МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій, обліку та фінансів
Кафедра комп'ютерних технологій
і моделювання систем

Кваліфікаційна робота
на правах рукопису

Саламатова Олександра Ілліча

УДК 004:72

КВАЛІФІКАЦІЙНА РОБОТА

Інформаційна технологія збереження інформації про об'єкти культурної спадщини

Спеціальності 122 "Комп'ютерні науки"

Подається на здобуття освітнього ступеня бакалавр

кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____

(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Молодецька Катерина Валеріївна
д.т.н., професор

Житомир – 2023

Висновок кафедри _____
за результатами попереднього захисту: _____

Протокол засідання кафедри _____
№ ____ від «_____» _____ 20____ р.

Завідувач кафедри комп'ютерних технологій і моделювання систем

_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)
«_____» _____ 20____ р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _____ захистив (ла)
(прізвище ,ім'я, по батькові)
кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____
за шкалою ECTS _____
за національною шкалою _____

Секретар ЕК

_____ _____ _____
(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

АНОТАЦІЯ

Випускна робота бакалавра представляє систему, яка дозволяє зберігати інформацію про об’єкти культурної спадщини України та надавати доступ до цієї інформації кінцевому користувачу. Пояснювальна записка до випускної роботи складається з 52 сторінок, 12 ілюстрацій та 7 таблиць.

Під час реалізації даного проєкту було визначено основні вимоги для кінцевої системи. Також було обґрунтовано вибір технологій для створення даної системи. Система буде реалізована за допомогою мови програмування PHP та фреймворку Laravel 10. Даний фреймворк дозволяє швидко та зручно створити web-сайт, використовуючи архітектурний шаблон MVC. В якості системи управління базами даних було обрано MySQL. Для реалізації графічного дизайну сайту було обрано CSS фреймворк Tailwind CSS. Він дозволяє значно пришвидшити роботу з написання CSS стилів. Разом з Tailwind CSS було обрано JavaScript фреймворк Alpine.js, що дозволило додати “реактивність” в браузерну логіку web-сайту. Система буде впроваджена за допомогою використання веб-серверу NGINX та під керівництвом ОС Linux.

КЛЮЧОВІ СЛОВА: PHP, LARAVEL, WEB-САЙТ, ІНФОРМАЦІЙНА СИСТЕМА, ІНФОРМАЦІЯ, ДАНІ.

SUMMARY

The bachelor's graduate work presents a system that allows to store information about heritage objects in Ukraine. Documentation for the bachelor′s graduate work contains 52 pages, 12 illustrations, and 7 charts.

The main requirements for the end system have been defined during the project realization. The choice of technologies for the system implementation has been substantiated as well. The system will be implemented using the PHP programming language and Laravel 10 framework. The framework allows us to create a website quickly and conveniently using the MVC architecture pattern. MySQL has been chosen as the database management system. Tailwind CSS has been chosen as a CSS framework to implement the website's graphic design. The framework allows us to increase the speed of CSS styling. Together with the Tailwind CSS, Alpine.js has been chosen, which allowed us to add a “reactivity” to the website's browser logic. The system will be implemented using the NGINX web server and Linux OS.

KEYWORDS: PHP, LARAVEL, WEBSITE, INFORMATION SYSTEM, INFORMATION, DATA.

ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій, обліку та фінансів

Кафедра комп'ютерних технологій і моделювання систем

Спеціальність 122 «Комп’ютерні науки»

“ЗАТВЕРДЖУЮ”

З завідувача кафедри комп’ютерних технологій і моделювання систем

_____О. М. Николюк

“___” _____ 2023 р

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Саламатова Олександра Ілліча

1. Тема кваліфікаційної роботи: «Інформаційна технологія збереження інформації про об'єкти культурної спадщини»».

затверджена наказом № 1196/ст від «04» жовтня 2022 р.

2. Термін подання роботи: 16.06.2023.

3. Предмет дослідження – є використання веб-технологій для інформатизації процесу зберігання інформації про об’єкти культурної спадщини.

4. Об’єкт дослідження: є процес інформатизації процесу зберігання інформації про об’єкти культурної спадщини України.

5. Методи дослідження: методи моделювання, методи аналізу, методи порівняння, методи проектування, методи комп'ютерного моделювання.

6. Інформаційна база дослідження: вітчизняні та зарубіжні навчально-наукові видання, інформаційні і довідникові видання, бібліографічні ресурси, інформаційні ресурси.

7. Зміст роботи: проектування архітектури, розробка алгоритмів та реалізація системи зберігання інформації про об’єкти культурної спадщини України.

8. Перелік графічного матеріалу: 7 табл., 12 Рисунків., 3 дод., 10 джерел.

9. Дата видачі завдання: 10.10.2022.

Керівник роботи

науковий ступінь, вчене звання _____ д.т.н., проф. К. В. Молодецька

Завдання прийняв
до виконання _____ О.І. Саламатов

КАЛЕНДАРНИЙ ПЛАН РОБОТИ

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1.	Підготовка тез для участі в конференції	04.02.2023	виконано
2.	Написання першого розділу	14.02.2023	виконано
3.	Написання другого розділу	29.03.2023	виконано
4.	Написання третього розділу	17.05.2023	виконано
5.	Оформлення списку літературних джерел	25.05.2023	виконано
6.	Оформлення додатків	26.05.2023	виконано
7.	Підготовка матеріалів до друку	15.06.2023	виконано
8.	Підготовка презентації для доповіді	18.06.2023	виконано

Здобувач вищої освіти _____ О.І. Саламатов

Керівник роботи

науковий ступінь, вчене звання _____ д.т.н., проф. К. В. Молодецька

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRUD – create, read, update, delete (*з англ.* створення, читання, оновлення, видалення)

MVC – Model–View–Controller (*з англ*. Модель–Представлення–Контролер)

SQL − structured query language (*з англ.* мова структурних запитів)

БД – база даних

ОС – операційна система

СУБД – система управління базами даних

ВСТУП

Актуальність теми. Збереження об'єктів культурної спадщини залишається надзвичайно актуальним завданням у сучасному світі. Ось кілька причин, чому системи для збереження об'єктів культурної спадщини продовжують бути важливими і корисними:

1. **Передача культурної спадщини майбутнім поколінням**. Об'єкти культурної спадщини відображають унікальну історію, традиції і цінності різних культур. Збереження цих об'єктів дозволяє передати цю спадщину майбутнім поколінням, зберігаючи інформацію про нашу історію і культуру.

2. **Захист від пошкоджень та знищення**. Багато об'єктів культурної спадщини піддаються ризику пошкоджень, знищення або втрати через природні катастрофи, війни, недбалість або незаконну діяльність. Системи збереження дозволяють вжити заходів для захисту цих об'єктів, включаючи фізичну охорону, контроль вологості та температури, моніторинг стану тощо.

3. **Дослідження і освіта**. Збереження об'єктів культурної спадщини створює можливості для досліджень, археологічних розкопок, реставрації та вивчення різних аспектів минулих цивілізацій. Це не тільки допомагає розширити наші знання про минуле, але і надає освітні та культурні ресурси для громадськості.

4. **Туризм і економіка**. Багато об'єктів культурної спадщини стають туристичними атракціями, які приваблюють відвідувачів з усього світу. Використання систем для збереження об'єктів культурної спадщини допомагає зберегти їх і забезпечує сталість туристичного потоку, що сприяє розвитку місцевої економіки.

Отже, системи для збереження об'єктів культурної спадщини мають значення як для збереження цінностей минулого, так і для забезпечення розвитку та економічної стійкості сучасного суспільства.

Мета і завдання роботи. Метою бакалаврської роботи є проектування архітектури, розробка алгоритмів та реалізація системи зберігання інформації про об’єкти культурної спадщини України.

Визначена мета роботи обумовлює появу наступних завдань:

– аналіз об’єктів культурної спадщини України;

– проектування складових частин системи та їх взаємодію;

– обрання засобів реалізації програмної системи;

– реалізація системи для зберігання інформації про об’єкти культурної спадщини України;

– тестування реалізованого програмного продукту;

– опис можливостей використання та налаштування системи.

За темою кваліфікаційної роботи було опубліковано наукові тезиси, а саме:

- Саламатов О. І. Концептуальне проєктування інформаційної системи збереження культурної спадщини Студентські наукові читання – 2022 : зб. тез доповідей науково-практичної конференції «Студентські наукові читання – 2022» за результатами І туру Всеукраїнського конкурсу студентських наукових робіт на факультеті інформаційних технологій, обліку та фінансів. Житомир : Поліський національний університет, 2022. С. 83–85;
- Саламатов О.І. Web-орієнтована інформаційна система збереження даних про культурну спадщину. Інформаційні технології та моделювання систем : збірник праць учасників Всеукраїнської науково-практичної конференції здобувачів вищої освіти і молодих вчених, 30 березня 2023 р. Житомир : Поліський національний університет, 2023. С. 113–114.

Об’єктом дослідження є процес інформатизації процесу зберігання інформації про об’єкти культурної спадщини України.

Предметом дослідження є використання веб-технологій для інформатизації процесу зберігання інформації про об’єкти культурної спадщини.

Структура та обсяг роботи. Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 10 найменувань (1 стор.), 3 додатки (19 стор.), 7 таблиць та 12 ілюстрацій по тексту. Повний обсяг роботи становить 52 сторінки, з них 20 сторінок основного тексту.

РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЙ ІНФОРМАТИЗАЦІЇ ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ ПРО ОБ'ЄКТИ КУЛЬТУРНОЇ СПАДЩИНИ

1.1 Особливості збереження інформації про об'єкти культурної спадщини

Збереження інформації про об'єкти культурної спадщини має свої особливості, оскільки вимагає детального дослідження, документування і захисту цінних артефактів і пам'яток

Разом з цим є ряд особливостей, які стосуються саме цифрового збереження об’єктів культурної спадщини. Дуже важливо враховувати формати файлів. Вони повинні бути зручні для довготривалого зберігання. Наприклад, JPEG для зображень і PDF для документів, забезпечують високу якість та широку сумісність. Також потрібно взяти до уваги зберігання метаданих про об’єкти культурної спадщини, адже вони грають велику роль у цифровому збереженні. До метаданих можуть входити інформація про походження об'єкта, дату, автора, контекст, історію та інші важливі деталі. Важливо розробити структуровану систему метаданих, щоб забезпечити легкий пошук, ідентифікацію та організацію цифрових ресурсів.

Не менш важливо реалізувати захист системи від втрати та пошкоджень. Цифрові дані піддаються ризику втрати або пошкодження внаслідок технічних проблем, хакерських атак, тощо. Для забезпечення безпеки важливо налагодити процеси резервного копіювання, а також мати стратегію відновлення даних у разі потреби.

Слід також врахувати проблему довготривалої доступності до збереженої інформації, що і є одним з основних завдань цифрового збереження інформації.

Зважаючи на всі перелічені особливості збереження інформації про об'єкти культурної спадщини, було вирішено обрати веб-технології, для реалізації даної системи. Адже саме ці технології в повній мірі дозволяють вирішити всі завдання, які були визначені в ході аналізу підходів збереження інформації.

Для створення інформаційної системи було обрано мову програмування PHP 8.2 та фреймворк Laravel 10. Дані технології дозволяють створити веб сайт, використовуючи архітектурний шаблон MVC.

MVC – достатньо розповсюджений архітектурний патерн, що досить часто використовується для побудови веб-додатків. Переваги застосування даної моделі є численними. Багато класів у додатках, які використовують MVC, зазвичай, повторно використовуються, та їх інтерфейси, як правило, краще визначені. Програмні продукти, що побудовані за допомогою даного патерну, також легше розширюються, ніж інші програми [10].

При правильному використанні принципів MVC, можна отримати наступні переваги в розробці:

– швидкий процес розробки;

– можливість реалізовувати додаток у вигляді модулів;

– можливість використовувати багато представлень;

– додавання представлень не змінює логіку моделі.

1.2 Аналіз інформаційних потреб предметної області

Аналіз інформаційних потреб предметної області, такої як збереження об'єктів культурної спадщини, є важливим кроком у розробці ефективної системи. Для розуміння цих потреб варто було звернути увагу на декілька аспектів, від яких залежить проектування та реалізація інформаційної системи.

Для початку необхідно було визначитися з типами даних, які пов'язані з об'єктами культурної спадщини. Це можуть бути фотографії, текстові документи, розмітка, 3D-моделі, аудіо- та відеозаписи, архіви тощо. Визначення різних типів даних дасть змогу, зрозуміти, на етапі проектування, які інструменти і технології потрібні для їх збереження і обробки.

Наступним кроком було визначення обсягу даних, а саме, які обсяги даних потрібно зберігати і обробляти. Розуміння обсягу і масштабу даних дозволить

вибрати потрібну апаратну і програмну інфраструктуру для збереження та обробки інформації.

Подальшим аспектом аналізу потреб предметної області був аналіз пошуку та доступу до інформації. Користувачі можуть бажати швидкого і зручного пошуку об'єктів за різними критеріями, включаючи географічне розташування, роки створення, типи матеріалів тощо. Аналіз потреб користувачів допоміг розробити ефективний та зручний дизайн системи.

Завершальним етапом аналізу був аналіз вимог до безпеки та конфіденційності даних. Розуміння цих вимог допомогло встановити відповідні рівні захисту даних, використовуючи шифрування, контроль доступу та інші методи.

Проаналізувавши інформаційні потреби даної предметної області, було сформовано такі функціональні та нефункціональні можливості системи:

- система повинна бути представлена у вигляді вебсайту;
- вебсайт повинен мати зручний пошук по ключовим словам , назві об’єкту, охоронному номеру тощо;
- всі посилання вебсайту повинні бути SEO-оптимізовані, для забезпечення його просування в пошукових системах;
- адміністрування системи та додавання інформації повинно бути здійснене за допомогою користувачів-адміністраторів;
- головний адміністратор системи повинен мати змогу створювати, редагувати та видаляти інших користувачів-адміністраторів;
- вхід в систему в ролі адміністратора повинен бути забезпечений за допомогою електронної пошти користувача та захищеного паролю;

Висновки до розділу 1

В даному розділі було сформовано основне завдання та мету впровадження програмної системи для збереження об'єктів культурної спадщини. Разом з цим було здійснено аналіз інформаційних потреб предметної області та визначено

особливості збереження інформації даної системи. Як результат, було сформовано вимоги до програмної системи, методи та засоби її реалізації.

Таким чином було прийнято рішення, що система повинна включати в себе наступну функціональність:

- ➢ пошук по ключовим словами, назві об’єкту, охоронному номеру;
- ➢ SEO-оптимізовані посилання на всі сторінки вебсайту;
- ➢ адміністрування системи за допомогою панелі адміністратора;
- ➢ вхід в систему за допомогою електронної пошти користувача та захищеного паролю;
- ➢ встановлення різного рівня доступу до інформації для користувачів-адміністраторів, які мають різні ролі.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ ПРО ОБ'ЄКТИ КУЛЬТУРНОЇ СПАДЩИНИ

2.1. Узагальнена структурна схема

Система для збереження об'єктів культурної спадщини має наступні цілі: вона розробляється для полегшення зберігання даних про об’єкти культурної спадщини та захисту цих даних від різних чинників.

Кінцевий користувач повинен мати змогу користуватися системою, виконуючи мінімум маніпуляцій та налаштувань. На додачу до цього, система мусить бути простою та швидкою в адмініструванні. Також необхідна підтримка одночасної роботи декількох користувачів-адміністраторів.

Зовнішні користувачі повинні мати змогу виконувати наступні дії:

1. Перегляд списку об’єктів культурної спадщини.
2. Перегляд детальної інформації про об’єкт.
3. Здійснення пошуку за ключовими словами назви об’єкту.
4. Здійснення пошуку за охоронним номером.
5. Здійснення пошуку за ключовими словами детальної інформації об’єкту.

6. Отримання більш детальної інформації про типи та види об’єктів культурної спадщини.

Користувачі-адміністратори повинні мати змогу додавати, редагувати та видаляти об’єкти культурної спадщини в системі. Нижче представлено список функцій для користувачів панелі керування:

1. Авторизація за допомогою логіна та пароля.

2. Редагування інформації профілю.

3. Редагування об’єктів культурної спадщини.

4. Додавання об’єктів культурної спадщини.

5. Видалення об’єктів культурної спадщини.

Функціональні вимоги:

1. Авторизація адміністраторів за допомогою логіну та паролю.

2. Можливість додавати користувачів-адміністраторів.

3. Можливість видаляти користувачів-адміністраторів.

4. Можливість змінювати дані профілю користувачів-адміністраторів.

5. Можливість додавати інформацію про об’єкти культурної спадщини.

6. Можливість редагувати інформацію про об’єкти культурної спадщини.

7. Можливість видаляти інформацію про об’єкти культурної спадщини.

8. Пошук інформації про об’єкт культурної спадщини за ключовими словами та охоронним номером.

Нефункціональні вимоги:

1. Навчання роботи з системою:

– час відповіді системи для запитів не повинен перевищувати 1 секунду;

– інтерфейс системи повинен бути інтуїтивно зручним для користувача та не вимагати додаткової підготовки;

– надійність;

– доступність – час, потрібний для обслуговування системи не повинен перевищувати 10% від загального часу роботи;

– середній час безперервної роботи – 7 робочих днів;

– максимальна норма помилок та дефектів в роботі системи – 1 помилка на 100 запитів користувача.

2. Продуктивність

Система повинна підтримувати мінімум 1000 одночасно працюючих користувачів.

Можливість експлуатації

- масштабування – система повинна мати можливість збільшувати потужності (продуктивність), зі збільшенням користувачів таким чином, щоб це негативно не відобразилося на її роботі;
- оновлення версій – оновлення версій повинно здійснюватися автоматично.

Для реалізації системи було обрано мову програмування PHP 8.2 та фреймворк Laravel 10. Laravel - це відкритий фреймворк для розробки веб-додатків, написаний на мові програмування PHP. Він надає зручні інструменти і структуру для швидкої та ефективної розробки веб-додатків з високою якістю коду [5].

У якості основного патерну проектування було обрано архітектурний шаблон MVC.

Система збереження інформації про об’єкти культурної спадщини має широкий спектр функцій, серед яких є авторизація користувачів-адміністраторів, пошук інформації за ключовими словами та охоронним номером, менеджмент даних зі сторони адміністратора. Всі ці можливості реалізуються класами та методами описаними вище.

2.2. Розроблення бази даних

База даних системи «heritage_database» включає в себе 5 таблиць, що містять у собі всю необхідну інформацію для роботи програми.

Таблиця 2.1

Назва та призначення таблиць БД «heritage_database»

Назва таблиці	Призначення
users	Таблиця адміністраторів
password_reset_tokens	Таблиця токенів для відновлення паролю
items	Таблиця об’єктів культурної спадщини
types	Таблиця типів об’єктів культурної спадщини
kinds	Таблиця видів об’єктів культурної спадщини

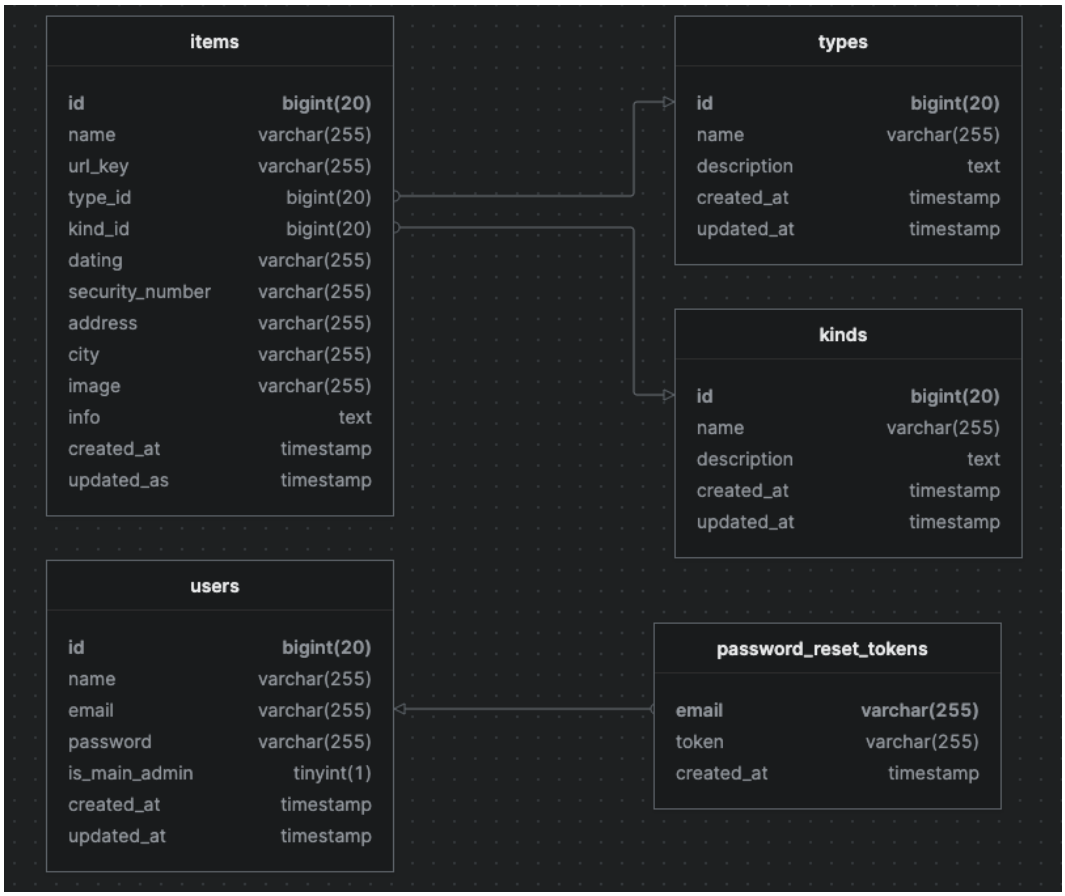


Рис. 2.4. Структурна схема бази даних

Тепер опишемо всі таблиці бази даних, які застосовуються в інформаційій технології збереження інформації про об'єкти культурної спадщини.

Таблиця «users» створена для зберігання даних про користувачів-адміністраторів. Структура таблиці наведена у додатку А (табл. А.1).

Таблиця «password_reset_tokens» використовується для зберігання токенів, які потрібні для процедури відновлення паролю, коли користувач натискає посилання “Забув пароль”. Структура таблиці наведена у додатку А (табл. А.2).

Таблиця «items» призначена для збереження основної інформації про об’єкти культурної спадщини. Структура таблиці наведена у додатку А (табл. А.3).

Таблиця «types» створена для зберігання типів об’єктів культурної спадщини. Структура таблиці наведена у додатку А (табл. А.4).

Таблиця «kinds» створена для зберігання видів об’єктів культурної спадщини. Структура таблиці наведена у додатку А (табл. А.5).

В результаті проектування БД, було створено базу даних, яка містить 5 таблиць. Отримана база даних є реляційною, так як містить зв’язки між таблицями, реалізовані за допомогою первинних та зовнішніх ключів. Типи даних полів були влучно підібрані, зважаючи на тип інформації, яка повинна зберігатися в цих полях.

2.3. Моделювання бізнес-процесів

Одним з основних процесів системи є авторизація користувача-адміністратора в даній системі за допомогою електронної пошти та паролю.

Після запуску вебсайту користувачу необхідно перейти на сторінку авторизації, яка знаходиться за шляхом “/admin”. Після цього перед користувачем відобразиться форма, в якій наявні 2 текстових поля “Email” та “Пароль”, а також кнопка “Вхід”, яка саме і здійснює відправку форми на сервер. Після того, як користувач заповнює дані текстові поля та натискає кнопку, відправляється POST запит на сервер, в якому наявні параметри з даними форми.

Наступним етапом авторизації є хешування паролю та виконання запиту в базу даних для перевірки коректності даних з форми.

Хешування паролю є дуже важливим перед відправкою в БД, адже всі паролі в БД є захешованими. І саме хеші паролів, а не самі паролі, порівнюються між собою.

У разі, якщо відповідний email та хеш паролю знайдено в базі даних, виконується генерування контенту головної сторінки панелі адміністратора. Відбувається перенаправлення користувача на дану сторінку та відображення контенту.

Якщо ж email та хеш паролю не знайдено в базі даних - користувача буде перенаправлено на сторінку авторизації та відображено помилку авторизації.

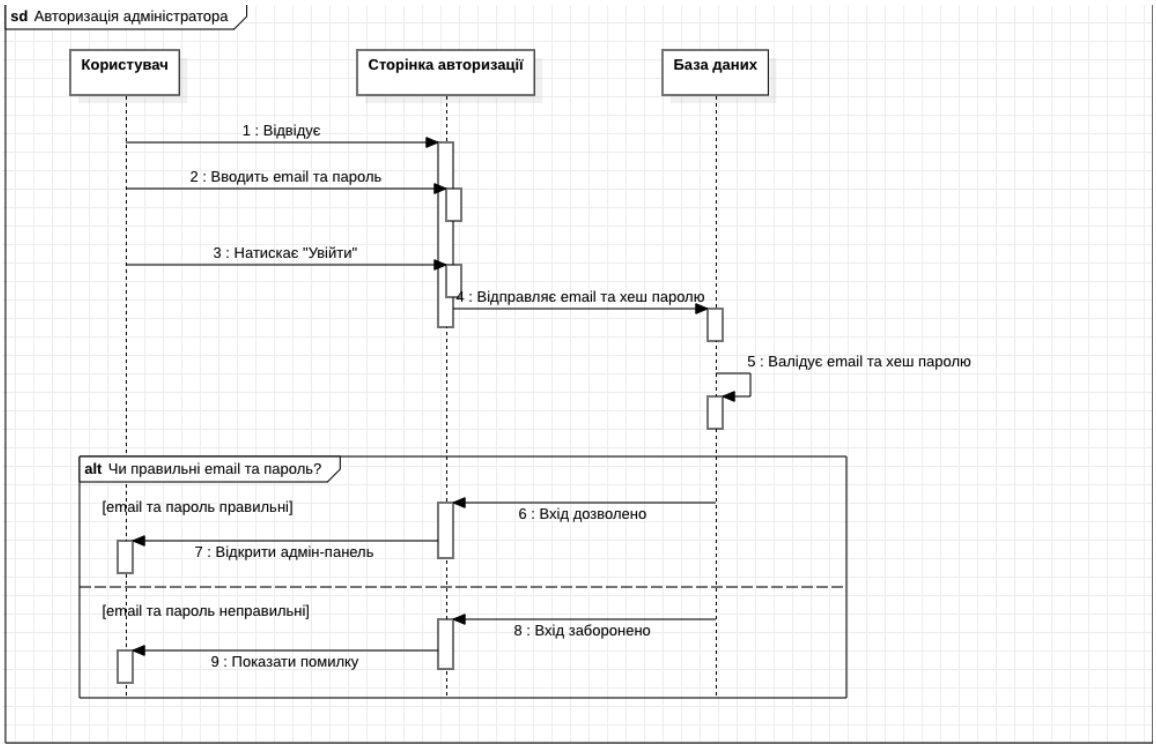


Рис. 2.5. Діаграма послідовності авторизації адміністратора

Зупинимось детальніше на процесі рендерингу веб сторінки авторизації, яка відкривається, коли користувач переходить за шляхом “/admin”. Для того, щоб система розуміла, яку сторінку відображати при переході за тією чи іншою адресою, використовується роутинг.

Роутинг (routing) - це механізм, який визначає, як веб-додаток відповідає на запити клієнта до конкретних URL-адрес. У контексті веб-розробки роутинг використовується для визначення шляхів (routes), які відповідають за обробку запитів і повернення відповідних результатів [7].

В фреймворці Laravel за роутинг відповідають PHP файли, які знаходяться в директорії “routes”, що лежить в корні системи [7].

За допомогою статичного методу “Route::middleware()”, що приймає теперішній статус користувача (в даному випадку “guest”, тобто не авторизований), ми визначаємо, що даний шлях буде виконуватись лише для не авторизованих користувачів. Наступним етапом є “реєстрація” шляху за допомогою методу “Route::get()”. Назва методу “get” визначає що шлях буде зареєстровано виключно для GET запитів. Для прикладу, щоб зареєструвати шлях для POST запиту, потрібно використовувати ″Route::post()” [7].

Перший параметр методу відповідає за назву шляху. Другим параметром є масив, першим елементом якого є клас контролеру, який буде обробляти запит, а другим елементом якого є назва методу даного контролеру.

Для спрощення читабельності коду, даному шляху присвоєно своє власне ім’я, за допомогою методу “name()”. Це дозволяє звертатися до даного шляху за назвою в будь якій частині програми, наприклад в темплейтах [9].

Метод “create()” повертає функцію “view()”, що приймає шлях до відповідного темплейту. Наступним етапом функція створює екземпляр класу “View” за допомогою відповідної фабрики та повертає даний екземпляр. Саме тому, при оголошенні методу ″create” ми явно вказуємо, що вона повинна повертати об’єкт класу “View”, використовуючи наступний синтаксис “create(): View” [9].

Шлях до темплейту (в нашому випадку “auth.login”) означає, що темплейт повинен бути знайдено в системі за наступним шляхом “resources/views/auth/login.blade.php” [9].

Для створення темплейтів в фреймворці Laravel використовується шаблонізатор Blade. Його перевага полягає в тому, що ми уникаємо копіювання коду в розмітці. Це дозволяє писати чистий та зрозумілий код [9].

Одним з найпоширеніших підходів до розробки розмітки є застосування Blade макетів (layouts). Макет - це своєрідна “обгортка” всього контенту веб сторінки. Він слугує для того, щоб помістити в нього HTML теги найвищого рівня, такі як <head> та <body>, та використовувати на всіх сторінках. Оскільки саме в <head> виконується підключення стилів та скрипів, ми уникаємо повторного підключення даних елементів на усіх сторінках вебсайту. Таких макетів може бути скільки завгодно [9].

Ще однією важливою перевагою у використанні Blade є компоненти (components). Вони також дозволяють уникати дублювання коду та робити його більш чистим та розширюваним [9].

Для прикладу розглянемо контент компоненту “confirm-password”, який знаходиться за шляхом “resources/views/auth/confirm-password.blade.php”:

Як можна побачити, даний компонент використовує макет “guest”, який підключається через тег “<x-guest-layout>”. Саме тому, нам не потрібно знову підключати стилі та скрипти, адже вони вже відключені в макеті.

Висновки до розділу 2

Система збереження об’єктів культурної спадщини реалізує необхідний функціонал, необхідний для роботи з інформацією про культурну спадщину України, а саме: авторизацію адміністраторів за допомогою логіну та паролю, пошук інформації про об’єкт за охоронним номером. В даному розділі описано структуру БД, що була створена відповідно до вимог реляційної моделі. БД забезпечує збереження та доступ до інформації системи. Вона містить 5 таблиць, пов’язаних між собою. Серед таблиць БД можна виокремити наступні: «users», «password_reset_tokens», «items», «types» та «kinds».

Було розглянуто та описано алгоритми та основні методи роботи системи збереження об’єктів культурної спадщини.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОТОТИПУ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1. Розроблення інтерфейсу та основних функцій

Розглянемо інтерфейс панелі керування. Для доступу до даного ресурсу необхідно перейти за посиланням виду «[web-site domain]/admin». В нашому випадку доменне ім’я, на якому розміщується панель керування – «https://heritage.test/admin». Після переходу за посиланням перевіряється, чи користувач авторизований в системі. Якщо ні – перед ним з’являється форма авторизації. Ілюстрація наведена у додатку Б (рис. Б.1).

Коли користувач успішно авторизувався в системі, використовуючи email та пароль, він переходить на головну сторінку панелі керування. На даній сторінці є меню навігації адмін-панелі та таблиця зі списком об’єктів культурної спадщини. В даній таблиці наявна вся необхідна інформація, а також список дій, які можна зробити з об’єктами. Ілюстрація наведена у додатку Б (рис. Б.2).

Після переходу до пункту меню “Список Адміністраторів”, головний адміністратор може переглянути всіх адміністраторів, що зареєстровані в системі. Також він може додати нового адміністратора, натиснувши кнопку “Створити Нового Адміністратора” та видалити будь якого іншого адміністратора, окрім себе, натиснувши кнопку “Видалити”. Ілюстрація наведена у додатку Б (рис. Б.3).

Щоб додати інформацію про новий об’єкт, необхідно натиснути кнопку “Додати об’єкт культурної спадщини”. Після цього відкриється сторінка створення нового об’єкту . ілюстрація наведена у додатку Б (рис. Б.4).

Форма створення об’єкту має всі необхідні поля, для зручного редагування даних. Поля “Тип" та “Вид” представлені у вигляді елемента dropdown, в якому містяться типи та види об’єктів культурної спадщини відповідно. Разом з цим при створенні об’єкту, необхідно обов’язково обрати зображення. Дане зображення можна зручно передати на сервер використовуючи filepicker.

Для видалення об’єкту культурної спадщини з системи, необхідно натиснути кнопку “Видалити”, яка знаходиться в таблиці, в стовпчику “Дії”. Ілюстрація наведена у додатку Б (рис. Б.5).

Далі розглянемо основну частину сайту, яка не відноситься до користувачів-адміністраторів. Для того щоб потрапити на головну сторінку сайту, необхідно перейти за посиланням даного сайту. У нашому випадку це “https://heritage.test”.

На головній сторінці відображено список об’єктів культурної спадщини, які відсортовані в порядку спадання, тобто останні додані розміщено на першому місці. Кожен об’єкт містить необхідну інформацію, а саме: назву, фото, охоронний номер та місце розташування. Ілюстрація наведена у додатку Б (рис. Б.6).

Для того, щоб відразу знайти інформацію по бажаному об’єкту культурної спадщини, на сайті присутній пошук. Він дозволяє шукати об’єкти за охоронним номером та ключовими словами, які присутні в назві об’єкту чи в описі об’єкту.

Для того щоб здійснити пошук за охоронним номером, необхідно ввести його повністю в поле пошуку веб-сайту. ілюстрація наведена у додатку Б (рис. Б.7).

Для того щоб переглянути більш детальну інформацію про об’єкт культурної спадщини, необхідно натиснути на блок об’єкту. Після цього відкриється сторінка об’єкту, на якій присутня вся інформації про нього. Ілюстрація наведена у додатку Б (рис. Б.8).

Нижче на сторінці присутня таблиця з короткими відомостями про даний об’єкт. В таблиці присутні такі дані: тип об’єкту, вид об’єкту, датування, охоронний номер, місто та адреса розміщення (якщо наявна). Ілюстрація наведена у додатку Б (рис. Б.9).

Для полів “Тип” та “Вид”, присутній короткий опис, який допомагає зрозуміти що саме означає даний тип або вид об’єкту культурної спадщини. Для

того щоб переглянути опис, необхідно натиснути на іконку “і” біля назви типу або виду. Після цього з’явиться спливаюче вікно з інформацією. Ілюстрація наведена у додатку Б (рис. Б.10).

Далі наведено таблицю відповідності посилань до сторінок вебсайту. Структура таблиці наведена у додатку Б (табл. Б.1).

3.2.Тестування програмної системи

Після реалізації програмної системи, необхідно було провести її тестування та пошук слабких місць. Під час тестування, було виділено такі його види:

1. Функціональне тестування;
2. Стрес тестування;
3. Тестування безпеки.

Якщо тестування та усунення слабких місць проводити належним чином, можна досягнути найвищої якості та продуктивності програмного продукту.

В момент роботи над проектом було застосовано інструменти для відлагодження програмного коду. Саме тому велика кількість проблем була виправлена на етапі розробки системи. Перед початком тестування, було проаналізовано існуючі функції та створено список браузерів, в яких потрібно провести перевірку, а саме: Firefox, Google Chrome, Opera та Safari. Разом з цим було перевірено роботу вебсайту в мобільних браузерах.

Функціональне тестування

Цілі: Тестування програмної системи на функціональні помилки, та відповідності між очікуваннями та результатом роботи програми.

Було здійснено функціональне тестування наступних складових системи:

1. Панель адміністратора:

– реєстрація та авторизація адміністраторів;

– управління об’єктами;

– редагування поточного профілю;

– управління користувачами адміністраторами.

2. Основна частина вебсайту:

– правильність побудови посилань на об’єкти;

– правильність відображення інформації;

– коректність стилів та скрипів;

– коректність роботи пошуку;

– коректність роботи пагінації;

– кешування сторінок.

Стрес-тестування

Мета: Проведення перевірки програмного продукту на стійкість до великого навантаження на сервер при одночасному використанні системи багатьма користувачами.

Основним складовим для даного виду тестування було обрано СУБД MySQL, оскільки даний компонент виконують роль збереження та передачі інформації.

Тестування безпеки

Мета: Перевірити вразливість системи до SQL ін’єкцій та XSS (міжсайтового скриптингу) та виправити знайдені незахищені місця.

Даний вид тестування є дуже важливим та актуальним в загальному тестуванні програмної системи. Адже втрата користувацьких даних є дуже великою проблемою в сучасному світі. Також, при «правильному» використанні вразливостей системи, можна мати значний вплив на її роботу.

Як результат, було проведено комплексне та повноцінне тестування системи збереження об’єктів культурної спадщини. Серед проведених тестувань можна виділити наступні: тестування безпеки, стресове тестування, тестування функціональності та продуктивності. Знайдені помилки було виправлено. Всі вразливі місця в безпеці системи було ліквідовано.

Висновки до розділу 3

В даному розділі було здійснено детальний опис роботи готової програмної системи.

Створений програмний продукт включає в себе інтерфейси адміністратора та користувача. Для адміністраторів система надає перелік можливостей, такі як: авторизція та реєстрація, управління профілем, управління інформацією про об’єкти культурної спадщини. З точки зору користувача, система обладнана необхідними функціональними можливостями, які допомагають користувачу отримувати інформацію про той чи інший об’єкт культурної спадщини України.

Завершальним етапом реалізації системи стало її тестування. Було здійснено наступні типи тестування: функціональне тестування, стресове тестування та тестування безпеки. Всі помилки та вразливості було виправлено. Також в результаті тестування було перевірено, що програмна система працює правильно та надійно.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено систему збереження інформації про об'єкти культурної спадщини. Для цього було виконано такі завдання:

1. Сформовано основне завдання та мету впровадження програмної системи для збереження об'єктів культурної спадщини. Разом з цим було здійснено аналіз інформаційних потреб предметної області та визначено особливості збереження інформації даної системи.

2. Для побудови системи було використано такі інструменти: мову розмітки HTML5 та шаблонізатор Blade; CSS фреймворк Tailwind CSS; фреймворк для браузерної мови програмування AlpineJS; мову програмування PHP 8.2; фреймворк Laravel 10; СУБД MySQL; середовище розробки PhpStorm 2023.1.

3. Було здійснено аналіз функціональних та нефункціональних вимог системи збереження інформації про об'єкти культурної спадщини. Також було створено діаграму варіантів використання, діаграму класів та діаграму послідовностей даної системи.

4. Реалізовано базу даних, що забезпечує збереження та зручний доступ до інформації програмної системи. БД складається з 5 таблиць: «users», «password_reset_tokens», «items», «types» та «kinds».

5. Розроблено та реалізовано інтерфейс системи. Дана система наявна у вигляді веб-сайту. Для управління всією інформацією в системі використовується панель адміністратора. Для звичайних користувачів наявна можливість зручного пошуку інформації про об'єкти культурної спадщини .

6. Здійснено тестування програмного продукту. Тестування системи включало в себе наступні типи: функціональне тестування, стресове тестування та тестування безпеки. Всі помилки та вразливості було виправлено.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Schwartz B. High Performance MySQL: Optimization, Backups, and Replication / B. Schwartz, P. Zaitsev, V. Tkachenko., 2012. – 826 с.
2. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship / Robert Martin., 2008. – 464 с.
3. Martin R. Clean Architecture: A Craftsman′s Guide to Software Structure and Design / Robert Martin., 2017. – 352 с.
4. Fowler M. UML distilled / Martin Fowler., 2003. – 208 с.
5. PHP Data Objects [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/manual/en/book.pdo.php.
6. Laravel 10 Installation [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/10.x/installation.
7. Laravel 10 Routing [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/10.x/routing.
8. Get started with Tailwind CSS [Електронний ресурс] – Режим доступу до ресурсу: https://tailwindcss.com/docs/installation.
9. Blade Templates [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/10.x/blade.
10. MVC Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/php-mvc-architecture.

ДОДАТКИ

Додаток А

Таблиця А.1

Опис полів таблиці «users»

Назва	Тип даних	PK	FK	Опис поля
id	bigint(20)	+	-	Ідентифікатор користувача
name	varchar(255)	-	-	Ім'я користувача
email	varchar(255)	-	+	Email користувача
password	varchar(255)	-	-	Пароль користувача
is_main_admin	tinyint(1)	-	-	Чи є адміністратор головним
created_at	timestamp	-	-	Дата створення
updated_at	timestamp	-	-	Дата оновлення

Таблиця А.2

Опис полів таблиці «password_reset_tokens»

Назва	Тип даних	PK	FK	Опис поля
email	varchar(255)	+	-	Email користувача
token	varchar(255)	-	-	Точен
created_at	timestamp	-	-	Дата створення

Таблиця А.3

Опис полів таблиці «items»

Назва	Тип даних	PK	FK	Опис поля
id	bigint(20)	+	-	Ідентифікатор об’єкту
name	varchar(255)	-	-	Назва об’єкту
url_key	varchar(255)	-	-	Ключ посилання
type_id	bigint(20)	-	+	Ідентифікатор типу
kind_id	bigint(20)	-	+	Ідентифікатор типу
dating	varchar(255)	-	-	Датування
security_number	varchar(255)	-	-	Охоронний номер
address	varchar(255)	-	-	Адреса
city	varchar(255)	-	-	Місто
image	varchar(255)	-	-	Шлях до зображення
info	text	-	-	Інформація
created_at	timestamp	-	-	Дата створення
updated_at	timestamp	-	-	Дата оновлення

Таблиця А.4

Опис полів таблиці «types»

Назва	Тип даних	PK	FK	Опис поля
id	bigint(20)	+	-	Ідентифікатор типу
name	varchar(255)	-	-	Назва типу
description	text	-	-	Опис типу
created_at	timestamp	-	-	Дата створення
updated_at	timestamp	-	-	Дата оновлення

Таблиця А.5

Опис полів таблиці «kinds»

Назва	Тип даних	PK	FK	Опис поля
id	bigint(20)	+	-	Ідентифікатор виду
name	varchar(255)	-	-	Назва виду
description	text	-	-	Опис виду
created_at	timestamp	-	-	Дата створення
updated_at	timestamp	-	-	Дата оновлення

Додаток Б

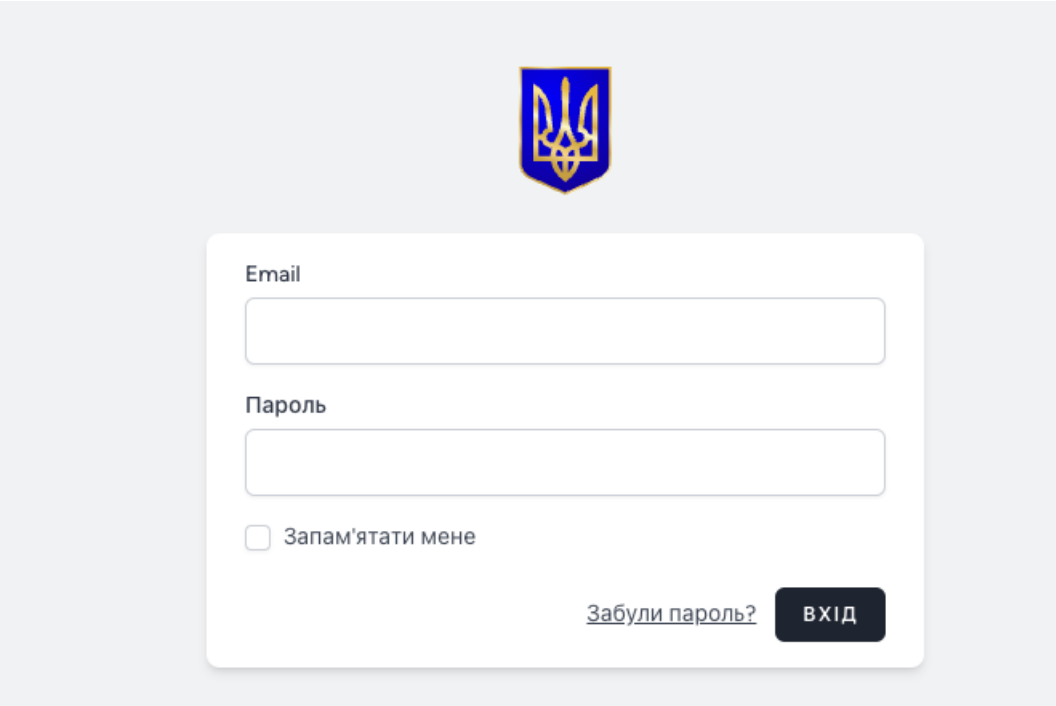


Рис. Б.1. Форма авторизації

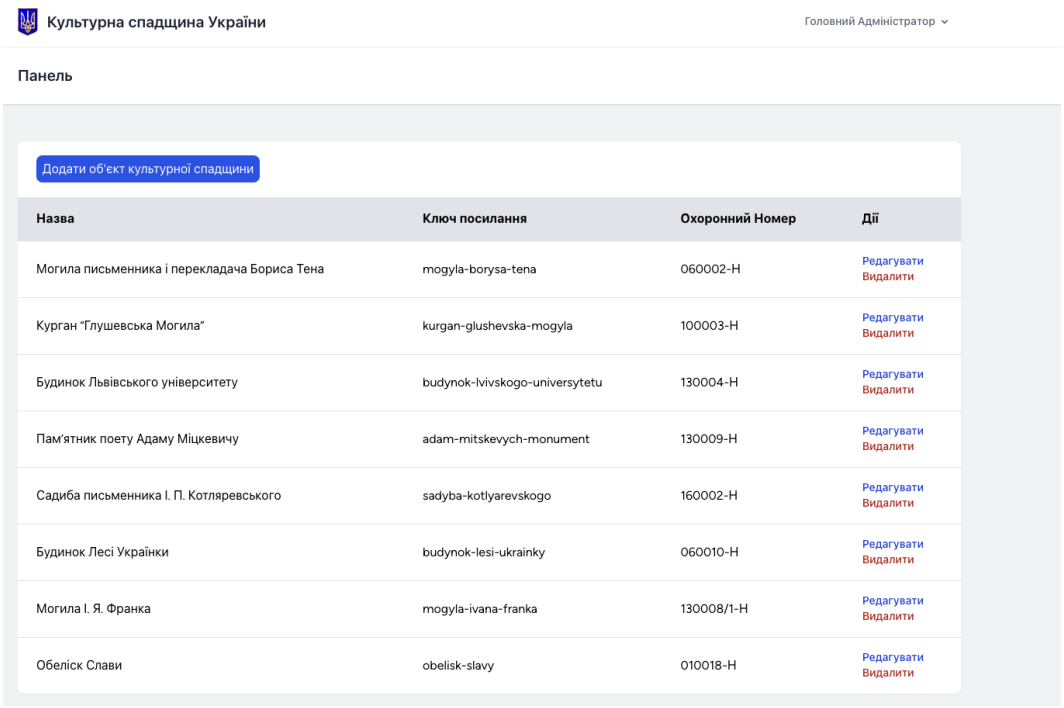
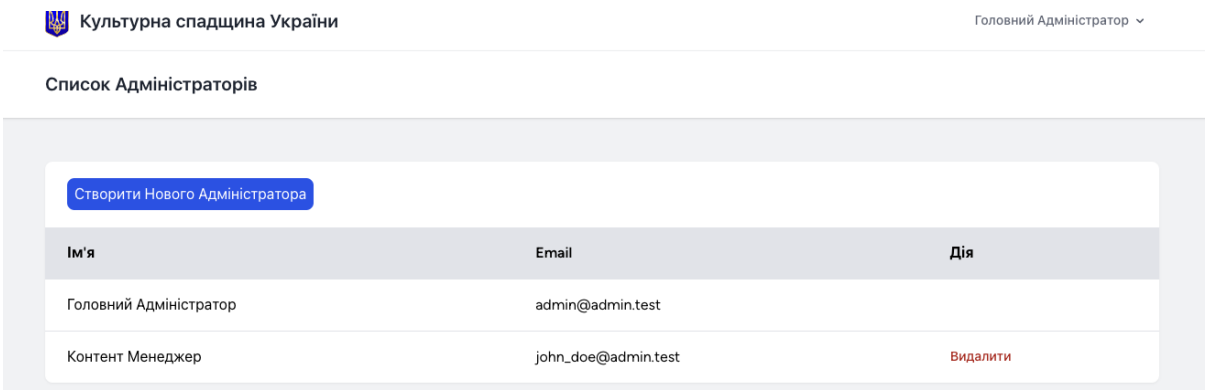


Рис. Б.2. Головна сторінка адмін-панеліРис.



Б.3. Управління адміністраторами

Рис.Б.4. Сторінка створення нового об’єкту

Будинок Лесі Українки	budynok-lesi-ukrainky	060010-Н	Редагувати Видалити
Могила І. Я. Франка	mogyla-ivana-franka	130008/1-Н	Редагувати Видалити

Рис. Б.5. Об’єкти культурної спадщини

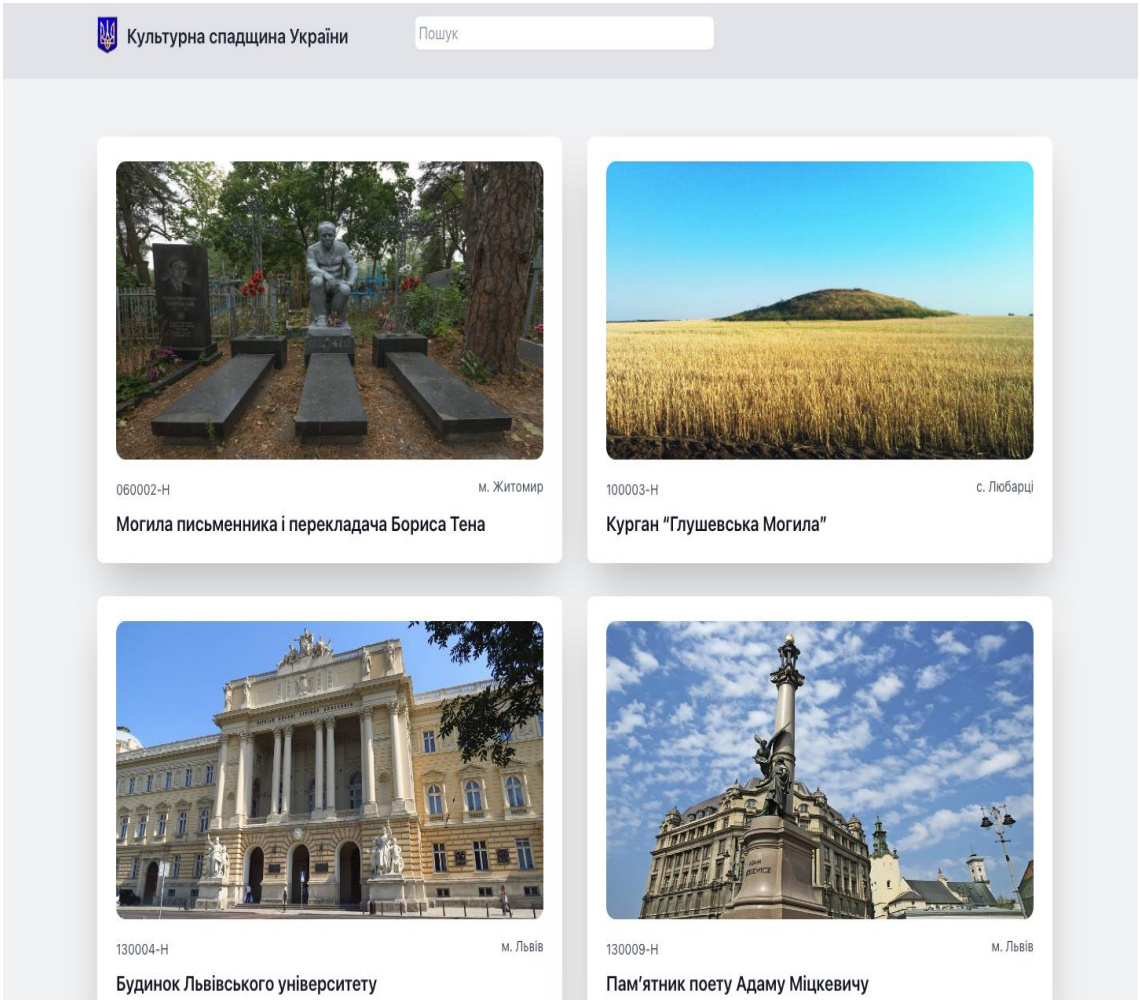


Рис. Б.6. Головна сторінка сайту

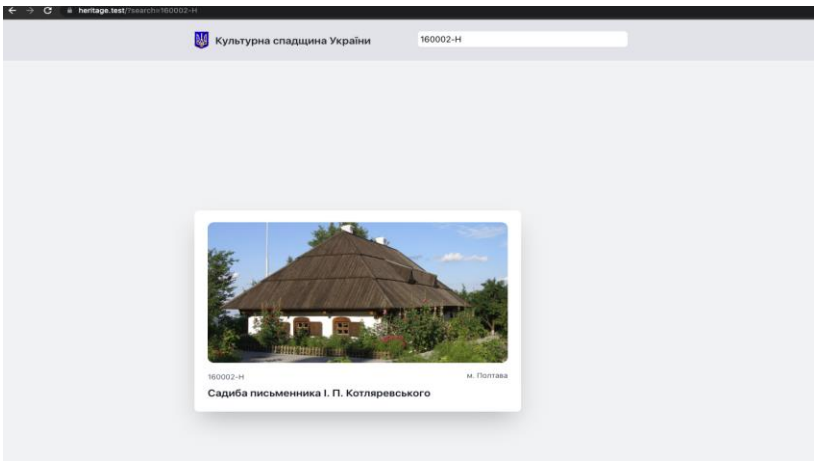


Рис. Б.7. Пошук за охоронним номером

Рис. Б.8. Головна сторінка об’єкту

Тип:	Визначні Місця
Вид:	Історичні
Датування:	1916 рік
Охоронний Номер:	130008/1-Н
Місто:	м. Львів
Адреса:	Личаківський цвинтар, поле 4

Рис. Б.9. Короткі відомості про об’єкт

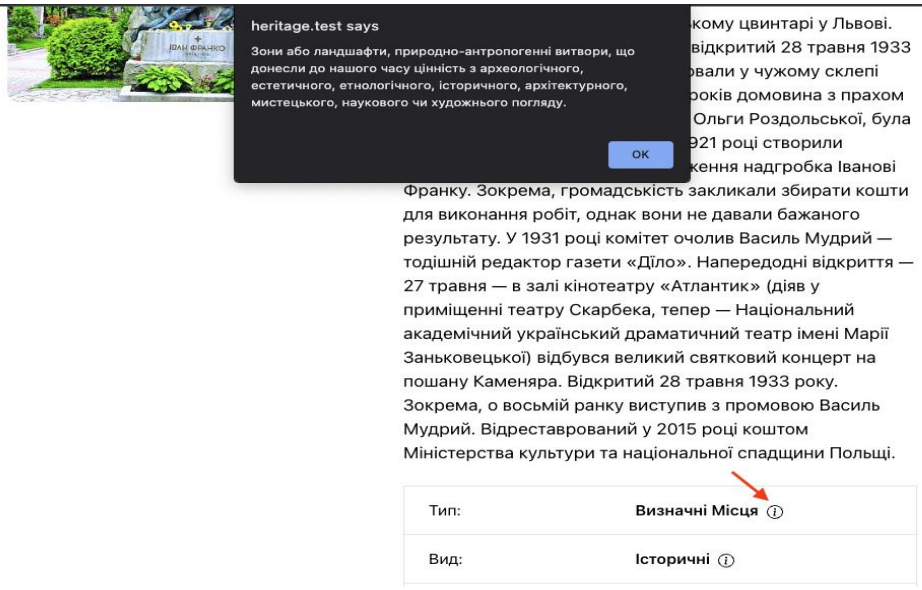


Рис. Б.10. Короткий опис типу “Визначні місця”

Таблиця Б.1.

Відповідність посилань до сторінок вебсайту

Сторінка сайту	**Посилання**
Головна сторінка	https://heritage.test/
Сторінка об’єкту	https://heritage.test/items/[url_key]/
Панель адміністратора	https://heritage.test/admin/
Сторінка авторизації	https://heritage.test/login/
Сторінка ”Забули пароль”	https://heritage.test/forgot-password/
Створити новий об’єкт	https://heritage.test/admin/items/
Редагувати об’єкт	https://heritage.test/admin/items[id]/
Редагувати профіль	https://heritage.test/admin/profile/
Список адміністраторів	https://heritage.test/admin/profiles/
Створити нового адміністратора	https://heritage.test/register/
Вихід	https://heritage.test/logout/

Додаток В

Лістинг коду

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];
}

<?php
```

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Item extends Model
{
    use HasFactory;

    public function scopeFilter($query, array $filters)
    {
        $query->when($filters['search'] ?? false, fn($query, $search) =>
            $query->where(fn($query) =>
                $query->where('name', 'like', '%' . $search . '%')
                    ->orWhere('info', 'like', '%' . $search . '%')
                    ->orWhere('security_number', '=', $search)
            )
        );
    }


    public function type(): BelongsTo
    {
        return $this->belongsTo(Type::class);
    }

    public function kind(): BelongsTo
    {
        return $this->belongsTo(Kind::class);
    }
}

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;
```

```
class Kind extends Model
{
    use HasFactory;

    public function item(): HasMany
    {
        return $this->hasMany(Item::class);
    }
}

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Type extends Model
{
    use HasFactory;

    public function item(): HasMany
    {
        return $this->hasMany(Item::class);
    }
}

<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Symfony\Component\HttpFoundation\Response;

class MainAdministrator
{
    public function handle(Request $request, Closure $next): Response
```

```
        {
            if (!Auth::user()->is_main_admin) {
                abort(Response::HTTP_FORBIDDEN);
            }

            return $next($request);
        }
    }

    <?php

    namespace App\Http\Controllers;

    use App\Models\Item;
    use Illuminate\Http\RedirectResponse;
    use Illuminate\Support\Facades\Redirect;
    use Illuminate\View\View;
    use Illuminate\Validation\Rule;

    class ItemController extends Controller
    {
        public function index(): View
        {
            return view('home', [
                'items'  =>  Item::latest()->filter(request([  'search'  ]))->paginate(6)-
>withQueryString()
            ]);
        }

        public function show(Item $item): View
        {
            return view('item.show', [ 'item' => $item ]);
        }

        public function create(): View
        {
            return view('item.create');
        }

        public function store(): RedirectResponse
        {
```

```
            Item::create(array_merge($this->validateItem(), [
                'image' => request()->file('image')->store('images')
            ]));

            return Redirect::to('/admin');
        }

        public function edit(Item $item): View
        {
            return view('item.edit', [ 'item' => $item ]);
        }

        public function update(Item $item): RedirectResponse
        {
            $attributes = $this->validateItem($item);

            if ($attributes['image'] ?? false) {
                $attributes['image'] = request()->file('image')->store('images');
            }

            $item->update($attributes);

            return Redirect::to('/admin');
        }

        public function destroy(Item $item): RedirectResponse
        {
            $item->delete();

            return Redirect::to('/admin');
        }

        private function validateItem(?Item $item = null): array
        {
            $item ??= new Item();

            return request()->validate([
                'name' => 'required',
                'url_key'    =>    ['required',    Rule::unique('items',    'url_key')-
>ignore($item)],
                'type_id' => ['required', Rule::exists('types', 'id')],
```

```
            'kind_id' => ['required', Rule::exists('kinds', 'id')],
            'dating' => 'required',
            'security_number' => 'required',
            'city' => 'required',
            'image' => ['required', 'image'],
            'info' => 'required'
        ]);
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProfileUpdateRequest;
use App\Models\Item;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
use Illuminate\View\View;

class ProfileController extends Controller
{
    public function index(): View
    {
        return view('dashboard', [
            'items' => Item::all()
        ]);
    }

    public function edit(Request $request): View
    {
        return view('profile.edit', [
            'user' => $request->user(),
        ]);
    }

    public function update(ProfileUpdateRequest $request): RedirectResponse
    {
        $request->user()->fill($request->validated());
```

```
        $request->user()->save();

        return Redirect::route('profile.edit')->with('status', 'profile-updated');
    }

    public function destroy(Request $request): RedirectResponse
    {
        $request->validateWithBag('userDeletion', [
            'password' => ['required', 'current_password'],
        ]);

        $user = $request->user();

        Auth::logout();

        $user->delete();

        $request->session()->invalidate();
        $request->session()->regenerateToken();

        return Redirect::to('/');
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\Http\RedirectResponse;
use Illuminate\View\View;

class ProfilesController extends Controller
{
    public function index(): View
    {
        return view('profile.show-all', [
            'users' => User::all()
        ]);
    }
```

```
        public function destroy(User $user): RedirectResponse
        {
            if (!$user->is_main_admin) {
                $user->delete();
            }

            return back();
        }

    }

    <?php

    use App\Http\Controllers\ItemController;
    use App\Http\Controllers\ProfileController;
    use App\Http\Controllers\ProfilesController;
    use App\Http\Controllers\Auth\AuthenticatedSessionController;
    use App\Http\Controllers\Auth\ConfirmablePasswordController;
    use App\Http\Controllers\Auth\NewPasswordController;
    use App\Http\Controllers\Auth\PasswordController;
    use App\Http\Controllers\Auth\PasswordResetLinkController;
    use App\Http\Controllers\Auth\RegisteredUserController;
    use Illuminate\Support\Facades\Route;

    Route::middleware('guest')->group(function () {

        Route::get('login',     [AuthenticatedSessionController::class,     'create'])-
>name('login');

        Route::post('login', [AuthenticatedSessionController::class, 'store']);

        Route::get('forgot-password',  [PasswordResetLinkController::class,  'create'])-
>name('password.request');

        Route::post('forgot-password',  [PasswordResetLinkController::class,  'store'])-
>name('password.email');

        Route::get('reset-password/{token}',  [NewPasswordController::class,  'create'])-
>name('password.reset');
```

```
        Route::post('reset-password',      [NewPasswordController::class,      'store'])-
>name('password.store');
    });

    Route::middleware('auth')->group(function () {

        Route::get('admin', [ProfileController::class, 'index'])->name('dashboard');

        Route::get('admin/profile',         [ProfileController::class,         'edit'])-
>name('profile.edit');

        Route::patch('admin/profile',       [ProfileController::class,       'update'])-
>name('profile.update');

        Route::delete('admin/profile',      [ProfileController::class,      'destroy'])-
>name('profile.destroy');

        Route::get('admin/profiles',        [ProfilesController::class,        'index'])-
>name('profiles.show');

        Route::delete('admin/profiles/{user:id}',            [ProfilesController::class,
'destroy'])->name('profiles.delete');

        Route::get('confirm-password',  [ConfirmablePasswordController::class,  'show'])-
>name('password.confirm');

        Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);

        Route::put('password',          [PasswordController::class,          'update'])-
>name('password.update');

        Route::post('logout',    [AuthenticatedSessionController::class,    'destroy'])-
>name('logout');

        Route::delete('admin/items/{item:id}',    [ItemController::class,    'destroy'])-
>name('item.destroy');

        Route::patch('admin/items/{item:id}',     [ItemController::class,     'update'])-
>name('item.update');
```

```
        Route::get('admin/items/{item:id}',       [ItemController::class,       'edit'])-
>name('item.edit');

        Route::get('admin/items',           [ItemController::class,           'create'])-
>name('item.create');

        Route::post('admin/items', [ItemController::class, 'store'])->name('item.store');
    });

    Route::middleware(['auth', 'main.admin'])->group(function() {

        Route::get('register',       [RegisteredUserController::class,       'create'])-
>name('register');

        Route::post('register', [RegisteredUserController::class, 'store']);
    });

    <?php

    use App\Http\Controllers\ItemController;
    use Illuminate\Support\Facades\Route;

    Route::get('/', [ItemController::class, 'index'])->name('home');

    Route::get('items/{item:url_key}',         [ItemController::class,         'show'])-
>name('item.show');


    require __DIR__.'/admin.php';


    <x-main-layout>
        <div class="relative sm:flex sm:justify-center sm:items-center min-h-screen bg-
dots-darker bg-center bg-gray-100 dark:bg-dots-lighter dark:bg-gray-900 selection:bg-
red-500 selection:text-white">
            <div class="max-w-7xl mx-auto p-6 lg:p-8">
                <div class="mt-6">
                    <div class="grid grid-cols-1 md:grid-cols-2 gap-6 lg:gap-8">
                        @if (count($items) === 0)
                            @if (!empty(request('search')))
```

```
                                <p class="text-center text-lg">{{ __('Sorry, using your request') }} <strong>'{{ request('search') }}'</strong> {{ __('can not find any results.') }}</p>
                            @else
                                <p class="text-center text-lg">{{ __('No items yet. Come back later!') }}</p>
                            @endif
                        @endif

                        @foreach($items as $item)
                            <a href="/items/{{ $item->url_key }}" class="scale-100 p-6 bg-white dark:bg-gray-800/50 dark:bg-gradient-to-bl from-gray-700/50 via-transparent dark:ring-1 dark:ring-inset dark:ring-white/5 rounded-lg shadow-2xl shadow-gray-500/20 dark:shadow-none flex motion-safe:hover:scale-[1.01] transition-all duration-250 focus:outline focus:outline-2 focus:outline-red-500">
                                <div>
                                    <div class="mb-4">
                                        <img src="{{ asset('storage/' . $item->image) }}" alt="{{ $item->name }}" class="rounded-xl h-72" width="550px">
                                    </div>

                                    <span class="text-sm text-gray-600">{{ $item->security_number }}</span>
                                    <span class="text-sm text-gray-600 float-right">{{ $item->city }}</span>

                                    <h2 class="mt-2 text-xl font-semibold text-gray-900 dark:text-white">{{ $item->name }}</h2>
                                </div>
                            </a>
                        @endforeach
                        {{ $items->links() }}
                    </div>
                </div>
            </div>
        </div>
    </x-main-layout>

    <x-admin-layout>
        <x-slot name="header">
            <h2 class="font-semibold text-xl text-gray-800 leading-tight">
```

```
                {{ __('Dashboard') }}
            </h2>
        </x-slot>

        <div class="py-12">
            <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
                <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
                    <div class="p-6 text-gray-900">
                        <a class="bg-blue-600 hover:bg-blue-800 text-white p-2 rounded-
lg" href="/admin/items">{{ __("Add New Heritage Item") }}</a>
                    </div>

                    <table class="min-w-full divide-y divide-gray-200">
                        <thead class="bg-gray-200 min-w-full divide-y divide-gray-200">
                        <tr>
                            <td class="px-6 py-4 font-bold whitespace-nowrap">{{ __('Item
Name') }}</td>
                            <td class="px-6 py-4 font-bold whitespace-nowrap">{{ __('Url
Key') }}</td>
                            <td   class="px-6   py-4   font-bold   whitespace-nowrap">{{
__('Security Number') }}</td>
                            <td   class="px-6   py-4   font-bold   whitespace-nowrap">{{
__('Actions') }}</td>
                        </tr>
                        </thead>
                        <tbody class="min-w-full divide-y divide-gray-200">
                        @foreach ($items as $item)
                            <tr>
                                <td  class="px-6  py-4  whitespace-nowrap">{{  $item->name
}}</td>
                                <td class="px-6 py-4 whitespace-nowrap">{{ $item->url_key
}}</td>
                                <td   class="px-6   py-4   whitespace-nowrap">{{   $item-
>security_number  }}</td>
                                <td  class="px-6  py-4  whitespace-nowrap  text-sm  font-
medium">
                                    <a   class="text-sm   text-blue-700   hover:underline"
href="/admin/items/{{ $item->id }}">{{ __('Edit') }}</a>
                                    <form    action="/admin/items/{{    $item->id    }}"
method="POST">
                                        @csrf
```

```
                                        @method('DELETE')

                                        <button        class="text-sm        text-red-700
hover:underline">{{ __('Delete') }}</button>
                                    </form>
                                </td>
                            </tr>
                        @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </x-admin-layout>
```