МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук,
доцент _____ Є. О. Давиденко
підпис

«___» _____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК ВІДОБРАЖЕННЯ РОЗВИТКУ ЕКОЛОГІЧНОГО ПРОЄКТУ ПРОГРАМИ ЖАНА МОНЕ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.21917111

Студентка _____ І. В. Полтавець
підпис

«__» _____ 2023 р.

Керівник ст. викладач _____ М. В. Фаленкова
підпис

«__» _____ 2023 р.

Консультант канд. техн. наук, доцент _____ А. О. Алексєєва
підпис

«__» _____ 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студентці групи 409 факультету комп'ютерних наук

_____Полтавець Ілоні Вадимівні_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Вебзастосунок відображення розвитку екологічного проєкту програми Жана Моне_____

_____

Затверджена наказом по ЧНУ від «17» _березня_ 2023 р. № 60

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є вебзастосунок з відображенням розвитку екологічного проєкту програми Жана Моне_____

_____

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до програмного забезпечення;
- визначення архітектури для проектування програмного забезпечення;
- моделювання та проектування програмного забезпечення;
- розробка програмного забезпечення;
- здійснення тестування роботи програмного забезпечення;

– проведення аналізу результатів розробки;

5. Перелік графічних матеріалів

__Презентація_____

6. Завдання до спеціальної частини

__Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій__

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи _ст._викл. кафедри ІПЗ Фаленкова Марина Володимирівна_____

(посада, прізвище, ім’я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____Полтавець Ілона Вадимівна_____

(прізвище, ім’я, по батькові)

_____

(підпис)

Дата видачі завдання «____» _____ 2023р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Вебзастосунок відображення розвитку екологічного проєкту програми Жана Моне_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	10.11.2022	11.11.2022	виконано
2.	Огляд літератури за темою роботи	12.11.2022	13.11.2022	виконано
3.	Складання календарного плану КРБ	14.11.2022	15.11.2022	виконано
4.	Аналіз предметної області	16.11.2022	16.11.2022	виконано
5.	Розробка проєктних рішень	17.11.2022	18.11.2022	виконано
6.	Моделювання та конструювання ПЗ	19.11.2022	23.11.2022	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	24.11.2022	02.02.2023	виконано
8.	Оформлення КРБ та презентації	01.05.2023	28.05.2023	виконано
9.	Розробка спеціальної частини з охорони праці	04.06.2023	05.06.2023	виконано
10.	Відгук керівника КРБ	06.06.2023	06.06.2023	виконано
11.	Попередній захист	07.06.2023	07.06.2023	виконано
12.	Завершення оформлення КРБ та презентації	08.06.2023	10.06.2023	виконано
13.	Рецензування	14.06.2023	14.06.2023	виконано
14.	Захист кваліфікаційної роботи	26.06.2023	26.06.2023	виконано

Розробив студент _____Полтавець Ілона Вадимівна_____ _____

(прізвище, ім’я, по батькові) (підпис)

«__» _____ 2023 р.

Керівник роботи _ст._викл. каф. ІПЗ Фаленкова Марина Володимирівна_ _____

(посада, прізвище, ім’я, по батькові) (підпис)

«__» _____ 2023 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок відображення розвитку екологічного проєкту програми Жана Моне»

Студентка 409 групи: Полтавець Ілона Вадимівна

Керівник: ст. викладач Фаленкова М. В.

Дана робота присвячена розробці вебзастосунку для підвищення обізнаності більш широкої аудиторії про природні ресурси, якість довкілля та зміну клімату завдяки інформації, що буде знаходитися на вебсторінці.

Об’єкт роботи – процес розробки вебзастосунку для відображення розвитку екологічного проєкту програми Жана Моне.

Предмет роботи – технології створення вебзастосунку для підвищення обізнаності більш широко аудиторії про природні ресурси, якість довкілля та зміну клімату.

Метою роботи є підвищення обізнаності більш широкої аудиторії про природні ресурси, якість довкілля та зміну клімату, шляхом створення вебзастосунку для відображення розвитку екологічного проєкту програми Жана Моне, що спрямовано на відображення актуальних новин та подій у сфері екології.

Кваліфікаційна робота складається з вступу, 4 розділів, висновків та переліку джерел посилань.

У вступі визначається актуальність теми, мета, предмет та об’єкт дослідження.

У першому розділі проведено аналіз існуючих вебзастосунків-аналогів, визначення функціоналу, переваг, недоліків, технології за допомогою яких було створено вебзастосунок. Формування та опис специфікації вимог вебзастосунку, що розробляється.

У другому розділі приведено моделювання та розробку структури застосунку, що розробляється.

У третьому розділі описується огляд мов, технологій та бібліотек, що використовуються для розробки, та процес проєктування програмного забезпечення.

У четвертому розділі описано процес розробки програмного забезпечення та його тестування.

У висновках проводиться аналіз виконаних робіт та отриманих результатів.

Кваліфікаційна робота бакалавра викладена на 62 сторінки, містить 4 розділи, 53 ілюстрації, 4 таблиці, 14 джерел в переліку посилань.

Ключові слова: *створення вебзастосунку, адаптивний інтерфейс, розробка на Laravel, екологічний проєкт, зміна клімату, природні ресурси.*

ABSTRACT

of the Bachelor's Thesis

«Web application displaying the development of the environmental project of the Jean Monet program»

Student: Ilona Poltavets

Supervisor: Senior Lecturer Marina Falenkova

This work is devoted to the development of a web application to raise the awareness of a wider audience about natural resources, environmental quality and climate change thanks to the information that will be on the web page.

The object of the work is the process of developing a web application to reflect the development of the environmental project of the Jean Monet program.

The subject of the work is the technology of creating a web application to raise the awareness of a wider audience about natural resources, the quality of the environment and climate change.

The purpose of the work is to increase the awareness of a wider audience about natural resources, the quality of the environment and climate change, by creating a web application to reflect the development of the environmental project of the Jean Monet program, which is aimed at reflecting current news and events in the field of ecology.

The qualification work consists of an introduction, 4 sections, conclusions and a list of reference sources.

The introduction defines the relevance of the topic, the purpose, subject and object of the research.

In the first section, an analysis of existing analog web applications, definition of functionality, advantages, disadvantages, technologies with the help of which the web application was created was carried out. Formation and description of the requirements specification of the web application under development.

The second chapter presents the modeling and development of the structure of the application under development.

The third chapter describes an overview of the languages, technologies and libraries used for development and the software design process.

The fourth chapter describes the process of software development and testing.

The conclusions analyze the work performed and the results obtained.

The bachelor's qualification work is laid out on 62 pages, contains 4 chapters, 53 illustrations, 4 tables, 14 sources in the list of references.

Keywords: *web application creation, adaptive interface, Laravel development, environmental project, climate change, natural resources.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	База даних
ПЗ	–	Програмне забезпечення
ОС	–	Операційна система
CRUD	–	Create, Read, Update, Delete
CSS	–	Cascading Style Sheet
JS	–	JavaScript
PHP	–	Hypertext Preprocessor
MVC	–	Model-View-Controller
UML	–	Unified Modeling Language
ORM	–	Object-Relational Mapping
WYSIWYG	–	What You See Is What You Get

ВСТУП

Актуальність теми. На сьогоднішній день для отримання потрібної інформації існує безліч джерел: книги, програмні застосунки на смартфон, групи та канали у месенджерах, ЗМІ та вебсайти. Кожне джерело можна обрати за своїм власним вподобанням. Вебсайти відрізняються від всіх інших тим, що для отримання актуальної інформації потрібні пристрій с браузером та доступ до інтернету. Вебсайти забезпечують:

- отримування інформації цілодобово;
- вільний вибір інформації;
- великий спектр тем, для обрання;
- легкодоступність інформації;
- заміну паперових джерел, таким чином зберігають екологію.

Таким чином попит та актуальність на вебсайти зростає, незалежно від його тематики.

Об’єкт роботи – процес розробки вебзастосунку для відображення розвитку екологічного проєкту програми Жана Моне.

Предмет роботи – технології створення вебзастосунку для підвищення обізнаності більш широко аудиторії про природні ресурси, якість довкілля та зміну клімату.

Метою кваліфікаційної роботи є підвищення обізнаності більш широкої аудиторії про природні ресурси, якість довкілля та зміну клімату, шляхом створення вебзастосунку для відображення розвитку екологічного проєкту програми Жана Моне, що спрямовано на відображення актуальних новин та подій у сфері екології.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- аналіз предметної сфери;
- обговорення деталей проєкту;

- створення UML-діаграм;
- моделювання mockup;
- розробка backend-частини;
- розробка дизайну за mockup;
- розробка адмінпанелі;
- тестування вебзастосунку;
- завантаження готового продукту на сервер.

Кваліфікаційна робота бакалавра викладена на 62 сторінок, містить 4 розділи, 53 ілюстрацій, 4 таблиць, 14 джерел в переліку посилань.

Ключові слова: *створення вебзастосунку, адаптивний інтерфейс, розробка на Laravel, екологічний проєкт, зміна клімату, природні ресурси.*

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд застосунків-аналогів

Одним з важливих етапів аналізу вимог до програмного забезпечення є аналіз вимог. Оскільки це дозволяє визначити, чи існують на ринку такі програмні засоби, які можуть задовольнити потреби користувачів, або які можуть бути використані в якості основи для створення нового ПЗ. Також це допомагає визначитись у критеріях функціональності, інтерфейсу користувача, швидкодії, масштабованості та інших.

Було розглянуто аналоги до розроблюваного вебзастосунку, для виявлення основних функцій, недоліків, що потрібно буде вирішити для покращення розроблюваного застосунку. Були розглянуті наступні аналоги вебзастосунок дослідження водних ресурсів (табл. 1.1) [1], Європейської агенції довкілля (табл. 1.2) [2] та Агенції з охорони довкілля США (табл. 1.3) [3].

Water security

Застосунок присвячений проблемам водної безпеки. На сайті можна знайти інформацію про проєкт та його цілі, також на сайті представлені матеріали та результати досліджень, що проводяться в рамках проєкту, а також різноманітні публікації, звіти та новини про водну безпеку. Сайт має зручний та зрозумілий інтерфейс (рис. 2.1), з інтуїтивно зрозумілою навігацією та простим доступом до матеріалів.

Таблиця 1.1 – Опис вебзастосунку Water security

Назва	**Water security**
Архітектура	Web application
Виробник	ЧНУ ім. Петра Могили
Мова реалізації	PHP з використанням CMS WordPress

Кінець таблиці 1.1

Функції	– Додавання новин; – додавання фото у галерею; – додавання ресурсів; – контактна інформація; – актуальна інформація про проєкт.
Недоліки	– Відсутній попередній перегляд фотографій; – при завантажені додаткових фото в галереї виникає помилка.
Переваги	– Актуальність інформації; – легкий доступ до інформації; – можливість завантаження матеріалів; – відкритість та прозорість; – наявність контактів; – адаптивність; – використання захисту даних; – доступність з різних частин світу; – наявність контактів та форми зворотного зв'язку.
Посилання	https://watersecurityproject.chmnu.edu.ua

European Environment Agency

Вебзастосунок є офіційним сайтом Європейського агентства з навколишнього середовища (EEA) та надає інформацію про стан навколишнього середовища в Європі. На сайті можна знайти новини та пресрелізи про екологічні проблеми, звіти та публікації про стан навколишнього середовища, а також інтерактивні карти, інструменти та бази даних, які допомагають візуалізувати та аналізувати дані про стан навколишнього середовища в Європі. Сайт англійською мовою, що робить його доступним для англомовної аудиторії та полегшує обмін інформацією та співробітництво між країнами Європи в галузі навколишнього середовища.

Таблиця 2.2 – Опис вебзастосунку European Environment Agency

Назва	**European Environment Agency**
Архітектура	Web application
Виробник	European Environment Agency
Мова реалізації	Python
Функції	– Додавання новин; – класифікація інформації; – контактна інформація; – актуальна інформація; – пошук за ключовими словами; – фільтрування новин.
Переваги	– Доступність; – актуальність; – надійність; – доступність для користувача; – функціональність; – співпраця з країнами Європи; – зручність використання.
Недоліки	– Складність матеріалу; – недоступність деяких матеріалів; – актуалізація.
Посилання	https://www.eea.europa.eu/

U.S. Environmental Protection Agency

Сайт є офіційним сайтом Агентства з охорони навколишнього середовища (EPA) США. Він надає доступ до інформації, ресурсів та інструментів, пов’язаних з охороною навколишнього середовища та забезпеченням здоров’я людини. Загалом сайт EPA є важливим ресурсом для всіх, хто цікавиться питаннями охорони навколишнього середовища та здоров'я людини у США.

Таблиця 2.3 – Опис вебзастосунку U.S. Environmental Protection Agency

Назва	**U.S. Environmental Protection Agency**
Архітектура	Web application
Виробник	U.S. Environmental Protection Agency
Мова реалізації	PHP
Функції	– Додавання новин; – актуальна інформація про законодавство та правила про навколишнє середовище США; – актуальна інформація, щодо забруднення; – доступ до наукової інформації; – освітні матеріали; – співпраця з державними організаціями та бізнесом.
Переваги	– Надійність та точність; – широкий спектр інформації; – простота використання; – ресурси для різних аудиторій; – зворотний зв'язок;
Недоліки	– Сайт може бути складним для новачків; – обмежена кількість мов; – машинний переклад.
Посилання	https://www.epa.gov/

Фактори на які було звернуто особливу увагу:

– доступність для користувачів у яких не має базових знань екології;
– доступність для широкої аудиторії (мультимовність);
– актуальність інформації;
– простота використання та інтуїтивний інтерфейс;
– доступність на будь-якій платформі.

1.2 Специфікація вимог

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якого розробляється ПЗ

Призначенням застосунку є підвищення обізнаності більш широкої аудиторії про природні ресурси за рахунок розробки вебзастосунку.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення ПЗ було використано додатковий фреймворк – Laravel.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 25.12.2022р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування:

Вебзастосунок призначений для надання інформації та ресурсів про екологічний проєкт у будь-який час.

Характеристика користувачів

Основні характеристики користувачів: наявність пристрою (комп’ютер, ноутбук, телефон тощо), доступ до Інтернету.

Загальна структура та склад системи.

Система складається з наступних частин:

1. Клієнтська частина (Front-end):
   - дизайн та інтерфейс вебсайту;
   - графічний контент та мультимедійні елементи;
   - клієнтська програмна логіка (JavaScript, jQuery).
2. Серверна частина (Back-end):
   - хостинг та база даних;
   - серверна логіка (Laravel, PHP);

– інтерфейс взаємодії з базою даних.

3. Адміністративна панель:
   – інтерфейс адміністратора;
   – функціонал керування контентом (додавання, редагування, видалення);
4. База даних:
   – збереження статей, новин, шляхів до зображень, файлів;
   – взаємодія з БД через ORM.

Загальні обмеження

Основне обмеження у використанні застосунку – доступ до інтернету.

ФУНКЦІЇ СИСТЕМИ

Додавання новини

Опис функції

Функція допомагає адміністратору надавати користувачам нову та актуальну інформацію.

Вхідна і вихідна інформація

Вхідна інформація – заголовок статті українською та англійською, вміст новини англійською і українською та, можливо, зображення.

Вихідна інформація – згенерована сторінка.

Функціональні вимоги

База даних, що зберігає новини та доступ до Інтернету.

Створення галереї

Опис функції

Додавання зображення у галерею, що підтверджує діяльність проєкту.

Вхідна і вихідна інформація

Вхідна інформація – файл та опис до нього.

Вихідна інформація – шлях до файлу, що знаходиться на сервері.

Функціональні вимоги

База даних, що зберігає зображення та доступ до Інтернету.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

В цьому ПЗ вхідні дані отримуються від адміністратора, що, заповнюючи форми, наповнює застосунок вмістом.

Нормативно-довідникова інформація (класифікатори, довідники тощо)

Вимоги до даного пункту відсутні.

Вимоги до способів організації, збереження та ведення інформації

Обмін даними відбувається через обробку запитів у контролері та використанням ORM. БД – MySQL.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Для розробки програмного забезпечення немає значних технічних обмежень. Але для підтримки необхідних програм та комфортної розробки бажано мати комп’ютер або ноутбук з оперативної пам’яті місткістю не менше 8 Гбайт.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Система складається з: клієнтської частини, серверної частини та БД.

Системне програмне забезпечення

Застосунок побудований з використанням фреймворку Laravel. Для написання клієнтської частини були використані: Bootstrap, OwlCarousel2, jQuery. Для серверної частини: Laravel, MySQL. В якості БД для застосунку обрано MySQL.

Мережне програмне забезпечення

Під час розробки було використано ОС Windows 10, для написання коду було використано IDE phpStorm, для перегляду застосунку веббраузер Google Chrome.

Програмне забезпечення ведення інформаційної бази

За допомогою ORM та CRUD-операцій виконується маніпуляції з БД.

Мова і технологія розробки ПЗ

Програмне забезпечення має розроблюватися з використанням фреймворку Laravel. Мови розробки – PHP та JavaScript.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс має бути інтуїтивно зрозумілим користувачеві, для цього були використані стандартні рішення при верстці.

Апаратний інтерфейс

Програмний застосунок має бути доступним на будь-якому пристрої з будь-яким розширенням екрану, тому інтерфейс має бути адаптивним.

Програмний інтерфейс

Laravel – є потужним фреймворком з великою кількістю інструменті, що допомагають та пришвидшують розробку. Вагомі переваги, що будуть використані при розробці: ORM, шаблонізатор Blade, простори імен що допомагають реалізувати паттерн MVC.

Комунікаційний інтерфейс

При хостингу вебзастосунку використовується протокол HTTP з криптографічним протоколом SSL

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Доступний для будь-якого користувача, що має бажання ознайомитися з екологічним проєктом, за наявності у нього пристрою з браузером та доступом до Інтернету.

Супроводжуваність

Вебзастосунок не потребує супроводжуваності з боку розробника, окрім моментів з заміною деталей та стилю дизайну застосунку, за бажанням замовника.

Переносимість

Застосунок доступний на будь-якому пристрої та ОС.

Продуктивність

Продуктивність залежить від якості Інтернет-з’єднання.

Надійність

Пароль користувача хешований.

Безпека

Доступ до редагування є лише у адміністратора.

Висновки до розділу 1

В першому розділі кваліфікаційної роботи бакалавра було оглянуто та проведено аналізовано існуючі застосунки-аналоги, після чого було створено та узгоджено з замовником специфікацію вимог, у якій було зазначено наступні вимоги, функції та властивості ПЗ:

- призначення та межі проєкту;
- загальний опис;
- функції системи;
- вимоги до інформаційного забезпечення;
- вимоги до технічного забезпечення;
- вимоги до програмного забезпечення;
- вимоги до зовнішніх інтерфейсів;
- властивості програмного забезпечення.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ

2.1 Створення сценаріїв

Use Case – це послідовний перелік дій (сценарій), відповідно якого користувач буде взаємодіяти з застосунком, для виконання якоїсь дії, що передбачене функціоналом застосунку. Це допоможе проаналізувати поведінку застосунку при певній дії, та передбачити дії користувача [5].

Короткий сценарій

Адміністратор вебзастосунку заходить на сайт. Виконує вхід у обліковий запис. Після чого відкривається адмін. Панель Додає новину про проведений захід. Переходить на головну сторінку, для перевірку коректного відображення. Потім знову повертається до адмінпанелі та завантажує зображення та ресурси. Все коректно додано. Переходить на головну сторінку для перевірки коректності. Все коректно відображається.

Середній сценарій

Головний сценарій (успішний):

Адміністратор вебзастосунку заходить на сайт. Виконує вхід у обліковий запис. Після чого відкривається адмін. панель, додає новину про проведений захід. Переходить на головну сторінку, для перевірки коректного відображення. Потім знову повертається до адмінпанелі та завантажує зображення та ресурси. Все коректно додано. Переходить на головну сторінку для перевірки коректності. Все коректно відображається.

Альтернативний сценарій:

1. Не коректний формат зображення для WYSIWYG-редактору, тому новина не додана.
2. Некоректний формат зображення для додавання у галерею.
3. Перевищений розмір для додавання файлів у ресурси.

4. Збій системи на сервері, через що вебзастосунок відмовляється працювати.

5. Адміністратор ввів не правильні дані користувача.

Повний сценарій

Таблиця 2.1 – Повний сценарій

Use Case Name	Додавання інформації
Scope	Вебзастосунок відображення розвитку екологічного проєкту
Level	Мета адміністратора
Primary Actor	Адміністратор
Stakeholders and interests	– Користувач: Зацікавлений у здобуті інформації про проєкт та його розвиток. – Адміністратор: Зацікавлений у перегляді коректності відображення інформації. – Керівник проєкту: Зацікавлений у коректності даних.
Preconditions	Адміністратор ввів коректні дані
Main Success Scenario	1. Після входу адміністратор опинився у адмінпанелі. 2. Перейшов на вкладнику керуванням новинами. 3. Переходить на сторінку додавання новини. 4. Вводить необхідну інформацію, та редагує вміст новини. 5. Зберігає новину. 6. Переходить на вкладнику керуванням зображенням. 7. Переходить у режим додавання зображень. 8. Додає зображення та вводить його опис. 9. Зберігає введену інформацію. 10. Виходить з адмін панелі.

Кінець таблиці 2.1

Extensions	1. Не коректний формат зображення для WYSIWYG-редактору, тому новина не додана. 2. Некоректний формат зображення для додавання у галерею. 3. Перевищений розмір для додавання файлів у ресурси. 4. Збій системи на сервері, через що вебзастосунок відмовляється працювати. 5. Адміністратор ввів не правильні дані користувача.
Special Requirements	1. Система має виконувати запити не довше 1 секунди. 2. Підтримка мультимовності.
Technology and Data Variations List	1. Для успішного виконання усіх кроків, необхідно мати стабільне інтернет-з'єднання. 2. Редактор новин, реалізовую новину саме так як виглядає у редакторі. 3. При завантаженні pdf- та pptx-файлів потрібно слідкувати за їх об'ємом, щоб він не перевищував 10 Мбайт.
Frequency of Occurrence	Система працює майже безперервно, але це залежить від електропостачання сервера та інтернету.
Miscellaneous	1. Чи виникають збої при виконанні завантаження файлів. 2. Редагування зображення при додаванні новини.

Перевагою табличних та текстових сценаріїв є те що їх легше читати та розуміти замовнику та легше змінювати ніж діаграми.

2.2 Створення use cases діаграм (варіантів використання)

За допомогою діаграм можна переглянути на всі можливі ролі у застосунку, та дії які вони можуть робити у рамках цього застосунку. Тому було розроблено діаграму варіантів використання (рис. 2.1) [4].

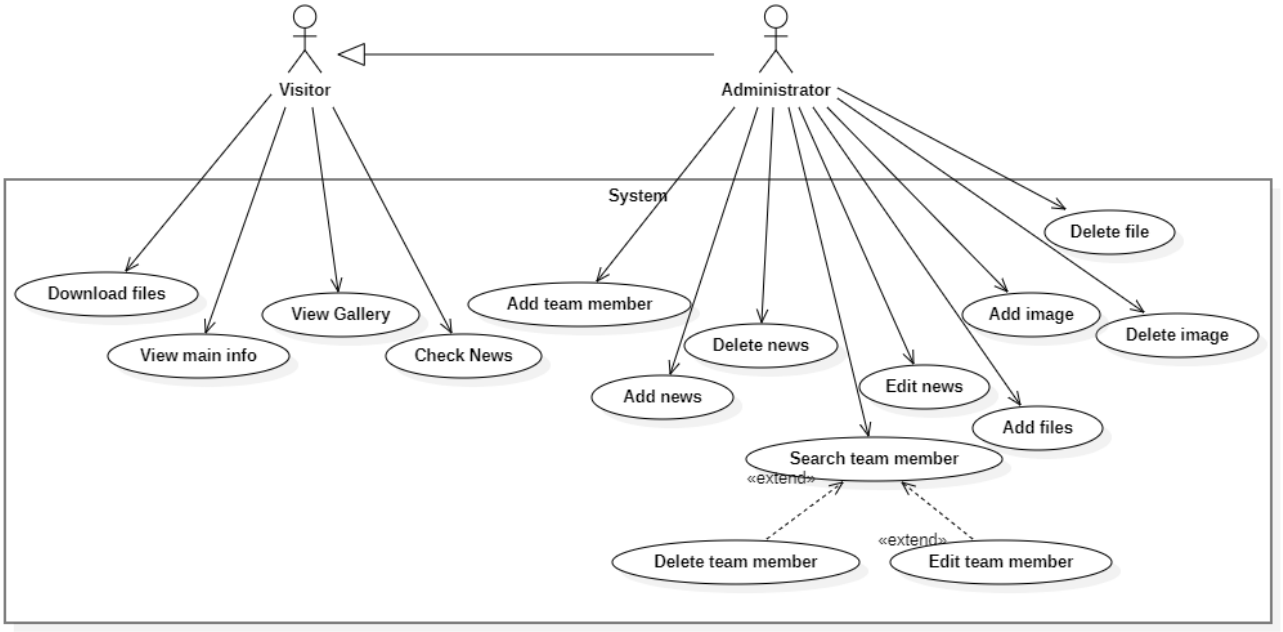

Рисунок 2.1 – Діаграма прецедентів

З даної діаграми можна переглянути усі функції відвідувача та адміністратора. Адміністратор маніпулює контентом на сторінці, додає новини оновлює, або видаляє. Відвідувач переглядає, та слідкує за новинами.

2.3 Побудова діаграми взаємодії

Після визначення функцій, що потрібно реалізувати, потрібно спланувати як виконується та чи інша дія. Для цього були побудовані діаграми взаємодії. На діаграмі взаємодії відображається інтерактивна поведінка система. Тобто на цій діаграмі буде зображено життєвий цикл прецедента. Так як дії CRUD мають схожий алгоритм, та відрізняються полями, тому для кожного об’єкту, тому діаграми були створені по одній на кожну дію [6].

При виконанні входу у систему для редагування даних (рис. 2.2) на якій користувач виконує вхід з введеними даними (1), у контролері виконується валідація даних (2), відправляється запит у модель для порівняння провалідованих даних з даними у БД (3), потім повертається об’єкт (4), виконується запит на генерацію

сторінки в залежності від повернутих даних (5), генерується сторінка (6), повертається (7) для перегляду результату користувачем (8).

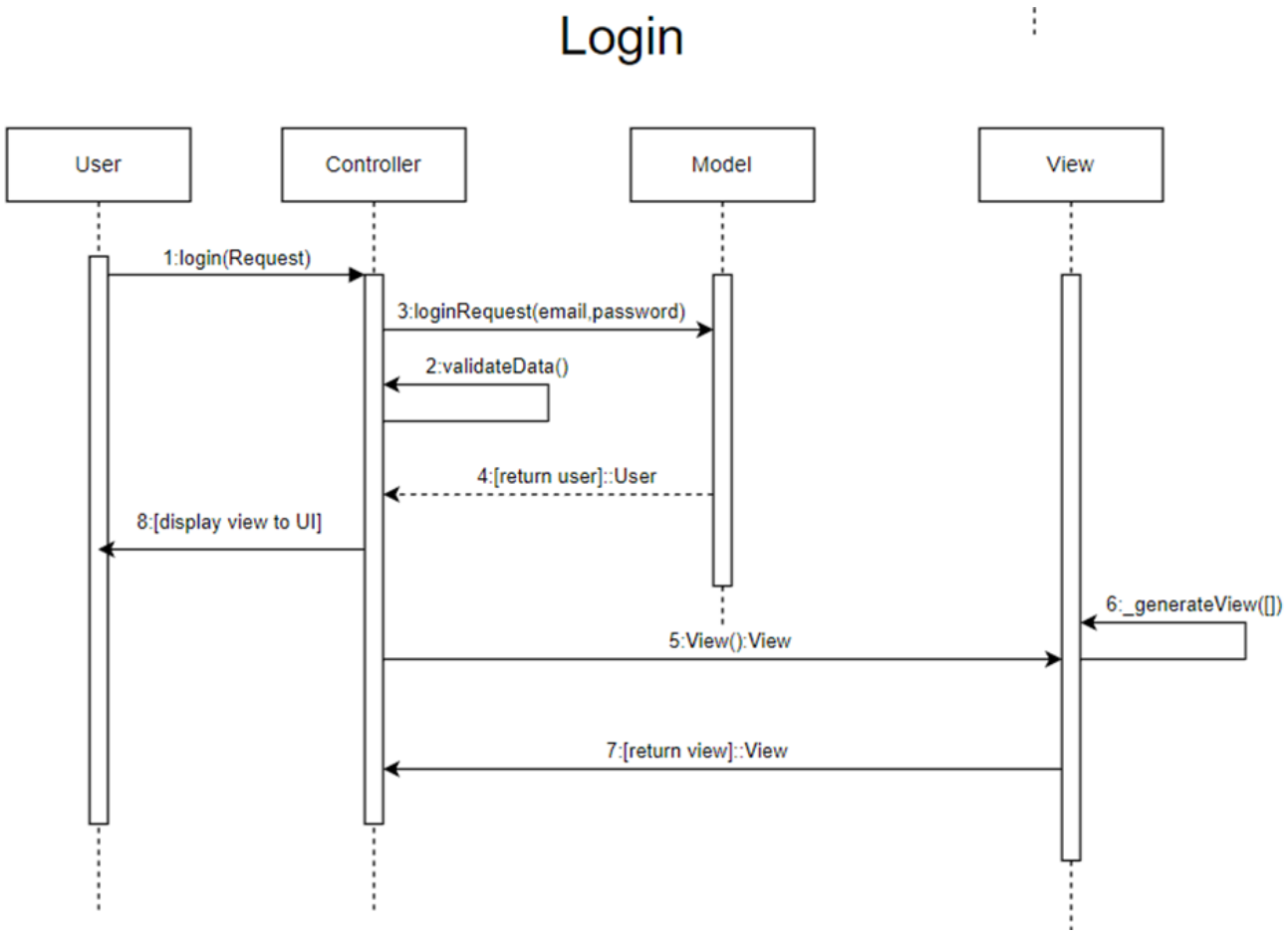

Рисунок 2.2 – Діаграма взаємодії при вході в обліковий запис

Діаграма створення новини має наступну послідовність (рис. 2.3): відправляються дані форми у запиті (1), валідація даних (2), збереження валідованих даних у БД через модель (3), повертається оновлений масив новин (4), запит на оновлення сторінки (5), генерація сторінки (6), повернення сторінки (7), відображення користувачу (8).

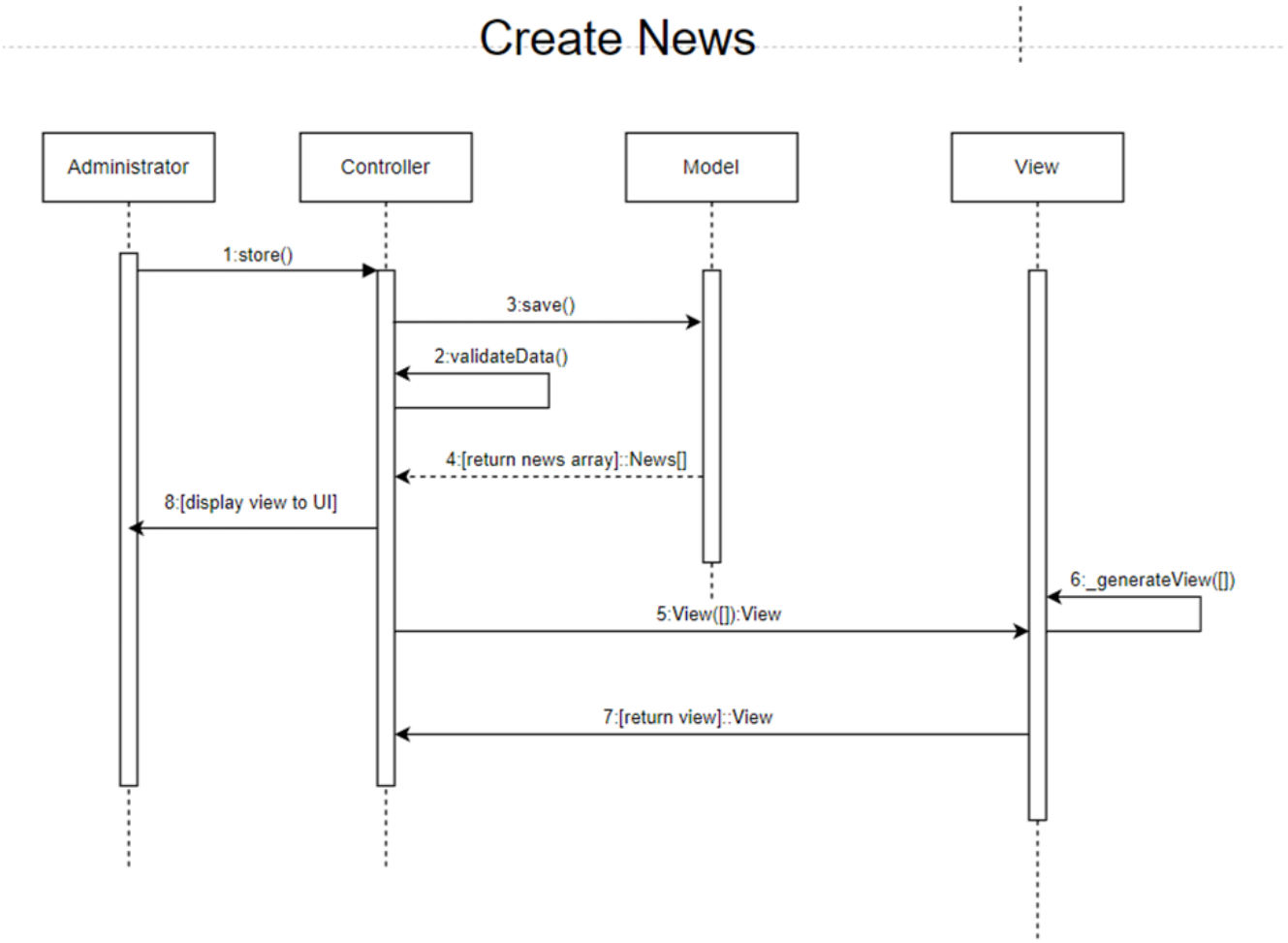

Рисунок 2.3 – Діаграма взаємодії при створені новини

Діаграма оновлення новини має наступну послідовність (рис. 2.4): відправляються дані форми у запиті (1), валідація даних (2), збереження валідованих даних у БД через модель після пошуку за індексом (3), повертається оновлений масив новин (4), запит на оновлення сторінки (5), генерація сторінки (6), повернення сторінки (7), відображення користувачу (8).

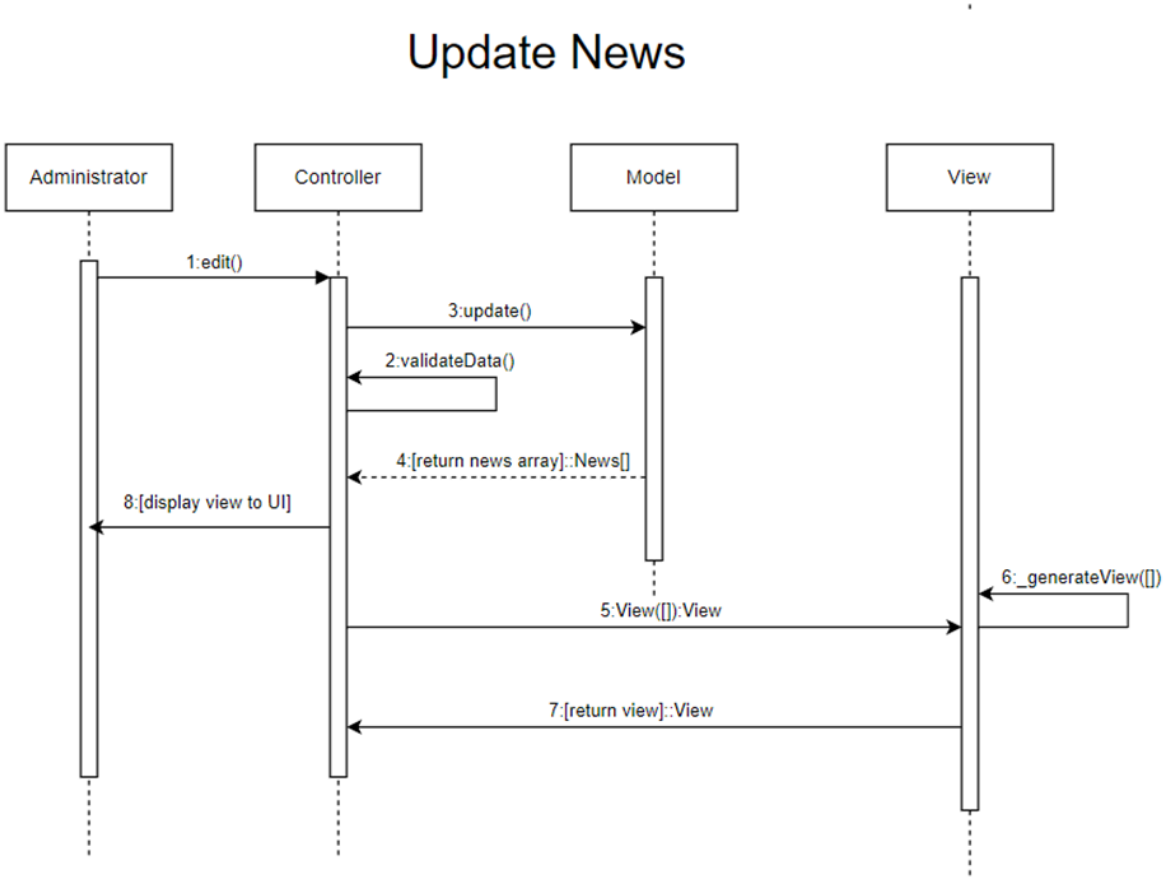

Рисунок 2.4 – Діаграма взаємодії при оновлені новини

Була побудована діаграма послідовності при видаленні (рис. 2.5). Виконується запит на видалення об’єкта (1), запит до БД через модель (2), що повертає оновлений масив новин (3), масив передається у View (4), що у свою чергу генерує сторінку (5), сторінка повертається (6) та відображається користувачу (7).

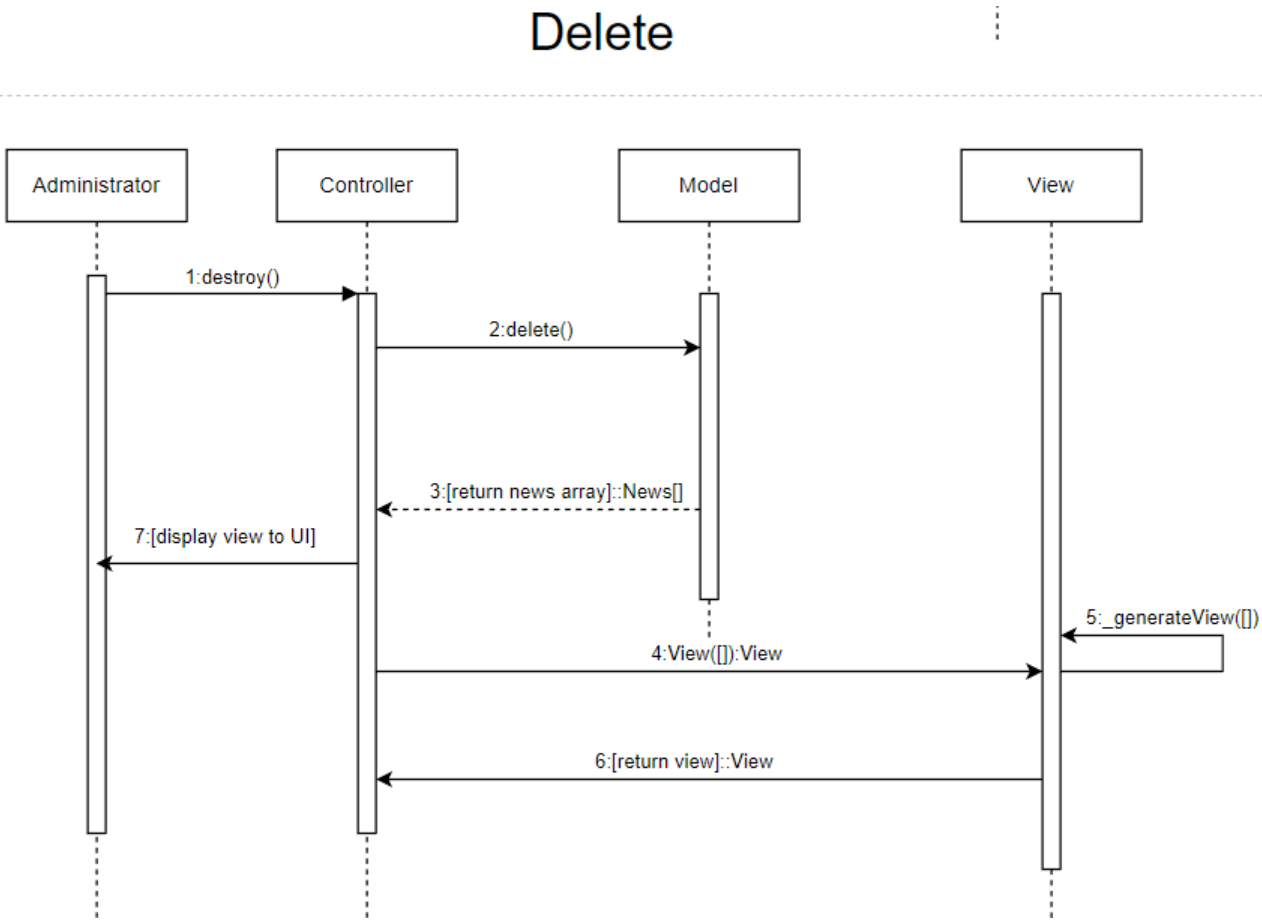

Рисунок 2.5 – Діаграма взаємодії при видалені

Була побудована діаграма послідовності при переході на сторінку з усіма новинами (рис. 3.6). Виконується запит на генерацію сторінки (1), запит до БД через модель (2), що повертає масив з усіма новинами (3), масив передається у View (4), що у свою чергу генерує сторінку (5), сторінка повертається (6) та відображається користувачу (7).

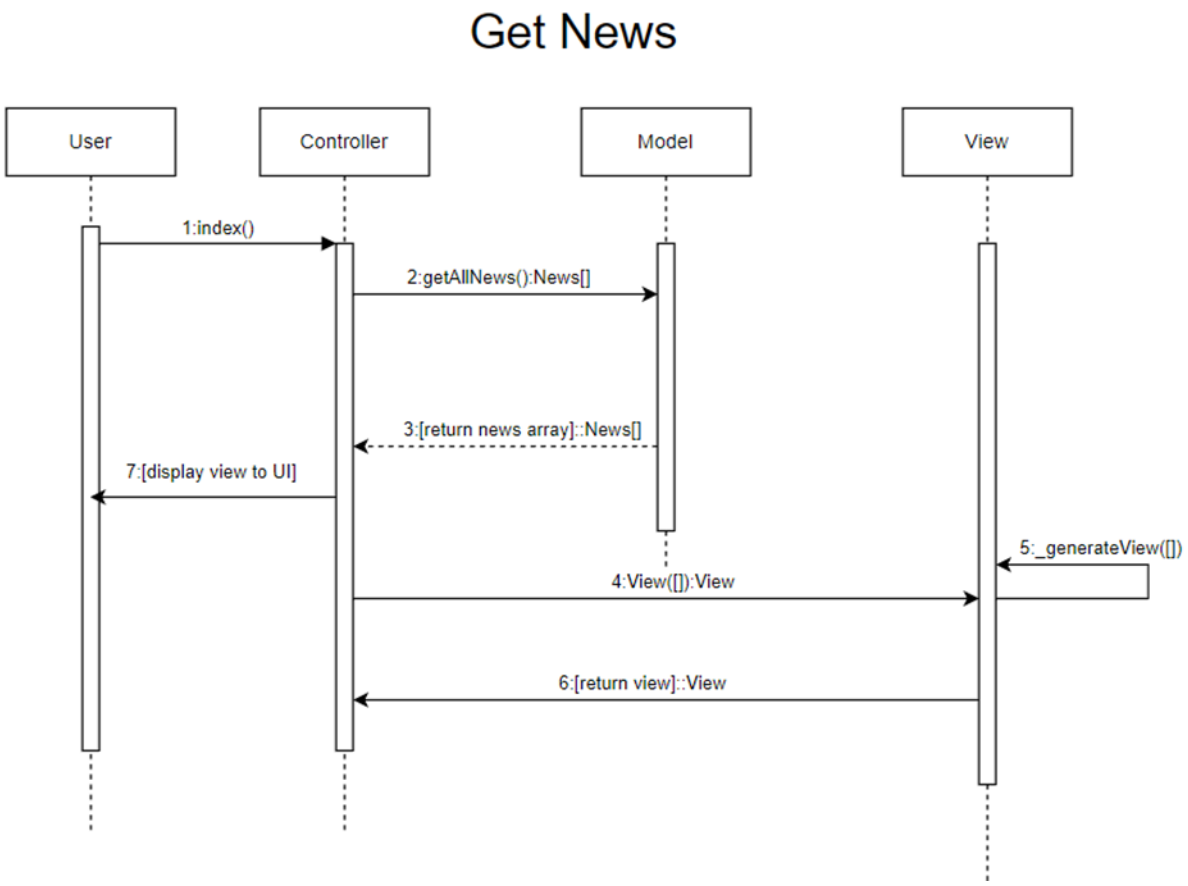

Рисунок 2.6 – Діаграма взаємодії при генерацій списку новин

Була побудована діаграма послідовності при перегляді новини повністю (рис. 3.7). Виконується запит на відображення новини (1), запит за ідентифікатором до БД через модель (2), що повертає об’єкт (3), об’єкт передається у View (4), що у свою чергу генерує сторінку (5), сторінка повертається (6) та відображається користувачу (7).

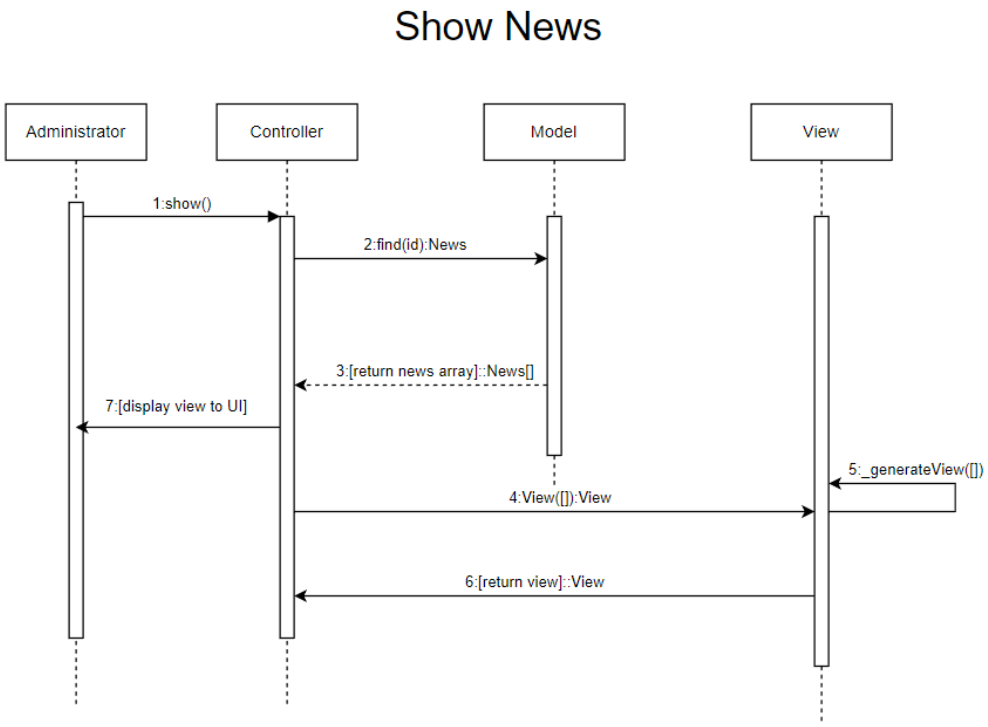

Рисунок 2.7 – Діаграма взаємодії детальний перегляд новини

Ці діаграми допоможуть краще розуміти як реалізувати ту чи іншу дію при кодуванні.

2.4 Побудова діаграми станів

Діаграма станів та переходів візуально показує, яка деякий об’єкт переходить з одного стану в інший. Було створено діаграму станів застосунку, об’єктом цієї діаграми є застосунок (рис. 2.8) [7]. Стан у точці входу «Користувач поза сайтом» потім користувач відкриває сайт, застосунок переходить у стан очікування запиту від користувача, користувач може, за допомогою інтерфейсу, відправити GET-запит до серверу (перехід на іншу вкладку сайту), тоді застосунок у стані обробки запиту, відправляє оброблені дані, тоді застосунок знаходиться у стані генерації результату, що відповідає обробленим даним, згенерований результат відображається користувачу. Також користувач може відправити POST-запит, він також оброблюється, генерується результат та повертається користувачу. Точкою виходу є

покидання користувачем вебзастосунку. Це може статися у будь-якому стані вебзастосунку.

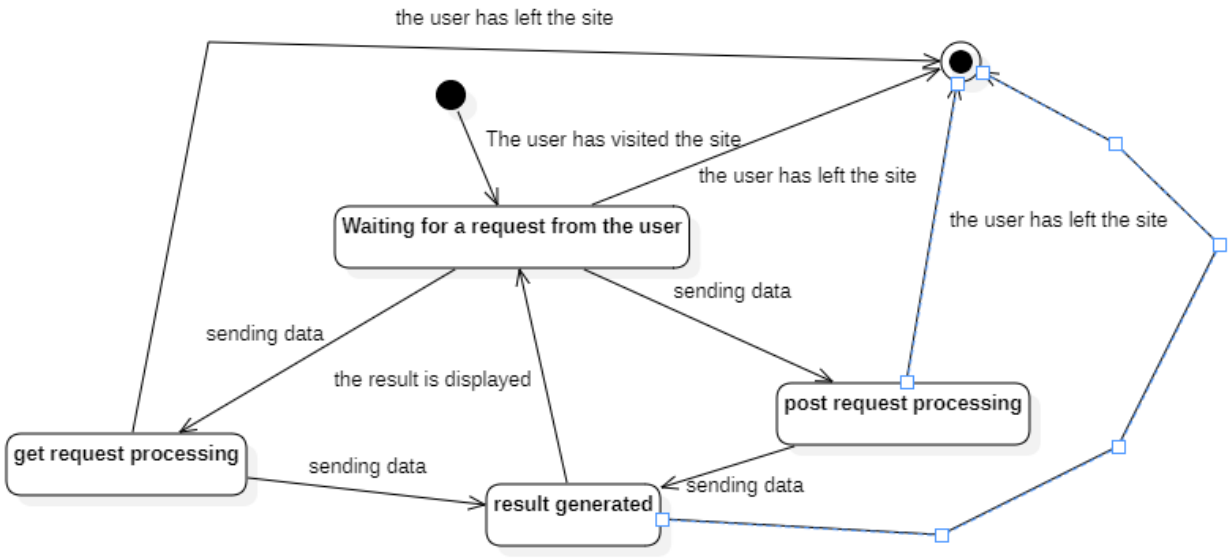

Рисунок 2.8 – Діаграма станів застосунку

Потім було створено діаграму станів для новини (рис. 2.9). Об’єктом є новина. Точкою входу є «Новина не існує», потім новина створюється та переходить у стан «Новина у процесі створення», після цього новина може бути у станах «Новина видалена», «Новина в процесі оновлення», «Новина в стані валідації», після ж валідації новина у стані збереження у БД, а з БД може бути видалення або у процесі оновлення. Точкою виходу є стан «Новина видалена».

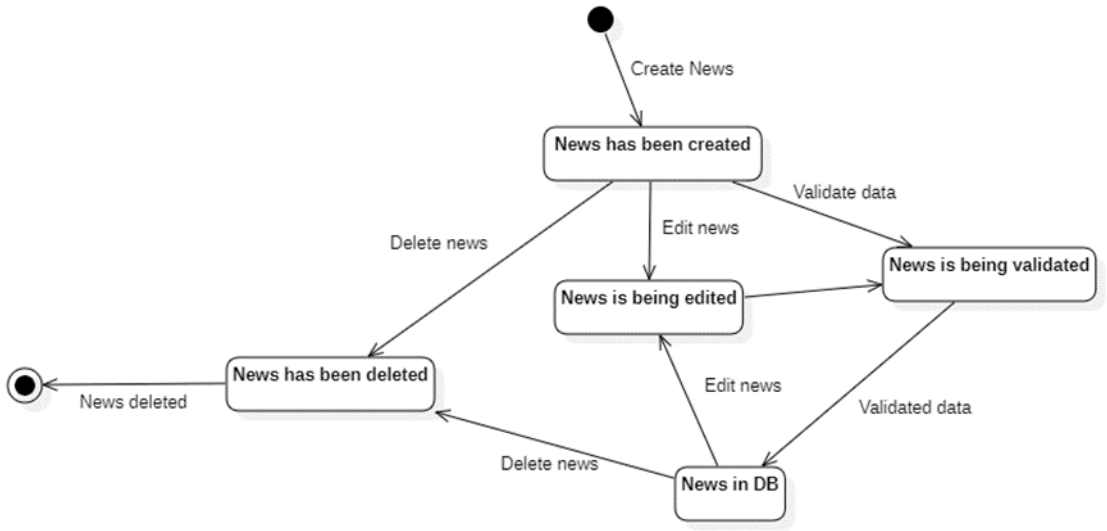

Рисунок 2.9 – Діаграма станів новини

Варто зазначити, що у цій діаграмі новина може бути замінена на будь-який інший об’єкт застосунку (картинка, член команди тощо).

Висновки до розділу 2

У другому розділі було описано моделювання та планування програмного забезпечення. Були побудовані наступні завдання:

- написання сценарію використання;
- побудова діаграми прецедентів;
- побудова діаграма взаємодії;
- побудова діаграми станів.

Завдяки цьому були закріплені навички побудови діаграм та планування ПЗ. Також це допоможе в подальшій розробці ПЗ.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ТА ОГЛЯД СТЕКУ ТЕХНОЛОГІЙ

3.1 Огляд технологій

При розробці вебзастосунку «Вебзастосунок відображення розвитку екологічного проєкту програми Жана Моне» використано наступні технології:

- мови програмування – PHP, JavaScript;
- фреймворк – Laravel;
- система управління базою даних – MySQL;
- інструмент для контролю версій – Git;
- вебсервер – Apache;
- бібліотеки для реалізацій деяких функцій – FancyBox, jQuery, Owl carousel 2, Bootstrap, TinyMCE.

Обрання стеку технологій грає важливу роль у процесі розробки програмного забезпечення, так як він визначає основі інструменти, що будуть використовуватися для створення проєкта.

Важливість обрання стеку технологій пов’язана з декількома факторами:

- Функціональність: кожна технологія має свої особливості та можливості і вибір стеку технологій повинен відповідати вимогам проєкту.
- Продуктивність: вибір стеку технологій може суттєво впливати на продуктивність застосунку. Правильний вибір технологій може допомогти оптимізувати продуктивність та прискорити роботи системи.
- Розробка: вибір технологій також може впливати на швидкість та комфортність розробки. Деякі технології та фреймворки можуть значно спростити процес розробки та прискорити процес створення застосунку.

- Спільнота та підтримка: вибір популярних та широко використовуваних технологій може забезпечити наявність розвиненої спільноти розробників та ресурсів підтримки, що може значно полегшити розробку та підтримку системи.
- Безпека: вибір стеку технологій може впливати на рівень безпеки системи. Деякі технології та фреймворки мають механізм захисту від уразливостей та атак, що може підвищити безпеку програми.

В цілому правильний вибір стеку технологій може допомогти створити високоякісний та продуктивний застосунок, а також полегшити процес розробки та підтримки.

3.2 Діаграма класів

Діаграма класів – це графічне представлення класів, що входять до складу системи і в’язки між ними. Вона використовується для моделювання структури програмного забезпечення та відображення взаємозв’язків між класами [8]. Це допомагає уникнути можливих помилок та проблем, які можуть виникнути в процесі розробки [8, 9].

Було побудовано діаграму класів для розроблюваного ПЗ (рис. 3.1).

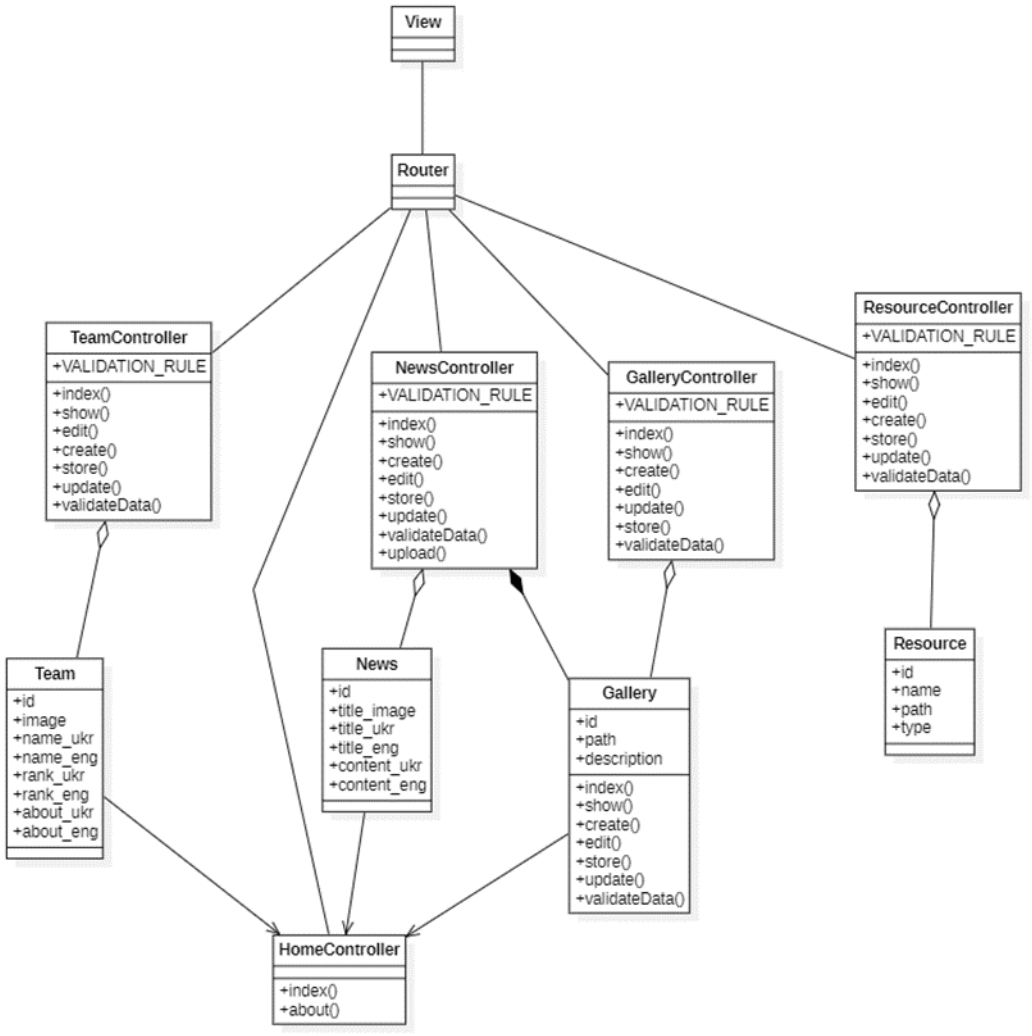

Рисунок 3.1 – Діаграма класів

Діаграма дещо узагальнена задля розуміння. На діаграмі зображені контролери, у яких генерується запит до БД через моделі та повертається результат з БД. З’єднання між контролерами та видом, виконується через вбудований інструмент фреймворку Laravel, що відстежує та передає посилання та дані. Та вид, що генерує сторінку за запитом та з отриманими даними. У діаграмі моделі відображаються іменником, а контролери же мають ключове слово Controller, усі моделі з’єднанні зі своїми контролерами (назви збігаються) за допомогою агрегації, тому що методи моделей (на діаграмі моделі не мають методів, так як вони є наслідниками вбудованого у Laravel

класу Model, що реалізує можливості ORM. A Gallery з’єднана з NewsController так як з Gallery викликається метод через делегування. Усі контролери мають двонаправлену асоціацію з роутером, бо роутер приймає результат обробки, та відправляє у контролери запити.

У контролерах є методи з однаковими назвами (за стандартом Laravel так повинно бути) є сім стандартних назв методів.

1. Index() – зазвичай використовується для генерації сторінки перегляду списку об’єктів моделі.
2. Show() – використовується для генерації детального відображення опису об’єкту.
3. Create() – використовується для генерації сторінки з формою створення об’єкту.
4. Edit() – використовується для генерації сторінки з формою для редагування сторінки.
5. Update() – обробка post-запиту на оновлення об’єкту.
6. Store() – обробка post-запиту на створення об’єкту.
7. Destroy() – видалення об’єкту з БД.

ValidateData() метод для спрощення бізнес-логіки та уникання дублюючого коду для перевірки введених даних.

3.3 Діаграма компонентів

Діаграма компонентів – це структурна діаграма, яка відображає компоненти системи та їх взаємодії. У контексті паттерну проєктування Model-View-Controller (MVC), діаграма компонентів відображає структуру системи з точки зору її складових частин: моделі (Model), представлення (View) та контролера (Controller) (рис. 3.2) [10].

Рисунок 3.2 – Діаграма компонентів

Розроблення діаграми компонентів для MVC допомагає краще зрозуміти структуру системи та забезпечити її правильну розробку та інтеграцію. Крім того, її можна використовувати для документування ПЗ.

3.4 Діаграма пакетів

Діаграма пакетів є структурним типом діаграми, який використовується для візуального відображення взаємозв'язків між пакетами (групами класів, інтерфейсів, функцій тощо) у програмному проєкті [11].

Було створено діаграму пакетів вебзастосунку mvc Laravel (рис. 3.3). На діаграмі взаємодіють три великих пакети, що відображають логіку взаємодії паттерну MVC. У пакеті моделей знаходяться моделі. У пакет контролер – контролери. У пакет виду знаходяться файли виду форм та сторінок для кожної моделі, стиль для сторінок та скрипти, для анімацій та взаємодії.

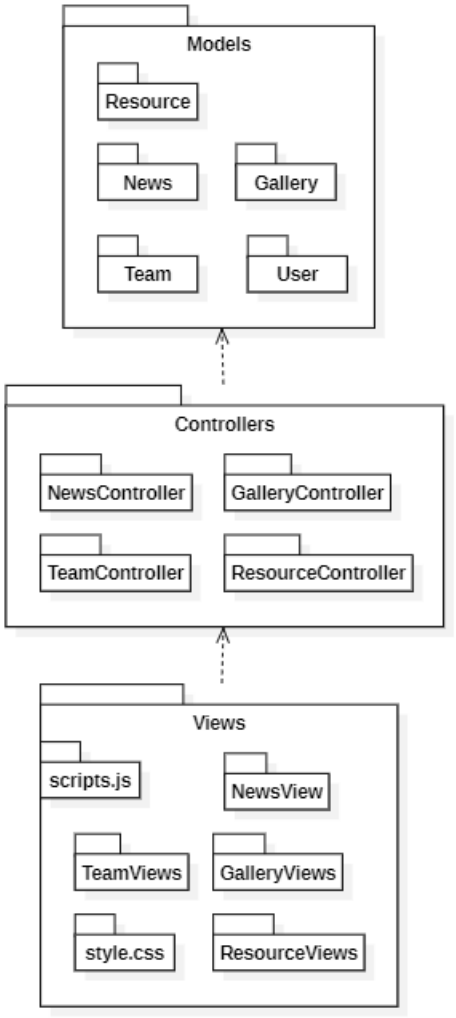

Рисунок 3.3 – Діаграма пакетів

3.5 Діаграма розгортання

Діаграма розгортання в MVC використовується для моделювання фізичного розміщення компонентів системи, їх залежності та взаємодії на рівні виконання. Діаграма допомагає визначити архітектуру системи на рівні виконання, тобто вказує, де будуть розміщені окремі компоненти, які мережеві протоколи будуть використані для взаємодії між ними, а також де буде розміщена база даних та інші важливі ресурси [12].

На діаграмі розгортання ПЗ (рис. 3.4) зображено три основні компоненти системи 3 вузли, що представляють сервер з базою даних, сервер на якому «хоститься» за стосунок та оброблює запити, та пристрій користувача, що має браузер. На стороні клієнта виконуються «скріпти» для анімації. На стороні «хосту» оброблюються запити зі сторони користувача та повертають результат з бази даних. База даних повертає результат запитів.

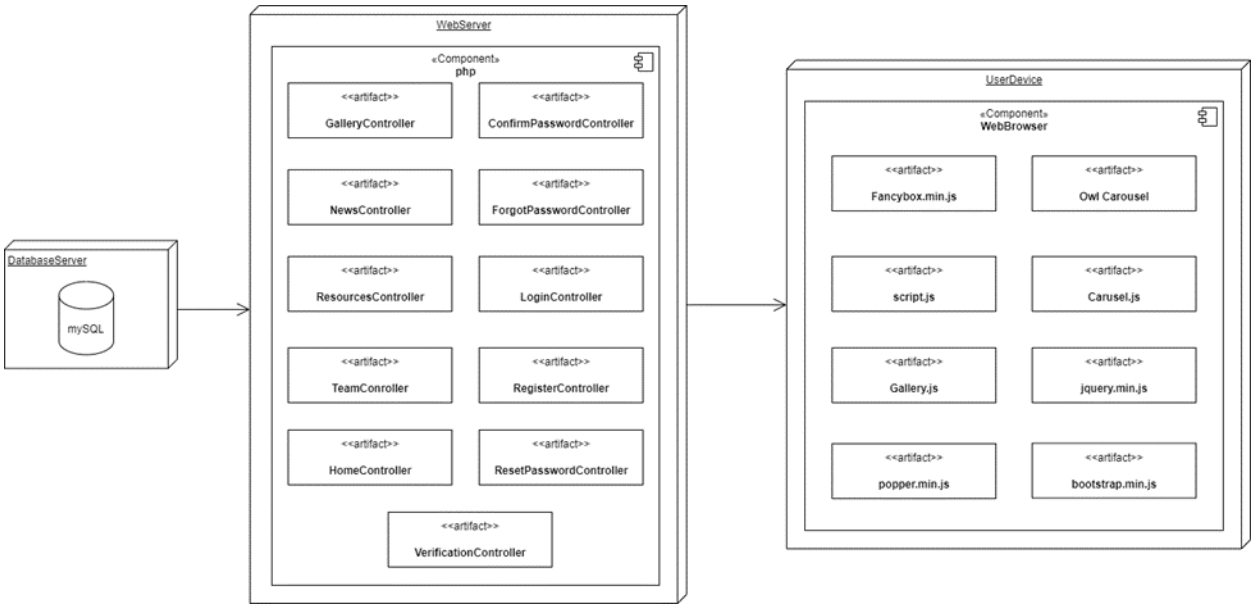

Рисунок 3.4 – Діаграма розгортання

Висновки до розділу 3

Було проведено роботу над проєктування програмного забезпеченням, що є останнім та важливим етапом перед реалізацією ПЗ, завдяки ньому при розробці час буде витрачено лише на реалізацію. Під час проєктування були виконані наступні завдання:

- побудова діаграми класів;
- побудова діаграми компонентів;
- побудова діаграми пакетів;
- побудова діаграми розгортання.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ТА ПРОВЕДЕННЯ ТЕСТУВАННЯ

4.1 Налаштування середовища та створення БД

Завчасно було встановлено портативний локальний WAMP/WNMP сервер OpenServer, що має у собі багатофункціональну керовану програму та великий вибір підключених компонентів, серед цих компонентів є усі необхідні, для створення Laravel-застосунку. За допомогою, вбудованої консолі було створено проєкт (рис. 4.1).

```
IP@DESKTOP-08R575L d:\OSPanel\domains
# composer create-project --prefer-dist laravel/laravel Ecology
```

Рисунок 4.1 – Створення проєкту

Після введення команди, розпочнеться встановлення пакетів, що забезпечують основні функції фреймворку.

Далі створено порожню базу даних з назвою ecology у phpMyAdmin. Потім було змінено файл конфігурації .env у який записано тип з’єднання, сервер, порт та назву БД також вказано ім’я користувача та пароль з’єднання (рис. 4.2).

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=ecology
DB_USERNAME=*********
DB_PASSWORD=*********
```

Рисунок 4.2 – Налаштування з'єднання з БД

Так як, записи матимуть зображення, то для їх збереження у файлі filesystems.php вказано шлях посилання (рис. 4.3) та за допомогою команди створено символьне посилання на директорію storage (рис. 4.4.), у яку за замовчуванням зберігаються завантажені файли, але звернутися до них можна лише через посилання.

```
'links' => [
    public_path( path: 'storage') => storage_path( path: 'app/storage'),
],
```

Рисунок 4.3 – Вказання посилання

```
IP@DESKTOP-08RS75L d:\OSPanel\domains\Ecology
# php artisan storage:link
```

Рисунок 4.4 – Створення посилання

Далі створено моделі, для кожної моделі створено контролер, міграцію та seeder (рис. 4.5).

```
IP@DESKTOP-08RS75L d:\OSPanel\domains\Ecology
# php artisan make:model News -cms
```

Рисунок 4.5 – Команда для створення моделі, контролера, міграції та seeder

Таким чином створені усі необхідні сутності.

За допомогою міграцій будуть автоматично створені таблиці у БД. Вміст файлів міграції зображено на рис. 4.6.

```
public function up()
{
    Schema::create( table: 'news', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'title_image');
        $table->string( column: 'titleUkr');
        $table->string( column: 'titleEng');
        $table->longText( column: 'contentUkr');
        $table->longText( column: 'contentEng');
        $table->timestamps();
    });
}
 Ilona-Poltavets
public function down()
{
    Schema::dropIfExists( table: 'news');
}
```

Рисунок 4.6 – Вміст міграції

Також було встановлено пакети для аутентифікації та авторизації користувачів, для цього виконано наступну команду, що зображена на рис. 4.7.

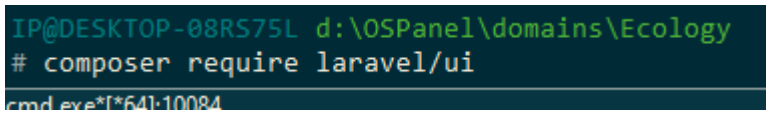

Рисунок 4.7 – Встановлення пакету

Після закінчення завантаження пакету, створено шаблони автентифікації. На рис. 4.8 команда для створення шаблонів аутентифікації та авторизації.

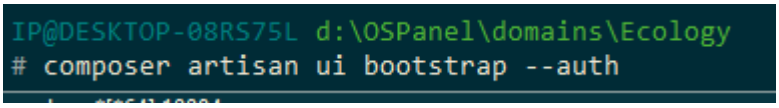

Рисунок 4.8 – Генерація шаблонів для входу/реєстрації

На рис. 4.9 зображена команда встановлення залежностей, та побудови ресурсів.

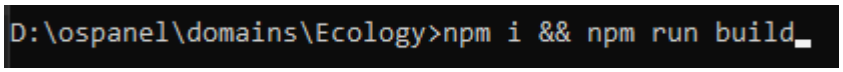

Рисунок 4.9 – Встановлення та побудова пакетів

На цьому етапі налаштування було завершено.

4.2 Реалізація CRUD, для усіх сутностей. Створення адміністративної панелі

Для маніпулювання даними, необхідно для кожної сутності створити контролер, що виконуватиме запити до БД та повертатиме результат виконання. Для забезпечення CRUD у Laravel за замовчуванням є 7 методів:

– index – не отримує вхідних параметрів, виконує вибірку з БД та повертає шаблон виду для відображення усіх обраних записів;

– show – отримує ідентифікатор об’єкту, для якого потрібно відобразити більш детальну інформацію, за ідентифікатором виконується пошук у БД;

– edit – отримує ідентифікатор об’єкту, що необхідно відредагувати, за отриманим ідентифікатором виконує пошук у БД, потім повертає шаблон для редагування з об’єктом;

– create – повертає шаблон для створення об’єкту;

– store – виконує валідацію та збереження об’єкту у БД, на вхід отримує дані з запиту, після успішного виконання збереження повертає користувача на вказану у return сторінку;

– update – виконує оновлення запису об’єкту у БД, за ідентифікатором;

– destroy – виконує видалення об’єкту з БД.

Ці методи, здебільшого генерують види, для кожної моделі є наступні шаблони.

– create.blade.php – шаблон для створення об’єкту;

– edit.blade.php – шаблон для редагування об’єкту;

– form.blade.php – необов’язковий шаблон, але він полегшує роботу, вміщуючи у собі форму, для заповнення вхідних даних, що використовується у edit та create;

– index.blade.php – шаблон у якому буде відображена таблиця, через яку адміністратор буде маніпулювати даними.

– show.blade.php – шаблон для відображення детальної інформації про об’єкт.

На рис. 4.10 зображено ієрархію шаблонів.

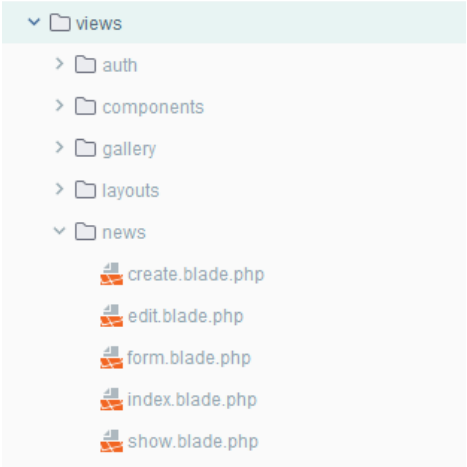

Рисунок 4.10 – Ієрархія об'єктів

На рис. 4.11 зображено index метод для моделі News. У ньому виконується запит до таблиці, що належить моделі News, отримані запити відсортовані за датою створення, від нового до старого, та отримані запити поділяються на сторінки, так щоб на кожній сторінці було до 10 об’єктів. Отримані записи відправляються на сторінку /news, що буде відображати таблицю об’єктів адміністратору.

```
Ilona-Poltavets
public function index()
{
    $data['news'] = News::orderBy('created_at', 'DESC')->paginate(10);
    return view( view: 'news.index', $data);
}
```

Рисунок 4.11 – Метод для відображення усіх записів

У додатку А вміст index.blade.php.

За допомогою методів edit та create викликається шаблон edit.blade.php та create.blade.php, відповідно. Та на рис. 4.13 та 4.14 відображено вміст цих файлів.

```
@extends('layouts.admin')
@section('title','Додати новину')
@section('content')
    <form method="post" enctype="multipart/form-data" action="{{route('news.store')}}">
        @method('POST')
        @csrf
        @include('news.form')
    </form>
@endsection
```

Рисунок 4.12 – Create.blade.php

```
@extends('layouts.admin')
@section('title','Редагувати новину')
@section('content')
    <form method="post" enctype="multipart/form-data" action="{{route('news.update',$post->id)}}">
        @method('PATCH')
        @csrf
        @include('news.form')
    </form>
@endsection
```

Рисунок 4.13 – Edit.blade.php

За допомогою директиви @method('PATCH') вказано HTTP-метод, а директива @csrf, генерує токен, для захисту вразливих ресурсів, що передаються. Та директива @include('news.form') викликає шаблон form.blade.php, що містить поля форми. Для внесення контенту новини використано WYSIWYG-редактор TinyMCE, за комфортного введення інформації.

Для забезпечення роботи TinyMCE, було завантажено бібліотеку локально та створено директорію views/components, у якій знаходились файли для налаштування редактору. У файлі views/components/tinymce-config.blade.php (рис. 4.14) знаходиться скрипт для вказання селектору, за яким будуть знаходитися елементи, що

використовуватимуть редактор, інструменти, що знаходитимуться на панелі інструментів.

```
tinymce.init({
    selector: 'textarea',  // change this value according to your HTML
    language: 'uk',
    plugins: 'image',
    toolbar: 'undo redo | bold italic underline superscript subscript strikethrough | forecolor backcolor fontsize
});
```

Рисунок 4.14 – Налаштування TinyMCE

Додатково вказано мову, якою будуть відображатися назви інструментів та через параметр plugins додано підтримку завантаження зображення. На рис. 4.15 зображено отриману форму для створення новини.

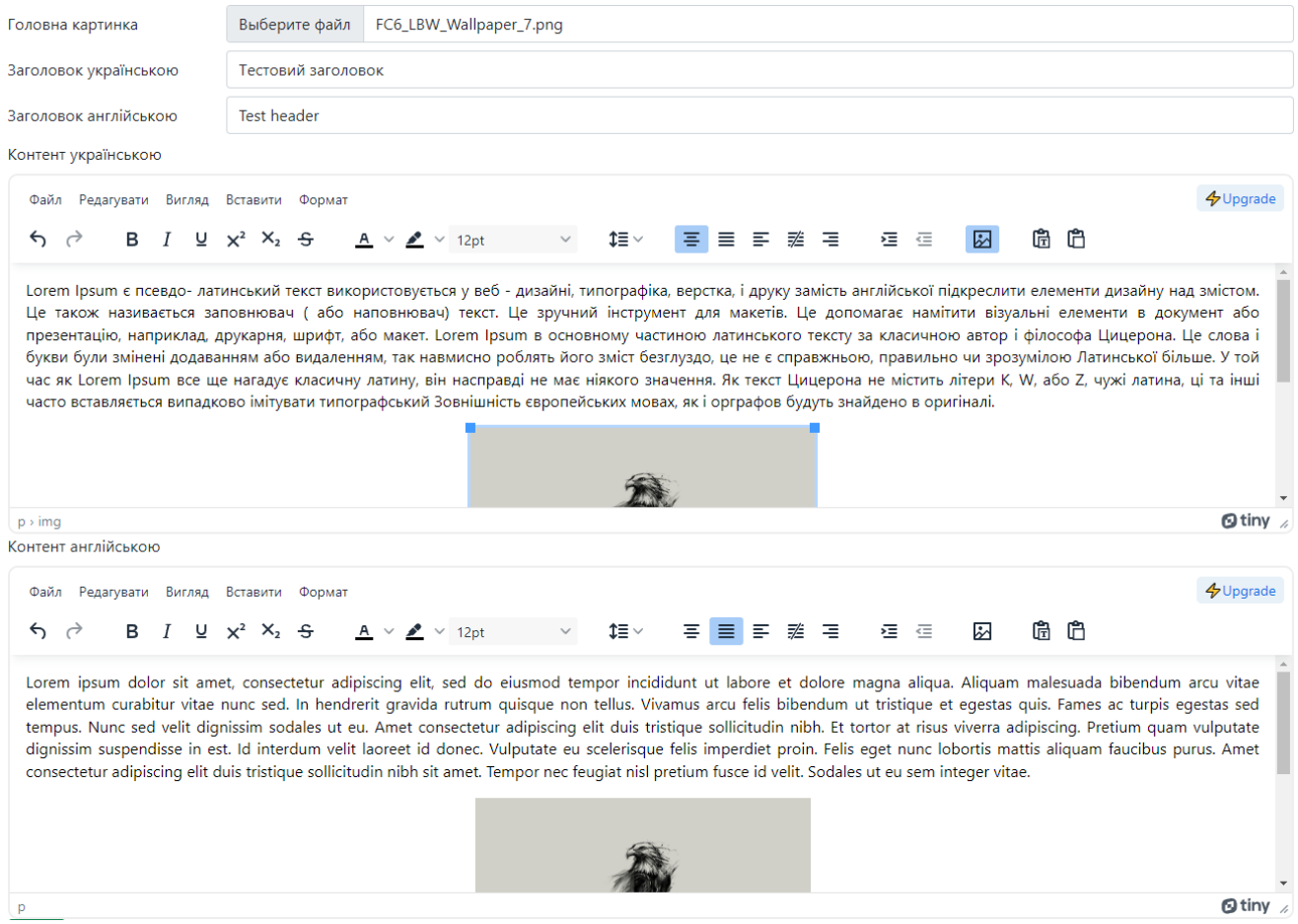

Рисунок 4.15 – Заповнена форма додавання новини

Для збереження нових даних у БД, створений метод store (рис. 4.16).

```
Ilona-Poltavets
public function store(Request $request)
{
    $post = new News();
    $validator = $this->validateData($request);
    if ($validator->fails()) {
        return redirect( to: 'news/create')
            ->withErrors($validator)
            ->withInput();
    }
    $this->save($post, $request);

    return redirect()->route( route: 'news.index');
}
```

Рисунок 4.16 – Метод створення нового запису у БД

Та метод для оновлення записів update (рис. 4.17).

```
Ilona-Poltavets
public function update(Request $request, $id)
{
    $post = News::find($id);
    $validator = $this->validateData($request);
    if ($validator->fails()) {
        return redirect( to: 'news/' . $id . '/edit')
            ->withErrors($validator)
            ->withInput();
    }
    $this->save($post, $request);

    return redirect()->route( route: 'news.index');
}
```

Рисунок 4.17 – Метод для оновлення записів

Дублюючий код збереження та валідації винесено в окремі методи save (рис. 4.18) та validateData (рис. 4.19), відповідно.

```php
function save($post, $request)
{
    $rootPath = ($post->title_image == null || $post->title_image == "") ? 'images/No_photo.png' : $post->title_image;
    if ($request->file('title_image') != null) {
        if ($post->title_image != 'images/No_photo.png') {
            Storage::delete($rootPath);
        }
        $rootPath = ($request->title_image)->store("storage/news");
    }

    $post->title_image = $rootPath;
    $post->titleUkr = $request->titleUkr;
    $post->titleEng = $request->titleEng;
    $post->contentUkr = $request->contentUkr;
    $post->contentEng = $request->contentEng;
    $post->save();
}
```

Рисунок 4.18 – Метод для винесення дублюючого коду, що відповідає за збереження даних у БД

```php
function validateData($request)
{
    $validator = Validator::make($request->all(), rules: self::VALIDATION_RULE);
    return $validator;
}
```

Рисунок 4.19 – Метод для винесення дублюючого коду, що відповідає за валідацію даних

Для повної реалізації CRUD також створений метод для видалення запису з БД destroy (рис. 4.20).

```
Ilona-Poltavets
public function destroy($id)
{
    $post = News::find($id);
    Storage::delete($post->title_image);
    $post->delete();
    return redirect()->route( route: 'news.index');
}
```

Рисунок 4.20 – Метод для видалення запису з БД

Щоб методи виконувалися, додано у файл wep.php маршрути (рис. 4.21). До кожного маршруту додано middleware за правилом auth, щоб маршрут був захищеним від неавторизованих користувачів.

```
Route::get( uri: '/news', action: "App\Http\Controllers\NewsController@index")->name( name: 'news.index')->middleware( middleware: 'auth');
Route::get( uri: '/news/create', action: "App\Http\Controllers\NewsController@create")->name( name: 'news.create')->middleware( middleware: 'auth');
Route::post( uri: '/news', action: "App\Http\Controllers\NewsController@store")->name( name: 'news.store')->middleware( middleware: 'auth');
Route::get( uri: '/news/{id}/edit', action: "App\Http\Controllers\NewsController@edit")->name( name: 'news.edit')->middleware( middleware: 'auth');
Route::patch( uri: '/news/{id}', action: "App\Http\Controllers\NewsController@update")->name( name: 'news.update')->middleware( middleware: 'auth');
Route::delete( uri: '/news/{id}', action: "App\Http\Controllers\NewsController@destroy")->name( name: 'news.destroy')->middleware( middleware: 'auth');
//Route::resource('news','App\Http\Controllers\NewsController')->middleware('auth');
Route::post( uri: '/upload',  action: 'App\Http\Controllers\NewsController@upload')->middleware( middleware: 'auth');
```

Рисунок 4.21 – Маршрути, що відповідають за функціонування CRUD

Для виведення списку об’єктів БД у шаблоні news.index використано бібліотеку bootstrap, дані виведено у звичайні таблиці (рис. 4.22).

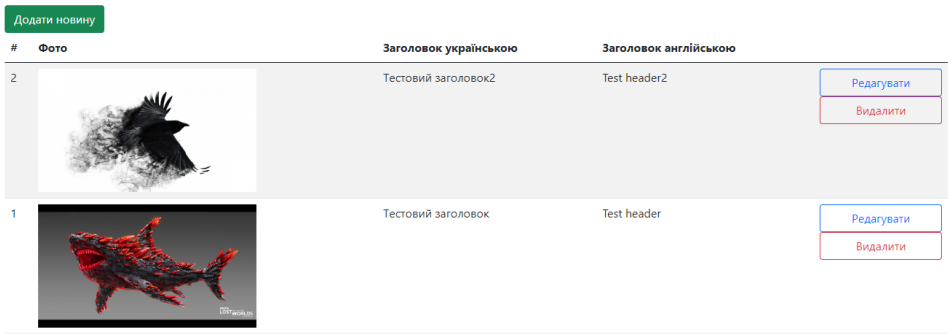

Рисунок 4.22 – Перегляд списку новин з панелі адміністратора

У додатку А вміст index.blade.php.

Для видалення створено модальне вікно, для підтвердження видалення запису (рис. 4.23).

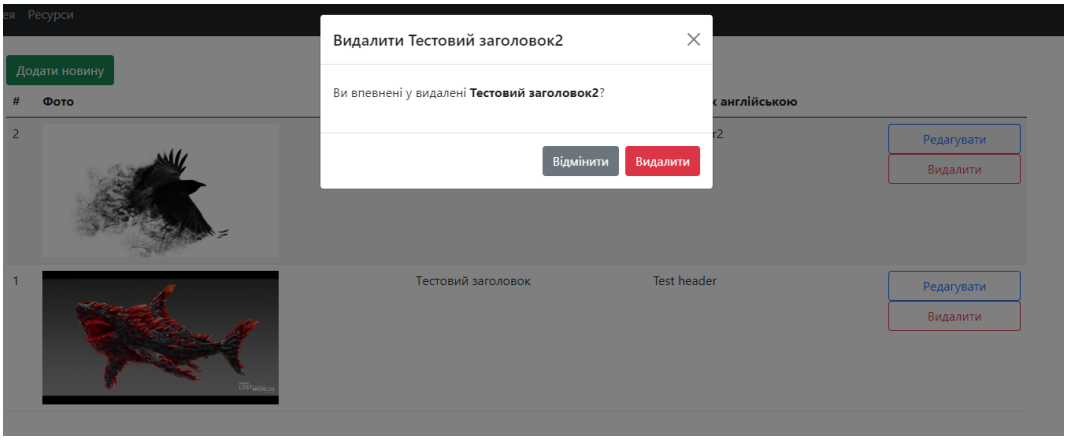

Рисунок 4.23 – Модальне вікно для підтвердження видалення

Таким же чином реалізовано CRUD для інших моделей. Більш детально можна розглянути реалізацію у github репозиторії [12].

Для даного вебзастосунку не було реалізовано реєстрацію для користувачів, так як замовнику вона не потрібна. Потрібна лише авторизація, для одного користувача.

4.3 Верстка основних сторінок

Зроблено верстку сторінок, що відображаються для для відвідувачів сайту, а саме:

– домашня сторінка;
– про проєкт;
– новини;
– галерея;
– ресурси.

Для усіх цих сторінок створено layout, у якому створено стандартну розмітку HTML, до неї додано секції, у яких буде міститься контент, що розширить цю схему. Код схеми можна розглянути у додатку Б.

Порядок розміщення елементів вказував замовник тому над панеллю навігації знаходяться логотипи партнерів. Для реалізації адаптивного меню використано

bootstrap. На рис. 4.24 зображено стиль меню навігації, що переписує стандартний стиль bootstrap. Меню на головній сторінці, відображається майже прозорим, а розмір тексту змінюється в залежності від розміру екрану.

```
.menu {
    z-index: 9999;
    width: 100%;
}

@media (max-width: 990px) {
    .nav-text {
        font-size: 30px;
    }

    .menu {
        background-color: rgba(0, 0, 0, 0.7);
    }
}

@media (min-width: 1200px) {
    .menu .container-fluid .collapse ul li a {
        margin: 0 calc(10px + 5 * ((100vw - 320px) / (1350 - 320)));
    }
}

@media (min-width: 990px) {
    .nav-text {
        font-size: calc(10px + 15 * ((100vw - 320px) / (1350 - 320)));
    }

    .menu {
        background-color: rgba(0, 0, 0, 0.2);
    }
}

.nav-text {
    font-size: calc(10px + 15 * ((100vw - 320px) / (1350 - 320)));
}
```

Рисунок 4.24 – Стиль меню

На рис. 4.25 зображено меню на головній сторінці.

Рисунок 4.25 – Меню навігації на головній сторінці

Для інших сторінок стиль зображено на рис. 4.26 для них меню має зображення фону.

```css
.menu {
    background: url("../images/background.jpg") no-repeat #ECE7EA;
    background-color: rgba(0, 0, 0, 0.2);
    width: 100%;
    z-index:1;
}
```

Рисунок 4.26 – Стиль меню для інших сторінок

Було створено банер для головної сторінки. Зображення було запропоновано замовником. Стиль банеру зображено на рис. 4.27. Над зображенням розміщено назву проєкту, назву університету та їх логотипи (рис. 4.28).

```css
.mainBackground {
    height: 100vh;
    width: 100%;
    object-fit: fill;
}

.logo-title1 {
    width: 100px;
    height: 100px;
    float: left;
    border: 0 black solid;
    border-radius: 10px;
}

.textBlock h2 {
    letter-spacing: 3px;
    line-height: 1.65;
    margin: 0;
    padding: 0;
    font-weight: bolder;
    font-size: calc(20px + 50 * ((100vw - 320px) / (1350 - 320)));
    text-transform: uppercase;
    text-align: left;
    display: inline-block;
    text-shadow: calc(3px + 2 * ((100vw - 320px) / (1280 - 320))) calc(3px + 2 * ((100vw - 320px) / (1280 - 320))) rgba(0, 0, 0, 0.7);
}

.textBlock {
    align-items: center;
    color: #cbe065;
    min-width: 200px;
    display: flex;
    position: absolute;
    top: 20vh;
    left: 2vw;
    background-color: rgba(0, 0, 0, 0);
}
```

Рисунок 4.27 – Стиль банеру

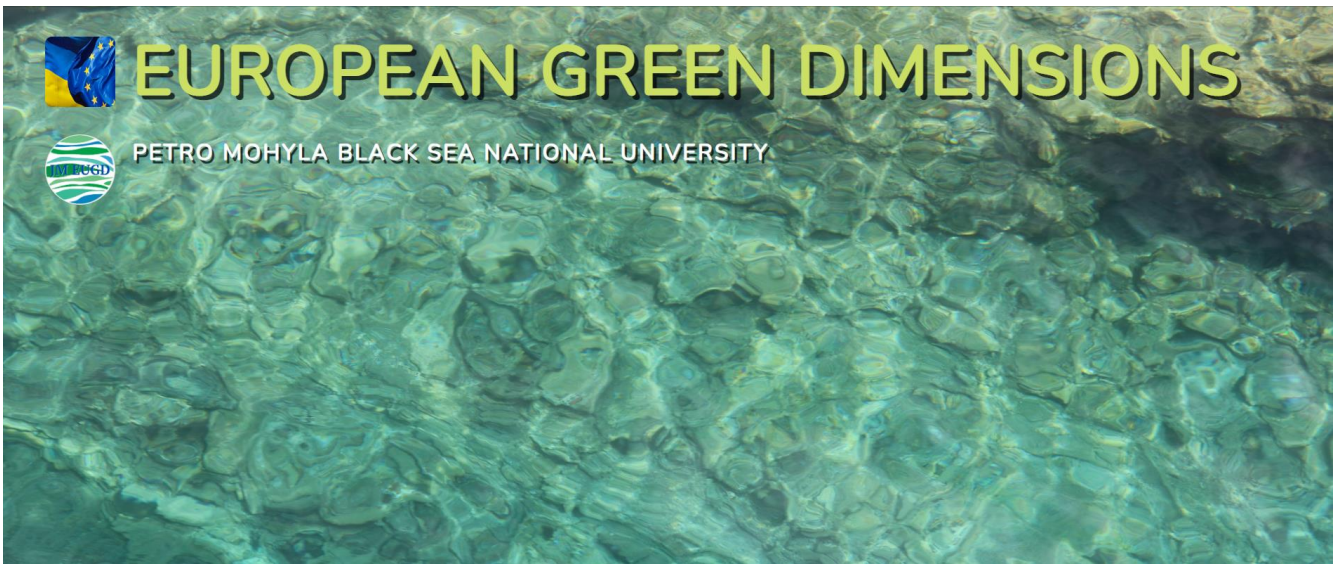

Рисунок 4.28 – Вигляд банеру

Далі на домашній сторінці розміщено «слайдер» з учасниками проєкту. Для його створення було використано бібліотеку owl carousel, на рис. 4.29 зображено параметри його налаштування.

```
$(document).ready(function() :void {
    $(".owl-carousel1").owlCarousel({
        autoplay:false,
        loop: true,
        center: true,
        margin: 0,
        // responsiveClass: true,
        items: 1,
        nav: true,
    });
});
```

Рисунок 4.29 – Налаштування слайдеру

Потім розміщено галерею. За замовчуванням завантажено останні чотири зображення, за бажанням користувач може довантажувати по 4 зображення за одне натискання кнопки «показати більше».

Для реалізації довантаження у контролері перед викликом шаблону масив з зображення перетворюється до формату JSON (рис. 4.30).

```php
public function index()
{
    $data['news']=News::orderby('created_at',"desc")->take(5)->get();
    $data['team']=Team::all();
    $data['images']=json_encode(Gallery::orderby('created_at','desc')->get());
    return view( view: 'home',$data);
}
```

Рисунок 4.30 – Виклик шаблону home та передача йому параметрів

Потім дані про зображення передаються у js-скрипт, що потім будуватиме html-розмітку (рис. 4.31) та на рис. 4.32 зображено саму секцію з галереєю.

```
<script>
    var len = 4;
    var maxLen = 0;
    var array = JSON.parse('<?php echo $images; ?>');
    maxLen = array.length - 4;

    function getPhotos() {
        var newHTML = [];
        for (var i = 0; i < len; i++) {
            newHTML.push('<div class="tile"><a data-fancybox="images" href="' + array[i].path + '"><img class="tile-image"
        }
        $("#photos").html(newHTML.join(""));
    }

    getPhotos();
    $('#viewmore').on('click', function () {
        if (maxLen - 4 < 0) {
            len = array.length;
            this.hidden = true;
        } else {
            len += 4;
            maxLen -= 4;
        }
        getPhotos();
    })
</script>
</body>
</html>
```

Рисунок 4.31 – Скрипт для реалізації довантаження зображень

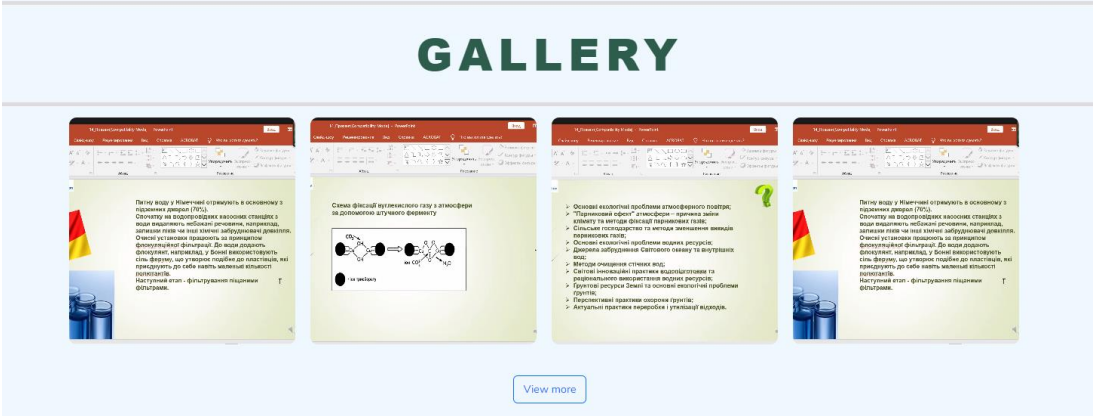

Рисунок 4.32 – Галерея на головній сторінці

Потім розміщені посилання на соціальні мережі проєкту та підвал сайту (рис. 4.33).

Рисунок 4.33 – Соціальні мережі та футтер сайту

Було додано статичну сторінку з інформацією про проєкт (рис. 4.34).

Рисунок 4.34 – Сторінка з інформацією про проєкт

Новини відображаються користувачу у виді плиток, при натисканні вона перенаправляє користувача на сторінку для перегляду новини. На плитках є обмеження на кількість рядків у назві, все що не вміщається обрізається та вкінці ставиться три крапки, для реалізації був використаний css, а саме завдяки параметрі line-clamp, overflow, text-overflow (рис. 4.35).

```css
.card__body h4 {
    font-size: 1.5rem;
    text-transform: capitalize;
    overflow: hidden;
    text-overflow: ellipsis;
    display: -webkit-box;
    -webkit-line-clamp: 4;
    line-clamp: 4;
    -webkit-box-orient: vertical;
    text-align: justify;
}
```

Рисунок 4.35 – Стиль для заголовку новини

На рис. 4.36 зображено новини, що відсортовані від нових до старих.

Рисунок 4.36 – Сторінка з новинами

При збережені вмісту новини tinyMCE увесь введений контент зберігається у вигляді html-розмітка, а зображення кодується у форматі base64. Для коректного відображення контенту та згенерованих html-тегів потрібно виводити через вираження {!! !!}. Таким чином контент не буде екранованим (рис. 4.37).

```
@extends('layouts.app')
@section('title',"post")
@section('content')
    <div class="container">
        <h1 style="...">{{App::isLocale('uk')?$post→titleUkr:$post→titleEng}}</h1>
        @if(App::isLocale('uk'))
            {!! $post→contentUkr !!}
        @else
            {!! $post→contentEng !!}
        @endif
    </div>
@endsection()
```

Рисунок 4.37 – Шаблон для виведення повної новини

Сторінка з галереєю працює з таким же скиптом для довантаження зображень. Також, за допомогою fancybox виконується збільшення зображення для зручного перегляду (рис. 4.38).

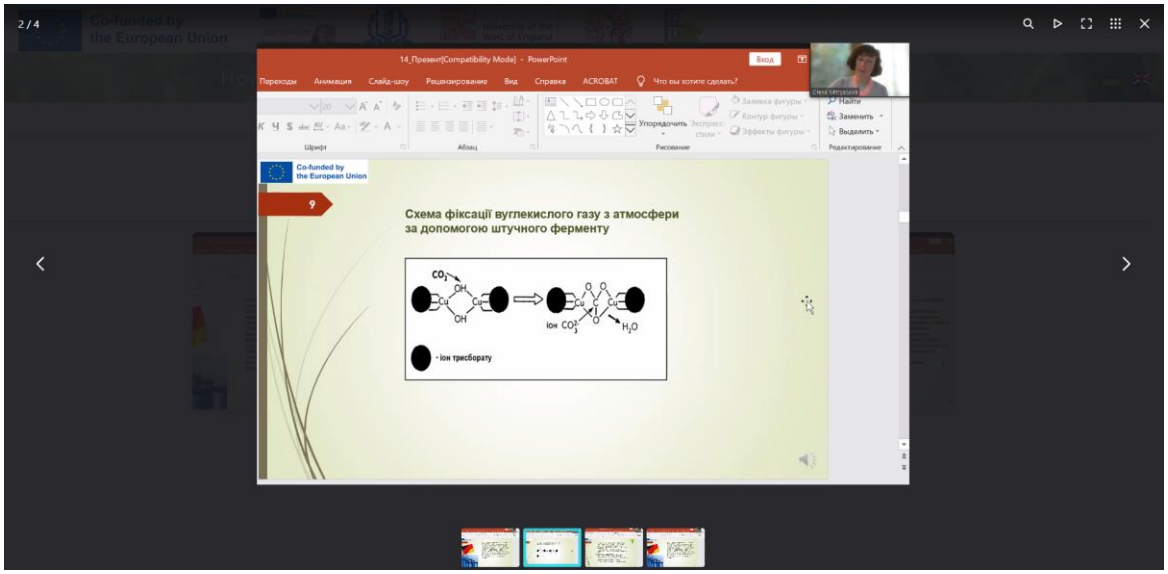

Рисунок 4.38 – Збільшення зображення

4.4 Локалізація

Для забезпечення двомовності вебзастосунку, було створено директорію у якій містяться файли локалізації. У файлах зберігається масив, що має ключ та значення, у значені зберігається текст, що потрібно відобразити за ключем. Для перекладу стандартних слів для входу, реєстрації, валідації використано файли з пакету локалізації, що знаходиться у git-репозиторії [12]. Було додано файл message.php, у якому зберігаються власноруч створені фрази та тексти, що відображається у інтерфейсі програмного забезпечення. На рис. 4.39 зображено файл для відображення контенту англійською, а на рис. 4.40 українською.

```php
<?php
return [
    "Home"⇒"Home",
    "Our Team"⇒"Our Team",
    "About"⇒"About",
    "News"⇒"News",
    "Gallery"⇒"Gallery",
    "Resources"⇒"Resources",
    "Bsnu"⇒"Petro Mohyla Black Sea National University",
    "View more"⇒"View more",
    "Keep in touch"⇒"Keep in touch",
    "footer_coordinator"⇒"Coordinator: prof. Olena Mitryasova",
    "footer_location"⇒"Mykolaiv, 68 Desantnykov Street 10, 54003",
    "footer_copyrights"⇒"This project has been funded with support from the European Commission. This publication [comm
    "footer_last_news"⇒"Latest news",
    "footer_contacts"⇒"Contacts",
    "title"⇒"EUROPEAN GREEN DIMENSIONS",
    ];
```

Рисунок 4.39 – Масив з текстом для локалізації англійською

```php
<?php
return [
    "Home"⇒"Головна",
    "Our Team"⇒"Команда",
    "About"⇒"Про проєкт",
    "News"⇒"Новини",
    "Gallery"⇒"Галерея",
    "Resources"⇒"Ресурси",
    "Bsnu"⇒"Чорноморський національний університет імені Петра Могили",
    "View more"⇒"Переглянути більше",
    "Keep in touch"⇒"Будьте на зв'язку",
    "footer_coordinator"⇒"Координатор: професор Олена Мітрясова",
    "footer_location"⇒"м.Миколаїв, вул. 68 Десантників, 10, 54003",
    "footer_copyrights"⇒"Цей проект фінансується за підтримки Європейської Комісії. Ця публікація [повідомлення] відобр
    "footer_last_news"⇒"Останні новини",
    "footer_contacts"⇒"Контактна інформація",
    "title"⇒"Європейські зелені виміри",
    ];
```

Рисунок 4.40 – Масив з текстом для локалізації українською

Для вказання фреймворку, що фразу потрібно локалізувати, використовується директива @lang та в дужках та в лапках вказується файл у якому шукати слово, потім крапка та ключ за яким шукати.

Після закінчення роботи вебзастосунок було протестовано, усі виявлені помилки були виправлені. Готовий проєкт було завантажено на сервер та на зараз є доступним та активно наповнюється контентом [14]. Також виконується підтримка застосунку та, за бажанням замовника, відбуваються зміни на сторінці.

Висновки до розділу 4

У четвертому розділі кваліфікаційної роботи бакалавра, описано виконання розробки програмного забезпечення. Реалізовано CRUD-функції з використанням контролерів та виконання обробки виведено у шаблони. Виконано верстку сторінок, створено динамічну бібліотеку за допомогою js та створено адміністративну панель для керуванням контентом вебзастосунку. Реалізовано двомовність сайту.

ВИСНОВКИ

При виконанні кваліфікаційної роботи бакалавра було розроблено вебзастосунок задля підвищення обізнаності більш широкої аудиторії про природні ресурси, якість довкілля та зміну клімату.

Для досягнення поставленої мети було виконано наступні завдання:

- досліджено предметну область;
- проаналізовано існуючі аналоги;
- сформовано специфікацію вимог до ПЗ;
- визначено архітектуру для проєктування;
- змодельовано та спроєктовано ПЗ;
- розроблено back-end частину;
- розроблено front-end частину;
- проведено тестування вебзастосунку.

Завдяки дослідженню предметної області визначено актуальність знань та інформованості людей у екологічних проблемах планети. За допомогою аналізу створено специфікацію вимог, для створюваного вебзастосунку, щоб зробити його максимально доступним для користувача та щоб він мав увесь необхідний функціонал. За створеною специфікацією вимог, створено технічне завдання.

При моделюванні та проєктуванні, створено 7 різних типів UML-діаграм, що допомогли при розробці програмного забезпечення. Дотримуючись технічного завдання було розроблено Laravel-вебзастосунок. Підчас тестування виявлені помилки були виправлені. Результатом виконання кваліфікаційної роботи бакалавра є вебзастосунок відображення розвитку екологічного проєкту програми Жана Моне, що можна переглянути за посиланням [13].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ecology. URL: https://watersecurityproject.chmnu.edu.ua/ (дата звернення: 31.03.2023).

2. European Environment Agency’s home page. URL: https://www.eea.europa.eu/en (Last accessed: 31.03.2023).

3. U.S. Environmental Protection Agency | US EPA. URL: https://www.epa.gov/ (Last accessed: 31.03.2023).

4. UML Use Case Diagram Tutorial | Lucidchart. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (Last accessed: 31.03.2023).

5. J. W. S. Whitla, *Visualising Business Transformation*. Routledge, 2022.

6. How to Model MVC Framework with UML Sequence Diagram?. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/how-to-model-mvc-with-uml-sequence-diagram/ (Last accessed: 31.03.2023).

7. Диаграмма классов (class diagram) PHP. URL: https://intellect.icu/diagramma-klassov-class-diagram-php-primenyaemye-v-oop-4835 (дата звернення: 31.03.2023).

8. B. Unhelkar, *Software Engineering with UML*. Auerbach Publications, 2017.

9. What is Component Diagram?. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/ (Last accessed: 31.03.2023).

10. What is Package Diagram?. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/ (Last accessed: 31.03.2023).

11. What is Deployment Diagram?. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/ (Last accessed: 31.03.2023).

12. Laravel-Lang/lang. URL: https://github.com/Laravel-Lang/lang (Last accessed: 24.11.2022).

13. BSNU_ecology. URL: https://github.com/Ilona-Poltavets/BSNU_ecology (Last accessed: 06.04.2023).

14. European Green Dimensions. URL: https://eugreendimensions.chmnu.edu.ua/ (Last accessed: 04.06.2023).

Додаток А

index.blade.php

```
@extends('layouts.admin')
@section('title','News')
@section('content')
    <a class="btn btn-success" href="{{route('news.create')}}">Додати
новину</a>
    <table class="table table-striped">
        <thead>
        <tr>
            <th>#</th>
            <th>Фото</th>
            <th>Заголовок українською</th>
            <th>Заголовок англійською</th>
            <th></th>
        </tr>
        </thead>
        <tbody>
        @foreach($news as $post)
            <tr>
                <td>{{$post->id}}</td>
                <td><img src="{{asset($post->title_image)}}" class="title-
image" alt=""></td>
                <td>{{$post->titleUkr}}</td>
                <td>{{$post->titleEng}}</td>
                <td>
                    <div class="d-grid">
                        <a href="{{route('news.edit', $post->id)}}"
                           class="btn btn-outline-primary">Редагувати</a>
                        <div class="d-grid">
                            <button type="button" class="btn btn-outline-
danger" data-bs-toggle="modal"
                                    data-bs-target="#exampleModal" data-bs-
name="{{$post->titleUkr}}"
                                    data-bs-id="{{$post->id}}">
                                Видалити
                            </button>
                        </div>
                    </div>
                </td>
            </tr>
        @endforeach
        </tbody>
    </table>

    {{$news->links()}}

    <div class="modal fade" id="exampleModal" tabindex="-1" aria-
```

```
labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog">
```

Продовження додатку А

```
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">New
message</h5>
                    <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                </div>
                <div class="modal-body">
                    <div class="mb-3">
                        <p class="col-form-label">Ви впевнені у видалені <b
id="recipient-name"></b>?</p>
                    </div>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Відмінити</button>
                    <form id="form_delete" action="" method="post">
                        @method('delete')
                        @csrf
                        <button type="submit" class="btn btn-
danger">Видалити</button>
                    </form>
                </div>
            </div>
        </div>
    </div>

    <script>
        var exampleModal = document.getElementById('exampleModal')
        exampleModal.addEventListener('show.bs.modal', function (event) {
            var button = event.relatedTarget
            var name = button.getAttribute('data-bs-name')
            var id = button.getAttribute('data-bs-id')

            var modalTitle = exampleModal.querySelector('.modal-title')
            var modalBodyInput = exampleModal.querySelector('.modal-body b')
            var modalForm=exampleModal.querySelector('#form_delete')

            modalTitle.textContent = 'Видалити ' + name
            modalBodyInput.textContent = name
            modalForm.action=window.location.origin+'/news/'+id
        })
    </script>
@endsection
```

Додаток Б

layouts/app.blade.php

```
<!doctype html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>@yield("title")</title>

    <!-- BOOTSTRAP 5 -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
          integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
            integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
            crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
            integrity="sha384-
cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
            crossorigin="anonymous"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.1/jquery.min.js"
            integrity="sha512-
aVKKRRi/Q/YV+4mjoKBsE4x3H+BkegoM/em46NNlCqNTmUYADjBbeNefNxYV7giUp0VxICtqdrbqU7iVaeZNXA=="
            crossorigin="anonymous" referrerpolicy="no-referrer"></script>

    <!-- Fancypps -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/@fancyapps/ui/dist/fancybox.css"/>

    <!-- Fonts -->
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.bunny.net/css?family=Nunito" rel="stylesheet">
    <link rel="stylesheet" href="{{url('css/loader.css')}}">
    <link rel="stylesheet" type="text/css" href="{{url('css/style.css')}}">
</head>
<body onload="myFunction()">
<div id="loader" class="body">
    <div class="loader">
        <span></span>
        <span></span>
        <span></span>
    </div>
</div>
<div style="display:none;" id="myDiv">
    <div class="logos">
        <img class="logo" src="{{asset("images/logos/logo1_v2.png")}}" alt="">
        <img class="logo" src="{{asset("images/logos/logo2.png")}}" alt="">
        <img class="logo" src="{{asset("images/logo_bsnu1.jpg")}}" alt="">
        <img class="logo" src="{{asset("images/logos/logo3.jpg")}}" alt="">
        <img class="logo" src="{{asset("images/logos/logo4.jpg")}}" alt="">
        <img class="logo" src="{{asset("images/logos/logo5.jpg")}}" alt="">
    </div>
```

```
    <nav class="navbar navbar-expand-lg navbar-dark menu position-absolute top-20">
        <div class="container-fluid">
```

Продовження додатку Б

```
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNavAltMarkup"
                    aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse"></div>
            <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
                <ul class="nav navbar-nav nav-text">
                    <li class="nav-item mx-2">
                        <a class="nav-link"
href="{{route("home")}}">@lang("messages.Home")</a>
                    </li>
                    <li class="nav-item mx-2">
                        <a class="nav-link"
href="{{route("about")}}">@lang("messages.About")</a>
                    </li>
                    <li class="nav-item mx-2">
                        <a class="nav-link" href="{{route("home")}}#team">@lang("messages.Our
Team")</a>
                    </li>
                    <li class="nav-item mx-2">
                        <a class="nav-link"
href="{{route('all_news')}}">@lang("messages.News")</a>
                    </li>
                    <li class="nav-item mx-2">
                        <a class="nav-link"
href="{{route('gallery')}}">@lang("messages.Gallery")</a>
                    </li>
                    <li class="nav-item mx-2">
                        <a class="nav-link"
href="{{route('resources',"pdf")}}">@lang("messages.Resources")</a>
                    </li>
                </ul>

                <ul class="navbar-nav ms-auto">
                    <a href="{{route('setlocale',['lang'=>'uk'])}}"><img class="flag
{{(App::isLocale('uk') ? 'active' : '')}}" alt="ukraine"
src="{{asset('images/langs/uk.png')}}"/></a>
                    <a href="{{route('setlocale',['lang'=>'en'])}}"><img class="flag
{{(App::isLocale('en') ? 'active' : '')}}" alt="english"
src="{{asset('images/langs/en.png')}}"/></a>
                </ul>
            </div>
        </div>
    </nav>

    <div class="py-4 main">
        @yield('content')
    </div>

    <hr>
    <div class="footer" style="clear: left">
        <div class="container-fluid">
            <div class="row">
                <div class="col">
                    <b>Copyrigths</b>
                    <p style="text-align: justify">@lang("messages.footer_copyrights")</p>
                </div>
```

```
                <div class="col">
                    <b>@lang("messages.footer_last_news")</b>
                    <ul class="">
```

Продовження додатку Б

```
                        @foreach($news as $post)
                            <li><a style="text-decoration: none;color:black"
                                   href="{{route("news.show",$post-
>id)}}">{{App::isLocale("uk")?$post->titleUkr:$post->titleEng}}</a>
                            </li>
                        @endforeach
                    </ul>
                </div>
                <div class="col">
                    <b>@lang("messages.footer_contacts")</b>
                    <ul class="list">
                        <li>@lang("messages.footer_coordinator")</li>
                        <li><i class="fa-solid fa-phone"></i> +38(095)2880479</li>
                        <li><i class="fa-solid fa-location-
dot"></i>@lang("messages.footer_location")</li>
                        <li><i class="fa-regular fa-envelope"></i> eco-terra@ukr.net</li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
</div>
<script src="{{asset('js/script.js')}}"></script>
<script src="https://cdn.jsdelivr.net/npm/@fancyapps/ui@4.0/dist/fancybox.umd.js"></script>
</body>
</html>
```