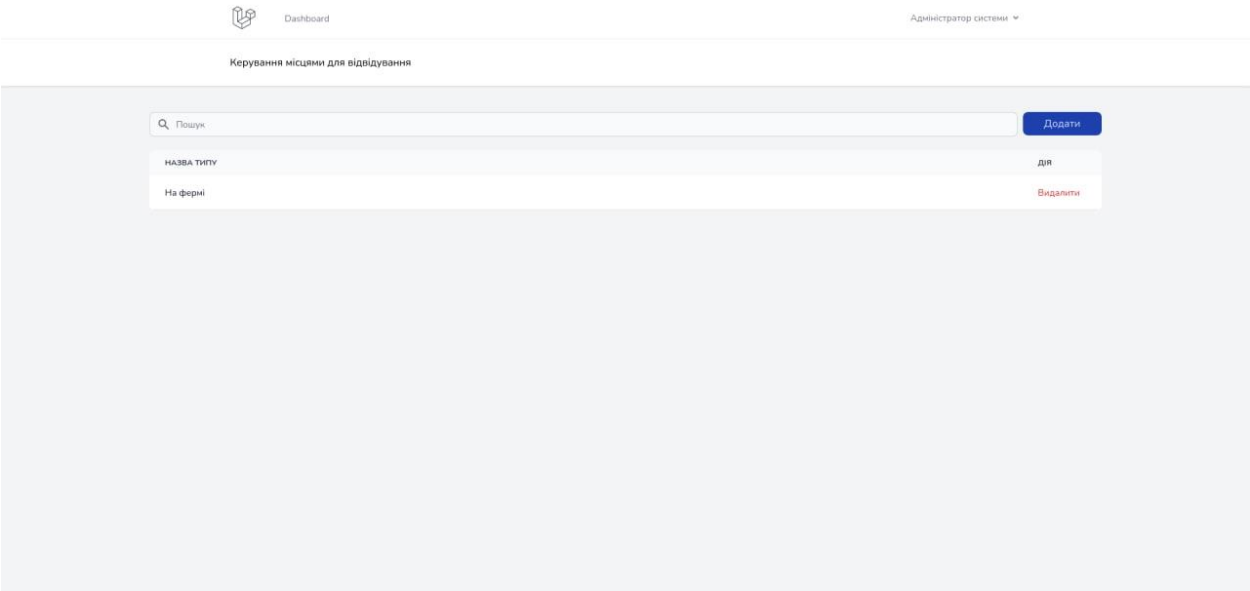

Рисунок 3.15 – Сторінка користувача

3.4 Middleware для аутентифікації

Практично на кожному сайті є розділ для авторизованих користувачів. І навіть більше того, всі їх дії обмежені авторизацією і ролями. Реалізація аутентифікації і авторизації на PHP – завдання не тривіальне і вимагає врахування безлічі винятків.

Так в Laravel включені middlewares [8] для перевірки аутентифікації користувача. Якщо користувач не авторизований, middleware перенаправляє його на сторінку логіна. Якщо ж в авторизацію – middleware не втручається в проходження запиту, передаючи його далі по ланцюжку middleware-посередників до власне додатком [9].

Так, як на рисунку 3.16, виглядає код отримання шляху, на який слід перенаправити користувача, коли він не авторизований.

```
class Authenticate extends Middleware
{
    protected function redirectTo($request)
    {
        if (!$request->expectsJson()) {
            return route('login');
        }
    }
}
```

Рисунок 3.16 – Текст коду шляху middleware

Також потрібно обробляти вхідний запит. За цей функціонал відповідає код, представлений на рис. 3.17.

```
class Authenticate extends Middleware
{
    protected function redirectTo($request)
    {
        if (!$request->expectsJson()) {
            return route('login');
        }
    }
}
```

Рисунок 3.17 – Текст коду вхідного запиту middleware

Останнє – це отримання шаблонів хостів, яким слід довіряти, а саме хосту, з яким треба працювати (див. рис. 3.18).

```
class RedirectIfAuthenticated
{
        public function handle($request, Closure $next, $guard = null)
    {
        if (Auth::guard($guard)->check()) {
            return redirect(RouteServiceProvider::HOME);
        }
        return $next($request);
    }
```

Рисунок 3.18 – Текст коду шаблонів хостів middleware

3.5 HTTP-контролери

Замість того, щоб визначати всю логіку обробки запитів у вигляді замикань в файлах Маршурт, ви можете організувати її за допомогою класів

контролерів. Контролери можуть групувати пов'язану з обробкою HTTP-запитів логіку в окремий клас. Контролери зберігаються в папці app / Http / Controllers [10].

Важливо пам'ятати, що при визначенні маршруту контролера не треба вказувати повне простір імен контролера, а тільки ту частину імені класу, яка слідує за «коренем» простору імен – App \ Http \ Controllers. Тому що RouteServiceProvider завантажує файли маршрутів разом з групою маршрутизації, що містить кореневий простір імен контролера.

Далі будуть приведені не складні контролери, які були застосовані в ході проєкту.

3.5.1 Контролер підтвердження

Цей контролер відповідає за обробку підтверджень паролів і використовує просту характеристику, щоб включити поведінку. Ви можете дослідити цю рису та замінити будь-які функції, які потребують налаштування (див. рис. 3.19).

```
{
    use ConfirmsPasswords;
    protected $redirectTo = RouteServiceProvider::HOME;
    public function __construct()
    {
        $this->middleware('auth');
    }
}
```

Рисунок 3.19 – Текст коду контролеру підтвердження

3.5.2 Контролер входу

Цей контролер обробляє автентифікацію користувачів програми та перенаправляє їх на головний екран. Контролер використовує ознаку, щоб зручно надати свої функції вашим додаткам (див. рис. 3.20).

```
{
    use AuthenticatesUsers;
    protected $redirectTo = RouteServiceProvider::HOME;
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
    }
}
```

Рисунок 3.20 – Текст коду контролеру входу

3.5.3 Реєстратор контролера

Цей контролер здійснює реєстрацію нових користувачів, а також їх перевірку та створення. За замовчуванням цей контролер використовує ознаку для забезпечення цієї функціональності, не вимагаючи додаткового коду (див. рис. 3.21).

```
{    use RegistersUsers;
    protected $redirectTo = RouteServiceProvider::HOME;
    public function __construct()
    {        $this->middleware('guest');    }
protected function validator(array $data)
    {        return Validator::make($data, [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],        ]);    }
    protected function create(array $data)
    {        return User::create([
            'name' => $data['name'],
            'phone' => $data['phone'],
            'admin' => 0,
            'email' => $data['email'],
            'password' => Hash::make($data['password']),        ]);    }}
```

Рисунок 3.21 – Текст коду контролеру реєстратору

3.5.4 Контролер перевірки електронної пошти

Цей контролер відповідає за перевірку електронної пошти будь-якого користувача, який нещодавно зареєструвався у програмі. Електронні листи також можуть бути надіслані повторно, якщо користувач не отримав оригінальне повідомлення електронної пошти (див. рис. 3.22).

```
{
    use VerifiesEmails;
    protected $redirectTo = RouteServiceProvider::HOME;
    public function __construct()
    {
        $this->middleware('auth');
        $this->middleware('signed')->only('verify');
        $this->middleware('throttle:6,1')->only('verify', 'resend');
    }
}
```

Рисунок 3.22 – Текст коду контролеру перевірки електронної пошти

3.5.5 Інші контролери туристичної інформаційної системи

В інші контролери сайту входить створювання туристичних місць, де адміністратор може створювати місця для користувачів, відображення типу місць, а також форму для редагування туристичного місця, оновлення вказаного туристичного місця у сховищі, також створення нового екземпляру контролера, контролер виду інформаційної панелі програм та інших контролери. Усі ці контролери можна переглянути у додатку А та у додатку Б.

ВИСНОВКИ

В процесі роботи над темою кваліфікаційної роботи були розширені і поглибленні знання з багатьох видів проєктування: процесу визначення архітектури, компонентів, інтерфейсів та інших характеристик системи.

Рівним чином, були отриманні практичні навички роботи з фреймворками, бібліотеками, різними інструментами роботи над проєктом, а також створення інформаційної туристичної системи, вебдодатку для візуалізації даних.

На прикладі розв’язання окремих задач, при створенні вебдодатку «Бібліотека» показана ефективність застосування фреймворку Laravel, Node.js та Tailwind UI та розв’язування прикладних задач.

Створений вебдодаток можна використовувати в туристичній та економічній галузі. Оскільки в сучасному світі туристичні системи є важливим атрибутом у разі швидкого виїзду за кордон або планування сімейної поїздки, розроблена туристична система дозволяє легко користуватися створеними місцями. Також завдяки унікально розробленому вебдодатку, користувачі можуть спокійно користуватися простим та ефективним способом пошуку туристичних місць, будь-якої країни та любого місця, а також наглядної побудови шляху, яким слідуватиме користувач у своєму турі. Туристична система надає вартість даного туру, а також можливість досконально дізнатися про всі додавання адміністратором місць, якими користувач може скористатися на своїй сторінці сайту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Стаффер М. Laravel. Полное руководство: 2-е издание. Санкт-Петербург, 2021. 512 с. URL: https://www.piter.com/collection/bestsellery-oreilly/product/laravel-polnoe-rukovodstvo-2-e-izdanie (дата звернення: 29.10.2022).
2. Дронов В. А. Laravel: быстрая разработка динамических Web-сайтов на PHP, MySQL, HTML и CSS. Санкт-Петербург, 2018. 768 с. URL: https://books.google.com.ua/books/about/Laravel_быстрая_разработ.html?id=psVDDwAAQBAJ&redir_esc=y (дата звернення: 29.10.2022).
3. Dockins K. Design Patterns in PHP and Laravel. Berkeley: Apress, 2017. 238 p. URL: https://doi.org/10.1007/978-1-4842-2451-9 (дата звернення: 29.10.2022).
4. Dangar H. Learning Laravel: 4 Application Development. Packt Publishing, 2013. 256 p. URL: https://www.amazon.com/Learning-Laravel-4-Application-Development/dp/1783280573 (дата звернення: 04.11.2022).
5. Сухов К. Node.js: Путеводитель по технологии. Санкт-Петербург: ДМК, 2015. 416 с. URL: https://codernet.ru/books/js/node_js_putevoditel_po_texnologii/ (дата звернення: 04.11.2022).
6. Get started with Bootstrap. Docs Bootstrap. URL : https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 04.11.2022).
7. Usage of server-side programming languages broken down by ranking; technologies overview. URL: https://w3techs.com/technologies/cross/programming_language/ranking (дата звернення: 04.11.2022).
8. Controllers – Laravel. The PHP Framework For Web Artisans. URL: https://laravel.com/docs/8.x/controllers (дата звернення: 04.11.2022).
9. Routing – Laravel. The PHP Framework For Web Artisans. URL:

https://laravel.com/docs/5.0/routing (дата звернення: 04.11.2022).

10. Jaenicke K. How To Create a Custom Middleware in Express.js. URL: https://www.digitalocean.com/community/tutorials/nodejs-creating-your-own-express-middleware (дата звернення: 04.11.2022).

ДОДАТОК А

Лістинг програми

```
namespace App\Http\Controllers;

use App\Models\Place;
use Illuminate\Http\Request;

class PlacesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
```

```
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    //
}

/**
 * Display the specified resource.
 *
 * @param  \App\Models\Place  $place
 * @return \Illuminate\Http\Response
 */
public function show(Place $place)
{
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  \App\Models\Place  $place
 * @return \Illuminate\Http\Response
 */
```

```
    public function edit(Place $place)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \App\Models\Place  $place
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Place $place)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Models\Place  $place
     * @return \Illuminate\Http\Response
     */
    public function destroy(Place $place)
    {
        //
    }
}
```

ДОДАТОК Б

Лістинг програми

```
namespace App\Http\Controllers;

use App\Models\TypesOfPlaces;
use Illuminate\Http\Request;

class TypesController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
```

```
 */
public function create()
{
    //
}

/**
 * Store a newly created resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function store(Request $request)
{
    //
}

/**
 * Display the specified resource.
 *
 * @param  \App\Models\TypesOfPlaces  $typesOfPlaces
 * @return \Illuminate\Http\Response
 */
public function show(TypesOfPlaces $typesOfPlaces)
{
```

```
    //
}

/**
 * Show the form for editing the specified resource.
 *
 * @param  \App\Models\TypesOfPlaces  $typesOfPlaces
 * @return \Illuminate\Http\Response
 */
public function edit(TypesOfPlaces $typesOfPlaces)
{
    //
}

/**
 * Update the specified resource in storage.
 *
 * @param  \Illuminate\Http\Request  $request
 * @param  \App\Models\TypesOfPlaces  $typesOfPlaces
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, TypesOfPlaces $typesOfPlaces)
{
    //
}
```

```
    /**
     * Remove the specified resource from storage.
     *
     * @param  \App\Models\TypesOfPlaces  $typesOfPlaces
     * @return \Illuminate\Http\Response
     */
    public function destroy(TypesOfPlaces $typesOfPlaces)
    {
        //
    }
}
```