

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Веб-орієнтований інформаційний додаток “Daily Bugle”»

Виконав студент групи ПЗ-19-2
ЛИМАРЬ Дмитро Андрійович

Керівник роботи:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
КРАВЧЕНКО Світлана Миколаївна

Житомир – 2023

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

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Лимарь Дмитро Андрійович.**

Керівник роботи: **Локтікова Тамара Миколаївна.**

Тема роботи: **«Веб-орієнтований інформаційний додаток “Daily Bugle”»,**
затверджена Наказом закладу вищої освіти від **«14» лютого 2023 р., №70с.**

Вихідні дані для роботи (зазначається предмет і об'єкт дослідження): об'єктом дослідження є інформаційні технології для розробки додатка. Предметом дослідження є використання веб-технологій для розробки додатка.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	14.02.2023р.	14.02.2023р.
2	Локтікова Т.М.	06.03.2023р.	06.03.2023р.
3	Локтікова Т.М.	05.05.2023р.	05.05.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	14 лютого 2023 – 05 березня 2023	Виконано
2	Проектування структури системи	06 березня 2023 – 12 квітня 2023	Виконано
3	Написання програмного коду	13 квітня 2023 – 04 травня 2023	Виконано
4	Тестування та відлагодження	05 травня 2023 – 04 червня 2023	Виконано
5	Написання пояснювальної записки	04 червня 2023 – 11 червня 2023	Виконано

РЕФЕРАТ

Випускна робота складається з програми інформаційного додатка та пояснювальної записки. Пояснювальна записка до випускної роботи містить 60 сторінок, 33 ілюстрації, 14 таблиць.

Метою роботи є створення інформаційного додатка, в якому реалізовані необхідні для повноцінної роботи модулі та методи. Додаток створювався для допомоги потерпілим від війни.

В роботі визначено завдання для розробки додатка, вивчено аналоги для вибраної теми, визначено патерн для реалізації системи - MVC. Обрано інструменти для розробки, а саме: мова програмування PHP8.1, мова розмітки HTML5, CSS3, база даних MySQL та середовище розробки PhpStorm. Були проаналізовані та обґрунтовані основні методи розробки додатка. Була описана та спроектована база даних. Наведено приклади роботи системи, та базового функціонала, загальна структура додатка.

КЛЮЧОВІ СЛОВА: БЛОГ, ПОСТ, АДМІН, ЧИТАЧ, MVC, ДОПОМОГА, ПРОГРАМИ ДОПОМОГИ.

					ІПЗ.КР.Б – 121 – 23 – ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Лимарь Д.А.			Веб-орієнтовний інформаційний додаток "Daily Bugle". Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник		Локтікова Т.М.					4	60
						Житомирська політехніка, група ІПЗ-19-2		
н. контр.								
Зав. каф.		Морозов А.В.						

ABSTRACT

The graduation thesis consists of a program, an informational appendix and an explanatory note. The explanatory note to the final thesis contains 60 pages, 33 illustrations, 14 tables.

The purpose of the work is to create an information application in which the modules and methods necessary for full-fledged work are implemented. The application was created to help war victims.

In the work, the task for the development of the application is defined, analogues for the selected topic are studied, and the pattern for the implementation of the system - MVC - is defined. The following development tools were selected: PHP8.1 programming language, HTML5 markup language, CSS3, MySQL database and PhpStorm development environment. The main methods of application development were analyzed and substantiated. The database was described and designed. Examples of system operation and basic functionality, the general structure of the application are given.

KEYWORDS: BLOG, POST, ADMIN, READER, MVC, HELP, HELP PROGRAMS.

					ИП3.КР.Б – 121 – 23 – П3	
						5

ЗМІСТ

ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНФОРМАЦІЙНОГО ДОДАТКА	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Вибір архітектури інформаційного додатка.....	13
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	17
Висновки до розділу 1	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ІНФОРМАЦІЙНОГО ДОДАТКА ..	20
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	20
2.2 Розробка бази даних системи.....	26
2.3 Проектування та реалізація алгоритмів роботи системи	31
2.4 Реалізація функціоналу інформаційного додатка.....	33
Висновки до розділу 2	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА	42
3.1 Порядок встановлення та налаштування системи.....	42
3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з веб- орієнтованим інформаційним додатком “Daily Bugle”	43
3.3 Тестування роботи програмного комплексу	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	55
ДОДАТКИ.....	57
ДОДАТОК А.....	58

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

L10 – Laravel 10 версії.

БД – база даних.

CRUD – create, read, update, delete (з англ. створити, прочитати, оновити, видалити).

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

MVP – Model–View–Presenter (з англ. Модель–Представлення–Презентер).

ООП - об'єктно-орієнтоване програмування.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						7

ВСТУП

Актуальність теми. В теперішніх реаліях велика кількість людей постраждали від військових дій росії. Багато людей та тварин потребують допомоги різного характеру, проте не мають змоги отримати та звернутись із проханням про цю допомогу. У більшості варіантів, люди повинні чекати занадто довго, щоб отримати будь-яку допомогу. Це являє собою проблему таких платформ через їх несвоєчасність.

Тож, обрана тема інформаційного додатка може мати велику вагу у розв'язання питань з допомогою, як звичайним людям, які опинились у скрутній ситуації, так і допомогою більш глобальною.

Метою випускною роботи є проектування архітектури, реалізація комплексу алгоритмів для розробки інформаційного додатка спрямованого на швидку та своєчасну допомогу тим, хто цього потребує та чекає. Виконана випускна робота буде результатом створення інформаційного додатка. Для виконання роботи необхідно виконати даний перелік задач:

Встановлена мета обумовлює наступні завдання:

- аналіз можливих задач додатка;
- вивчення аналогів;
- визначення структури та алгоритму додатка;
- вибір технічних засобів розробки;
- розробка додатка;
- опис роботи додатка;
- тестування інформаційного додатка.

Об'єктом дослідження є інформаційні технології для розробки додатка.

Предметом дослідження є використання веб-технології для розробки.

Розроблений додаток має базовий функціонал для розв'язання поставлених задач. В майбутньому можливо покращити та оновити наявний функціонал.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ІНФОРМАЦІЙНОГО ДОДАТКА

1.1 Постановка задачі

Основною метою реалізації додатка для розповсюдження потреб постраждалих була насамперед допомога. Велика кількість людей які втратили домівки, рідних, все разом. Ці люди потребують будь-якої допомоги. Даний додаток зможе допомогти в цьому.

У роботі необхідно реалізувати додаток, який зможе поширювати серед людей загальні програми для допомоги та представляти прохання про допомогу від звичайних людей. Для реалізації поставленого завдання, основними етапами є:

1. Розробити дизайн користувацького інтерфейсу.

1.1. Створити загальну концепцію дизайну користувацького інтерфейсу.

1.1.1. Головна сторінка.

1.1.2. Вхід, реєстрація.

1.1.3. Відображення програм допомоги.

1.1.4. Відображення постів про допомогу.

1.1.5. Персональний кабінет користувача.

1.1.6. Сторінки для - Збережені, Коментарі та Допомога.

1.2. Створити загальну концепцію дизайну адміністративного інтерфейсу.

1.2.1. Головна сторінка адміністративного кабінету.

1.2.2. Сторінки CRUD для (Користувачів, статей, тегів, категорій).

2. Проєктування та розробка основних модулів.

2.1. Модуль керування користувачами.

2.2. Модуль керування статтями.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						9

2.3. Модуль керування тегами.

2.4. Модуль керування категоріями.

2.5. Модуль керування допомогою.

3. Інтегрування сторонніх сервісів у додаток.

3.1. Система для посилення листів на e-mail (mailtrap).

1.2 Аналіз аналогів програмного продукту

Інформаційний додаток повинен забезпечити зручність створення програм підтримки та закликів про допомогу їх відображення та сприймання.

Для розгляду аналогів були відібрані максимально схожі та популярні наступні додатки: peopleinneed.net, fauna-flora.org, clovekvhrozeni.sk. Їх порівняльна характеристика наведена у табл. 1.1.

clovekvohrozeni.sk. - онлайн платформа для допомоги українським постраждалим. Також для охочих є можливість допомогти за загальними програмами.

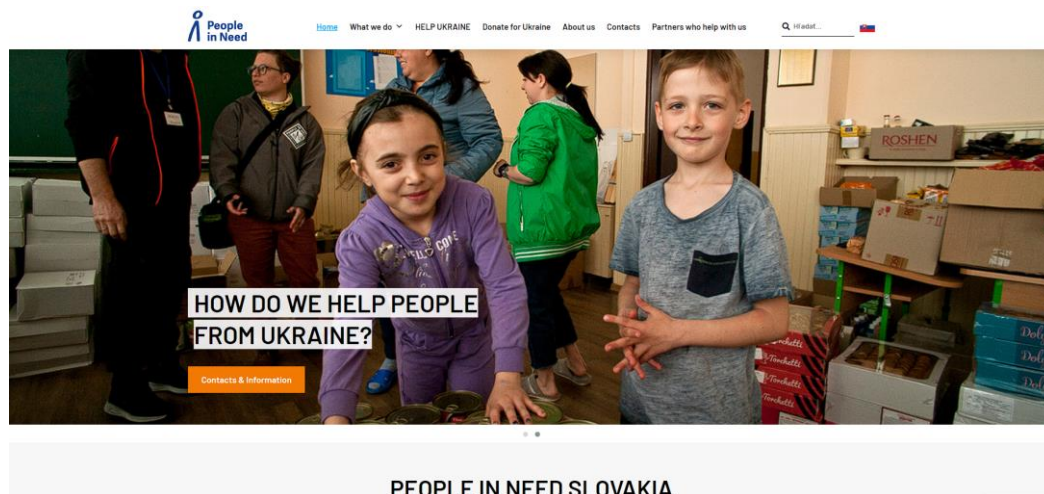


Рисунок 1.1 - Головна сторінка додатка clovekvhrozeni.sk.

Основними перевагами clovekvhrozeni є:

1. Різні варіанти допомоги;
2. допомога українцям;
3. простий користувацький дизайн.

Недоліками є:

1. Відсутність допомоги людині напряду;
2. відсутність персонального кабінету;
3. відсутня можливість подати заявку на допомогу.

peopleinneed.net. - додаток, який має за мету збір допомоги для українського населення. Проте загальна мета підтримка програм з допомоги людям у всьому світі.

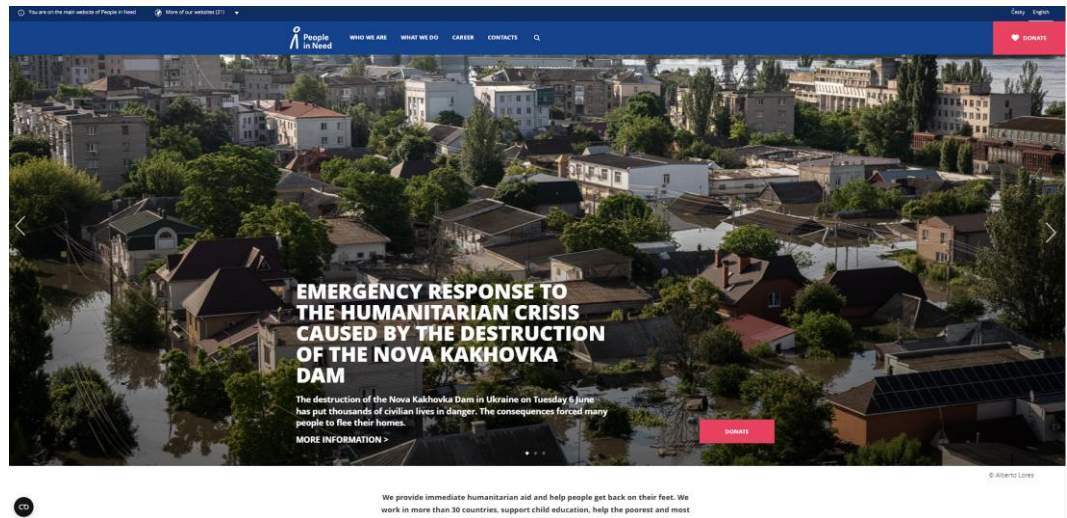


Рисунок 1.2 - Головна сторінка додатка peopleinneed.net.

Основними перевагами peopleinneed є:

1. Допомога українцям та всім постраждалим;
2. дизайн не викликає труднощів;
3. можливість донатів з карти.

Недоліками є:

1. Відсутність допомогти людині напряду;
2. відсутність персонального кабінету.

fauna-flora.org. - Додаток який має на меті допомогу тваринам. Повністю спеціалізується на такій допомозі. Має велику кількість програм по допомозі різним тваринам.

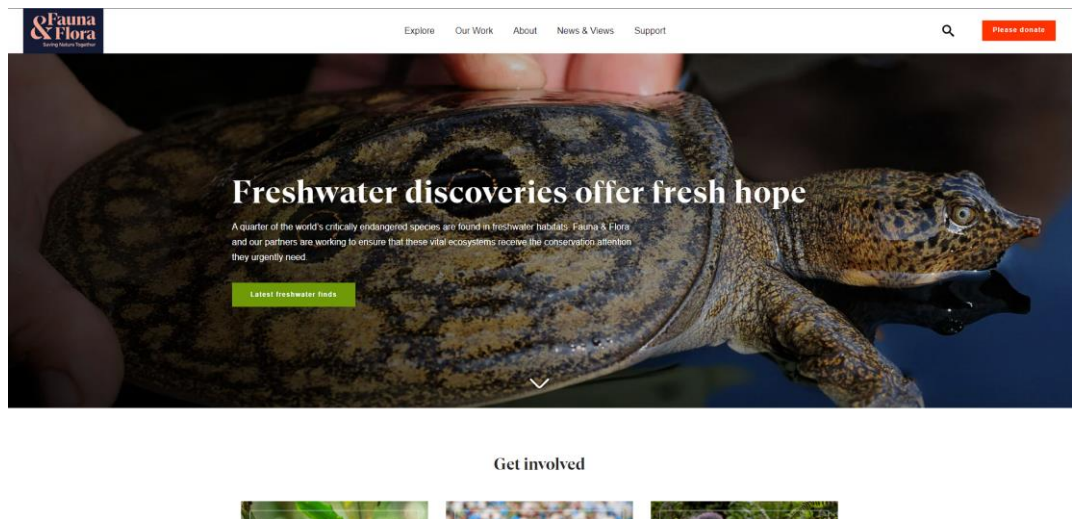


Рисунок 1.3 - Головна сторінка додатка fauna-flora.org.

Основними перевагами fauna-flora є:

1. Великий вибір варіантів допомоги;
2. можливість переглянути інформацію про необхідну допомогу;
3. можливість доната с карти.

Недоліками є:

1. Відсутність персонального кабінету.

Таблиця 1.1

Характеристика	clovekvohr rozeni	peopleinneed	fauna-flora	Власний додаток
Сума допомоги	20 / 50 / 125 євро	100 / 200 / 500 / 1000 євро	10 / 50 / 100 доларів	Власна сума
Допомога напряму людині	Ні	Ні	Ні	Так
Персональн ий кабінет	Ні	Ні	Ні	Так
Складність інтерфейсу	3/5	2/5	1/5	2/5

Переглянувши аналоги інформаційних додатків можна помітити, що майже всі сайти мають схожий базовий функціонал, що дозволяє взаємодіяти з автором та читачами.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						12

Судячи з представлених аналогів основними функція системи повинні бути: приємний дизайн, стрічка статей, можливість взаємодіяти з ними та персональний кабінет.

1.3 Вибір архітектури інформаційного додатка

До варіантів вибору архітектури було взято такі патерни: MVC, MVVM, MVP. Серед представлених патернів було обрано - MVC. Розглянемо особливості та відмінності кожного з представлених патернів.

MVP (Model-View-Presenter) є патерном програмування, який використовується для розділення логіки додатка на три компоненти: модель (дані), представлення (інтерфейс) та презентер (логіка) [2]. MVP базується на концепції, де модель і представлення не взаємодіють безпосередньо, а залежать від презентера для обміну даними та керування подіями.

Використання MVP дозволяє забезпечити відокремлення логіки додатка від його графічного інтерфейсу. Модель відповідає за управління даними, представлення відображає дані на екрані, а презентер обробляє події, такі як користувацькі дії. Презентер служить посередником між моделлю та представленням, забезпечуючи логіку взаємодії між ними.

Переваги MVP включають зменшення залежностей між компонентами, полегшення тестування та покращення розширюваності. Оскільки логіка додатка розділена на окремі компоненти, кожен з них можна тестувати окремо. Крім того, зміни в одному компоненті майже не впливають на інші, що полегшує розширення функціонала додатка.

Недоліком MVP може бути збільшена складність у порівнянні з іншими патернами. Перекладання взаємодії між компонентами на презентер може збільшити кількість коду і потребувати додаткової уваги при розробці. Крім того, для малих проєктів MVP може бути зайвим та вважатися зайвим надмірним шаром абстракції.

Загалом, MVP є потужним патерном програмування, який дозволяє відокремити логіку додатка від його інтерфейсу, полегшує тестування та підвищує розширюваність. Однак, він вимагає додаткових зусиль для його ефективного використання та може бути переважним для складних проєктів.

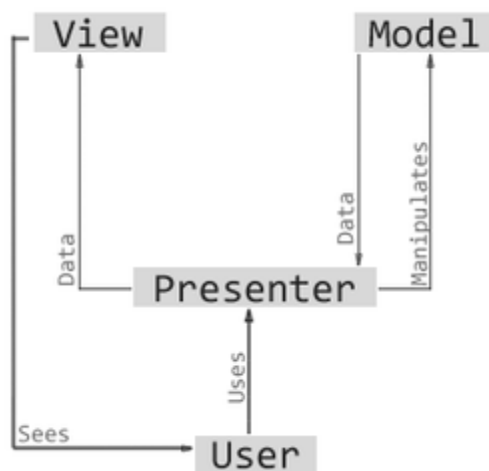


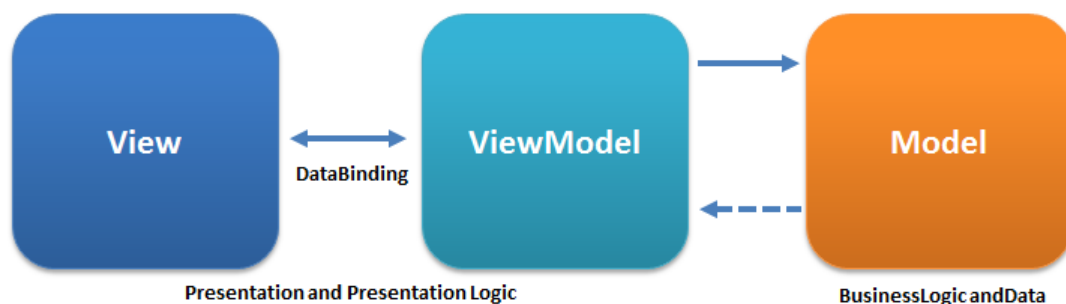
Рисунок 1.4 - Узагальнена структура MVP

MVVM (Model-View-ViewModel) - є патерном програмування, що дозволяє розділити логіку додатка на три компоненти: модель (дані), представлення (інтерфейс) та модель представлення (посередник) [3]. У MVVM модель представляє дані, представлення відображає ці дані на екрані, а модель представлення забезпечує комунікацію між моделлю та представленням.

Основна перевага MVVM - це відокремлення логіки додатка від його графічного інтерфейсу. Це сприяє більшій гнучкості та покращує розширюваність коду. Крім того, MVVM сприяє легкості тестування, оскільки модель представлення можна легко протестувати незалежно від інтерфейсу.

Недоліком MVVM може бути певна складність, особливо для початківців. Додаткова робота потрібна для налаштування зв'язків між моделями, представленнями та моделями представлення. Крім того, для простих проєктів MVVM може бути зайвим, оскільки він додає додатковий рівень абстракції.

Загалом, MVVM є потужним патерном програмування, який дозволяє забезпечити кращу організацію та керованість великих проєктів, але вимагає розуміння та додаткової праці для його ефективного використання.



Рисинук 1.5 - Узагальнена структура MVVM

MVC (Model-View-Controller) є популярним патерном програмування, який дозволяє організувати структуру додатка на три компоненти: модель (дані), представлення (інтерфейс) та контролер (логіка) [1]. Кожен компонент виконує свої відповідальності та співпрацює з іншими, забезпечуючи розподіл обов'язків.

Використання MVC дозволяє забезпечити розділення логіки додатка від його графічного інтерфейсу. Модель відповідає за управління даними та бізнес-логікою, представлення відповідає за відображення інформації на екрані, а контролер забезпечує взаємодію між моделлю та представленням.

Однією з переваг MVC є легкість тестування. Оскільки логіка додатка розділена на окремі компоненти, можна легко тестувати кожен з них окремо. Крім того, MVC сприяє полегшенню розширюваності проєкту, оскільки зміни в одному компоненті не впливають на інші.

Недоліком MVC може бути певна складність управління великими проєктами. З ростом розміру проєкту може збільшуватися кількість взаємодій між компонентами, що може ускладнити його розуміння та підтримку.

Загалом, MVC є ефективним патерном програмування, що дозволяє розділити логіку додатка на логічні компоненти, полегшуючи його тестування та розширюваність. Однак, для успішного використання MVC потрібно розуміння його принципів та вміння встановлювати зв'язки між компонентами.

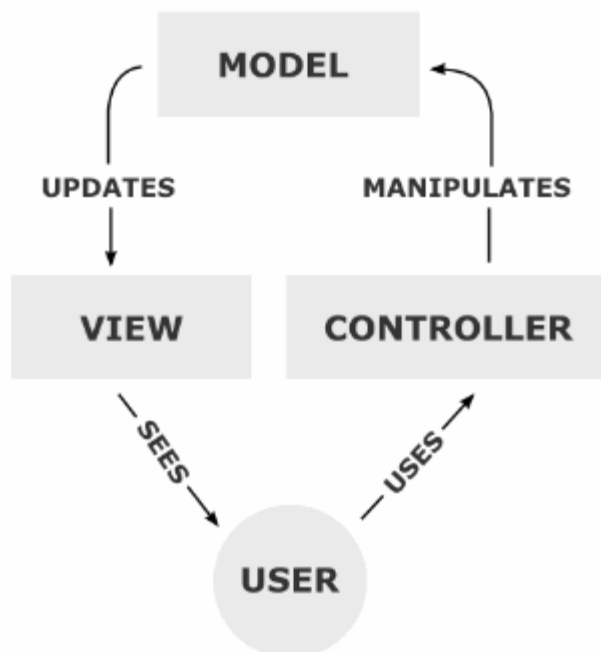


Рисунок 1.6 - Узагальнена архітектура MVC

Вибір MVC для створення блогу є доцільним з кількох причин.

Перш за все, MVC дозволяє чітко відокремити логіку додатка від його представлення. Це сприяє покращенню організації коду і забезпечує більшу читабельність і легкість утримання. Модель відповідає за управління даними додатка, представлення відображає їх на екрані, а контролер здійснює логіку взаємодії між ними.

Крім того, MVC сприяє більшій гнучкості та розширюваності блогу. Кожен компонент можна змінювати окремо, не впливаючи на решту. Це дає можливість легко додавати нові функції, оновлювати наявні та змінювати зовнішній вигляд додатка без значних змін в інших частинах системи.

Нарешті, MVC дозволяє забезпечити більшу тестованість додатка. Оскільки модель, представлення та контролер виконують різні функції та мають чітко визначені обов'язки, їх можна тестувати окремо. Це полегшує процес виявлення та виправлення помилок.

Загалом, вибір MVC для створення інформаційного додатка є раціональним через його здатність до розділення логіки, гнучкості та тестованості. Він допоможе створити чистий, організований та легко розширюваний код для вашого блогу.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для створення вебсайту, доцільно застосувати: мову розмітки гіпертекстових документів HTML, каскадні таблиці стилів CSS, браузерну мову програмування JavaScript, мову програмування PHP, вебсервер Apache з підтримкою PHP, MySQL-бази даних та HTML-редактор

HTML (HyperText Markup Language) - мова розмітка, яка лежить в основні більшості сайтів. Являє собою каркас для розробки подальшого функціонала, відображення вмісту сторінки, тексту, зображень тощо [4].

CSS (Cascading Style Sheets) - файли стилів які використовуються разом з HTML для візуального оформлення вебсторінок, тексту тощо. Особливістю є те, що не може працювати без використання HTML, тобто ці технології працюють тільки разом [5].

JavaScript - мова програмування, яка використовується переважно у браузерних додатках. Являється мовою з ООП і використовується для виконання скриптів які своєю чергою допомагають «оживити» сайт [6].

PHP (Hypertext Preprocessor) - мова схожа до синтаксису мови C. Це скриптова мова яка використовується у веброзробці. Має велику популярність серед розробників і декілька причин цьому є наявність великої кількості вбудованих засобів розробки й додаткових модулів розробки додатків. Актуальна версія - 8.2 [7].

JetBrains PhpStorm - універсальне середовище розробки, але з нахилом на розробку на мові PHP. Середовище, яке допомагає працювати з кодом за допомогою інтелектуального редактора, рефакторингу, підтримки генераторів і поліпшення синтаксису, підтримка фреймворків на PHP, гнучке налаштування плагінів, візуальна стилізація під користувача, редактор мови розмітки (HTML, CSS), слідування за стилями коду (PSR1/PSR2), потужні інструменти для роботи з базами даних [7].

Порівнявши можливі реляційні бази даних (MySQL, PostgreSQL) рішення було прийняте у користь MySQL, тому що MySQL відрізняється високою швидкодією та ефективністю, дозволяючи оптимально обробляти навіть великі обсяги даних [13]. Він підтримує розподілені транзакції та має потужну систему кешування для покращення продуктивності. На додаток, MySQL має велику спільноту користувачів та широку підтримку, що робить його легким у використанні та навчанні. Однак, на ринку існує інша популярна система - PostgreSQL. Вона відома своєю розширюваністю та повнотою функціонала. PostgreSQL надає багато додаткових можливостей таких, як усунення дублікатів, операції з JSON-даними та географічні запити. Ця система також має розширені можливості управління доступом та реалізації ролей користувачів.

Загалом, MySQL відзначається високою продуктивністю та простотою використання, що робить його чудовим вибором для багатьох проєктів. PostgreSQL, зі своїми розширеними можливостями та повнотою функціонала, підходить для вимогливих додатків, які потребують розширеного аналізу даних та гнучкості. Обидві системи мають свої унікальні особливості, і вибір між ними залежить від конкретних потреб проєкту [7].

Розглянувши варіанти технологій було обрано:

- Мову розмітки документів HTML5;
- стилі CSS3;
- мову програмування JavaScript та бібліотеку jQuery;
- фреймворк Laravel 10 (PHP8.1);
- середовище керування базами даних MySQL;
- Windows 10, Linux, MacOS;
- 32-розрядний або 64-розрядний процесор з тактовою частотою 1 ГГц та вище;
- 4 Гбайти ОЗП;
- 24 Гбайти, за можливості більше вільного місця на жорсткому диску;
- комп'ютер користувача виконує роль сервера.

Висновки до розділу 1

Проведений аналіз дозволив визначити базові напрямки для реалізації інформаційного додатка.

Розглянувши аналоги можна помітити, що основними функціями є: головна сторінка зі стрічкою статей, можливість перегляду статей за темою, можливість реєстрації та спілкування всередині додатку

Визначившись з базовою архітектурою, був обраний необхідний та достатній набір програмних продуктів для реалізації поставлених задач.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ІНФОРМАЦІЙНОГО ДОДАТКА

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

Інформаційний додаток має за мету: спрощення та автоматизацію поширення закликів про допомогу від людей, які потребують цього та розповсюдження інформації про глобальні благодійні збори.

Вимоги користувачів.

Гість має доступ до наступних функцій:

1. До інформації з головної сторінки (програми допомоги, інформацію про організацію);
2. до перегляду окремої програми допомоги та інформації про неї (дату створення, к-сть переглядів);
3. до коментарів під окремою програмою допомоги;
4. до перегляду всіх прохань про допомогу та їх детальну інформацію;
5. до реєстрації або авторизації на сайті.

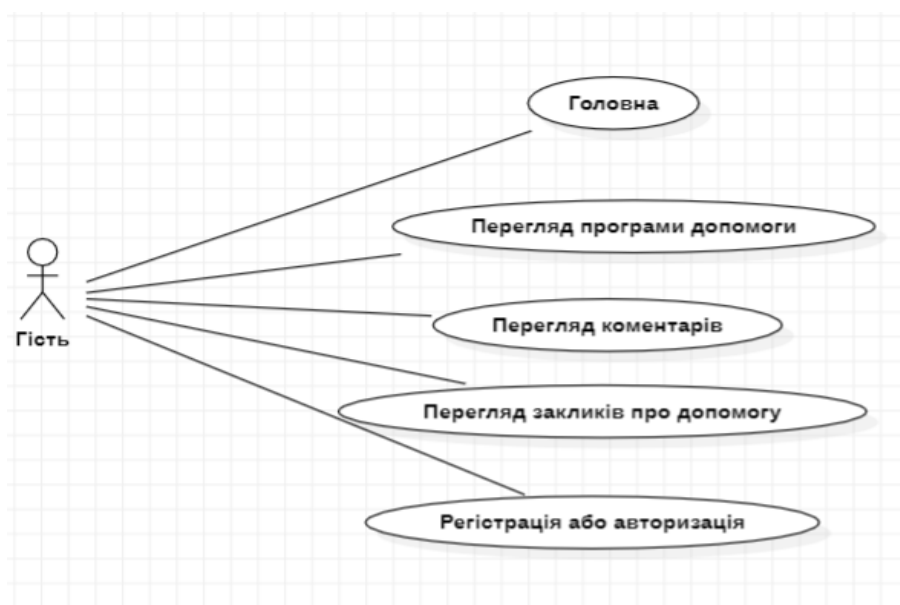


Рисунок 2.1 - Діаграма використання гостя

Авторизований користувач має доступ до наступних функцій:

1. До інформації з головної сторінки(програми допомоги, інформацію про організацію);
2. до перегляду окремої програми допомоги та інформації про неї (дату створення, к-сть переглядів);
3. до коментарів окремою програми допомоги та їх створенню;
4. до можливості збереження програми в особистому кабінеті;
5. до особистого кабінету;
6. до CRUD зі збереженими;
7. до CRUD з коментарями;
8. до CRUD з закликів допомоги;
9. до виходу с облікового запису.

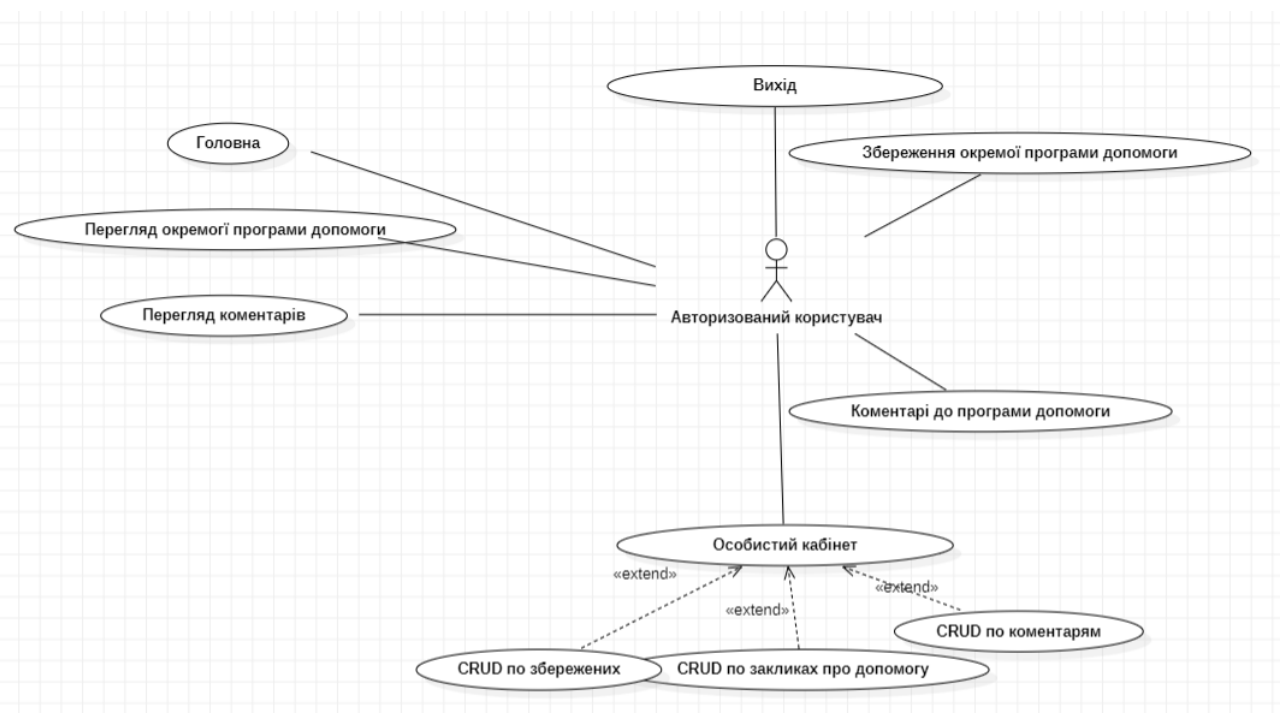


Рисунок 2.2 - Діаграма використання авторизованого користувача

Адміністратор має доступ до наступних функцій:

1. До адмінкабінету;
2. до CRUD з категорій
3. до CRUD з тегів;
4. до CRUD з постів;

5. до CRUD з користувачів;
6. до CRUD з допомоги;
7. до інформації з головної сторінки (програми допомоги, інформацію про організацію);
8. до перегляду окремої програми допомоги та інформації про неї (дату створення, к-сть переглядів);
9. до коментарів окремою програми допомоги та їх створенню;
10. до коментарів окремою програми допомоги та їх створенню.

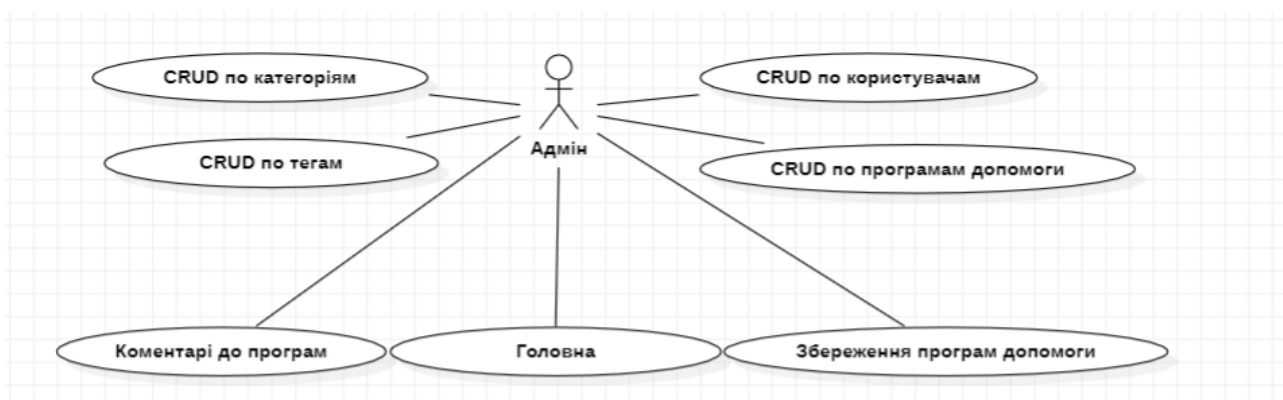


Рисунок 2.3 - Діаграма використання адміністратора

Функціональні вимоги:

1. Авторизація. Для користувачів необхідна можливість реєстрації. Необхідна реалізація можливості додавання користувачів з адміністративного кабінету.
2. Створення, редагування, перегляд та видалення програм допомоги, тегів, категорій та користувачів з адмін кабінету.
3. Система повинна зберігати конфіденційність користувачів.
4. Система повинна коректно та автоматично відображати необхідну інформацію.

Нефункціональні вимоги:

Сприйняття

1. Сприйняття:
 - Час, потрібний для початку роботи - 3 хв.
 - Час відповіді системи для звичайних запитів не повин перевищувати

0.540 мс, а для більш складних запитів - 0.840 мс.

– Інтерфейс повинен бути зрозумілим для користувачів.

2. Надійність:

– Середній час безперервної роботи - 55 год.

– Максимальна норма помилок та дефектів в роботі програми - 1 помилка на 3000 запитів користувача.

3. Швидкість. Сайт повинен швидко виконувати запити користувачів.

4. Можливості використання:

– Масштабування - після додавання функціонала не повинно бути фатальних помилок.

Аналіз функціональних вимог дозволив виявити наступні ключові елементи, що забезпечать розробку програмного комплексу для цієї системи. На рис. 2.4 наведена діаграма класів, що представляє контролери системи. Детальна структура представлень розглянута у третьому розділі.

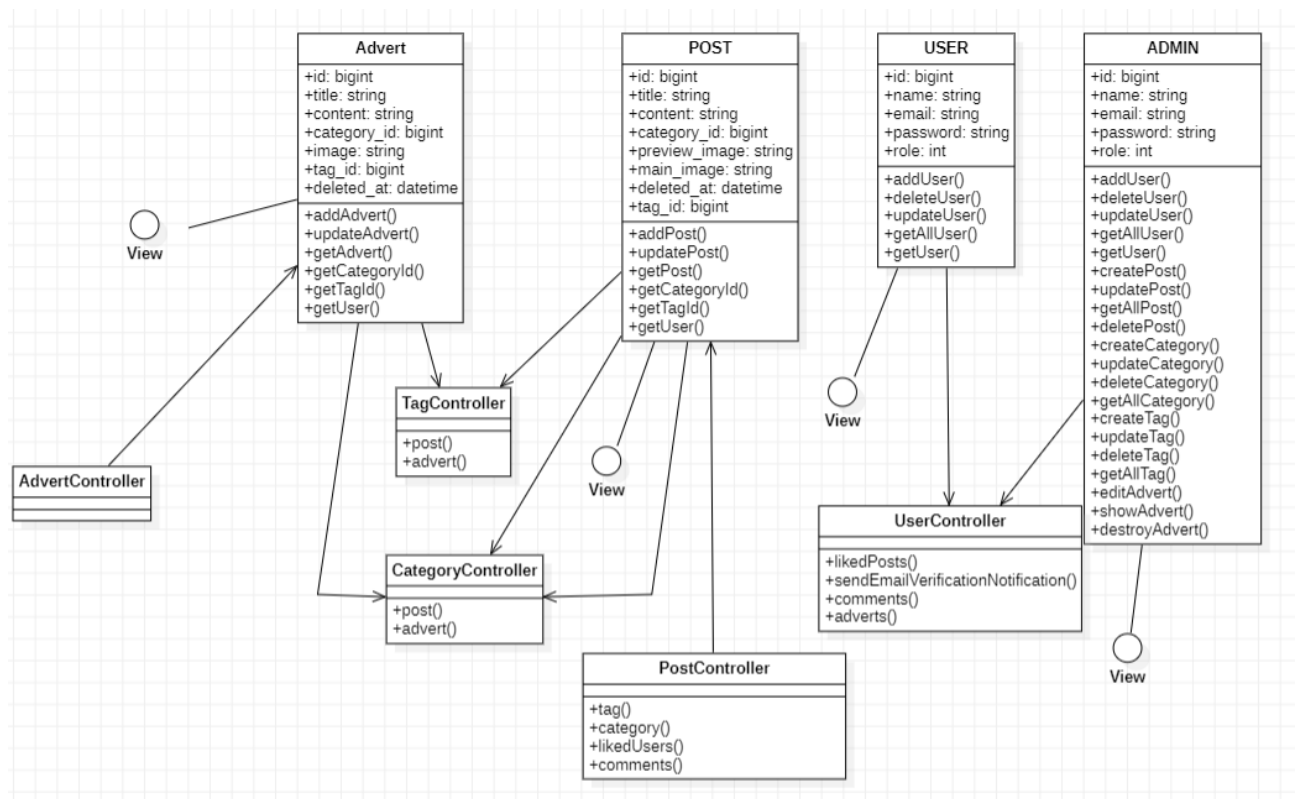


Рисунок 2.4 - Діаграма класів

На даному рисунку можна виділити наступні класи:

Admin – клас, який містить методи для роботи зі всіма сутностями додатка.

Поля:

- id (тип bigint) – унікальний ідентифікатор;
- name (тип string) – ім'я користувача;
- email (тип string) – е-мейл користувача;
- password (тип string) – пароль користувача;
- role (int) – роль користувача.

Методи:

- addUser() – додавання користувача;
- deleteUser() – видалення користувача;
- updateUser() – редагування користувача;
- getAllUser() – отримання всіх користувачів з бази даних;
- addCategory() – додавання категорії;
- deleteCategory() – видалення категорії;
- updateCategory() – видалення категорії;
- getAllCategory() – отримання всіх категорій з бази даних;
- addTag() – створення тега;
- updateTag() – редагування тега;
- getAllTag() – отримання всіх тегів з бази даних;
- editAdvert() – редагування заклику;
- showAdvert() – перегляд вибранного заклику;
- destroyAdvert() – видалення заклику.

POST – клас, який містить методи для роботи із програмами допомоги.

Поля:

- id (тип bigint) – унікальний ідентифікатор;
- title (тип string) – назва програми допомоги;
- content (тип string) – контент програми допомоги;
- preview_image (тип string) – прев'ю фото;
- main_image (тип string) – головне фото;
- category_id (тип bigint) – зовнішній ключ таблиці categories;

- tag_id (тип bigint) – зовнішній ключ таблиці tags;
- deleted_at (тип datetime) – дата та час створення.

Методи:

- getCategoryId() – отримання зовнішнього ключа категорії;
- getTagId() – отримання зовнішнього ключа тега;
- addPost() – додати пост до бази даних;
- updatePost() – редагувати пост;
- destroyPost() – видалити пост;
- getPost() – отримання поста;
- getAllPost() – отримання всіх постів з бази даних.

Advert – клас, який містить методи для роботи з закликами про допомогу.

Поля:

- id (тип bigint) – унікальний ідентифікатор;
- title (тип string) – назва програми допомоги;
- content (тип string) – контент програми допомоги;
- preview_image (тип string) – прев'ю фото;
- main_image (тип string) – головне фото;
- category_id (тип bigint) – зовнішній ключ таблиці categories;
- tag_id (тип bigint) – зовнішній ключ таблиці tags;
- deleted_at (тип datetime) – дата та час створення.

Методи:

- addAdvert() – створення заклика про допомогу;
- updateAdvert() – редагування заклика про допомогу;
- getAdvert() – отримання заклика про допомогу;
- getCategoryId() – отримання зовнішнього ключа категорії;
- getUser() – отримати інформацію про користувача, який створив залик;
- getTagId() – отримання зовнішнього ключа тега.
- destroyAdvert() – видалення заклика про допомогу

2.2 Розробка бази даних системи

Для збереження та подальшого використання даних було розроблено таблиці бази даних для кожного зі змінюваних сутностей, таких як:

1. users – таблиця в якій знаходиться інформація про користувачів;
2. posts – таблиця в якій знаходиться інформація про програми допомоги;
3. categories – таблиця в якій знаходиться інформація про категорії;
4. tags – таблиця в якій знаходиться інформація про теги;
5. comments – таблиця в якій знаходиться інформація про коментарі;
6. advertising – таблиця в якій знаходиться інформація про заклики на допомогу;
7. post_user_likes – таблиця, яка пов'язує posts та users на тему збереження;
8. user_adverts – таблиця яка пов'язує users та advertising.

Тож, база даних складається з 8 таблиць: категорії, теги, користувачі, заклики про допомогу, програми допомоги, коментарів.



Рисунок 2.5 - Діаграма бази даних

Нижче буде наведено опис кожної з таблиць бази даних (див. табл. 2.1 - 2.8).

Таблиця 2.1

users		
Поле	Тип	Призначення
Id	BIGINT, PK	Унікальний ідентифікатор

Продовження таблиці 2.1

name	VARCHAR(255)	Ім'я
email	VARCHAR(250)	Пошта
email_verified_at	TIMESTAMP	Дата та час верифікації пошти
password	VARCHAR(255)	Пароль
role	BIGINT	Роль користувача
deleted_at	TIMESTAMP	Дата та час видалення

Таблиця 2.2

posts		
Поле	Тип	Призначення
id	Uniqueidentifier, PK	Унікальний ідентифікатор
title	VARCHAR(255)	Заголовок
content	TEXT	Опис
category_id	BIGINT,FK	Зовнішній ключ категорії
main_image	VARCHAR(255)	Назва завантаженого головного фото
preview_image	VARCHAR(255)	Назва завантаженого прев'ю фото
tag_id	BIGINT,FK	Зовнішній ключ тега

Таблиця 2.3

comments		
Поле	Тип	Призначення
Id	BIGINT, PK	Унікальний ідентифікатор
message	VARCHAR(255)	Повідомлення
user_id	BIGINT, FK	Ідентифікатор користувача
post_id	BIGINT, FK	Ідентифікатор поста

Таблиця 2.4

tags		
Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
title	VARCHAR(255)	Заголовок
deleted_at	TIMESTAMP	Дата та час видалення

Таблиця 2.5

post_user_likes		
Поле	Поле	Поле
id	BIGINT, RK	Ідентифікатор роботи

Продовження таблиці 2.5

user_id	BIGINT, FK	Ідентифікатор користувача
post_id	BIGINT, FK	Ідентифікатор поста

Таблиця 2.6

advertising		
Поле	Тип	Призначення
id	BIGINT, PK	Унікальний ідентифікатор
title	VARCHAR(255)	Заголовок
content	TEXT	Опис
category_id	BIGINT,FK	Зовнішній ключ категорії
image	VARCHAR(255)	Назва завантаженого фото
tag_id	BIGINT,FK	Зовнішній ключ тега
deleted_at	TIMESTAMP	Дата та час видалення
add_info	VARCHAR(255)	Додаткова інформація
is_working	tinyint(1)	Статус показу

Таблиця 2.7

user_adverts		
Поле	Тип	Призначення
Id	BIGINT, PK	Унікальний ідентифікатор
user_id	BIGINT, FK	Ідентифікатор користувача
advertising_id	BIGINT, FK	Ідентифікатор поста

Таблиця 2.8

categories		
Поле	Тип	Призначення
Id	Uniqueidentifier, PK	Унікальний ідентифікатор
title	VARCHAR(255)	Заголовок
deleted_at	TIMESTAMP	Дата та час видалення

Реалізувавши базу даних, користувач і адміністратор додатка зможуть маніпулювати даними в ньому та автоматизувати свої дії. База даних складається з 8 основних таблиць.

2.3 Проектування та реалізація алгоритмів роботи системи

Головними компонентами в додатку є маніпуляції зі статтями автора (адміністратором) та дії зі статтями зі сторони користувачів. Розглянемо алгоритм роботи адміністратора:

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						31

- Автентифікація в обліковому записі з роллю Адміністратор;
- створення постів, користувачів, тегів, категорій;
- збереження створених сутностей у базу даних;
- перегляд, редагування, видалення певних елементів;
- збереження оновленої інформації.

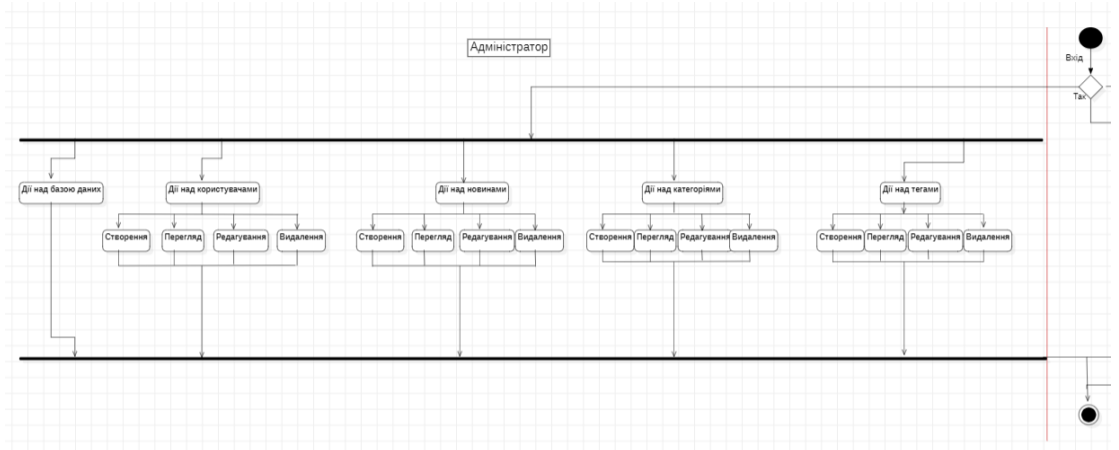


Рисунок 2.6 - Активність адміністратора

Розглянемо алгоритм роботи користувача:

- Автентифікація в обліковому записі з роллю читач;
- взаємодія зі статтями(перегляд, лайки, коментарі);
- збереження інформації про виконані дії у базу даних;
- перегляд, редагування, видалення збережених чи коментарів у персональному кабінеті;
- збереження оновленої інформації.

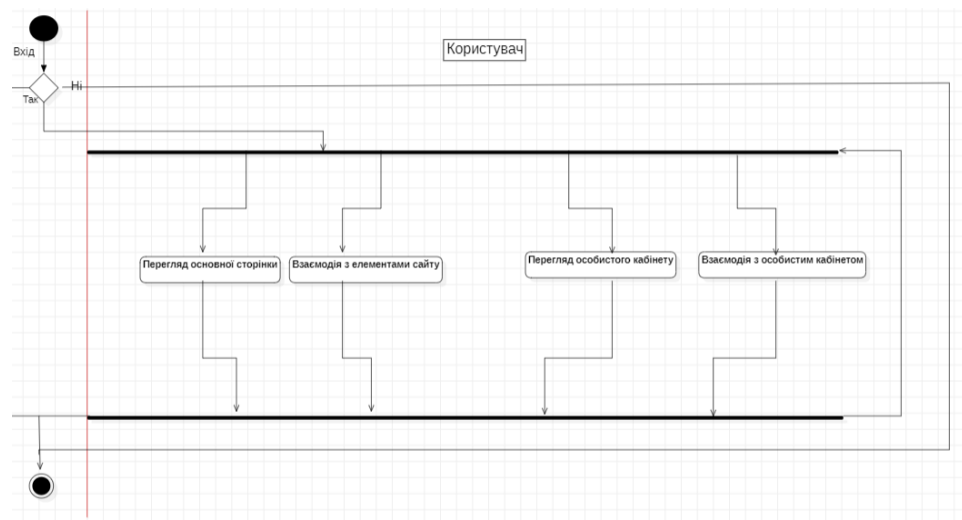


Рисунок 2.7- Активність користувача

Було розглянути можливі варіанти взаємодії з додатком зі сторони адміністратора та користувача.

2.4 Реалізація функціоналу інформаційного додатка

Для реалізації додатка було вибрано фреймворк Laravel. Laravel використовує розширення залежностей, такі як: маршрути, контролери, моделі та представлення [16]. Компоненти взаємодіють між собою за допомогою вбудованих методів.

Першим кроком до побудови додатка буде реалізація маршрутів. Код представлено у ДОДАТКУ А. Побудувавши ієрархію маршрутів переходимо до створення контролерів. Розглянемо контролери зі сторони адміністратора.

Контролер UserController відповідає за створення нового користувача з адмінпанелі.

```
class UserController extends Controller
{
    public function index()
    {
        $users = User::all();
        return view('admin.users.index', compact('users'));
    }
    public function create()
    {
        $roles = User::getRoles();
        return view('admin.users.create', compact('roles'));
    }
    public function store(StoreRequest $request)
    {
        $data = $request->validated();
        StoreUserJob::dispatch($data);
        return redirect()->route('admin.users.index');
    }
    public function show(User $user)
    {
        return view('admin.users.show', compact('user'));
    }
    public function edit(User $user)
    {
        $roles = User::getRoles();
        return view('admin.users.edit', compact('user', 'roles'));
    }
    public function update(UpdateRequest $request, User $user)
    {

```

```

        $data = $request->validated();
        $user->update($data);
        return view('admin.users.show', compact('user'));
    }
    public function destroy(User $user)
    {
        $user->delete();
        return redirect()->route('admin.users.index');
    }
}

```

Методи UserController

Таблиця 2.9

Назва	Опис та призначення
1	2
index	Метод для відображення всіх користувачів у базі даних
create	Метод, який передає дані для створення у метод Store
store	Метод створення нового користувача
show	Метод для перегляду певного користувача
edit	Метод, який передає дані для редагування у метод Update
update	Метод, який оновлює попередні дані у базі даних
destroy	Видалення обраного користувача. Дані про видалення зберігаються у базі даних за допомогою SoftDelete

Ресурсний Контролер PostController відповідає за роботу зі статтями, їх створення, редагування, перегляд та видалення.

```

class PostController extends Controller
{
    public function index()
    {
        $posts = Post::all();
        return view('admin.posts.index', compact('posts'));
    }
    public function create()
    {
        $categories = Category::all();
        $tags = Tag::all();
        return view('admin.posts.create', compact('categories',
'tags'));
    }
    public function store(StoreRequest $request)
    {

```

```

        try {
            $data = $request->validated();
            DB::beginTransaction();
            if (isset($data['tag_ids'])) {
                $tagIds = $data['tag_ids'];
                unset($data['tag_ids']);
            }
            $data['preview_image'] = Storage::disk('public')-
>put('/images/preview', $data['preview_image']);
            $data['main_image'] = Storage::disk('public')-
>put('/images/main', $data['main_image']);
            $post = Post::firstOrCreate($data);
            if (isset($tagIds)) {
                $post->tags()->attach($tagIds);
            }
            DB::commit();
        } catch (Exception $exception) {
            DB::rollBack();
            abort(500);
        }
        return redirect()->route('admin.posts.index');
    }
    public function show(Request $request, Post $post)
    {
        $tags = $post->tags;
        return view('admin.posts.show', compact('post', 'tags'));
    }
    public function edit(Post $post)
    {
        $categories = Category::all();
        $tags = Tag::all();
        return view('admin.posts.edit', compact('post',
'categories', 'tags'));
    }
    public function update(UpdateRequest $request, Post $post)
    {
        try {
            $data = $request->validated();
            DB::beginTransaction();
            if (isset($data['tag_ids'])) {
                $tagIds = $data['tag_ids'];
                unset($data['tag_ids']);
            }
            if (isset($data['preview_image'])) {
                $data['preview_image'] = Storage::disk('public')-
>put('/images/preview', $data['preview_image']);
            }
            if (isset($data['main_image'])) {
                $data['main_image'] = Storage::disk('public')-
>put('/images/main', $data['main_image']);
            }
            $post->update($data);
            if (isset($tagIds)) {

```

```

        $post->tags()->attach($tagIds);
    }
    DB::commit();
} catch (Exception $exception) {
    DB::rollBack();
    abort(500);
}
$tags = $post->tags;
return view('admin.posts.show', compact('post', 'tags'));
}
public function destroy(Post $post)
{
    $post->delete();
    return redirect()->route('admin.posts.index');
}
}

```

Методи PostController

Таблиця 2.10

Назва	Опис та призначення
1	2
index	Метод для відображення всіх статей у базі даних
create	Метод, який передає дані для створення у метод Store
store	Метод створення нової статті, використовуються стандартні функції PHP, для збереження фото використовується спосіб збереження назви з подальшим відображення
show	Метод для перегляду певної статті
edit	Метод, який передає дані для редагування у метод Update
update	Метод, який оновлює попередні дані у базі даних
destroy	Видалення обраної статті. Дані про видалення зберігаються у базі даних за допомогою SoftDelete

Ресурсний Контролер CategoryController відповідає за зміни у таблиці бази даних categories.

```

class CategoryController extends Controller
{
    public function index()
    {
        $categories = Category::all();
        return view('admin.categories.index',
compact('categories'));
    }
}

```

					ІПЗ.КР.Б – 121 – 23 – ПЗ	36

```

    }
    public function create()
    {
        return view('admin.categories.create');
    }
    public function store(StoreRequest $request)
    {
        $data = $request->validated();
        Category::firstOrCreate(['title' => $data['title']]);
        return redirect()->route('admin.categories.index');
    }
    public function show(Category $category)
    {
        return view('admin.categories.show', compact('category'));
    }
    public function edit(Category $category)
    {
        return view('admin.categories.edit', compact('category'));
    }
    public function update(UpdateRequest $request ,Category
$category)
    {
        $data = $request->validated();
        $category->update($data);
        return view('admin.categories.show', compact('category'));
    }
    public function destroy(Category $category)
    {
        $category->delete();
        return redirect()->route('admin.categories.index');
    }
}

```

Методи CategoryController

Таблиця 2.11

Назва	Опис та призначення
1	2
index	Метод для відображення всіх категорій з бази даних
create	Метод, який передає дані для створення у метод Store
store	Метод створення нової категорії
show	Метод для перегляду певної категорії
edit	Метод, який передає дані для редагування у метод Update
update	Метод, який оновлює попередні дані у базі даних
destroy	Видалення обраної категорії. Дані про видалення зберігаються у базі даних за допомогою SoftDelete

Ресурсний Контролер TagController відповідає за CRUD тегів.

```
class TagController extends Controller
{
    public function index()
    {
        $tags = Tag::all();
        return view('admin.tags.index', compact('tags') );
    }

    public function create()
    {
        return view('admin.tags.create');
    }

    public function store(StoreRequest $request)
    {
        $data = $request->validated();
        Tag::firstOrCreate($data);
        return redirect()->route('admin.tags.index');
    }

    public function show(Tag $tag)
    {
        return view('admin.tags.show', compact('tag'));
    }

    public function edit(Tag $tag)
    {
        return view('admin.tags.edit',compact('tag') );
    }

    public function update(UpdateRequest $request ,Tag $tag)
    {
        $data = $request->validated();
        $tag->update($data);
        return view('admin.tags.show', compact('tag'));
    }

    public function destroy(Tag $tag)
    {
        $tag->delete();
        return redirect()->route('admin.tags.index');
    }
}
```

Методи TagController

Таблиця 2.12

Назва	Опис та призначення
1	2
index	Метод для відображення всіх тегів з бази даних
create	Метод, який передає дані для створення у метод Store
store	Метод створення нового тега
show	Метод для перегляду певного тега
edit	Метод, який передає дані для редагування у метод Update
update	Метод, який оновлює попередні дані у базі даних
destroy	Видалення обраного тега. Дані про видалення зберігаються у базі даних за допомогою SoftDelete

Контролер AdvertisingController створений як ресурсний, проте деякі методи не використовуються.

```
class AdvertisingController extends Controller
{
    public function index()
    {
        $advertising = Advert::all();
        return view('admin.advert.index', compact('advertising'));
    }

    public function create()
    {
        //
    }

    public function store(Request $request)
    {
        //
    }

    public function show(Advert $advertising)
    {
        return view('admin.advert.show', compact('advertising'));
    }

    public function edit(Advert $advertising)
    {
        return view('admin.advert.edit', compact('advertising'));
    }
}
```

```

public function update(UpdateRequest $request, $id)
{
    $advertising = Advert::findOrFail($id);
    $advertising->is_working = $request->has('is_working');
    $advertising->save();
    return redirect()->route('admin.advertising.index',
compact('advertising'));
}

public function destroy(Advert $advertising)
{
    $advertising->delete();
    return redirect()->route('admin.advertising.index');
}
}

```

Методи AdvertisingController

Таблиця 2.13

Назва	Опис та призначення
1	2
index	Метод для відображення всіх закликів про допомогу з бази даних
create	-
store	-
show	Метод для перегляду певного заклику
edit	Метод, який передає дані для редагування у метод Update
update	Метод, який оновлює попередні дані у базі даних
destroy	Видалення обраного заклику. Дані про видалення зберігаються у базі даних за допомогою SoftDelete

Контролер IndexController використовується для відображення певної інформації про статті, користувачів, теги та категорії.

```

class IndexController extends Controller
{
    public function index()
    {
        $data = [];
        $data['usersCount'] = User::all()->count();
        $data['postsCount'] = Post::all()->count();
        $data['categoriesCount'] = Category::all()->count();
        $data['tagsCount'] = Tag::all()->count();
        return view('admin.main.index', compact('data'));
    }
}

```

Висновки до розділу 2

Інформаційний додаток реалізовано з дотриманням необхідних для виконання задач функціоналом.

В другому розділі було продемонстровано створення бази даних з необхідним для сайту функціонала. База даних складається з 8 таблиць, які взаємодіють між собою.

Було реалізовано і розглянуто основні методи та технології для продуктивної роботи з додатком.

В пункті 2.4 детально розглядається частина функціонала, точніше контролери які відповідають за створення сутностей з адміністративного кабінету.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКА

3.1 Порядок встановлення та налаштування системи

Для роботи з додатком на локальній машині необхідні певні налаштування комп'ютера. Для коректної роботи Laravel та його вбудованих методів необхідні такі встановлення: npm/yarn, node.js, PHP (версії 8.1), composer, MySQL, PhpStorm. Розглянемо більше детально встановлення цих технологій.

1. Для встановлення npm на Windows необхідно завантажити архів з офіційного сайту [10] та слідувати інструкціям інстальатора.

2. Для встановлення node.js потрібно перейти на сайт [11] для завантаження інстальатора та виконувати дії, необхідні для встановлення.

3. Для встановлення composer необхідно слідувати інструкціям з встановлення на офіційному сайті [12].

4. Встановлення PHP виконується завантаженням архіву з офіційного сайту. Обравши необхідну версію, після завантаження, необхідно додати шлях до файлу php.exe у реєстр Windows [7].

5. Встановлення MySQL виконується за допомогою вказівок у завантаженому інстальаторі з офіційного сайту [9].

Виконавши необхідні налаштування та встановлення, необхідно розвернути наш проєкт у середовищі розробки, у нашому випадку - PhpStorm. Потрібно створити базу даних для проєкту, зробити це можна за допомогою терміналу MySQL або візуального середовища MySQL Workbench. Після створення бази даних, в проєкті потрібно змінити конфігураційний файл .env у поля, які відповідають за конфігурацію бази даних необхідно написати дані, які були створені:

```
DB_DATABASE=example
DB_USERNAME=root
DB_PASSWORD=root
```

Після змін необхідно проініціалізувати проєкт, написавши команду **npm install**. Після успішного встановлення всіх залежностей, необхідно виконати міграції, відбувається це за командою **php artisan migrate**.

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						42

Після виконання цих дій, проєкт готовий до роботи та змін. Перевірити стан та запустити сервер можна за допомогою команди **php artisan serve**.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з веб-орієнтованим інформаційним додатком “Daily Bugle”

Головна сторінка знаходиться за адресою <http://127.0.0.1:8000/posts>. Користувач може бачити в шапці перелік основних посилань. Користувач має можливість переглянути зміст головної сторінки, перейти за посиланнями у шапці або увійти до облікового запису. Демонстрація роботи сайту проводиться з акаунту суперадміністратора.

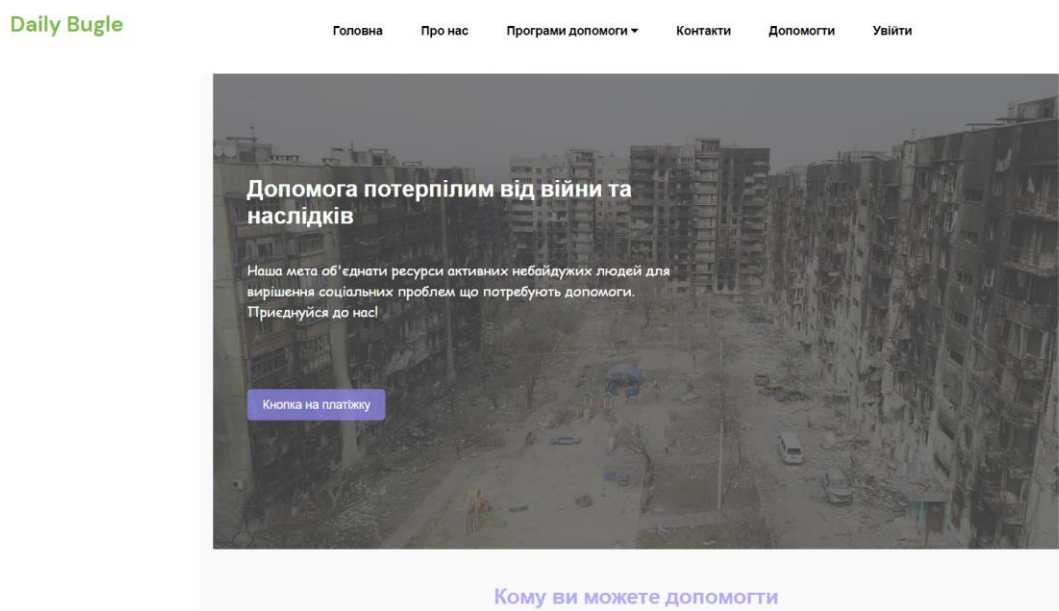


Рисунок 3.1 - Головна сторінка “Daily Bugle”

На головній сторінці користувач може вибрати програму з допомоги.

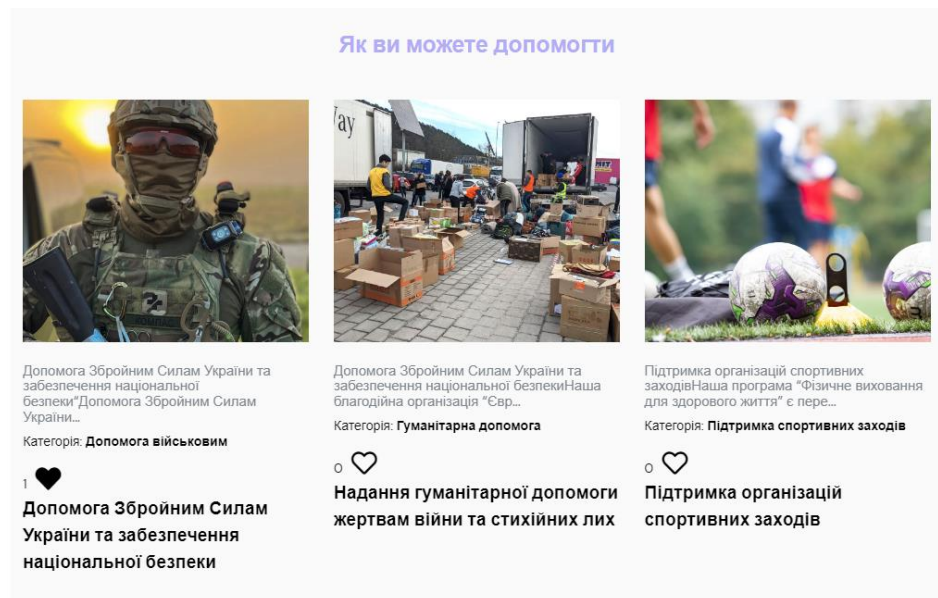


Рисунок 3.2 - Програми допомоги

Користувач може переглянути обрану допомогу та зберегти її в особистий кабінет.

Відкривши певну програму, користувач буде бачити основну інформацію про створення поста (дата та час), детальну інформацію щодо програми допомоги.

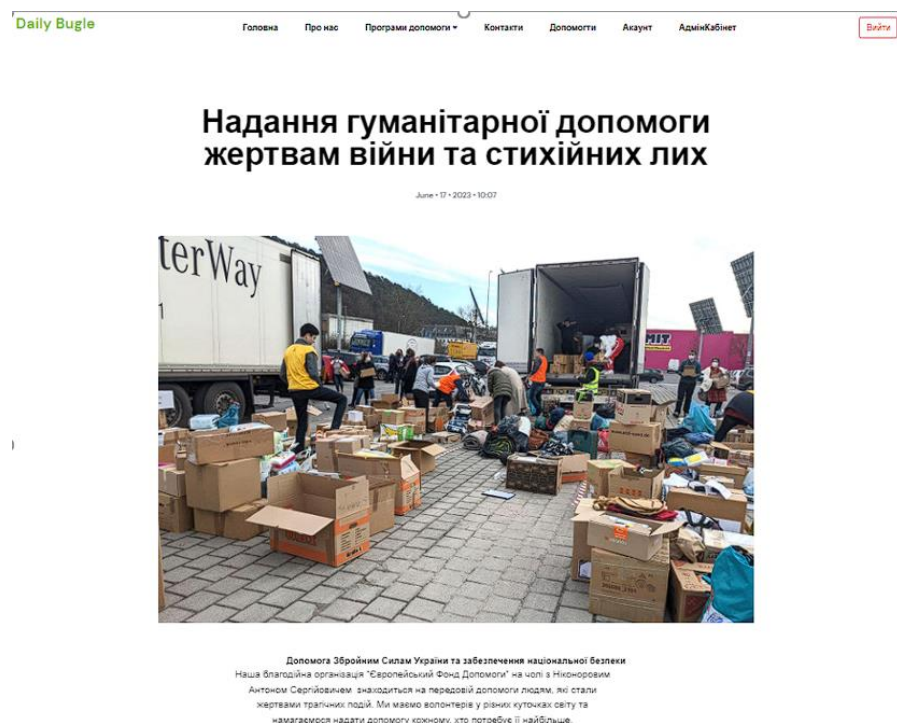


Рисунок 3.3 - Детальний зміст програми допомоги

Прогорнувши пост до кінця, користувач побачить кнопку з переходом на сторінку донату.

обладнання, будівництва житла для постраждалих тощо.



Допомогти

Коментарі (0)

Рисунок 3.4 - Сторінка для відправки допомоги

Перейшовши на вкладку “Допомогти” користувач перейде за адресою <http://127.0.0.1:8000/help>, де він побачить перелік закликів про допомогу від людей, які створили запит в особистому кабінеті.

Заклики про допомогу

У зв'язку з нещодавними затопленнями, нашою родиною був втрачений будинок. Наразі шукаємо, де можна... Втрата будинку Кому допомога: Пошук житла Місто: Херсон, Херсонська обл.	Категорії
	Допомога військовим Гуманітарна допомога
	Підтримка спортивних заходів Пошук житла
	Тварини Евакуація Допомога речами
	Загальна допомога
	Міста
	Київ, Київська обл. Херсон, Херсонська обл.
	Харків, Харківська обл. Житомир, Житомирська обл.
	Суми, Сумська обл. Чернігів, Чернігівська обл.
	Дніпропетровськ, Дніпропетровська обл.
	Запоріжжя, Запорізька обл. Донецьк, Донецька обл.

У наслідок останніх подій, під водою опинилось багато тварин які не можуть самостійно вибратись з па... Допомога постраждалим тваринам Кому допомога: Тварини Місто: Херсон, Херсонська обл.
Допомога в евакуації людей з обмеженими можливостями з затоплених районів. Збираємо інформація про т... Евакуація людей з обмеженими можливостями Кому допомога: Евакуація Місто: Херсон, Херсонська обл.
Наша родина вимушені були швидко покинути свою домівку через велику небезпеку затоплення. Більшість... Допомога речами для переселенців Кому допомога: Допомога речами Місто: Херсон, Херсонська обл.

Рисунок 3.5 - Сторінка “Допомогти”

На сторінці допомоги користувач має можливість детально переглянути заклик про допомогу. Також користувач може вибрати заклики за категорією чи містом. Розглянемо детально заклик. Користувач бачить основну інформацію

про створення заклика (час та дата), інформацію про проблему, автора та його контакти.

Втрата будинку

June • 14 • 2023 • 13:24

У зв'язку з нещодавними затопленнями, нашою родиною був втрачений будинок. Наразі шукаємо, де можна покинути деякий час. Якщо в когось є така можливість, будь ласка сповістіть. Родина з 4х людей.

Кому потрібна допомога

admin1
admin1@gmail.com

Додаткова інформація від автора

Номер телефону 380983301456

Якщо ви маєте можливість та бажання допомогти цій людині зверніться за контактами які були вказані у відповідних полях.

Рисунок 3.6 - Детальний перегляд заклику

При виборі закликів за категорією чи тегом користувач отримає перелік закликів за його вибором. Виберемо варіант міста Херсон, Херсонська обл.

Заклики про допомогу за містом: Херсон, Херсонська обл.

У зв'язку з нещодавними затопленнями, нашою родиною був втрачений будинок. Наразі шукаємо, де можна...

Втрата будинку

Кому допомога: Пошук житла

Місто: Херсон, Херсонська обл.

У наслідок останніх подій, під водою опинилось багато тварин які не можуть самостійно вибратись з па...

Допомога постраждалим тваринам

Кому допомога: Тварини

Місто: Херсон, Херсонська обл.

Допомога в евакуації людей з обмеженими можливостями з затоплених районів. Збираємо інформація про т...

Евакуація людей з обмеженими можливостями

Кому допомога: Евакуація

Місто: Херсон, Херсонська обл.

Категорії

Допомога військовимГуманітарна допомога

Підтримка спортивних заходівПошук житла

ТвариниЕвакуаціяДопомога речами

Загальна допомога

Міста

Київ, Київська обл.Херсон, Херсонська обл.

Харків, Харківська обл.Житомир, Житомирська обл.

Суми, Сумська обл.Чернігів, Чернігівська обл.

Дніпропетровськ, Дніпропетровська обл.

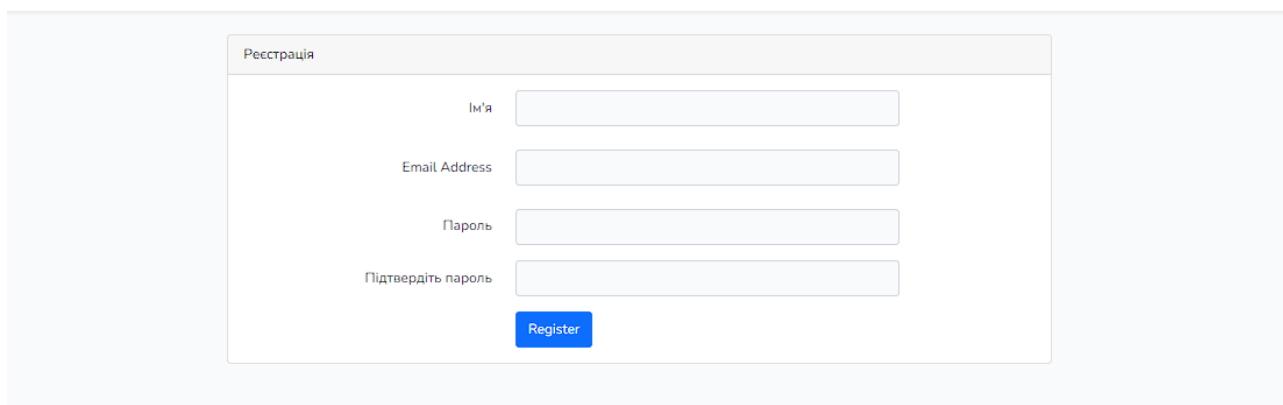
Запоріжжя, Запорізька обл.Донецьк, Донецька обл.

Рисунок 3.7 - Перелік закликів у місті Херсон, Херсонська обл.

Гість має можливість зареєструватися.

Daily Bugle

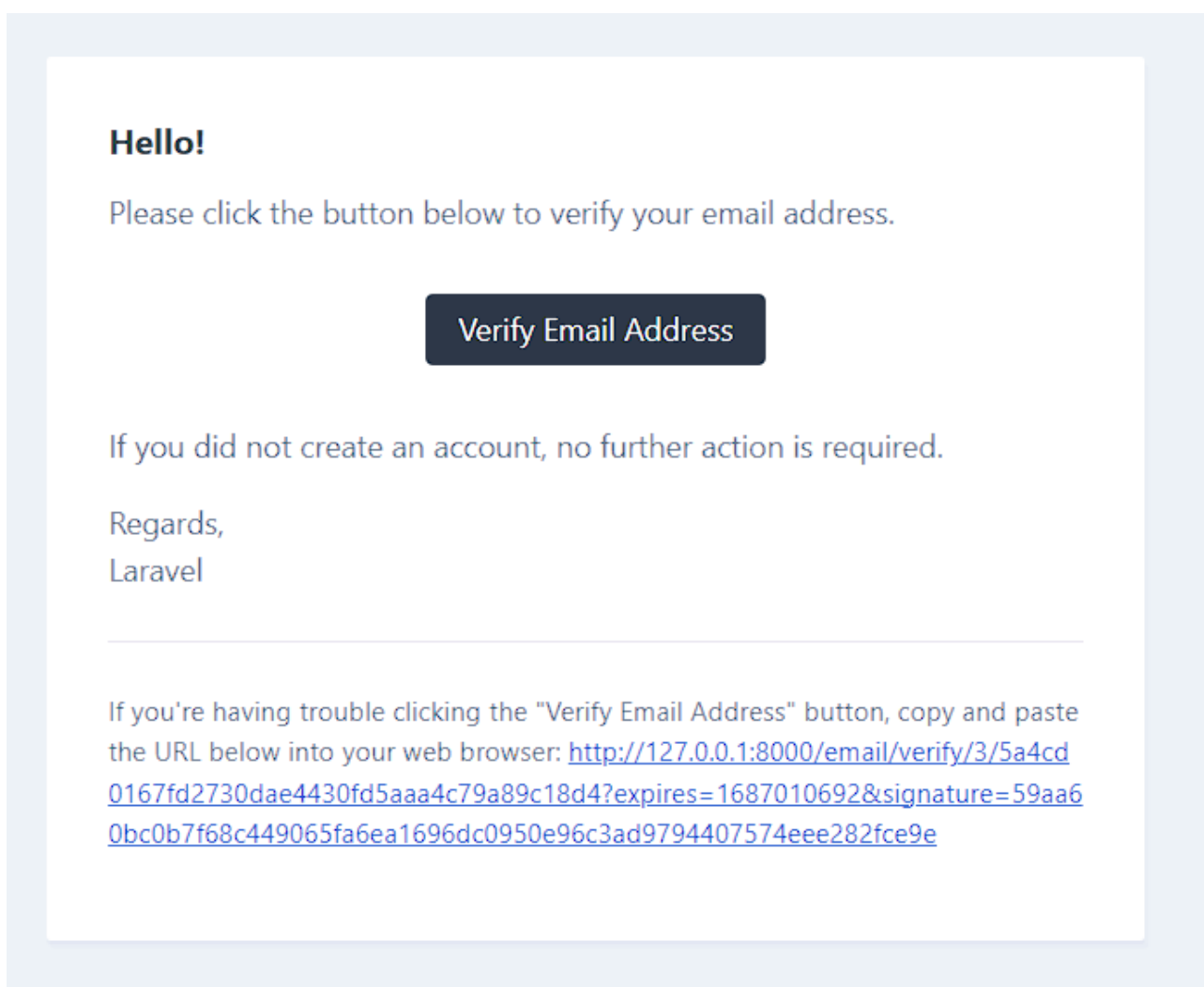
Регістрація



The image shows a registration form titled "Регістрація" (Registration) within a light blue container. The form itself has a white background and a light gray header with the title. It contains four input fields: "Ім'я" (Name), "Email Address", "Пароль" (Password), and "Підтвердіть пароль" (Confirm password). Below these fields is a blue "Register" button.

Рисунок 3.8 - Форма реєстрації

Для успішної реєстрації користувачу необхідно підтвердити свою пошту перейшовши по посиланню у листі, який надійде.



The image shows an email verification template. It starts with "Hello!" in bold. Below it is the text "Please click the button below to verify your email address." followed by a dark blue button with the text "Verify Email Address". Then it says "If you did not create an account, no further action is required." followed by "Regards, Laravel". At the bottom, it provides instructions on how to verify the email by clicking the button or copying a long URL into a web browser. The URL is: <http://127.0.0.1:8000/email/verify/3/5a4cd0167fd2730dae4430fd5aaa4c79a89c18d4?expires=1687010692&signature=59aa60bc0b7f68c449065fa6ea1696dc0950e96c3ad9794407574eee282fce9e>

Рисунок 3.9 - Лист для верифікації

При успішній верифікації пошти відкриється головна сторінка додатка.

Користувач має персональний кабінет, до якого він може потрапити за посиланням у шапці сайту. На головній сторінці користувач бачить лічильник своїх збережених програм допомоги та коментарів та має можливість детально переглянути їх.

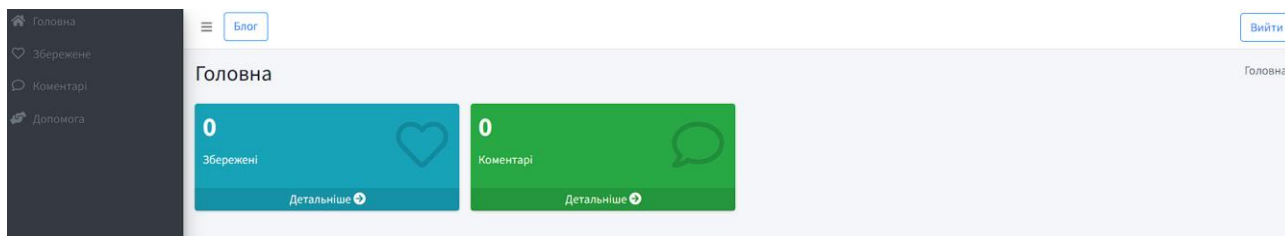


Рисунок 3.10 - Головна сторінка персонального кабінету

Перейшовши до вкладки “Збережені”, користувач побачить список статей, які він зберіг. Користувач має змогу переглянути їх та видалити.

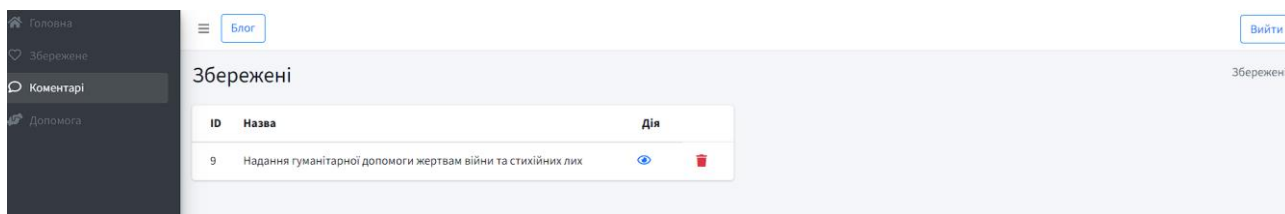


Рисунок 3.11 - Вкладка “Збережені” у персональному кабінеті

Перейшовши у вкладку “Допомога” користувач бачить перелік створених ним закликів про допомогу та кнопку для створення. Натиснувши на неї користувач буде відправлений на форму зі створенням заклику про допомогу.

Рисунок 3.12 - Форма створення заклику

Після створення заклик буде відображатися на головній сторінці допомоги. Користувач має можливість переглянути, відредагувати та видалити заклик. Заклик відобразиться на сторінці з закликами про допомогу, після того, як адміністратор підтвердить це.

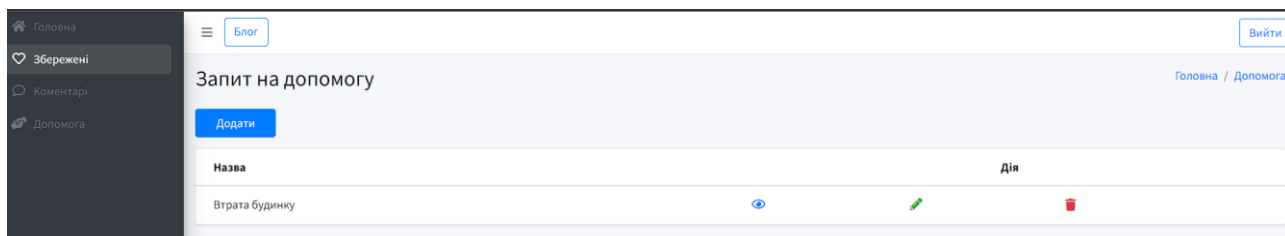


Рисунок 3.13 - Відображення створених закликів

Розглянемо можливості та функціонал адміністративного кабінету. При переході за посиланням з шапки сайту або за адресою <http://127.0.0.1:8000/admin>, буде відкрита головна сторінка адміністративного кабінету. Головна сторінка відображає лічильники основних сутностей.

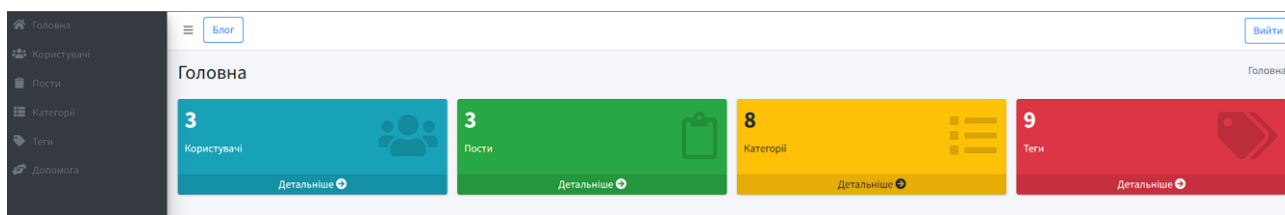


Рисунок 3.14 - Головна сторінка адміністративного кабінету

Розглянемо декілька можливих варіантів роботи з сутностями. Наприклад, “Користувачі”. Адміністратор при переході на вкладку “Користувачі” бачить перелік всіх зареєстрованих. Має можливість додати користувача, переглянути, редагувати та видалити.

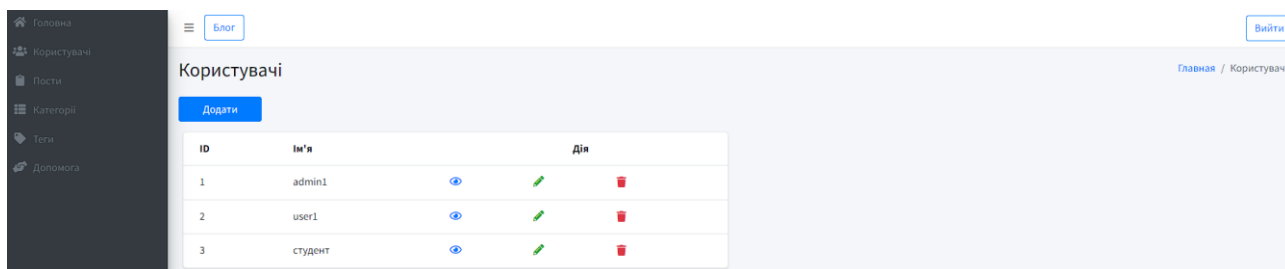


Рисунок 3.15 - Сторінка “Користувачі”

Натиснувши кнопку “Додати” здійсниться перехід на форму додавання користувача. У додатку реалізовано відправку листів за допомогою стороннього сервісу. Адміністратор вказує ім’я та пошту користувача, обирає роль і тоді на пошту користувача надходить два листи: один з паролем для входу, інший для верифікації.

Рисунок 3.16 - Форма додавання користувача

Розглянемо “Пости”. Реалізовано CRUD, на головній сторінці відображаються необхідні посилання для маніпулювання даними.

ID	Назва	Дія
8	Допомога Збройним Силам України та забезпечення національної безпеки	View Edit Delete
9	Надання гуманітарної допомоги жертвам війни та стихійних лих	View Edit Delete
10	Підтримка організацій спортивних заходів	View Edit Delete

Рисунок 3.17 - Сторінка “Пости”

При додаванні адміністратор повинен заповнити всі обов’язкові поля.

Рисунок 3.18 - Створення програми допомоги

Перейшовши на вкладку “Допомога”, адміністратор має можливість переглянути, видалити та підтвердити заклик про допомогу.

ID	Названіе	Статус	Дія
2	Втрата будинку	Показується	
3	Допомога постраждалим тваринам	Показується	
4	Евакуація людей з обмеженими можливостями	Не показується	
7	Допомога речами для переселенців	Не показується	
8	Просимо допомоги	Показується	
11	Втрата будинку	Не показується	

Рисунок 3.19 - Сторінка закликів про допомогу

При натисканні на кнопку “Показується” чи “Не показується” адміністратор потрапляє у форму перевірки заклику, де має можливість змінити статус для показу.

Статус	ID	Тема	Контент	Фото	Додаткова інформація
☒	2	Втрата будинку	У зв'язку з нещодавними затопленнями, нашою родиною був втрачений будинок. На...		Номер телефону 380983301456

Рисунок 3.20 - Перегляд окремого заклика

3.3 Тестування роботи програмного комплексу

Для коректної роботи будь-якого додатка необхідно проводити велику кількість тестів, що можуть виявити слабкі місця та проблеми у додатку. Для перевірки працездатності виконаємо дії, які повинен виконувати користувач.

1. Реєстрація облікового запису:

передумови: користувач який потрапив на головну сторінку має можливість зареєструватись;

кроки: натиснувши “Зареєструватись”, користувач заповнює форму та відправляє;

очікуваний результат: користувач отримує лист з верифікацією на пошту, після верифікації обліковий запис ініціалізований в системі;

фактичний результат: користувач пройшов верифікацію.

2. Користувач заходить в персональний кабінет:

передумови: користувач переходить до акаунту з головної сторінки;

очікуваний результат: користувач перейшов до особистого кабінету і взаємодіє з його елементами;

фактичний результат: користувач потрапив до особистого кабінету.

3. Взаємодія з сутностями (Збережені, Коментарі, Допомога):

передумови: користувач повинен мати змогу для керування даними;

кроки: перехід на вкладку сутності (Збережені, Коментарі, Допомога), де користувач може переглянути, додати, змінити та видалити обрані дані;

очікуваний результат: користувач змінює необхідні йому дані й вони зберігаються;

фактичний результат: користувач має можливість взаємодіяти зі всіма елементами персонального кабінету.

4. Взаємодія з елементами головної сторінки:

передумови: користувач, який знаходиться на головній сторінці, повинен мати змогу: переглядати програми допомоги, сторінку “Про нас”, “Контакти”, “Допомога” та мати можливість взаємодіяти з інформацією;

передумови: користувач, який знаходиться на головній сторінці, відкриває посилання на обрану програму допомоги;

очікуваний результат: користувачу відкривається повна версія програми з можливістю зберегти чи написати коментар;

фактичний результат: користувач має змогу взаємодіяти з необхідними елементами.

5. CRUD сутностей (Програми допомоги, Міста, Користувачі, Категорії,

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						52

Допомога) користувачем з роллю адміністратор:

передумови: користувач (адмін), перейшовши до адміністративного кабінету, повинен мати можливість для маніпулювання необхідними даними;

кроки: обравши додавання користувача, адміністратор побачить форму з необхідними для заповнення полями;

очікуваний результат: після відправлення форми, користувач збережений у базі даних;

фактичний результат: після створення користувач збережений у базі даних.

Висновки до третього розділу

В даному розділі було розглянуто інструкцію щодо розгортання проєкту на локальній машині, які технології та дії для цього потрібні.

Було детально розглянуто принцип роботи додатка. Його компоненти, сторінки та методи. Розглянуто для кожної з ролей (гість, користувач, адміністратор). Сам опис являє собою візуальну інструкцію щодо використання додатка.

Після інформації про роботу інтерфейсу, було проведе тестування. Тестування відбувалося з метою виявлення помилок та їх усунення.

ВИСНОВКИ

При розробці випускної роботи бакалавра було реалізовано веб-орієнтований інформаційний додаток "Daily Bugle" за допомогою PHP фреймворку Laravel. При створенні додатка були вирішені наступні питання:

1. Вибір веб-орієнтованої архітектури для розробки додатка. Чим MVC переважає над MVP та MVVC. Чому вибір MVC є раціональним для створення інформаційного додатка. Цей вибір дозволяє досягти гнучкості та розділення логіки.

2. Було вибрано та обґрунтовано вибір технологічних інструментів для розробки, чому краще використовувати мову програмування PHP, базу даних MySQL та мову розмітки HTML та CSS.

3. Було реалізовано та розглянуто основні методи та функції, які присутні в додатку. Було представлено алгоритми роботи на сайті та інші можливості. Відповідно до обраної архітектури та поставлених вимог були обрані інструментальні засоби для реалізації проекту.

4. Внаслідок виконання бакалаврської роботи був реалізований додаток, що функціонує та має потенціал в розвитку додаткових методів та варіантів роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. “MVC патерн”, freeCodeCamp, 2023. Доступ за посиланням <https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/>
2. “MVP патерн”, Brander, 2023. Доступ за посиланням <https://brander.ua/technologies/mvp>
3. “MVVM патерн”, TechTarget, 1999-2023. Доступ за посиланням <https://www.techtarget.com/whatis/definition/Model-View-ViewModel>
4. “Bootstrap Documentation”, @mdo and @fat, 2010-2023. [Електронний ресурс]. Доступ за посиланням <https://getbootstrap.com/docs/5.3/getting-started/parcel/>
5. “CSS Documentation”, mozilla.org, 1998–2023. [Електронний ресурс]. Доступ за посиланням <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
6. ”JavaScript ”, mozilla.org, 1998–2023. [Електронний ресурс]. Доступ за посиланням <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
7. “PHP Documentation”, The PHP Group, 2001-2023. [Електронний ресурс]. Доступ за посиланням <https://www.php.net/docs.php>
8. “IDE PhpStorm”, JetBrains s.r.o., 2000-2023. Доступ за посиланням <https://www.jetbrains.com/phpstorm/>
9. “Installing MySQL”, Oracle, 1995-2023. [Електронний ресурс]. Доступ за посиланням <https://dev.mysql.com/doc/refman/8.0/en/windows-installation.html>
10. “npm install”, npmjs, 2023. [Електронний ресурс]. Доступ за посиланням <https://www.npmjs.com/package/npm>
11. “nodejs install”, OpenJS Foundation, 2023. [Електронний ресурс]. Доступ за посиланням <https://nodejs.org/en/download>

12. “composer install”, Nils Adermann, Jordi Boggiano, 2023. [Электронный ресурс]. Доступ за посиланням <https://getcomposer.org/doc/00-intro.md>
13. “MySQL Documentation” , Oracle, 1995-2023. [Электронный ресурс]. Доступ за посиланням <https://dev.mysql.com/doc/>
14. “Bootstrap Templates”, BootstrapDash, 2023. [Электронный ресурс]. Доступ за посиланням <https://www.bootstrapdash.com/>
15. “Email Delivery Platform - Mailtrap”, Railware Products Studio LLC, 2023. Доступ за посиланням <https://mailtrap.io/>
16. “laravel”, Laravel LLC, 2011-2023. [Электронный ресурс]. Доступ за посиланням <https://laravel.com/>

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 23 – ПЗ	
						57

Код маршрутів додатка

```

Route::get('/', function () {
    return view('welcome');
});
Route::get('/', [App\Http\Controllers\IndexController::class,
'__invoke']);

Route::group(['prefix' => 'admin', 'middleware' => ['auth',
'admin', 'verified']], function () {
    Route::get('/',
[\App\Http\Controllers\Admin\IndexController::class, 'index'])->
name('admin.main.index');

    Route::resource('categories', CategoryController::class, ['as'
=> 'admin'])->except('update');
    Route::post('categories/{category}',
[CategoryController::class, 'update'])->
name('admin.categories.update');

    Route::resource('tags', TagController::class, ['as' =>
'admin'])->except('update');
    Route::post('tags/{tag}', [TagController::class, 'update'])->
name('admin.tags.update');

    Route::resource('users', UserController::class, ['as' =>
'admin'])->except('update');
    Route::post('users/{user}', [UserController::class,
'update'])->name('admin.users.update');

    Route::resource('posts', PostController::class, ['as' =>
'admin'])->except('update');
    Route::post('posts/{post}', [PostController::class,
'update'])->name('admin.posts.update');
});

Route::group(['prefix' => 'personal', 'middleware' => ['auth',
'admin', 'verified']], function () {
    Route::resource('advertising', AdvertisingController::class,
['as' => 'personal'])->except('update');
    Route::post('advertising/{advertising}',
[AdvertisingController::class, 'update'])->
name('personal.advertising.update');
});

Route::group(['prefix' => 'admin', 'middleware' => ['auth',
'admin', 'verified']], function () {
    Route::resource('advertising',
\App\Http\Controllers\Admin\AdvertisingController::class, ['as' =>
'admin'])->except('update');

```

```

Route::post('advertising/{advertising}',
[\App\Http\Controllers\Admin\AdvertisingController::class,
'update'])->name('admin.advertising.update');

});

Route::group(['namespace' => 'Main'], function () {
Route::get('/',
[\App\Http\Controllers\Main\IndexController::class, '__invoke'])->name('main.index');
});
Route::group(['prefix' => 'posts'], function () {
Route::get('/',
[\App\Http\Controllers\Post\IndexController::class, '__invoke'])->name('post.index');
Route::get('/{post}', [ShowController::class, 'show'])->name('post.show');
});
Route::group(['prefix' => '{post}/comments'], function () {
Route::post('/',
[\App\Http\Controllers\Post\Comment\StoreController::class, 'store'])->name('post.comment.store');
});
Route::group(['prefix' => '{post}/likes'], function () {
Route::post('/',
[\App\Http\Controllers\Post\Like\StoreController::class, 'store'])->name('post.like.store');
});

Route::group(['namespace' => 'Personal', 'prefix' => 'personal', 'middleware' => ['auth', 'verified']], function () {
Route::group(['namespace' => 'Main', 'prefix' => 'main'], function () {
Route::get('/',
[\App\Http\Controllers\Personal\Main\IndexController::class, '__invoke'])->name('personal.main.index');
});
Route::group(['namespace' => 'Liked', 'prefix' => 'liked'], function () {
Route::get('/',
[\App\Http\Controllers\Personal\Liked\IndexController::class, '__invoke'])->name('personal.liked.index');
Route::delete('/{post}',
[\App\Http\Controllers\Personal\Liked>DeleteController::class, '__invoke'])->name('personal.liked.delete');
});
Route::group(['namespace' => 'Comment', 'prefix' => 'comment'], function () {
Route::get('/',
[\App\Http\Controllers\Personal\Comment\IndexController::class, '__invoke'])->name('personal.comment.index');
Route::get('/{comment}/edit',

```

```

[\App\Http\Controllers\Personal\Comment\EditController::class,
'edit'])->name('personal.comment.edit');
    Route::post('/{comment}',
[\App\Http\Controllers\Personal\Comment\UpdateController::class,
'update'])->name('personal.comment.update');
    Route::delete('/{comment}',
[\App\Http\Controllers\Personal\Comment\DeleteController::class,
'delete'])->name('personal.comment.delete');
    });
});

Route::group(['namespace' => 'Category', 'prefix' =>
'categories'], function () {
    Route::get('/',
[\App\Http\Controllers\Category\IndexController::class,
'__invoke'])->name('category.index');

    Route::group(['namespace' => 'Post', 'prefix' =>
'{category}/posts'], function () {
        Route::get('/',
[\App\Http\Controllers\Category\Post\IndexController::class,
'__invoke'])->name('category.post.index');
    });
});

Auth::routes(['verify' => true]);

Route::get('/home', [\App\Http\Controllers\HomeController::class,
'index'])->name('home');

```