Хмельницький національний університет

Факультет програмування
та комп'ютерних і телекомунікаційних систем

Кафедра інженерії програмного забезпечення

ДИПЛОМНИЙ ПРОЕКТ

Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт

Назва теми

Рівень вищої освіти Перший(бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр ДПІПЗ.170120.01.17.ПЗ

Виконав студент IV курсу група ІПЗ-17-1 _____ В. В. Стьопич
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ Г. І. Радельчук
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Радельчук
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

7 06 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
5 . 02 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Стьопичу Владиславу Валерійовичу
Прізвище, ім'я, по батькові студента

1. Тема проєкту (роботи) Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт
Керівник проєкту (роботи) Радельчук Галина Іванівна
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
кандидат технічних наук, доцент
Затверджена наказом ректора університету від 05.02.2021 р. № 11
2. Строк подання студентом проєкту (роботи) на кафедру 01.06.2021 р.
3. Вихідні дані до проєкту (роботи) Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування Інтернет-платформи, програмна реалізація, тестування програми
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди, 16 шт)

6. Консультанти розділів дипломного проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Радельчук Г. І., доцент кафедри ІПЗ		
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ		

7. Дата видачі завдання «05» лютого 2021 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Приміт
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних там ДП	01.12 – 30.12.2020	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2021	
3 Проектування програмного забезпечення	01.02 – 28.02.2021	
4 Програмна реалізація	01.03 – 10.04.2021	
5 Тестування програмного забезпечення	11.04 – 30.04.2021	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2021	
7 Попередній захист ДП	Травень 2021 (згідно графіка)	
8 Перевірка ДП на плагіат, нормконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки)	26.05 – 30.05.2021	
9 Підготовка до захисту та захист ДП	з 01.06.2021	

Студент _____ В. В. Стьопич
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Г. І. Радельчук
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт.

Автор проекту: Стьопич Владислав Валерійович.

Керівник проекту: Радельчук Галина Іванівна.

Пояснювальна записка: 105 с., 39 рис., 6 табл., 3 дод., 13 джерел.

Графічна частина: 16 презентаційних слайдів.

ІНТЕРНЕТ-ПЛАТФОРМА, БЮРО ЗНАХІДОК, ЗНАХІДКА, ВТРАТА, NGINX, MARIADB, PHP, YII2.

Метою проекту є створення Інтернет-платформи «Бюро знахідок» з підтримкою та використанням Google-карт.

У дипломному проекті проведено аналіз предметної області, ринку на наявність схожого програмного забезпечення, розроблено модель бази даних, порівняні стилі архітектури програмного забезпечення, розглянуті та обрані інструменти для реалізації.

Для розробки програмної системи використано мову програмування PHP, сервер бази даних MariaDB і Web-сервер Nginx.

У результаті проектування здійснена програмна реалізація Інтернет-платформи «Бюро знахідок» з підтримкою та використанням Google-карт.

1.06.2021р.
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	ДПІПЗ.170120.01.17.ПЗ	Пояснювальна записка	105		
2	А4		Завдання на дипломний проект	1		
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні слайди	16		

Змін	Арк.	№ докум.	Підпис	Дата	ДПІПЗ.170120.01.17.ПЗ		
Виконав		Стьопич В.В.		1.06	Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт	Літ.	Арк. 1	Аркушів 1
Керівник		Равелюк Г.І.		3.06				
Н. Контр.		Равелюк Г.І.		4.06	Відомість документів	ХНУ, ІПЗ-17-1		
Зав. Каф.		Бобрилюк Л.Р.		5.06				

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
СУБД	–	Система Управління Базами Даних
API	–	Application programming interface
CMS	–	Content Management System
HTTP	–	HyperText Transfer Protocol
UML	–	Unified Modeling Language
CSS	–	Cascading Style Sheets
PHP	–	Hypertext Preprocessor
CRUD	–	Create, Read, Update, Delete
UI	–	User Interface
URL	–	Uniform Resource Locator
IDEF	–	Integrated DEFinition
DFD	–	Data Flow Diagrams
UML	–	Unified Modeling Language
ER	–	Entity-Relationship
РСУБД	–	Реляційні Системи Управління Базами Даних
DTO	–	Data Transfer Object
SSH	–	Secure Shell
MIT	–	Massachusetts Institute of Technology
SEO	–	Search Engine Optimization
VPS	–	Virtual Private Server
VDS	–	Virtual Dedicated Server
IP	–	Internet Protocol
SSD	–	Solid-State Drive
HDD	–	Hard Disk Drive
GIT	–	Global Information Tracker

ВСТУП

Інтернет сьогодні – це не тільки спосіб весело провести дозвілля. Основна задача глобальної мережі – це допомога у вирішенні значної кількості задач. Найчастіше такими задачами є пошук і викладення будь-якої інформації у будь-якому вигляді. Нерідко така інформація не фільтрується, а тому для всіх користувачів видається багато непотрібної їм інформації.

Але серед цього всього знаходиться прохання про допомогу, наприклад, пошук втрачених речей, тварин тощо. Даний вид інформації відноситься до такої категорії, яку користувач переглядає і гортає далі. І це ще добре, якщо прочитають, а не прогортають. Саме це стає причиною того, що власник загубленої речі більше ніколи її не побачить.

Актуальність теми дипломного проекту (ДП) полягає в тому, що на сьогоднішній день існує така проблема: користувачі соціальних мереж та різноманітних месенджерів часто бачать такі типи постів, як: «Шукаю (річ, домашнього улюбленця, людину, документ)» або «Знайшов ..., поверну».

Оскільки ці пости розміщуються у стрічці, то вони мають таку властивість як губитися. Буде достатньо, щоб загубитися десь в стрічці звичайного користувача з кількістю підписників більше 100. А отже шанс того, що втрачена/знайдена річ повернеться до власника, зменшується.

Саме тому, потрібно створити спеціальну платформу, де буде зберігатися такий тип даних. Особливо зручною формою для реалізації буде веб-сайт, оскільки для роботи з ним не потрібно встановлювати додаткового програмного забезпечення (ПЗ); потрібні лише браузер і доступ до мережі Інтернет. Зручність використання такої Інтернет-платформи буде помітна відразу: можна дістати телефон, зайти на платформу і тут же оформити заяву про втрачену річ чи знахідку.

Отже, метою проекту є створення Інтернет-платформи, яка призначена для допомоги громадянам України у віднаходженні втрачених речей.

Основними завданнями при виконанні дипломного проекту є:

– провести аналіз предметної області та виявити її особливості;

– провести аналіз ринку на наявність готового вирішення поставленої проблеми дипломного проекту;

– обрати стиль серверної архітектури розроблюваної Інтернет-платформи;

– обрати інструменти та технології для розробки, які зможуть задовільнити поставлені вимоги;

– реалізувати серверну та клієнтську частини Інтернет-платформи;

– написати інструкцію користування Інтернет-платформою;

– розписати про процес запуску Інтернет-платформи на хостингу;

– провести модульне тестування системи на наявність недоліків та відповідність поставленим вимогам.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Інтернет-платформа «Бюро знахідок» – це система, в якій зберігається архів даних про будь-які речі, які були втрачені або знайдені. Так як це Інтернет-платформа, то це означає, що доступ до системи може отримати будь-який пристрій з доступом до мережі інтернет.

Розглянемо в чому ж заключаються переваги та недоліки використання саме веб-версії перед традиційним бюро знахідок. Переваги:

– доступ в будь-який час, в будь-який момент, в будь-якому місці;

– швидкий доступ до будь-якого оголошення завдяки зручному фільтру пошуку на сайті;

– так як це веб-версія, то непотрібно шукати приміщення для зберігання речей, достатньо купити місце на хостигу.

Недоліком веб-версії можна вважати постмодерацію. Платформа може обслуговувати команда модераторів але цього може бути все одно мало оскільки дія додатку розповсюджується на всю територію країни. А отже і кількість заяв може сягати великих значень. Саме через це публікація заяви може зайняти деякий час.

Далі буде наданий графічний опис системи. Для цього буде використана методологія IDEF0. З її допомогою можна дослідити функції системи [1].

Взаємодія робіт із зовнішнім світом і між собою описується за допомогою стрілок. Стрілки являють собою якусь інформацію (або об’єктами) й називаються іменниками. В IDEF0 розрізняють п'ять видів стрілок.

Вхід – об'єкти, що використовуються і перетворюються роботою для одержання результату (виходу). Стрілка входу зображається як вхідна в ліву частину роботи на діаграмі.

Керування – інформація для управління роботою. Будь-яка робота повинна мати хоча б одну стрілку керування, що зображується як вхідна у верхню частину роботи.

Вихід – об'єкти, в які будуть перетворені вхідні об’єкти. Кожна робота повинна мати хоча б одну стрілку виходу, що зображається як вихідна із правої частини роботи.

Механізм – об’єкти, що виконують роботу. Стрілка механізму зображається як вхідна в нижню частину роботи. Діаграма може не містити стрілок механізмів.

Виклик – стрілка, що вказує на іншу модель роботи. Стрілка виклику зображається як вихідна з нижньої частини роботи й використовується для вказівки того, що деяка робота виконується за межами, системи, яка моделюється. На рисунках буде зображено контекстну IDEF0 діаграму та діаграми декомпозицій (рисунки 1.1 – 1.3).

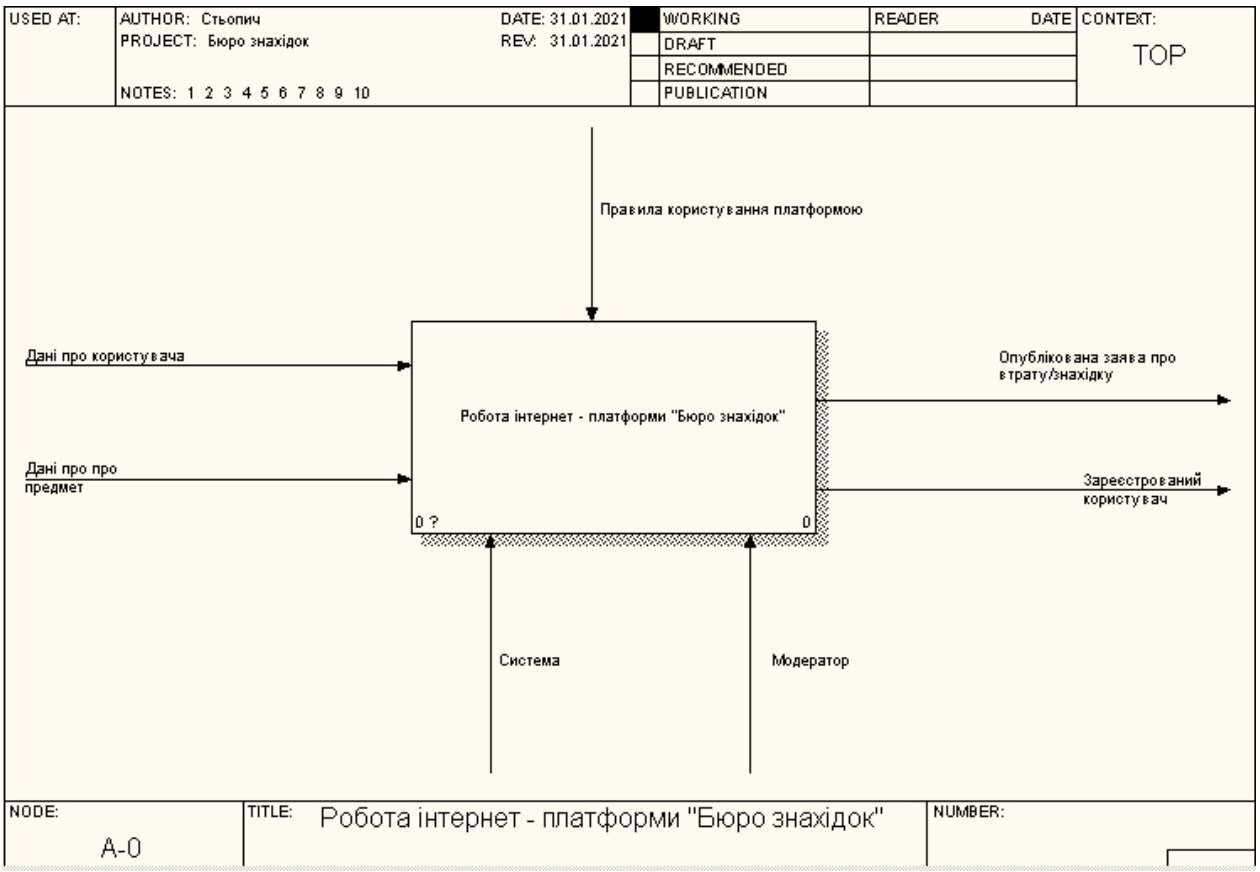


Рисунок 1.1 – Контекстна IDEF0-діаграма роботи «Бюро знахідок»

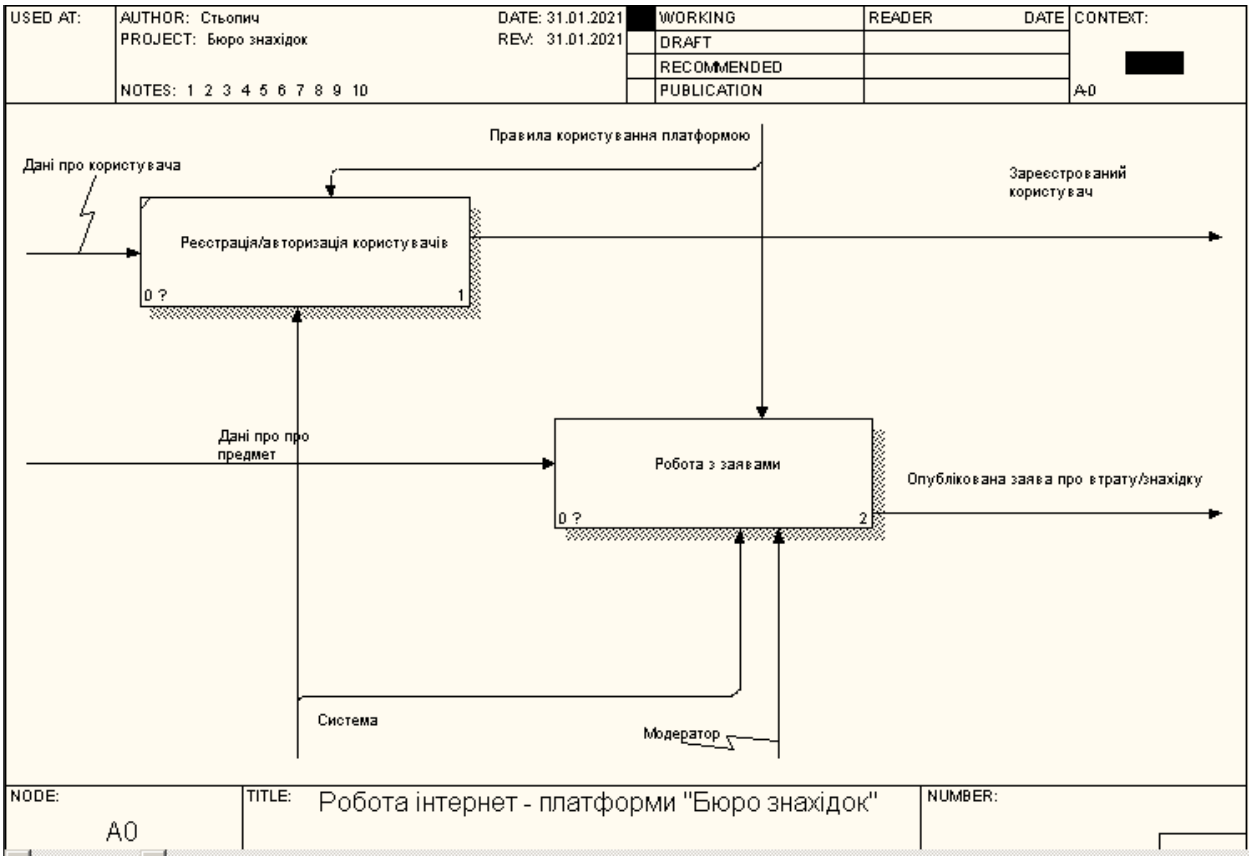


Рисунок 1.2 – IDEF0-діаграма декомпозиції першого рівня

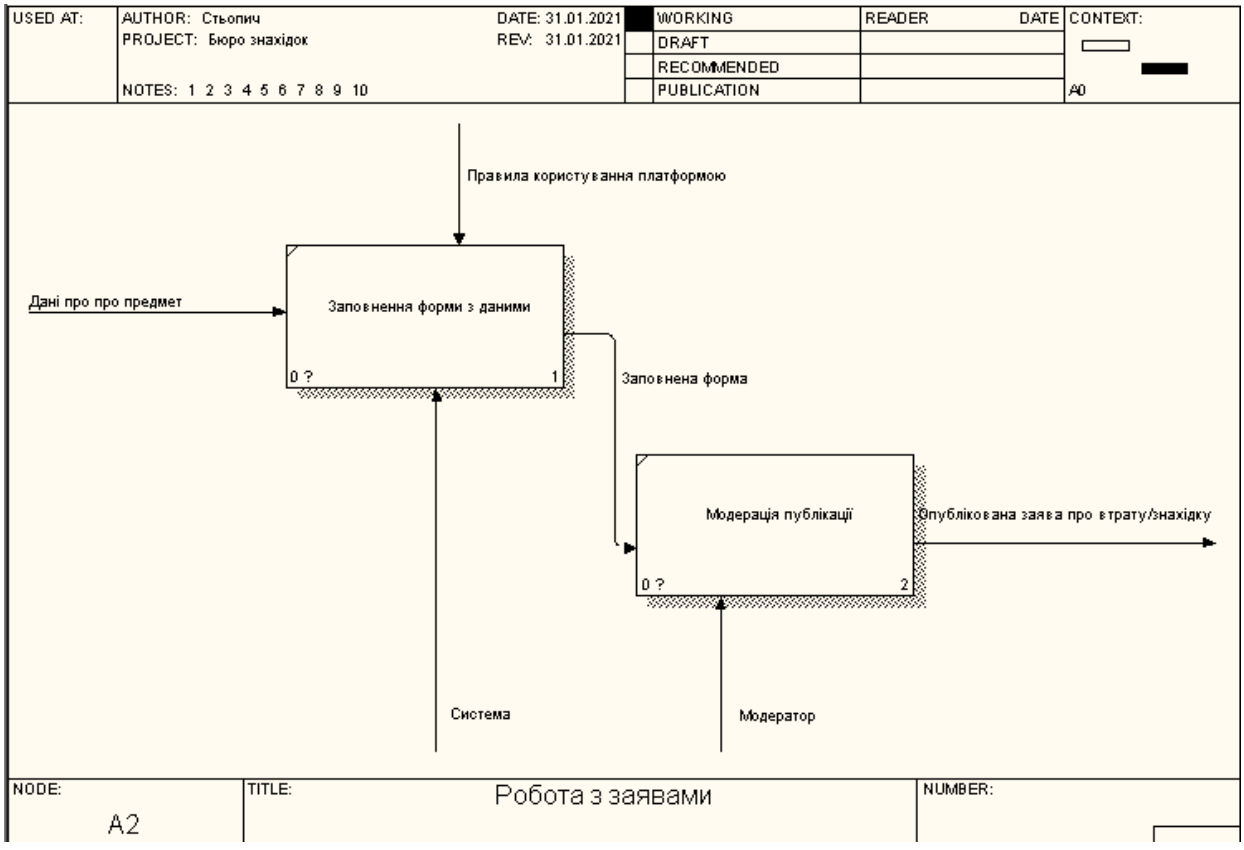


Рисунок 1.3 – IDEF0-діаграма декомпозиції другого рівня

Для відображення потоку даних в системі будуть спроектовані DFD-діаграми. Для кожного процесу була спроектована своя діаграма. На рисунках 1.4 та 1.5 показана DFD-діаграма для процесу авторизації / реєстрації.

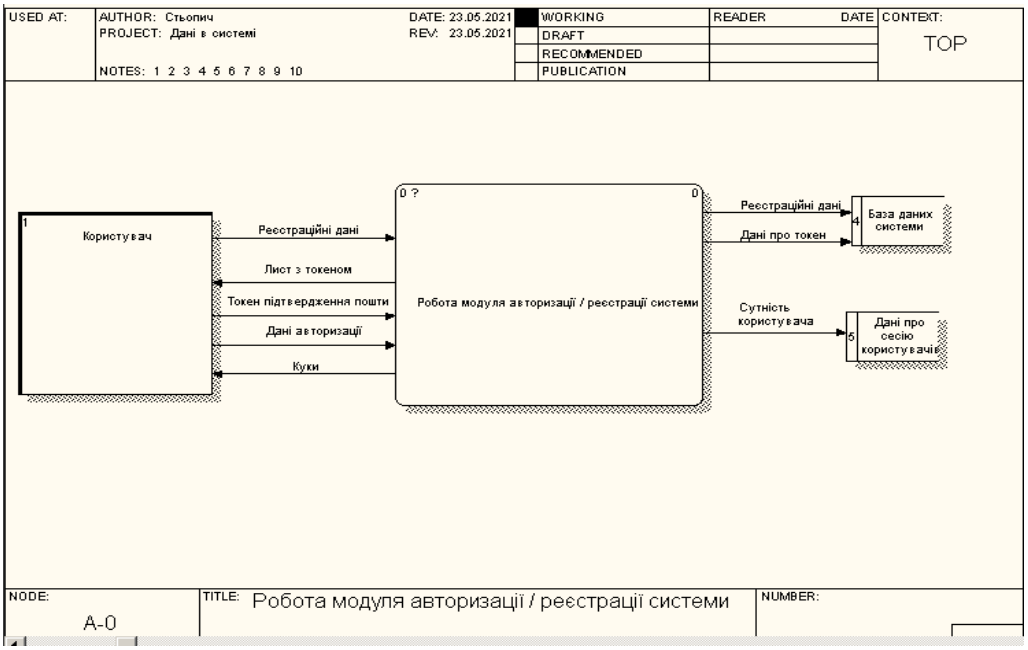


Рисунок 1.4 – DFD-діаграма верхнього рівня декомпозиції для процесу авторизації / реєстрації

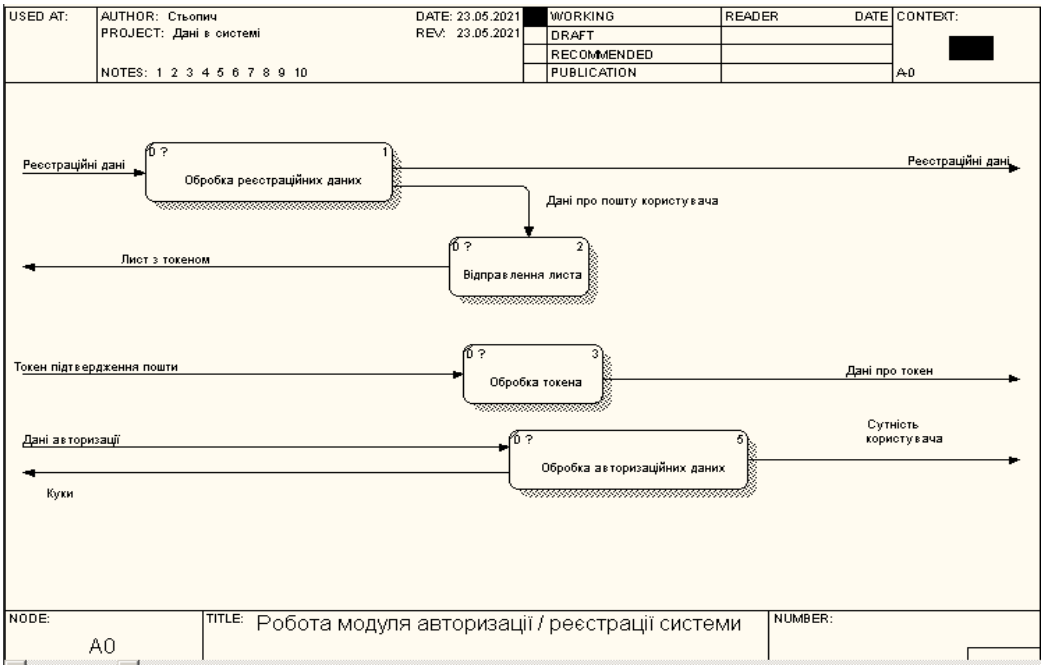


Рисунок 1.5 – DFD-діаграма другого рівня декомпозиції для процесу авторизації / реєстрації

Процес публікації оголошення відбувається з допомогою декількох модулів системи. На рисунках 1.6 та 1.7 зображений процес публікації від початку створення оголошення до його появи на сторінці сайту.

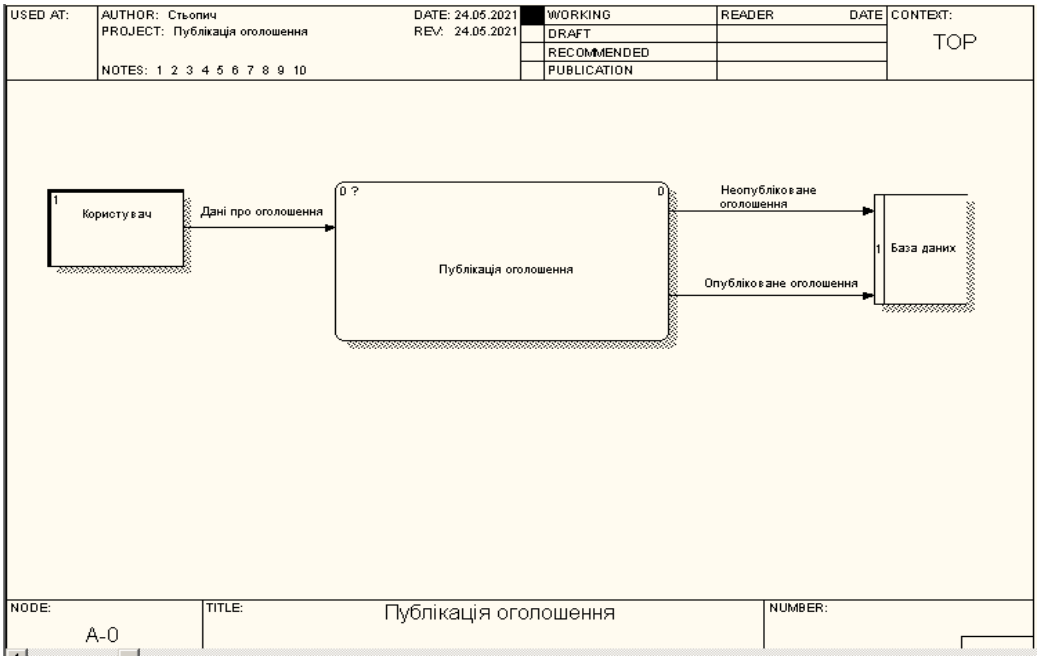


Рисунок 1.6 – DFD-діаграма верхнього рівня декомпозиції для процесу публікації оголошення

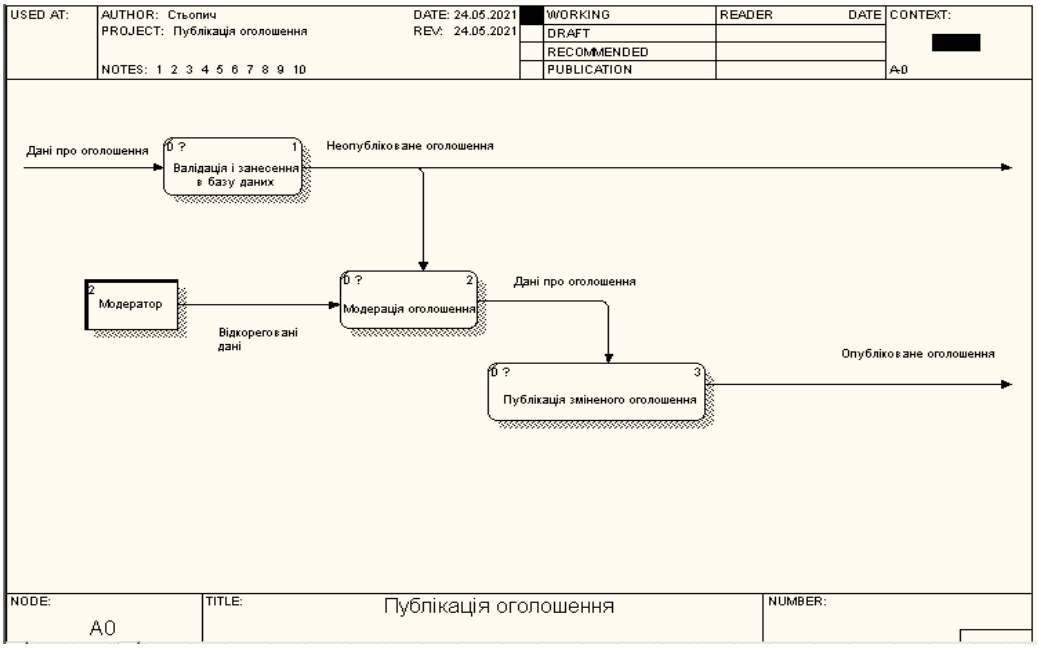


Рисунок 1.7 – DFD-діаграма другого рівня декомпозиції для процесу публікації оголошення

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

На сьогоднішній день, спеціалізованих інструментів для пошуку втрачених речей на ринку дуже небагато. Зазвичай, люди використовують соціальні мережі або розміщують оголошення на онлайн-дошках в Інтернет-сервісах. Розглянемо приклади таких рішень.

OLX.ua – одна з лідерів серед онлайн-дошок для об’яв, яка відома більше ніж в 40 країнах світу. OLX надає швидкий доступ до покупки, продажу або обміну товарів шляхом публікації заяв. Як ви вже зрозуміли, OLX – це не зовсім те місце, де можна було б розміщувати заяви про втрачені речі, однак на сайті можна заявити про втрачених тварин (рисунок 1.8). На жаль, таке доступне тільки для розділу з тваринками. Для того, щоб залишити заяву, потрібну зареєструватися на платформі.

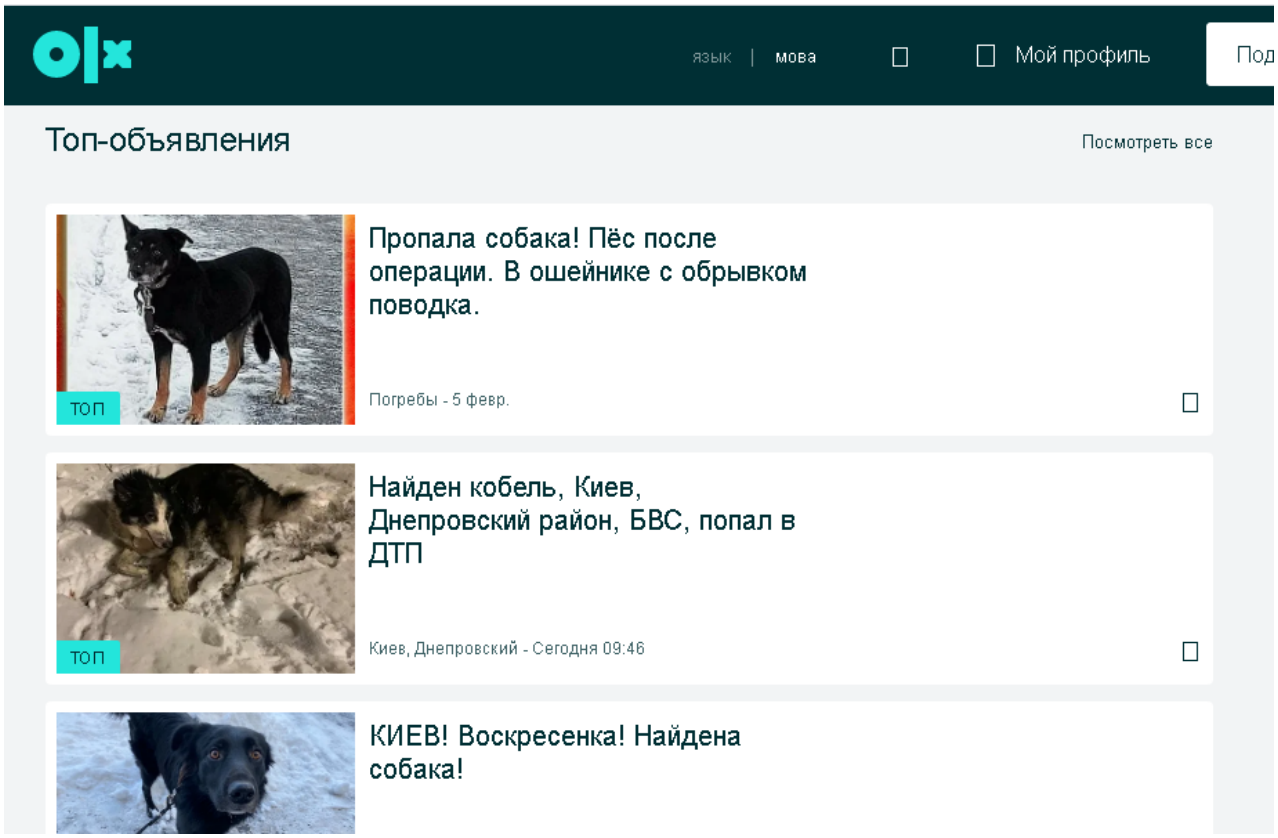


Рисунок 1.8 – Сторінка на OLX.ua

Тепер розглянемо як виглядає пошук втрачених речей в месенджерах або соціальних мережах. Візьмемо, до прикладу, онлайн-сервіс Інстаграм, а також аккаунт khmfind, так як в ньому часто роблять оголошення, зв'язані з пропажами. (рисунки 1.9 – 1.11). Специфіка роботи такого сервісу полягає в наступному: оголошення Інстаграму мають властивість зникати після 24-ох годин. Із–за цього, якщо об'єкт не був знайдений, доводиться робити ще одне таке ж оголошення. Адміністрація не зобов'язана слідкувати за статусом пошуку, а тому, зазвичай, робиться максимум одне оголошення.

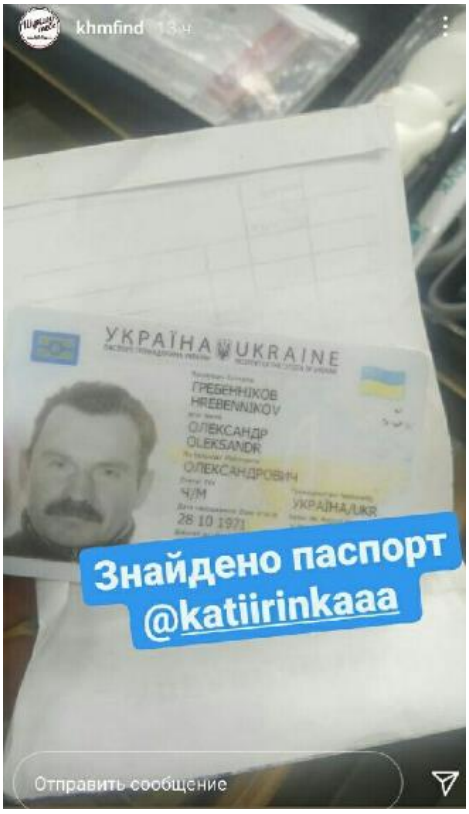


Рисунок 1.9 – Втрачений паспорт

Проведемо короткий аналіз:

Месенджери / соціальні сторінки: пости мають властивість губитися / втрачатися. Постмодерація може заняти значно більше часу, якщо це не спеціалізована група / аккаунт. Мала поширеність.

Онлайн дошки оголошень: не на всіх дошках є можливість робити оголошення про втрату / знахідку так як, зазвичай, дошки – це місце для бізнесу.

Рисунок 1.10 – Втрачений домашній улюбленець

Рисунок 1.11 – Втрачені ключі

1.3 Визначення вимог до програмного забезпечення та розробка технічного завдання

Для визначення вимог до програмного забезпечення використовується UML – мова моделювання, призначена для побудови діаграм, визначення, візуалізації і документування моделей, орієнтованих на об’єкти програмного забезпечення. За допомогою UML можна переглянути, як скомпонувати систему, а також полегшити співпрацю розробників [2].

Діаграма варіантів використання описує зв’язки і залежності між групою варіантів використання і акторами, що беруть участь у процесі. На діаграмах зображується, що повинна робити система.

У таблиці 1.1 наведено опис основних акторів, а в таблиці 1.2 – опис варіантів використання Інтернет-платформи і акторів, що взаємодіють з ними.

Таблиця 1.1 – Опис акторів

Актор	Короткий опис
Незареєстрований користувач	Може шукати і переглядати інформацію на сайті, може зв’язатися з автором посту, не може робити пости.
Зареєстрований користувач	Даний користувач може виконувати такі ж дії, що й незареєстрований. Але крім цього, він може створювати власні оголошення в своєму кабінеті, керувати ними.
Адміністратор	Управляє системою в цілому: має можливість управляти ролями користувачів в системі, назначати модераторів, банити юзерів, робити CRUD операції з постами.
Модератор	CRUD операції з постами, перегляд постів.

Таблиця 1.2 – Опис варіантів використання

Актор	Найменування ВВ	Опис ВВ
Незареєстрований користувач	Реєстрація в системі	Користувач може зареєструватися в системі, ввівши свої дані.
Зареєстрований користувач / Модератор / Адміністратор	Авторизація в системі	Користувач може авторизуватися в системі, ввівши дані.
Зареєстрований користувач	Редагування інформації в особистому кабінеті	Користувач може редагувати особисту інформацію, змінювати пароль.
Всі ролі	Пошук/перегляд інформації	Всі можуть переглядати інформацію з постів.
Адміністратор	Редагування ролей	Адміністратор може додавати, редагувати і видаляти права і ролі для користувачів.
Модератор / Адміністратор	Модерація постів	Модератор / Адміністратор можуть додавати, редагувати і видаляти пости.
Модератор / Адміністратор	Модерація користувачів	Модератор / Адміністратор можуть заблоковувати облікові записи користувачів.
Зареєстрований користувач	Створення і редагування постів	Зареєстрований користувач може додавати і редагувати пости.

Відповідно до описаних акторів і варіантів використання побудовано діаграму, зображену на рисунку 1.12.

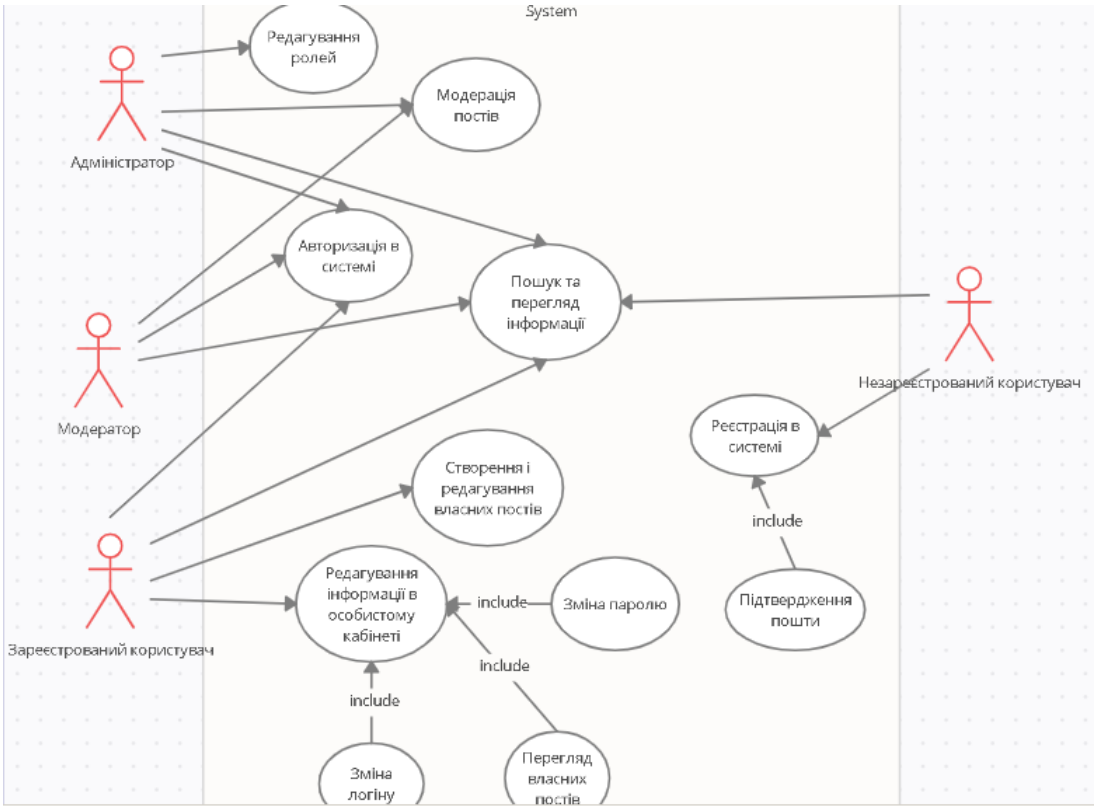


Рисунок 1.12 – Діаграма варіантів використання Інтернет-платформа «Бюро знахідок»

На основі проведеного аналізу поставлених вимог до ПЗ було розроблено технічне завдання, яке описане у додатку А.

Отже, у розділі проведений аналіз існуючого програмного забезпечення, яке допомогло б вирішити проблему за темою дипломного проекту. В результаті аналізу було визначено, що на ринку майже немає такого рішення, яке повністю задовольнило б поставлені вимоги. Також сформовані функціональні і нефункціональні вимоги до розроблюваної системи, описані основні варіанти використання та потоки даних в системі.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та вибір архітектури Інтернет-платформи

Для реалізації додатку була обрана клієнт-серверна архітектура. Основою такої архітектури є спілкування між клієнтом (будь–який пристрій, який може посилати запити) та сервером (пристроєм, який обробляє запити і віддає відповіді для клієнтів). Сервер може обробляти декілька запитів одночасно, однак якщо їхня кількість стає занадто великою, то вони ставляться в чергу.

Для комунікації між собою клієнт та сервер використовують протоколи передачі даних. Найпоширенішими є HTTP та WebSocket [3].

WebSocket – протокол зв’язку між клієнтом та сервером, який призначений для встановлення двостороннього зв’язку, тобто клієнт та сервер обмінюються повідомленнями в реальному часі. Найчастіше його використовують, коли дані треба передавати моментально. Наприклад, його використовують в чатах, відеоіграх, на стрімінгових платформах.

HTTP – протокол, який застосовується для передачі даних по принципу запит-відповідь. Кожен запит клієнта – це новий сеанс, а це означає, що для ідентифікації клієнта, йому необхідно разом з даними посилати ще й ідентифікаційні дані як, наприклад, куки в браузері.

Кожен HTTP запит містить в собі заголовки та тіло. Заголовки містять в собі різну технічну інформацію та метадані для запиту. Наприклад, це може бути адреса запиту, мова браузера користувача, інформація про браузер, тощо. Також в заголовки входить така інформація, як HTTP-метод, який вказує, що необхідно зробити з даними. Розглянемо основні методи:

– GET – запит вмісту вказаного адресу. Запит може містити в собі додаткові параметри, які розміщуються в URI стрічці;

– POST – передає дані користувача заданому ресурсу. Дані будуть знаходитися в тілі запиту;

– PATCH – завантажує певну частину даних на сервер;

– PUT – завантаження вказаного ресурсу на сервер;

– DELETE – видалення вказаного ресурсу.

На основі огляду протоколів спілкування між клієнтом та сервером був зроблений висновок, що протокол HTTP є кращим вибором для реалізації Інтернет-платформи, оскільки для обміну даними не потрібна можливість постійного зв’язку.

2.2 Реалізація серверної архітектури

Для того, щоб створити проект, який може забезпечити потрібний функціонал і при цьому не завдати великих збитків при подальшій його підтримці, потрібно правильно обрати архітектуру. Розглянемо монолітну та мікросервісну стилі архітектури.

Монолітна архітектура (рисунок 2.1) – концепція цієї архітектури полягає в тому, що всі компоненти додатку об’єднані в одну платформу. Такий додаток складається з бази даних, клієнтського інтерфейсу та серверного додатку. Монолітна архітектура зручна, якщо планується розробка невеликого додатку і на який планується виділити невелику групу розробників.

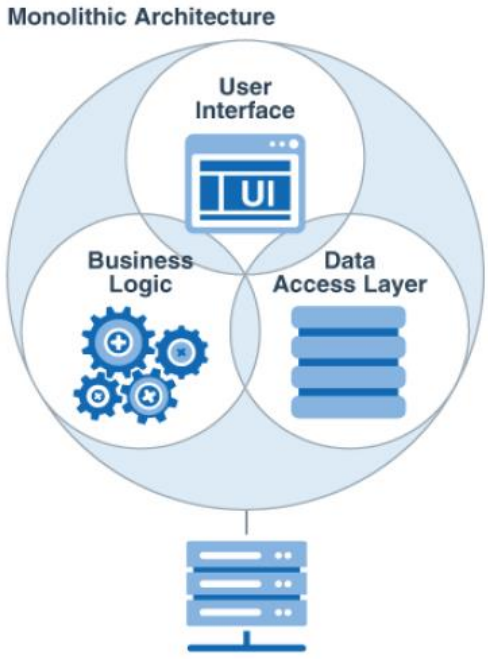


Рисунок 2.1 – Монолітний стиль архітектури

Розглянемо переваги:

– простіша розробка та розгортання; всі компоненти розташовуються в одному місці, а оновлення, якщо і відбувається, то для всієї системи цілком;

– краща продуктивність; якщо правильно побудувати додаток, то його ефективність та працездатність будуть вищими ніж, наприклад, мікросервісна архітектура, за рахунок спільного використання ресурсів машини програмою.

Недоліком монолітної архітектури можна назвати погану розширюваність кодової бази. Будь-який проект має властивість розширюватися в плані функціоналу. Чим більше нового функціоналу в проекті, тим складніше стає розбиратися в проекті.

Мікросервісна архітектура (рисунок 2.2) – концепція даної архітектури заключається в тому, що додаток складається з декількох незалежних компонентів, які спілкуються через API.

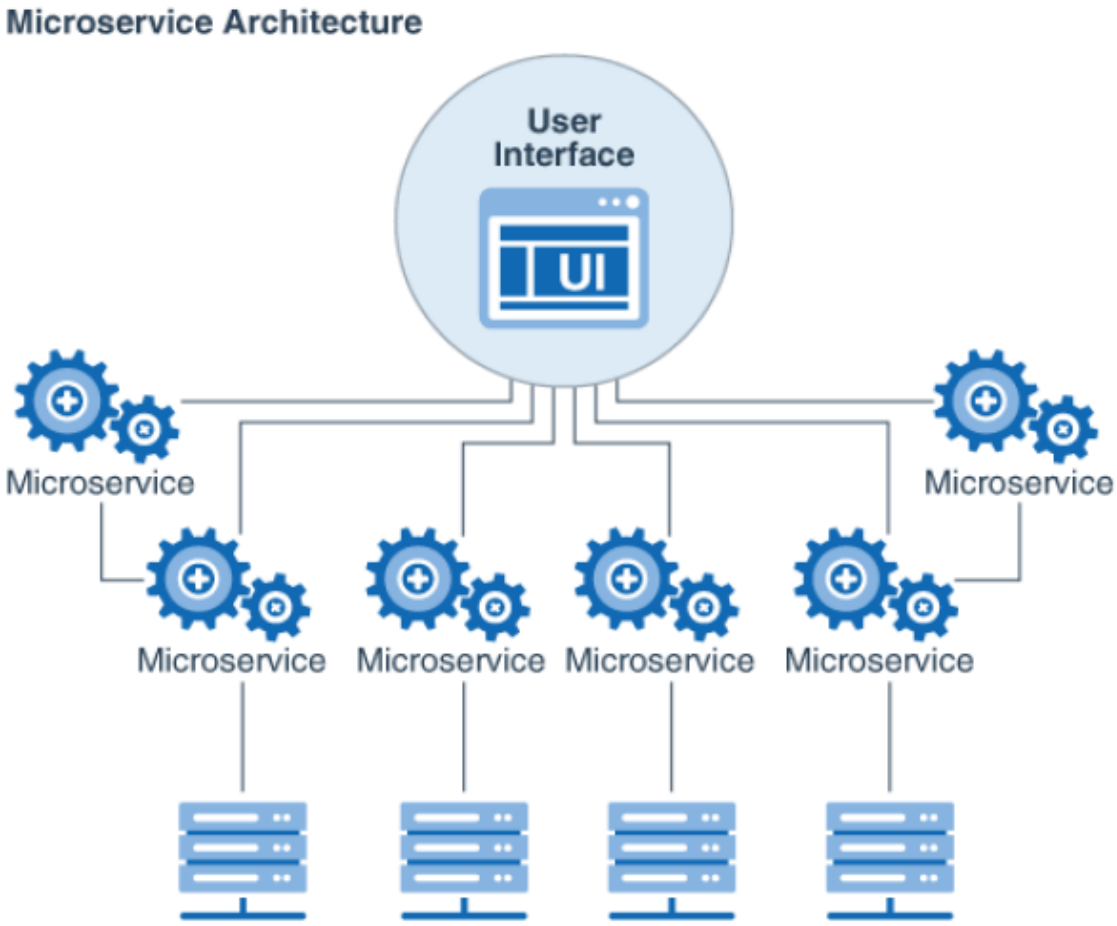


Рисунок 2.2 – Мікросервісний стиль архітектури

Головною перевагою мікросервісів можна назвати їхню незалежність один від одного. Дана властивість дозволяє використовувати різні технології для кожного мікросервіса в проекті. Також незалежність надає таку властивість як стабільність системи. Якщо один мікросервіс перестане працювати, то це ніяк не повпливає на працездатність системи в цілому.

Однак незалежність мікросервісів є і головним недоліком. По-перше, систему складніше протестувати. Для початку потрібно провести тестування кожного мікросерсвісу, а вже потім всієї системи в цілому. По-друге, для функціонування системи, потрібно налаштувати коректну комунікацію між мікросервісами, що може зайняти багато часу. Чим більше мікросервісів використовує система, тим важче налаштовувати взаємодію.

Висновок: зробивши огляд даних стилів побудови архітектури, а також проаналізувавши їхні переваги та недоліки [4], була обрана монолітна архітектура оскільки вона не потребує зайвих витрат ресурсів на розгортання та розробку. Також дипломний проект не є дуже великим, а тому не буде відчутно мінусів використання даного стилю архітектури.

2.3 Опис структури даних та моделі бази даних

Модель «сутність-зв'язок» (ER-модель) – модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків [5]. ER-модель – це мета-модель даних, тобто засіб опису моделей даних. Існує ряд моделей для представлення знань, але одним з найзручніших інструментів уніфікованого представлення даних, незалежного від програмного забезпечення що його реалізує, є модель «сутність-зв'язок».

ER-модель зазвичай реалізується у вигляді баз даних. В реляційних базах даних, кожен рядок таблиці являє собою екземпляр сутності. В не реляційних БД, сутністю буде окремий документ. Для зв’язку з іншими таблицями використовується зовнішній ключ-ідентифікатор.

Сутність – об’єкт, що має значення для аналізованої предметної області,

інформація про який підлягає збереженню. Кожна сутність має унікальний ідентифікатор. Кожний екземпляр сутності однозначно ідентифікується і відрізняється від усіх інших екземплярів даного типу сутності.

Зв'язок – це асоціація між двома або більше сутностями, при якій, як правило, кожний екземпляр однієї сутності, може асоціюватися із довільною кількістю екземплярів залежної сутності, при цьому залежна сутність може асоціюватися тільки з одним батьківським екземпляром сутності.

Атрибут – будь-яка характеристика сутності, призначена для кваліфікації, ідентифікації, класифікації, кількісної характеристики або відображення стану сутності. Кожна сутність (якщо це не слабка сутність) має мати мінімальний набір унікальних атрибутів, що зветься первинним ключем. Первинний ключ може складатися як з одного атрибуту так і з декількох. В другому випадку ключ буде називатися композитним.

На початковому етапі проектування логічної моделі програмної системи були виділені декілька основних сутностей.

User – користувач, який надав свої персональні дані для реєстрації в системі. Атрибути сутності:

– id – унікальний ідентифікатор користувача;

– username – ім'я користувача;

– email – пошта користувача;

– password_hash – хеш паролю;

– auth_key – ключ авторизації;

– status – статус користувача (0 – якщо це заблокований аккаунт, 1 – має статус активного користувача);

– phone – номер мобільного користувача;

– created_at – дата створення аккаунту;

– updated_at – дата оновлення данних аккаунту.

User_token – сутність для токену користувачу. Використовується для видачі користувачам токенів підтвердження пошти. Атрибути сутності:

– id – унікальний ідентифікатор токена;

– type – тип токена;

– token – сам токен;

– expire_at – час, скільки токен може жити;

– status – статус токена;

– created_at – дата створення токена;

– updated_at – дата оновлення токена;

– user_id – сутність користувача.

Posts – сутність оголошення втрати / знахідки, яку може опублікувати користувач. Атрибути сутності:

– id – унікальний ідентифікатор посту;

– title – назва посту;

– text – опис посту;

– coordinates – Координати для Google-карт;

– type – за допомогою цього атрибуту можна визначити, до якого типу (втрата / знахідка) належить оголошення;

– reward – винагорода;

– created_at – дата створення посту;

– updated_at – дата оновлення посту;

– status – (0 – якщо неактивний, 1 - активний);

– user_id – сутність користувача.

Files – сутність завантаженого користувачем файла. Атрибути сутності:

– id – унікальний ідентифікатор файлу;

– base_path – URL домену;

– name – шлях до файлу;

– type – тип файлу;

– size – розмір файлу;

– entity_id – ідентифікатор сутності оголошення посту.

Таким чином, визначившись з основними сутностями розроблюваної Інтернет-платформи та їх атрибутами, була сформована модель бази даних, зображена на рисунку 2.3.

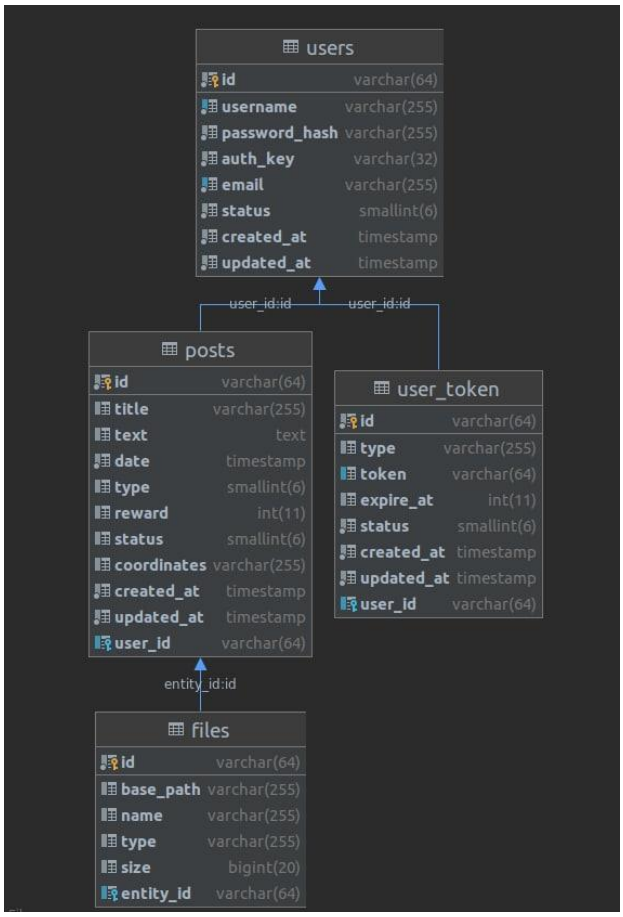


Рисунок 2.3 – Діаграма моделі бази даних

2.4 Аналіз та вибір бази даних

Для того, щоб обрати базу даних для збереження інформації, необхідно провести аналіз функціональності, який вони надають. На даний момент бази даних діляться на два типи: реляційні та не реляційні.

Реляційна база даних – це така база, яка зберігає дані з зумовленими зв’язками даних між ними. Ці дані організовані у вигляді таблиць. Стовбцями таблички є атрибути таблиць, які мають свій тип даних та значення. Рядки таблиці представляють собою набір значень, зв’язаних з одним об’єктом або сутністю. Кожна стрічка повинна мати унікальний ідентифікатор. Для зв’язку між таблицями використовуються зовнішні ключі.

Тепер розглянемо переваги даного типу баз даних:

– неймовірно велика кількість документації, підтримка спільноти;

– користувачу легше зрозуміти, як зберігаються дані, коли вони чітко та строго структуровані.

Найголовнішою перевагою РСУБД є зручний механізм транзакцій – виконання декількох операцій за один запит. У випадку успішної операції в БД відбудуться зміни для всіх запитів в транзакції. У випадку помилки – ніяких змін не буде зроблено. Щоб забезпечити підтримку транзакцій, БД повинні бути притаманні наступні властивості:

– Atomicity – транзакція виконується або повністю або ніяк;

– Consistency – до початку виконання транзакції та після її завершення, система має перебувати в узгодженому стані;

– Isolation – зміни, які були виконані під час транзакції, повинні бути недоступні для всіх;

– Durability – гарантія того, що у випадку збоїв в системі, всі запити транзакції будуть збережені.

Дані властивості називаються абревіатурою ACID – перша літера кожної властивості. Механізм транзакцій є надзвичайно важливим для систем, які працюють з складними даними.

Прикладами таких баз даних є MySQL, PostgreSQL, MariaDB.

Нереляційні бази даних – це такі бази даних, які можуть зберігати в собі дані не у вигляді таблиць, а різних видах структур. Наприклад, дані можуть зберігатися у вигляді документів json, відео-файлів, тощо.

Даний тип систем є більш зручним у використані за рахунок того, що в них відсутня необхідність в проектуванні таблиць та їх атрибутів. Замість цього, всі дані зберігаються в вказаному документі. Недоліком даних БД є відсутність повноцінного механізму транзакцій. Транзакції виконуються в межах тільки одного документа. Найпопулярнішими NoSQL базами даних можна віднести MongoDB, DynamoDB.

Проаналізувавши типи баз даних, було прийнято рішення обрати реляційний тип баз даних так як для виконання дипломного проекту необхідний

механізм транзакцій для збереження агрегатів даних.

Серед великого розмаїття реляційних баз даних було вирішено обрати MariaDB. Основною її перевагою у виборі стало те, що вона легша, ніж інші представники реляційних БД, а це означає, що система в буде витрачати менше ресурсів на підтримку працездатності. Окрім цього, MariaDB використовує кращі рушії Aria та XtraDB, які забезпечують більш стабільну та швидку роботу.

2.4 Проектування серверної частини Інтернет-платформи

Перед проектуванням серверної частини Інтернет-платформи, треба ознайомитися з таким поняттям як чиста архітектура [6].

Чиста архітектура – концепція розробки, яка має наступні ідеї:

– логіка додатку не залежить від інструментів розробки;

– логіка додатку не залежить від інтерфейсу клієнту;

– логіка додатку не знає про використовувану базу даних;

– логіка додатку взагалі нічого не знає про середовище проектування.

Дані принципи використовуються як основа для побудови складних додатків в найрізноманітніших сферах.

Така гнучкість досягається за рахунок розподілення додатку на шари:

– frameworks and drivers;

– interface adapters;

– use cases;

– entities.

Frameworks and drivers – середовище, в якому планується розроблювати проект. Є зовнішнім шаром, який складається з фреймворків, бібліотек, баз даних, веб-сервісів. Все, що знаходиться в цьому шарі, не повинно ніяким чином впливати на наступні шари.

Interface Adapters – шар адаптерів, які використовуються для конвертації отриманих від клієнта даних в такий формат, який зручно буде використовувати для шарів Use Cases і Entities. І так само навпаки. Наприклад, в цей сервіс

входить повністю весь шар представлення MVC моделі. Контролери, представлення, презентери – ключові складові цього шару.

Use Cases – код цього рівня складається з бізнес – правил. Він реалізує всі можливі варіанти використання системи. Use Cases відповідають за потік даних з Entities і навпаки для виконання необхідних дій. Єдині зміни, які можуть повпливати на цей шар – це зміни бізнес-правил. Всі інші зміни не повинні впливати на цей шар;

Entities – відповідає за бізнес-логіку проекту. Це можуть бути як і об'єкти з певними методами так і набір структур даних (до прикладу, DTO – Data Transfer Object) і функцій. В цьому шарі зосереджена основна високорівнева логіка. При будь-яких змінах, Entities – це останнє, чого ці зміни повинні торкнутися. В ідеалі, потрібно побудувати проект так, щоб Entities взагалі нічого не знали про зовнішній світ і так само потрібно зробити, щоб і зовнішній світ спілкувався з Entities тільки через Use Case.

Використовуючи ці принципи були спроектовані наступні модулі Інтернет-платформи:

– app – модуль, в якому буде описана логіка додатку, конфігураційні файли, міграції для баз даних (класи, з допомогою яких можна контролювати структуру бази даних);

– models – модуль, в якому будуть знаходитися моделі. Також тут будуть знаходитися всілякі частини моделей агрегатів (наприклад, модель поста є агрегатом, в якому містяться основні атрибути, а також модель, яка містить атрибути для представлення файлів користувача).;

– services – тут будуть знаходитися класи-сервіси, з допомогою яких можна керувати даними.;

– helpers – допоміжний функціонал системи. Наприклад, хелпери – статичні функції для допоміжної обробки даних;

– widgets – віджети системи. З їх допомогою можна формувати додаткові статичні дані, які будуть відображатися в представленнях;

– views – модуль, в якому знаходяться представлення.

2.6 Проектування клієнтської частини Інтернет-платформи

Кожен веб-додаток, складається з частин, які являються загальними для всіх сторінок та активних діалогів з користувачем. До таких частин належать:

– хедер (шапка сайту);

– основна частина;

– футер (підвал сайту).

Хедер – це верхня частина будь-якого сайту. В маркетингу і веб-дизайні шапка сайту виступає першим елементом на який звертають увагу користувачі. Саме тому важливо, щоб хедер був водночас легким для запам'ятовування та простим в плані функціоналу адже саме в шапці зазвичай розміщують такі елементи як меню сайту, навігаційну форму, кнопки для керування авторизацією користувача.

Основна частина – це основна частина сайту. Сюди вставляють весь контент сайту. Містить в собі різноманітні форми, тексти, інтерактивні одиниці (як, наприклад, всілякі пости користувачів).

Футер – розміщується в самому низу сайту. Зазвичай, в ньому розміщують посилання на інші ресурси сайту, дублюють основні пункти меню, форма підписки по пошті, посилання на соціальні мережі.

Кожна з цих структурних одиниць повинна мати таку властивість як відцентрованість на сторінці. Центрування контенту допомагає користувачу не сильно розсіювати свою увагу, а це в свою чергу означає менше навантаження на зорову систему та економія часу на перегляд контенту.

Тепер приступимо до проектування UI. Для адміністративної панелі було вирішено використати вже готовий шаблон AdminLTE (рисунок 2.3), який розповсюджується за MIT-ліцензією.

Даний шаблон включає в собі готові стилі Bootstrap та готовий пакет JQuery скриптів для функціонування шаблону. Також для управління оголошеннями та користувачами буде використовуватися стандартна Bootstrap форма AdminLTE. Для проектування дизайну сторони користувача буде

використовуватися метод прототипування. Для початку, буде описаний хедер. В хедері повинні знаходитися логотип та пункти меню (рисунок 2.4). В футері повинен знаходитися копірайт сайту (рисунок 2.5).

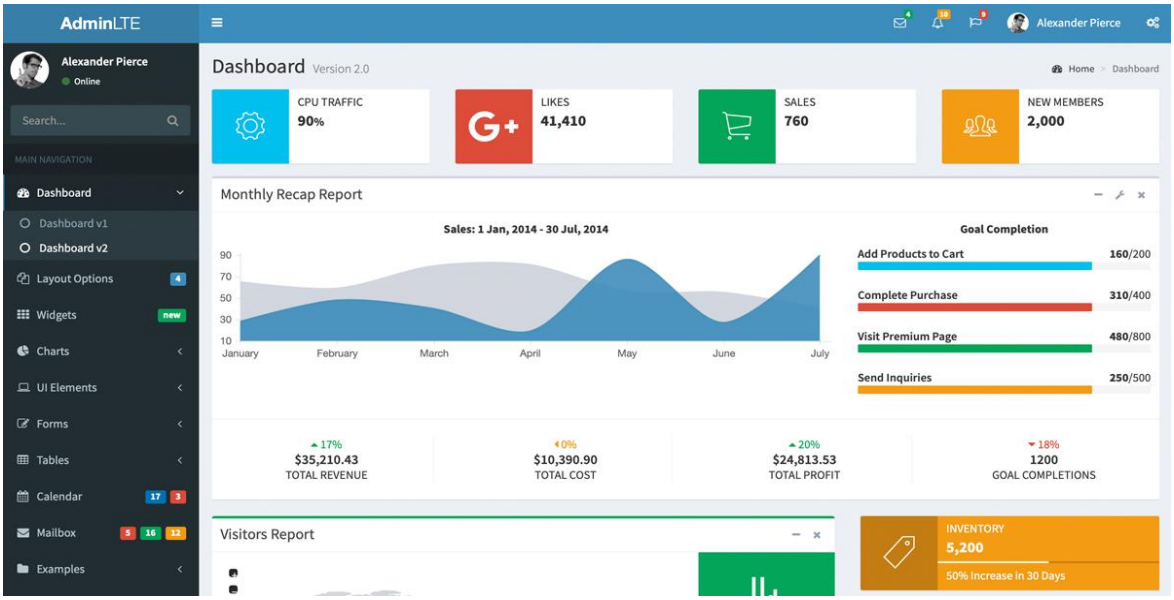


Рисунок 2.3 – Загальний вигляд шаблону AdminLTE

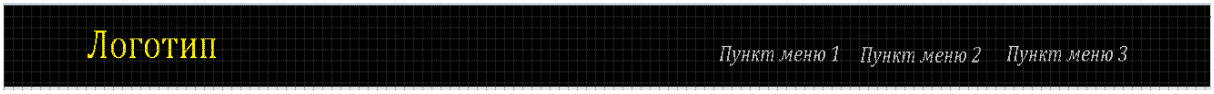


Рисунок 2.4 – Хедер сайту

Рисунок 2.5 – Футер сайту

Далі будуть спроектовані такі структурні одиниці як контейнери з оголошеннями. Взагалі їх буде два. Перший контейнер (рисунок 2.6) – буде знаходитися на головній сторінці Інтернет-платформи.

– зверху є назва даного контейнеру, яка оточена чорними стрічками;

– карточки оголошень. Кожна з них має фото, назву, дату публікації і кнопку-посилання з підписом «Детальніше»;

– кнопка посилання на сторінку з всіма оголошеннями.

Другий контейнер (рисунок 2.7) – знаходиться на сторінці з усіма оголошеннями і на сторінці особистого кабінету користувача. Цей контейнер має наступні особливості:

– зверху має підпис, що тут знаходяться всі оголошення користувачів;

– збоку зліва є фільтр, для кращої навігації;

– було вирішено виводити по 6 оголошень на сторінку задля зменшення навантаження на швидкість завантаження сторінки;

– використовуються такі ж карточки оголошень, що й в попередньому розробленому контейнері;

– знизу використовується Bootstrap панель пагінації, яка допоможе в переході між сторінками.

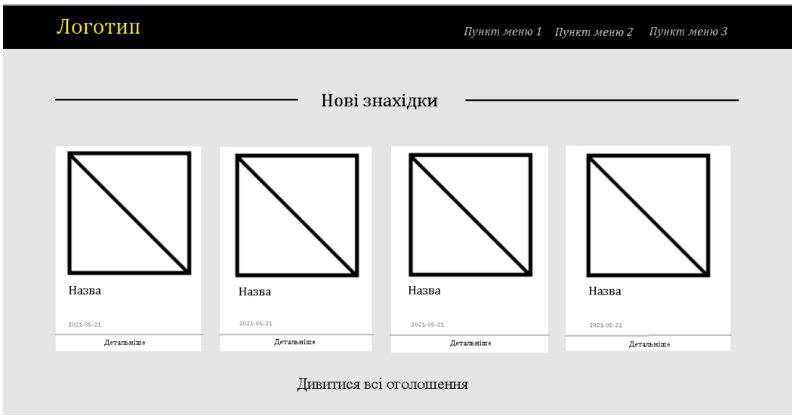


Рисунок 2.6 – Дизайн контейнеру на головній сторінці

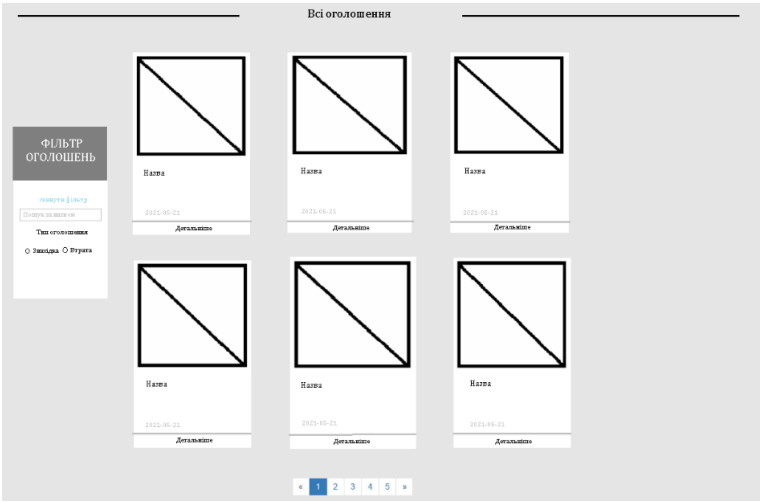


Рисунок 2.7 – Контейнер для сторінки з усіма оголошеннями

Для сторінки кабінету було вирішено використовувати той же самий контейнер, що й для сторінки з усіма оголошеннями, але є пара відмінностей:

– в фільтрі появиться пункт «Статус оголошення». Завдяки йому, можна глянути, які оголошення знаходяться на модерації, а які вже опубліковані;

– надпис зверху буде змінений на «Ваші оголошення».

Для перегляду контенту оголошення користувача було запропоновано використовувати наступну структуру (рисунок 2.8):

– назва оголошення;

– фото, які завантажив користувач;

– опис оголошення;

– винагорода;

– дата публікації;

– телефон;

– електронна пошта;

– місце на карті.

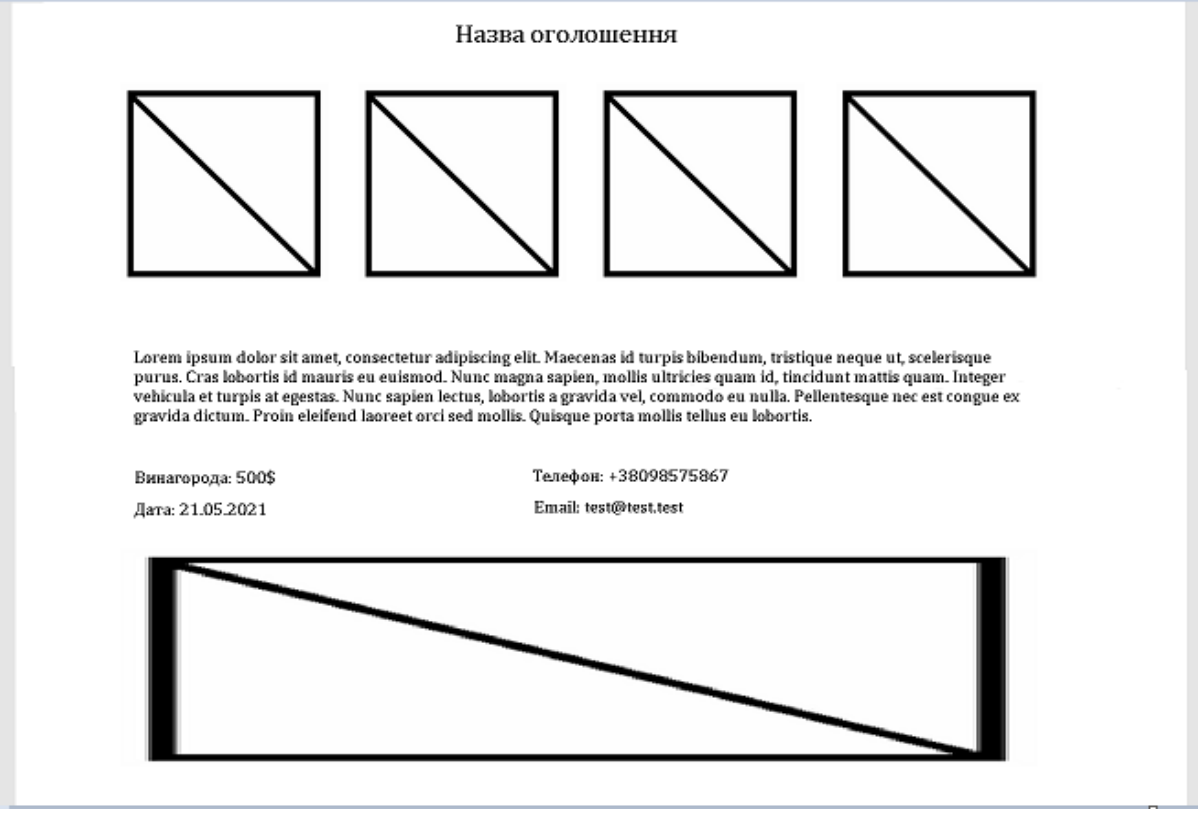


Рисунок 2.8 – Прототип сторінки перегляду контенту оголошення

2.7 Аналіз та вибір технологій і методів реалізації Інтернет-платформи

Перед розробкою потрібно вирішити в якому середовищі буде працювати додаток. Розглянемо такі інструменти як Vagrant та Docker.

Docker – програмне забезпечення для автоматизації розгортання і керування додатками в середовищах, які підтримують контейнеризацію. Контейнеризація – це такий метод віртуалізації, при якому ядро операційної системи підтримує декілька ізольованих екземплярів середовища користувача замість одного [7]. За допомогою нього, можна формувати список модулів та налаштувань для контейнеризованого середовища, збілдити його і користуватися додатком. Контейнеризація дуже сильно економить час на розробку і розгортання додатку.

Vagrant – програмне забезпечення для створення віртуального середовища розробки. При налаштуванні vagrant'а, можна обрати, яка операційна система буде використовуватися, які модулі треба поставити, та як це все сконфігурувати [8]. Мінусом данної системи є те, що середовище розгортається повністю з нуля, що в умовах обмеженої пам'яті на машині є неприйнятним.

Отже, для формування середовища додатку буде використовуватися система контейнеризації Docker.

Мова програмування, на якій буде написаний додаток – PHP. Коротко пройдемося по його основним характеристикам.

Перш за все, PHP дозволяє максимально швидко і недорого розробляти веб-додатки, що стає для будь-якого клієнта з малим – середнім бізнесом вирішальним фактором, оскільки час – це гроші.

Швидко і недорого не означає, що і підтримувати кодову базу також стане тяжче. Сучасний PHP дає безмірно великий асортимент інструментів для розробки, який дозволить швидко впроваджувати новий функціонал в додаток [9].

Проведемо аналіз основних таких інструментів.

CMS (Content Management System) – це системи управління контентом сайту. Суть cms в тому, що для розробки свого сайту навіть не потрібно знати програмування. Все поставляється в зручному веб-інтерфейсі, достатньо зробити

декілька кліків мишкою і готово. Сайт з простенькою формою реєстрації, кошиком для товарів, коментарі під постами, та гарний кабінет користувача – можливості всіх сучасних cms. Однак є одне але. Хоча cms і надають широкий асортимент функціоналу, але по суті на цьому можливості користувача закінчуються. Спроби інтегрувати власний функціонал, який буде відрізнятися від стандартів обраної cms скоріше за все, що призведе до проблем з швидкістю виконання скриптів, виникнуть труднощі з SEO-оптимізацією.

Фреймворки – по суті, готовий скелет додатку, який можна модифікувати під свою бізнес-логіку. Надає повну свободу дій для виконання будь-якої забаганки. Отже, для виконання поставлених бізнес-задач був обраний фреймворк. Але ще потрібно обрати, який саме, оскільки для PHP існує дуже багато таких інструментів. Вибір впав на Yii Framework. По-перше, Yii має легку і зрозумілу архітектуру. Але це не означає, що фреймворк нав'язує даний стиль. Його можна налаштовувати так, як цього вимагають бізнес-правила проекту. По-друге, Yii надзвичайно розширюваний. Можна налаштувати або замінити практично будь-яку частину основного коду.

Дані властивості [10] чудово підійдуть для реалізації як серверної так і клієнтської частини дипломного проекту.

Тепер перейдемо до вибору фронтенд технологій. Інтернет-платформа являється відносно простим додатком з монолітною архітектурою, тому будуть використовуватися прості технології. А саме HTML, CSS, Bootstrap – для побудови компонентів сторінки, JS та бібліотека jQuery – для надання інтерактивності для Інтернет-платформи.

HTML (мова гіпертекстової розмітки) – це код, який використовується для структурування і відображення веб-сторінки та її контенту. Весь контент сторінок буде рендеритися на сервері з допомогою PHP та віддаватися клієнту у вигляді HTML-сторінки.

Cascading Stylesheets (CSS, каскадні таблиці стилів) – це код, що використовується для стилізації сайту. Верстка додатку буде виконуватися з використанням властивості css float та технології flexbox.

Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу. Завдяки йому пришвидшується розробка інтерфейсу додатку.

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування. Використовується для створення сценаріїв які, надають можливість на боці клієнта (наприклад, браузер), асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки, керувати браузером [11].

jQuery – популярна JavaScript-бібліотека з відкритим кодом. Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

Основне завдання jQuery – це надання розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також дана бібліотека надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Таким чином, було проведено аналіз технологій та сформовано програмний стек, за допомогою якого буде розроблятись програмна система.

Отже, в даному розділі було проведено аналіз архітектурних стилів програмного забезпечення таких як моноліт та мікросервіси, були визначені переваги та недоліки кожного з них, та визначено, що моноліт найкраще підходить для розробки дипломного проекту. Були проаналізовані бази даних , та вирішено, що в проекті буде використовуватися MariaDB, так як вона легша ніж інші РСУБД, а також вона краще оптимізована за рахунок використання кращих рушіїв.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Спроектувавши модель бази даних у другому розділі, можна приступати до її розробки в проекті. Однак перш за все, варто було б задатися питанням: а як же правильно керувати структурою бази даних? Як правильно створювати таблиці, міняти атрибути кортежів і щоб у випадку необхідності не потрібно було лізти в базу даних і міняти там все руками? Тут на допомогу прийде механізм міграцій.

Міграції – це щось на кшталт системи контролю версії для бази даних. За допомогою такого механізму, можна слідкувати і контролювати, що відбувається з структурою бази даних. Окрім цього, міграції дозволяють не тільки зручно вносити будь-які зміни, а ще й автоматизувати процес побудови потрібної структури бази даних.

Фреймворк Yii2, який був обраний для розробки проекту, надає зручний спосіб роботи з міграціями. Для того, щоб почати роботу, треба виконати команду php yii migrate/create {назва_міграції}. Перша міграція, яка буде виконана, перш за все створить табличку migration в базі даних. В ній буде зберігатися ім’я міграції, та час її створення. Порядок в якому будуть йти записи в таблиці грає роль: у випадку відкату змін, буде взятий останній запис, після цього фреймворк спробує знайти клас міграції з заданим іменем, віднайде метод, який буде описувати інструкції для виконання і спробує їх виконати. У разі помилки, буде викладена її причина та суть. Фрагмент коду міграції для створення таблиці users:

```
class m210419_150617_users_tables extends Migration
{
    public function safeUp()
    {
        //опції таблиці
        $tableOptions = 'CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci ENGINE=InnoDB';
        //створення таблиць
        $this->createTable('{{%users}}', [
            'id' => $this->string(64),
            'username' => $this->string()->notNull()->unique(),
```

```
            'password_hash' => $this->string()->notNull(),
            'auth_key' => $this->string(32)->notNull(),
            'email' => $this->string()->notNull()->unique(),
            'status' => $this->smallInteger()->notNull()->defaultValue(0),
            'created_at' => $this->timestamp()->defaultExpression('CURRENT_TIMESTAMP'),
            'updated_at' => $this->timestamp()->defaultExpression('CURRENT_TIMESTAMP'),
        ], $tableOptions);
        //додаємо унікальний ідентифікатор
        $this->addPrimaryKey('pk_id', '{{%users}}', 'id');
            }
}
```

Нижче наведений код для видалення таблички з бази даних:

```
public function safeDown()
{
   $this->dropTable('{{%users}}');
}
```

Таким чином, використовуючи міграції, буде створено структура бази даних, описана в другому розділі.

3.2 Розробка серверної частини Інтернет-платформи

Після того, як була спроектована монолітна архітектура, можна приступати до реалізації. Для того, щоб почати роботу над проектом, потрібно установити сам фреймворк YII2. Після установки фреймворку, буде доступна структура папок і файлів:

– assets – папка з асетами, які будуть зареєстровані в шарі представлення. З допомогою асетів будуть підключатися файли css та js;

– commands – папка з консольними командами, з допомогою яких можна керувати системою, використовуючи CLI;

– config – папка з налаштуваннями додатку;

– controllers – папка з контролерами. Тут можуть знаходитися REST та WEB контролери;

– mail/layouts – папка з представленнями для відправки листів;

– models – папка, в якій знаходяться моделі предметної області. Сюди входять пакети класів програм, які були описані другому розділі;

– runtime – папка, в якій знаходяться кеш, логи і дані для дебагу (відловлювання помилок) системи;

– tests – папка призначена для зберігання тестів системи;

– vagrant – папка, в якій зберігається налаштування для розгортання віртуальної системи. Так як при проектуванні дипломного проекту буде використовуватися Docker, то файли Vagrant використовуватися не будуть;

– views – папка, в якій будуть знаходитися файли, які репрезентують шар представлення в системі. Тут будуть знаходитися файли – шаблони і файли для динамічного виводу даних. Файли-шаблони – це такі файли, які є статичними і при відображенні на сайті вони якщо й змінюються то дуже рідко. Для підвищення продуктивності системи, дані файли будуть кешуватися;

– web – папка, в якій зберігається мультимедіа сайту. Сюди входить css, js, fonts, pictures, папка з закешованими системними асетами. Також в цій папці буде зберігатися найголовніше – фото з постів користувачів. Також в цій папці знаходиться файл index.php, який являється вхідною точкою системи;

– widgets – папка з віджетами та представленнями для них;

– docker-compose.yml – файл для збирання та описування docker-контейнерів і їхнє налаштування;

– yii – файл для ініціалізації виконання консольних команд;

– composer.json – файл для менеджера пакетів композера. Тут описується, які бібліотеки треба встановити, який розширення PHP повинні використовуватися в проекті;

– readme.md – файл для опису проекту;

– .gitignore – файл, в якому описується, які файли / папки не потрібно індексувати і відповідно не заливати в віддалений репозиторій GIT.

Тепер варто глянути на структуризацію пакетів проекту. Проект було поділено на пакети: admin, auth, cabinet. В пакеті admin будуть знаходитися моделі предметної області для роботи тільки з адміністративною панеллю. Пакет auth – пакет модулів для роботи з авторизацією та реєстрацією. Cabinet – пакет роботи модулів для кабінету користувача.

Тепер можна приступати до реалізації проекту. Для початку потрібно налаштувати проект. Для цього буде створений файл config.php, в якому будуть знаходитися необхідні параметри для роботи Інтернет-платформи:

```
<?php

$params = require __DIR__ . '/params.php';
$db = require __DIR__ . '/db.php';

$config = [
    'id' => 'basic',
    'name' => 'Blog',
    'basePath' => dirname(__DIR__),
    'bootstrap' => [
        'log',
    ],
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm'   => '@vendor/npm-asset',
    ],
    'components' => [
        'request' => [
            'cookieValidationKey' => 'hElkpnBu5P2_09t3gBV-ZDZ0WuxomGcS',
            'baseUrl' => '',
        ],
        'cache' => [
            'class' => 'yii\caching\FileCache',
        ],
        'user' => [
            'identityClass' => 'app\models\entities\UserIdentity',
            'loginUrl' => ['site/auth/login'],
            'enableAutoLogin' => true,
        ],
        'errorHandler' => [
            'errorAction' => 'frontend/blog/error',
        ],
        'mailer' => [
            'class' => 'yii\swiftmailer\Mailer',
            'useFileTransport' => true,
        ],
        'log' => [
            'traceLevel' => YII_DEBUG ? 3 : 0,
            'targets' => [
                [
                    'class' => 'yii\log\FileTarget',
                    'levels' => ['error', 'warning'],
                ],
            ],
        ],
        'db' => $db,
        'urlManager' => [
            'enablePrettyUrl' => true,
            'showScriptName' => false,
            'rules' => $rules,
        ],
    ],
    'params' => $params,
];
return $config;
```

Тут будуть знаходитися: налаштування для кешування, підключення залежностей для роботи фронтенду; підключення поштового сервісу; використання компоненту User, що використовується для ідентифікації користувача в системі, а також багато інших модулів.

Робота Інтернет-платформи починається з файлу index.php. Веб-сервер перенаправляє всі запити на нього. Тут підключаються всі залежності та конфігурації проекту. Фрагмент коду точки входу:

```
<?php

defined('YII_DEBUG') or define('YII_DEBUG', true);
defined('YII_ENV') or define('YII_ENV', 'dev');
require __DIR__ . '/../vendor/autoload.php';
Dotenv\Dotenv::createImmutable(__DIR__ . '/../')->load();
require __DIR__ . '/../vendor/yiisoft/yii2/Yii.php';
require __DIR__ . '/../tools/helpers.php';
$config = require __DIR__ . '/../config/web.php';
(new yii\web\Application($config))->run();
```

Всі запити, які надійшли від користувача, обробляються в класі Application. В ньому розбирається HTTP запит та його параметри. При проектуванні структури можливих запитів було вирішено використовувати маршрути, представлені нижче:

```
<?php
return
[
    '' => 'site/site/index',
    '<a:(login|logout)>' => 'auth/auth/<a>',
    '<a:(signup)>' => 'auth/signup/signup',

    'admin' => 'admin/site/index',
    'admin/users' => 'admin/users/index',
    'admin/users/<id:\d+>' => 'admin/users/view',
    'admin/users/update/<id:\d+>' => 'admin/users/update',
    'admin/users/delete/<id:\d+>' => 'admin/users/delete',

    'admin' => 'admin/posts/index',
    'admin/posts' => 'admin/posts/index',
    'admin/posts/<id:\d+>' => 'admin/posts/view',
    'admin/posts/update/<id:\d+>' => 'admin/posts/update',
    'admin/posts/delete/<id:\d+>' => 'admin/posts/delete',

    'cabinet' => 'cabinet/index',
    'cabinet/posts’ => 'cabinet/post/index'
    'cabinet/posts/<id:[\w]+>’ => 'cabinet/post/view',
    'cabinet/posts/update/<id:\d+>' => 'cabinet/posts/update',
    'cabinet/posts/delete/<id:\d+>' => 'cabinet/posts/delete',
];
```

В кожному маршруті є префікс. Цей префікс означає, в якому пакеті з модулями знаходиться той чи інший контролер з потрібним методом для обробки. Наприклад, admin – контролер для адміністративної частини Інтернет-платформи, cabinet – контролер кабінету користувача, auth – контролер реєстрації та авторизації користувача.

Для того, щоб показати весь потік роботи в системі, а заодно і показати по якому принципу буде реалізована система, буде використаний сценарій реєстрації користувача.

Для обробки HTTP запиту, потрібно описати методи контролерів. Кожен метод контролера починатися з слова action, інакше фреймворк не зможе зрозуміти, що саме треба виконати. Фрагмент коду контролера для методу реєстрації користувача:

```
/**
 * Signup action.
 *
 * @return Response|string
 */
public function actionSignup()
{
   $model = new SignupForm();
   if ($model->load(Yii::$app->request->post()) && $model->validate()) {
      try {
         $this->service->signup($model);
         return $this->goBack();
      } catch (\RuntimeException $e) {
         Yii::$app->errorHandler->logException($e);
         Yii::$app->session->setFlash('error', $e->getMessage());
       }
    }
    $model->password = '';
    return $this->render('signup', [
    'model' => $model,
    ]);
 }
```

В даному методі створюється модель форми реєстрації користувача. Вона призначена для того, щоб зафіксувати атрибути , необхідні для заповнення користувачем та описати правила валідації цих атрибутів:

```
class SignupForm extends Model
{
   public ?string $username = null;
   public ?string $email = null;
```

```
    public ?string $password = null;
    public ?string $password_repeat = null;

    public function rules(): array
    {
        return [
            ['username', 'filter', 'filter' => 'trim'],
            ['username', 'required'],
            ['username', 'match', 'pattern' => '#^[\w_-]+$#i'],
            ['username', 'unique', 'targetClass' => User::class, 'message' => 'This username is already taken'],
            ['username', 'string', 'min' => 4, 'max' => 255],

            ['email', 'filter', 'filter' => 'trim'],
            ['email', 'required'],
            ['email', 'email'],
            ['email', 'unique', 'targetClass' => User::class, 'message' => 'This male is already taken'],

            ['password', 'required'],
            ['password', 'string', 'min' => 8],

            ['password_repeat', 'compare', 'compareAttribute' => 'password'],
            ['password_repeat', 'required'],
        ];
    }
}
```

Зобразимо моделі форм на діаграмі класів (рисунок 3.2).

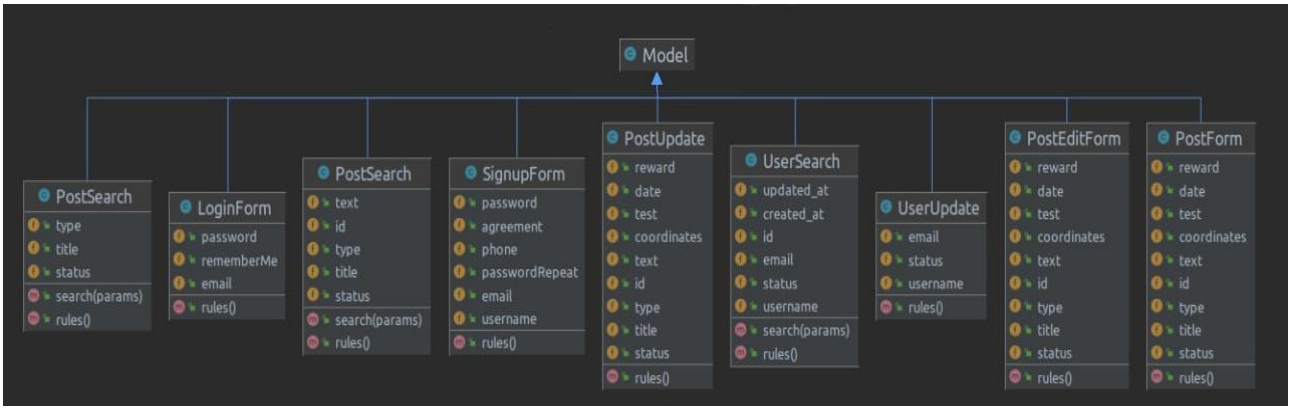


Рисунок 3.2 – Діаграма класів для моделей форм

Після того, як форма була відправлена і провалідована, дані з форми передаються далі в сервіс. Сервіс – це такий шаблон проектування, який інкапсулює бізнес-логіку додатку та вирішує проблему дублювання коду. Завдяки цьому, можна звертатися до будь-якого сервісу, який має виконати одне і теж з різних місць додатку. Далі буде показаний сервіс, який опрацьовує дані користувача, створює йому токен та відправляє лист на вказану пошту для підтвердження. Фрагмент сервісу реєстрації:

```
class SignupService
{
    private UserRepository $users;
    private UserTokenRepository $userTokens;
    public function __construct(
        UserRepository $users,
            UserTokenRepository $userTokens
    ) {
        $this->users = $users;
            $this->userTokens = $userTokens;
    }
    public function signup($form): void
    {
        $user = User::create(
            $form->username,
            $form->email,
            $form->password
        );

            $userToken = UserToken::create(
            $user->id
        );

            Yii::$app->mailer->send($user->email);

        $transaction = \Yii::$app->db->beginTransaction();
        try {
            $this->users->save($user);
                    $this->userTokens->save($userToken);
            $transaction->commit();
        } catch (\Exception $ex) {
            $transaction->rollBack();
            throw $ex;
        }
    }
}
```

На рисунку 3.5 зображена діаграма класів для сервісів проекту.

Рисунок 3.5 – Діаграма класів для моделей сервісів

Для подальшої обробки необхідно створити моделі реляційного відображення таблиць бази даних. Приклад класу User, який є моделлю користувача, буде поданий у додатку А. Усі класи успадковуються від класу ActiveRecord. Це означає, що клас не повинен містити властивості, оскільки при виконанні будь-якого запиту з використанням даної моделі, будуть створюватися віртуальні властивості моделей. Для полегшеної роботи з моделями, було описано всі віртуальні властивості в анотаціях класу. Також варто створити статичні конструктори (це такі методи, які будуть повертати об’єкт класу в контексті яких викликався статичний конструктор). Така необхідність зв’язана з тим, що при виклику запитів на знаходження певних записів методом find(), який йде базовим для всіх наслідників ActiveRecord, буде намагатися створити екземпляр класу, який був викликаний. Якщо описати конструктор класу, передаючи йому аргументи, то це викличе помилку, оскільки при запиті на виборку, потрібно буде також створювати об’єкт передаючи йому аргументи. А це означає, що треба вже лізти і правити ядро фреймворку. Тому було вирішено обійтися статичними методами-конструкторами, щоб не виникало конфліктів з фреймворком. Для того, щоб міняти скалярні значення сутності буде використовуватися метод edit(). Фрагмент коду такого конструктора для сутності користувача та методу для зміни значень сутності:

```
public static function create($username, $email, $password): User
{
   $user = new User();
   $user->username = $username;
   $user->email = $email;
   $user->setPassword($password);
   $user->status = User::STATUS_ACTIVE;
   $user->generateAuthKey();
   return $user;
}

public function edit(string $username, string $email, int $status): void
{
   $this->username = $username;
   $this->email = $email;
   $this->status = $status;
   $this->updated_at = time();
}
```

На рисунку 3.2 зображена діаграма класів для моделей Active Record.

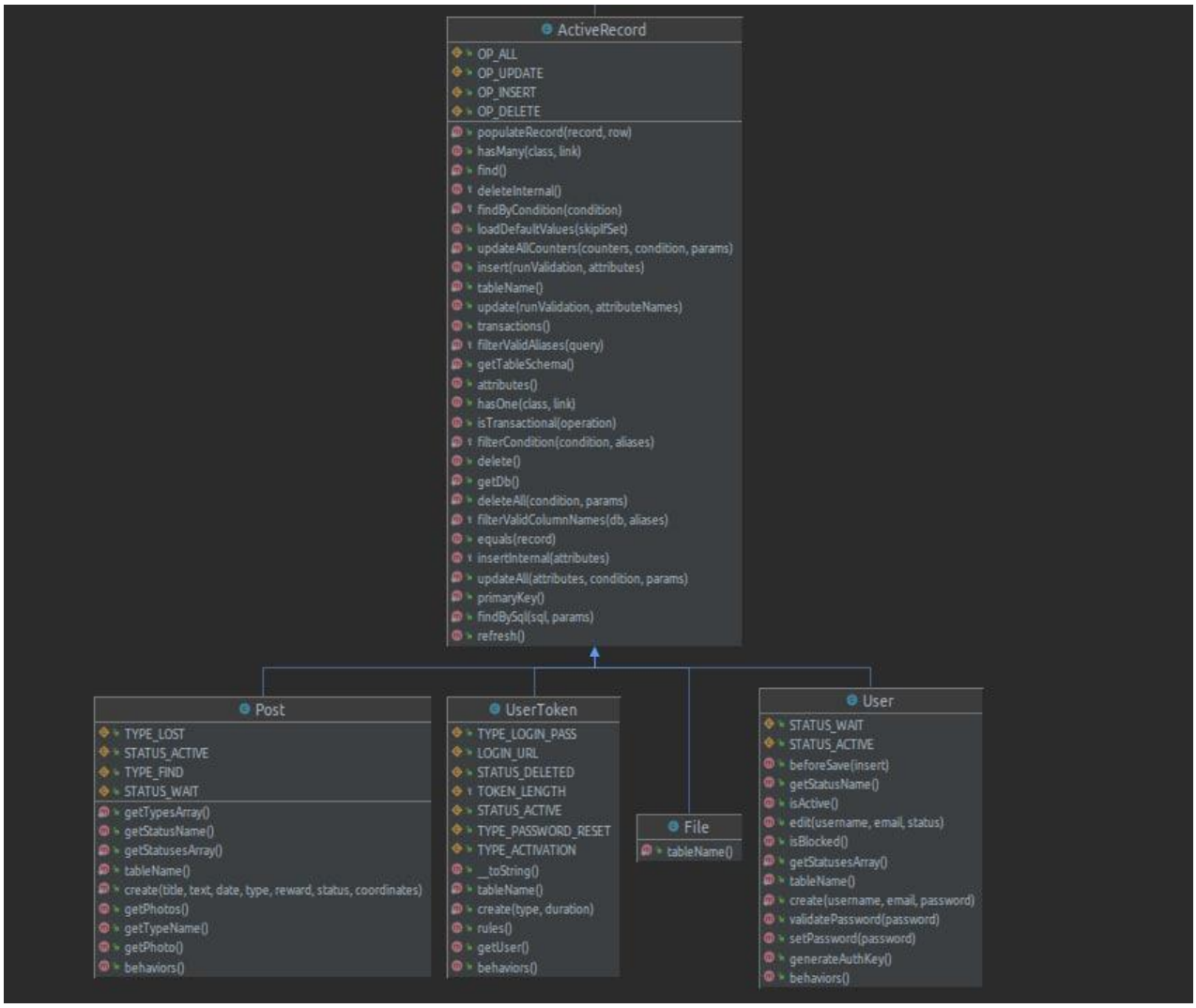


Рисунок 3.2 – Діаграма класів для моделей ActiveRecord

Після створення сутності, її потрібно зберегти. Для цього буде використовуватися такий патерн проектування як «Репозиторій». Для кожної моделі-сутності, яка наслідує ActiveRecord, був створений свій репозиторій. Опишемо репозиторій для користувача:

```
<?php
namespace app\models\repositories;
class UserRepository
{
        public function find($id): ?User
        {
          return User::findOne($id);
        }
```

```
    public function findAll(): array
    {
      return User::findAll();
    }

    public function save(User $user): void
    {
      if (!$user->save()) {
        throw new \RuntimeException('Saving error.');
      }
    }

    public function remove($id): void
    {
      $user = $this->find($id);
      if (!$user->delete()) {
        throw new \RuntimeException('Removing error.');
      }
    }

    public function findByEmail($value): ?User
    {
      return User::findOne(['email' => $value]);
    }

    public function findByUsername($value): ?User
    {
      return User::findOne(['email' => $value]);
    }

    private function getBy(array $condition)
    {
      $user = User::find()->andWhere($condition)->limit(1)->one();
      if (null === $user) {
        throw new \DomainException('User not found.');
      }

      return $user;
    }
}
```

Для кожного репозиторію системи мають бути описані наступні методи:

– find () – знайти сутність;

– getAll() – отримати всі записи;

– save() – створення/редагування запису в базі даних;

– remove() – видалення запису;

– getBy() – знаходження одного запису за заданими параметрами.

Для деяких репозиторіїв потрібно реалізувати додаткові методи, оскільки того потребують функціональні вимоги дипломного проекту. Зобразимо репозиторії на діаграмі класів (рисунок 3.2).

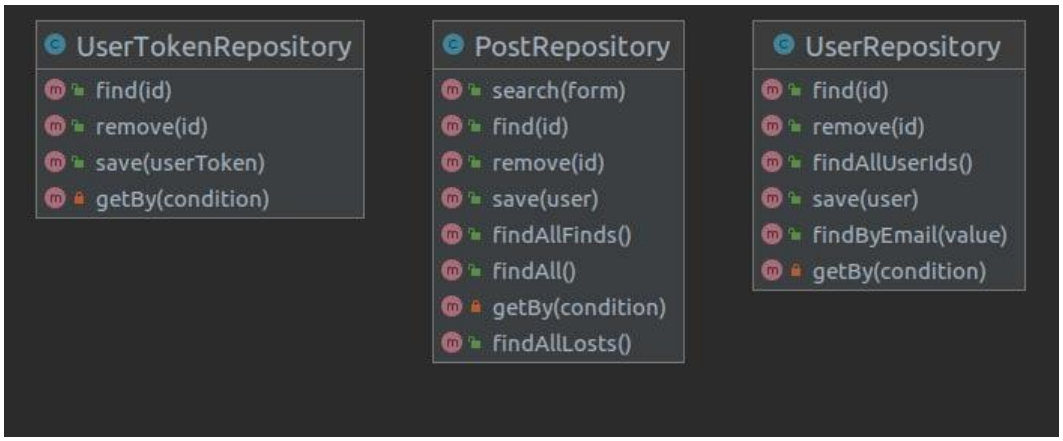


Рисунок 3.4 – Діаграма класів для моделей репозиторіїв

3.3 Керівництво користувача

Розпочнемо даний розділ з інструкції користування Інтернет-платформою для користувача.

Для того, щоб була змога користуватися Інтернет-платформою, потрібно відкрити браузер на будь-якому приладі (наприклад, телефон або комп’ютер) і перейти на відповідну адресу. Перед користувачем відкриється головна сторінка сайту (рисунок 3.6).

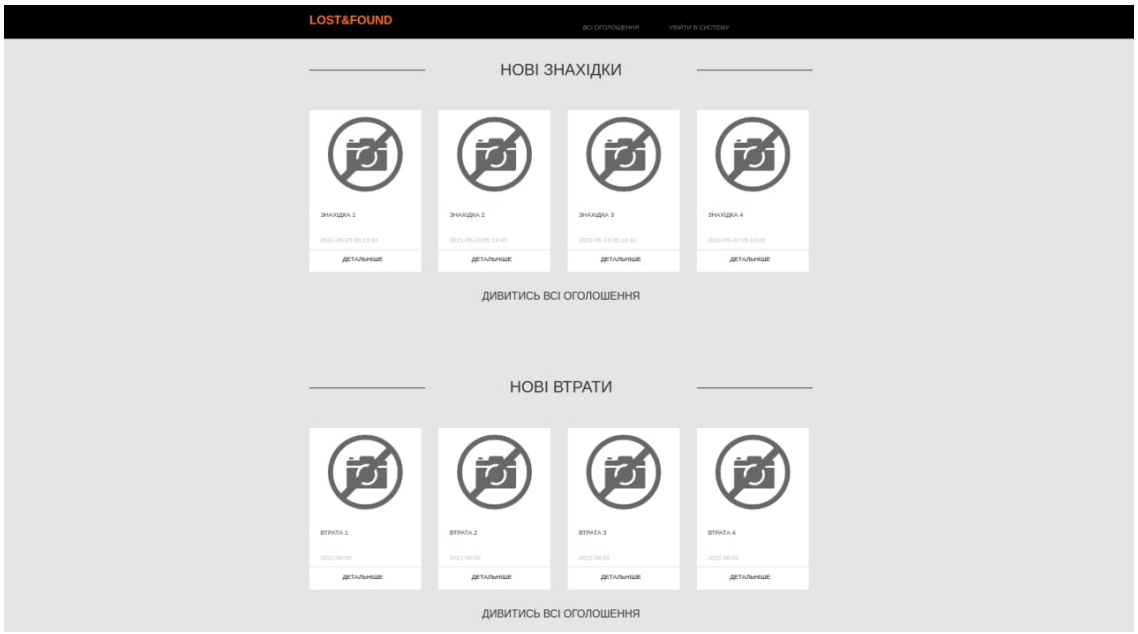


Рисунок 3.6 – Головна сторінка

На головній сторінці відображаються свіжі пости, які були відмодеровані опубліковані модератором.

Для того, щоб зробити свій власний пост, користувачу потрібно спершу зареєструватися. Це робиться задля безпеки в системі. Для того щоб це зробити, потрібно натиснути на кнопку зверху в хедері «Увійти в систему». Після цього користувача перекине на сторінку з формою для авторизації (рисунок 3.7). Нижче під формою є посилання на форму з реєстрацією (рисунок 3.8). Після заповнення всіх необхідних даних, користувача перекине на головну сторінку і він буде повідомлений, що на вказану ним пошту був відправлений лист з підтверджувальним посиланням. Тільки після того, як користувач підтвердить свою пошту, він зможе повністю працювати з системою.

Рисунок 3.7 – Форма авторизації

Рисунок 3.8 – Форма реєстрації

Після успішної авторизації для користувача відкривається доступ до власного кабінету, з допомогою якого можна переглянути власні оголошення (рисунок 3.9), та створювати їх (рисунок 3.10).

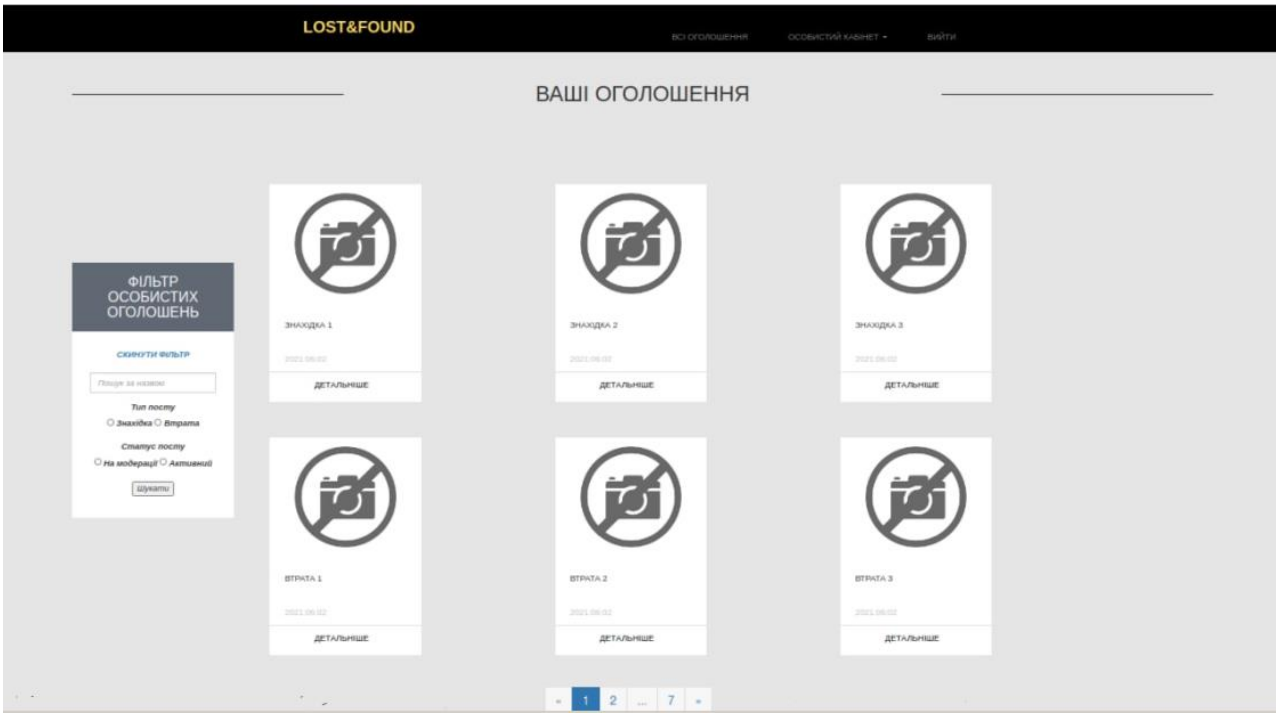


Рисунок 3.9 – Сторінка власного кабінету

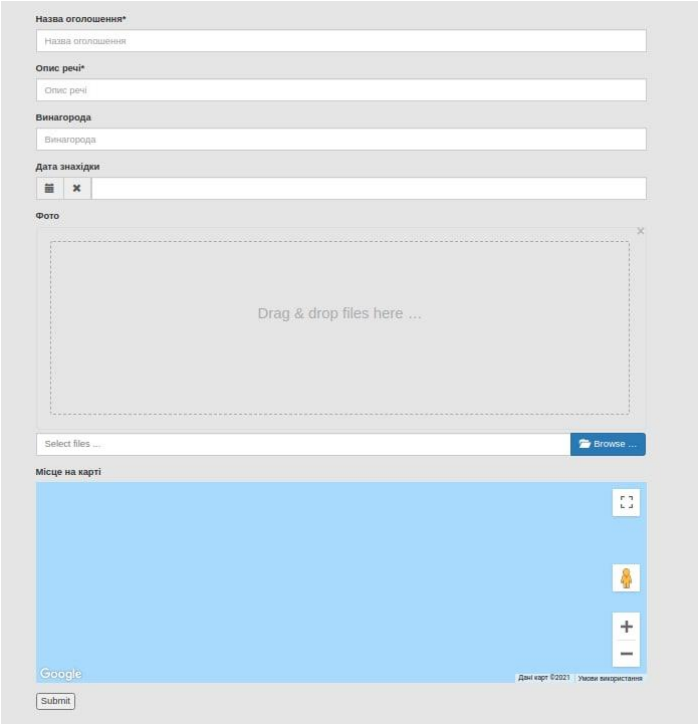


Рисунок 3.10 – Форма заповнення даних оголошення

Для того, щоб переглянути пости інших користувачів треба перейти на сторінку «усі оголошення» (рисунок 3.11).

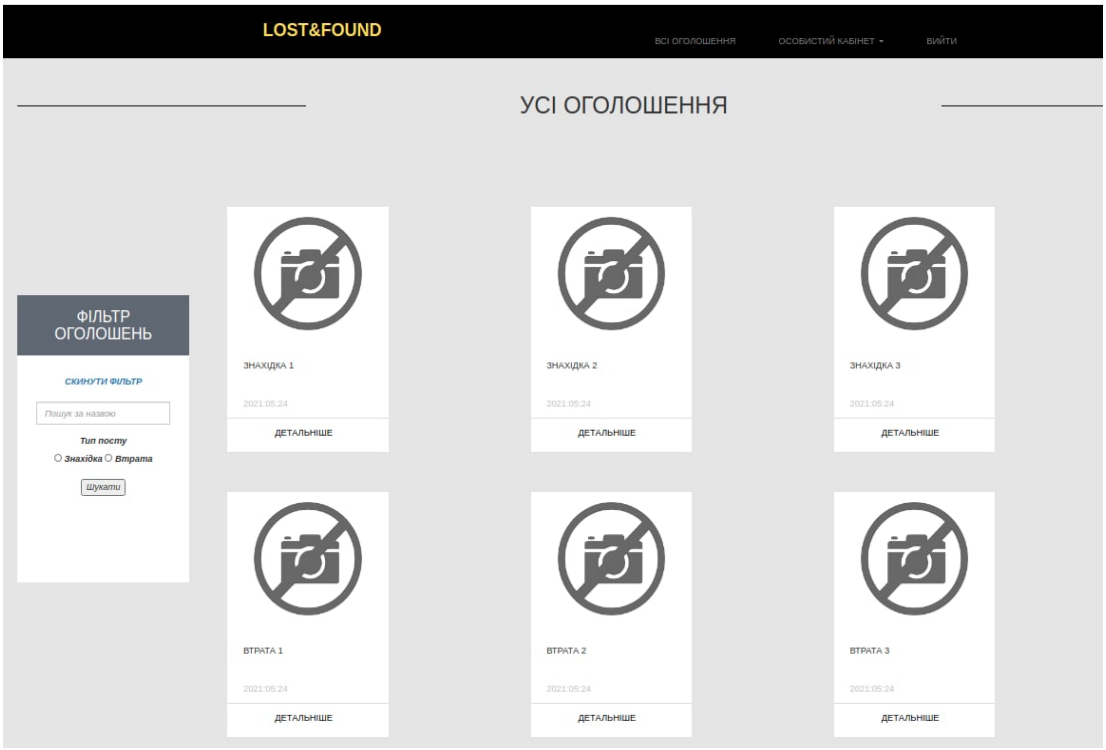


Рисунок 3.11 – Сторінка з усіма оголошеннями

Тепер розглянемо інструкцію користування Інтернет-платформою для адміністратора / менеджера.

Для початку роботи адміністрації на Інтернет-платформі, перш за все, потрібно, щоб вони зареєструвалися на платформі. Після цього, необхідно звернутися до адміністратора серверу, який надасть права для цих користувачів.

Тепер, адміністратору або менеджеру доступна адміністраторська панель сайту. Щоб туди перейти треба вказати в адресній стрічці браузера шлях /admin. Перед ними відкриється головна сторінка (рисунок 3.12).

Для управління користувачами, потрібно перейти на вкладку з користувачами. Відкриється сторінка з списком користувачів (рисунок 3.13).

Нажавши на олівець збоку справа відкриється сторінка, на якій можна відредагувати дані про користувача (рисунок 3.14). Після завершення редагування, потрібно натиснути на кнопку «Зберегти».

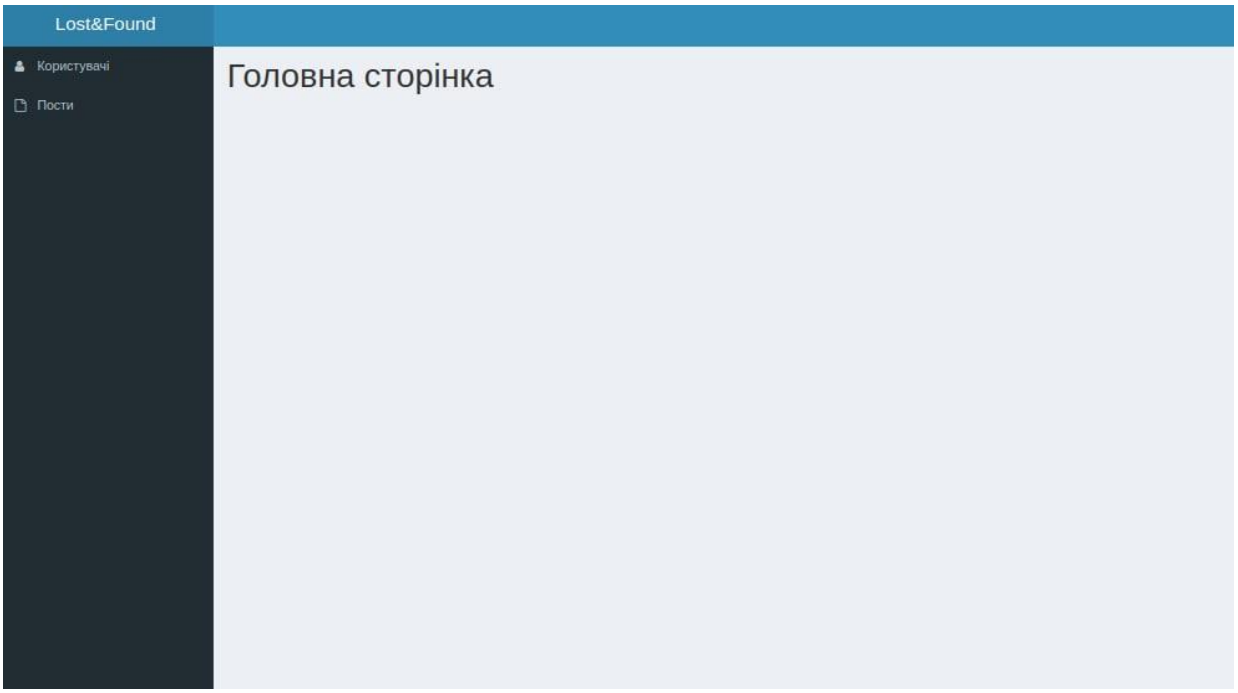


Рисунок 3.12 – Головна сторінка адміністраторської панелі

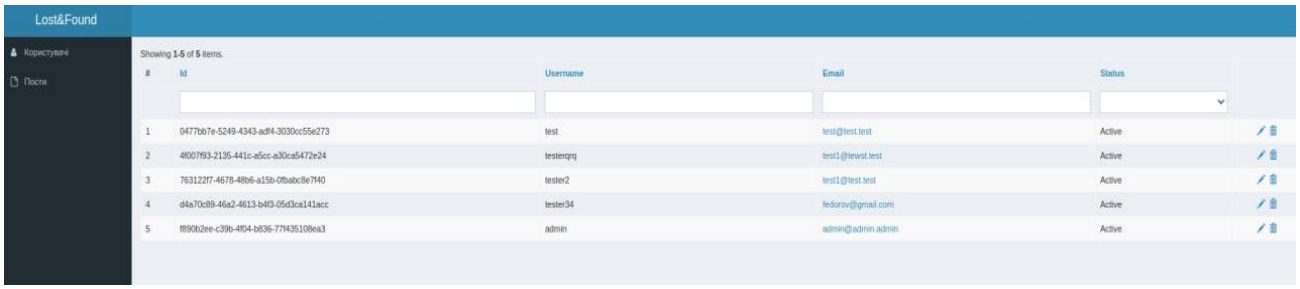


Рисунок 3.13 – Сторінка з списком користувачів

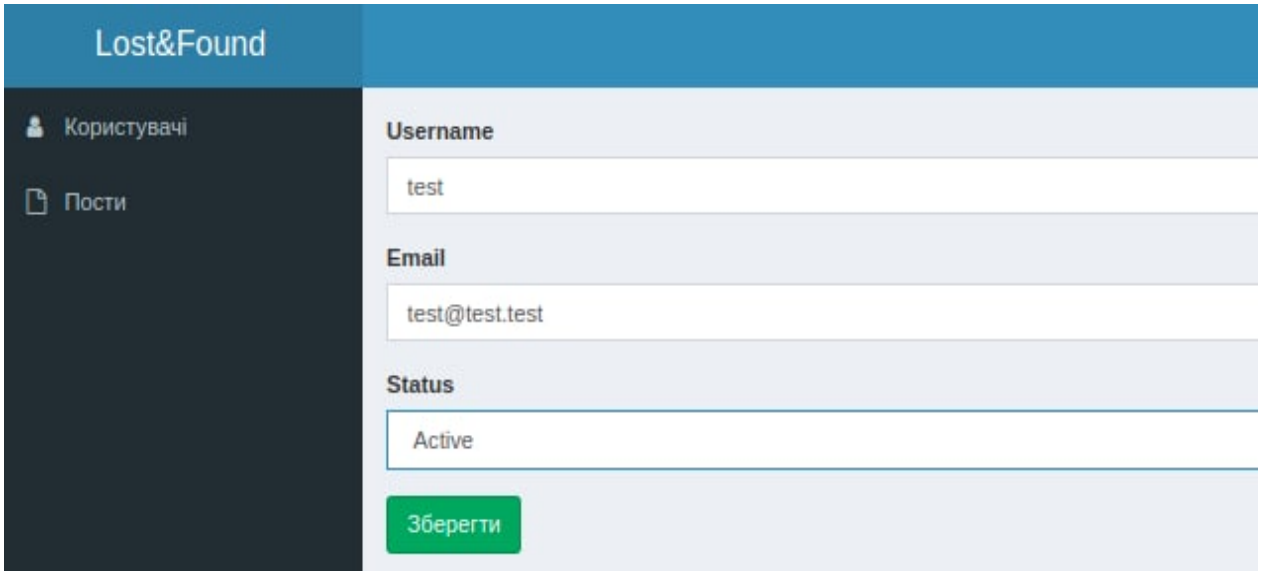


Рисунок 3.14 – Форма редагування даних користувача

Аналогічно з постами користувачів. Перейшовши на сторінку з записами (рисунок 3.15), можна побачити їх список.

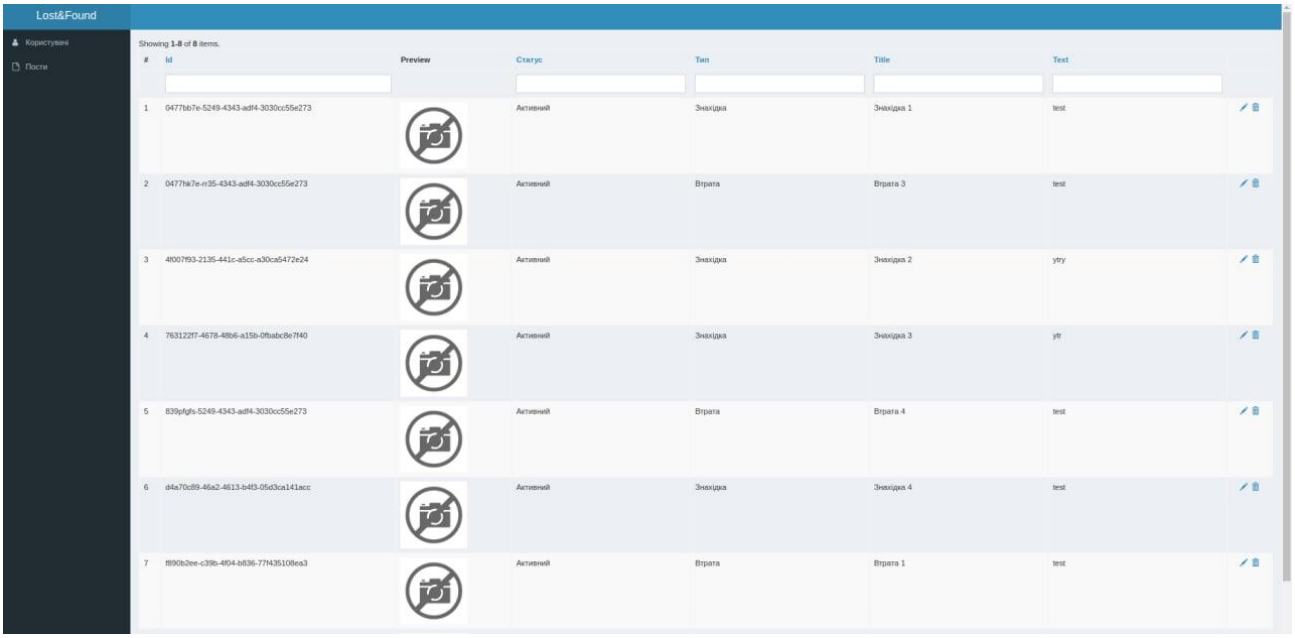


Рисунок 3.15 – Сторінка з постами користувачів

Перейшовши в режим редагування, можна побачити форму (рисунок 3.16). Після завершення модерації, потрібно натиснути на кнопку Submit.

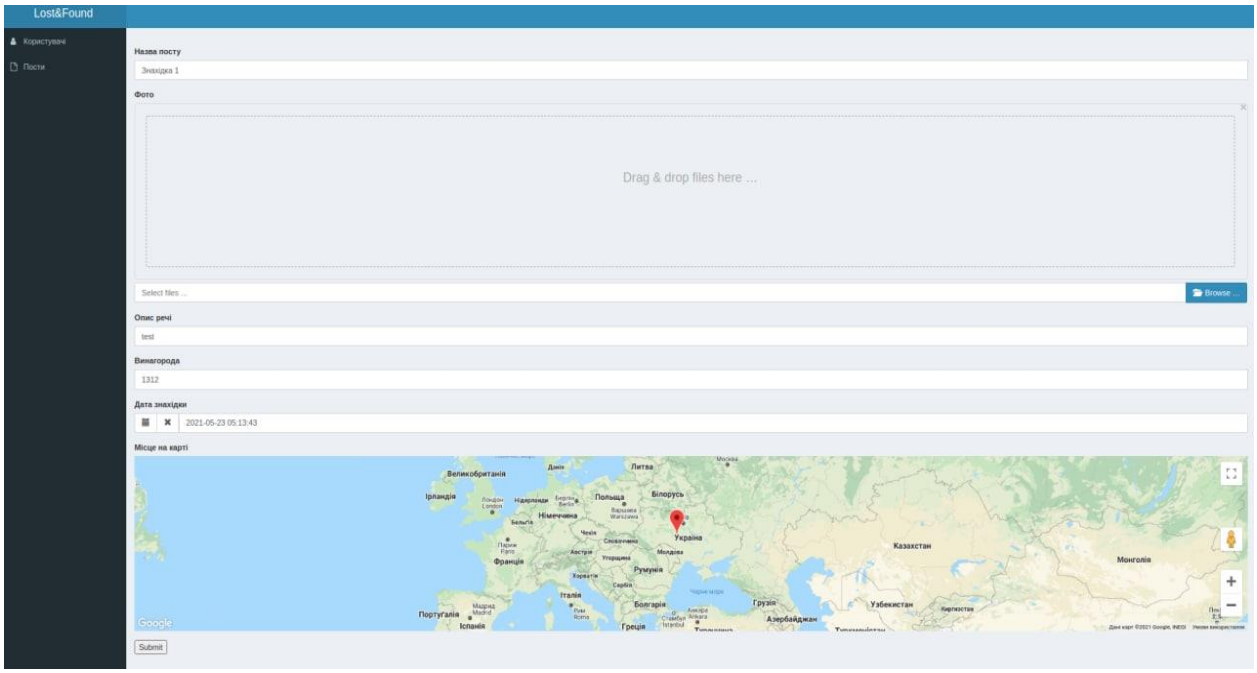


Рисунок 3.16 – Форма редагування посту в адміністраторській панелі

3.4 Технічні характеристики Інтернет-платформи

Для того, щоб забезпечити безперебійну працездатність системи на стороні серверу, варто використовувати техніку, яка задовольняє наступним технічним характеристикам:

– 4 ГБ оперативної пам’яті;

– 4 ГБ на SSD;

– трафік 4 ТБ;

– процесор на 4 ядра.

Для комфортного користування Інтернет-платформою на стороні клієнта, варто використовувати техніку, яка задовольняє наступним характеристикам:

– 2 ГБ оперативної пам’яті;

– 1 ГБ HDD або SSD пам’яті;

– процесор не нижче Intel Pentium 4.

3.5 Завантаження Інтернет-платформи на хостинг

Настав час для того, щоб завантажити Інтернет-платформу, на хостинг. Для початку треба обрати, на який тип хостингу буде відбуватися завантаження.

Віртуальний хостинг – це такий сервер, ресурси якого розподілені між декількома обліковими записами в системі. Зазвичай, на одну машину надається загальний доступ до веб-серверу, поштовий сервер, бази даних. Надається вже готовий функціонал (тобто заздалегідь встановлені модулі для мови програмування, баз даних), адміністративну панель для керування своїм ресурсом. Цей спосіб підійде тим користувачам, які бажають створити легкий веб-додаток, який не містить особливої бізнес-логіки.

Віртуальний приватний сервер – в даному випадку, кожен користувач отримує власний веб-сервер, сервер бази даних та поштовий сервер. Даний вид хостингу надає можливості для розгортання власного програмного продукту з нестандартними модулями. Такі сервери надають root доступ і користувачі з

досвідом роботи з системами Linux можуть власноруч поставити собі будь-який компонент, необхідний для роботи додатку.

Було вирішено обрати віртуальний приватний сервер, оскільки він надасть більше можливостей для роботи з платформою, а також, що немало важливо, дозволить користуватися системою контейнеризації Docker.

Хостингом VPS був обраний ресурс hostinger.com. Він надає послуги як для віртуального хостингу так і VPS. Був обраний трафік VPS 4, який включає в себе наступний пакет виділених ресурсів:

– 4 ГБ оперативної пам’яті;

– 80 ГБ місця на SSD-диску;

– 4 ТБ трафіку;

– виділений IP;

– root-доступ.

Надані ресурси цілком задовольняють потреби, необхідні для розгортання та користування додатком.

Тепер розглянемо процес встановлення та налаштування додатку на VPS хостингу. Для початку потрібно зайти в саму систему. Це буде виконано за допомогою SSH протоколу. Після цього буде підключення до репозиторію GIT, в якому знаходиться розроблений дипломний проект і скачування його на VPS. Після цього, потрібно створити файл .env, з допомогою якого можна налаштовувати логіни і паролі для модулів системи (рисунок 3.17).

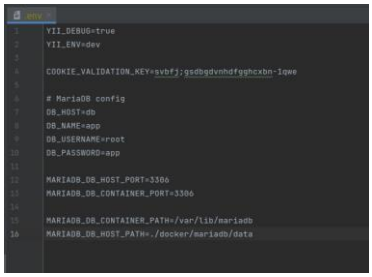


Рисунок 3.17 – Конфігураційні дані для додатку

Після цього будуть виконані команди для скачування відповідних образів модулів системи з офіційних Docker-репозиторіїв та їхня побудова в якості

контейнерів. Збирання контейнерів відбувається за допомогою файлу docker-compose.yml, в якому описані додаткові налаштування додатку (рисунок 3.18).

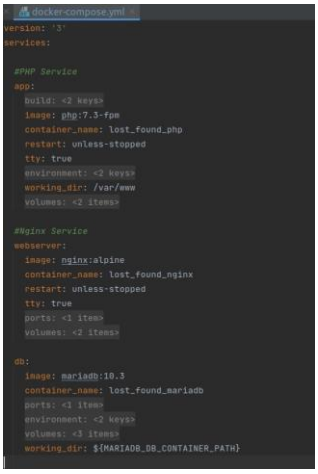


Рисунок 3.18 – Файл збирання додатку в Docker-контейнери

В третьому розділі пояснювальної записки було проведене проектування бази даних та Інтернет-платформи. Була виконана розробка програмних модулів, описана взаємодія між ними. Була створена інструкція користувача як для користувача так і для менеджера / адміністратора Інтернет-платформи. Були визначені вимоги для запуску Інтернет-платформи як на стороні сервера так і для клієнтів. Також був описаний процес розгортання системи на сервері.

4 ТЕСТУВАННЯ ІНТЕРНЕТ-ПЛАТФОРМИ

4.1 Вибір та обґрунтування методів тестування Інтернет-платформи

Тестування ПЗ – це процес перевірки працездатності, поведінки, та загальну відповідність до бізнес-вимог. Тестування є необхідною частиною розробки проекту, оскільки це дозволить не тільки прибрати дефекти при роботі системи, а й надати гарантію, що при розширенні проекту функціонально, всі процеси не зупиняться аварійно [12].

Спроектувавши програмний продукт, потрібно перевірити його на коректність роботи. Для цього існують модульні тести.

Модульне тестування розробляються і застосовуються програмістами. Зазвичай результати такого тестування показують, що код відповідає заданим вимогам до архітектури та має очікувану поведінку.

Найважливішою частиною додатку є саме серверна частина. Тестування буде проводитися від низькорівневого компонента до високорівневого.

Для проведення тестування буде використовуватися такий інструмент, як Codeception. Codeception – фреймворк, який дозволяє виконувати різні функціональні тести [13]. З його допомогою будуть написані модульні тести.

4.2 Модульне тестування Інтернет-платформи

Модульне тестування проводиться з допомогою бібліотеки PHPUnit – популярної бібліотеки для юніт тестів, яка включена в фреймворк Codeception.

Всі тести розташовані в папці tests. В цій же папці знаходяться bash-скрипти для запуску тестів і конфігураційні файли.

Для того, щоб почати тестувати систему потрібно визначитися, що треба тестувати. Перш за все, потрібно протестувати моделі ActiveRecord, які є релятивним відображенням записів в базі даних. До прикладу, спробуємо створити об’єкт користувача системи, який щойно зареєструвався в системі

через статичний метод-конструктор і перевіримо чи дійсно він створився. Фрагмент коду створення сутності користувача:

```
class SignupTest extends Unit
{
   public function testSuccess()
   {
      $user = User::requestSignup(
         $username = 'username',
         $email = 'email@site.com',
         $phone = '70000000000',
         $password = 'password'
      );

      $this->assertEquals($username, $user->username);
      $this->assertEquals($email, $user->email);
      $this->assertEquals($phone, $user->phone);
      $this->assertNotEmpty($user->password_hash);
      $this->asserEquals($password, $user->password_hash);
      $this->assertNotEmpty($user->created_at);
      $this->assertNotEmpty($user->auth_key);
      $this->assertFalse($user->isActive());
      $this->assertTrue($user->isWait());
   }
}
```

Тут перевіряється, чи правильно передалися дані про користувача, такі як ім’я, електронна пошта та телефон. Була проведена перевірка, чи згенерувався хеш паролю, чи відпрацював спеціальний тригер для генерації дати створення та оновлення моделі користувача, а також перевірка статусу новоствореного облікового запису. Суть виконання такого тесту однакова для всіх моделей ActiveRecord: перевіряється чи коректно була створена модель. Далі були створені тести, за допомогою яких можна протестувати взаємодію клієнта та системи через веб-форму. Фрагмент коду форми реєстрації:

```
class SignupFormTest extends \Codeception\Test\Unit
{
   public function testCorrectSignup()
   {
      $model = new SignupForm([
         'username' => 'some_username',
         'email' => 'some_email@example.com',
         'phone' => '70000000005',
         'password' => 'some_password',
      ]);

      expect_that($model->validate());
            expect_that($model->getErrors('username'));
      expect_that($model->getErrors('email'));
```

```
        expect($model->getFirstError('username'))
            ->equals('This username has already been taken.');
        expect($model->getFirstError('email'))
            ->equals('This email address has already been taken.');
    }
}
```

Далі, потрібно протестувати шар доступу до даних, а саме методи для реалізації створення запитів до бази даних. Вся справа в тому, що тестування повинно проводитися в вакуумному середовищі, тобто тести повинні бути незалежними від зовнішніх даних. Для того, щоб реалізувати тестування цього шару додатку існує два методи.

Створення додаткової локальної бази даних, які буде ідентичною до рельної бази. Плюсом є те, що mock-запити будуть ідентичними до реальних, а отже відтворюється і реальна поведінка.

Недоліки:

– тести стають залежними від бази даних;

– база даних може перестати працювати, і тоді тести будуть неуспішними;

– в базі можуть бути як і хороші дані, так і погані, а тому тести можуть видавати неочікуваний результат і постійно падати.

Розглянемо другий варіант тестування. Для того, щоб виконати тести, можна створити фікстури – такі собі невеликі заготовки з даними.

Переваги:

– немає залежності від бази даних, оскільки дані готуються в фікстурах;

– дані не пересікаються між собою, оскільки можна підготувати фікстури з поганими даними і з хорошими окремо;

– не потрібно використовувати міграції для БД;

– запити робляться не в базу даних, що може займати час, а в фікстури.

Недоліком є те, що іноді в базах даних може знаходитися логіка, яка лежить на рівні бази. Наприклад, функції підрахування записів.

Очевидним вибором способу для тестування шару доступу до даних стає другий варіант як найбільш оптимальний та легкий в супроводженні і використанні. Фрагмент коду для перевірки наявності запису в базі-фікстурі:

```
class UserInDbCest extends \Codeception\Test\Unit
{
   public function _before(AcceptanceTester $I)
   {
      if(Fixtures::exists('users')){
         /*
          * Завантаження даних в фікстуру
          */

         foreach (Fixtures::get('users') as $item) {
            $I->haveInDatabase('users', $item);
         }
      }
   }
   public function checkUserInDb(AcceptanceTester $I)
   {
     $I->seeInDatabase ('users', array ('name' => 'testName1', 'email' => 'testName1@gmail.com'));
   }
}
```

Тепер варто протестувати контролери, які входять в шар Interface Adapters. Для прикладу, буде обраний метод update, який призначений для оброблення даних для оновлення посту. Вони повинні повертати HTTP статус з кодом двісті у випадку, якщо потрібний action відпрацює без помилок. Фрагмент коду для контролеру оголошень для методу update, який повинен повертати код статусу двісті:

```
public function testPostController()
{
   $appConfig = [];
   $this->mockApplication($appConfig);
   $params = ['baafb124-31b3-4f8e-8ce0-1d89af8a199b'];
   $controller = new PostController('id', Yii::$app);
   $result = $controller->run('update', $params);
   $this->assertEquals(HTTP::OK, $result);
}
```

Або повертати статус з кодом п’ятсот, якщо винила помилка. Фрагмент коду для контролеру оголошень для методу update з кодом статусу п’ятсот:

```
public function testPostController()
{
   $appConfig = [];
   $this->mockApplication($appConfig);
   $params = ['$&&#%^&TIUF8t#TR#'];
   $controller = new PostController('id', Yii::$app);
   $result = $controller->run('update', $params);
   $this->expectException(ServerErrorHttpException::class);
}
```

В результаті проведеного тестування був створений загальний список всіх тест-кейсів для важливих частин системи:

– для контролерів – таблиця 4.1;

– для сутностей – таблиця 4.2;

– для сервісів – таблиця 4.3;

– для репозиторіїв – таблиця 4.4.

Таблиця 4.1 – Тестування контролерів

Назва тесту	Клас, який перевіряють	Опис тесту
testSiteControllerActionIndex	SiteController	Метод для обробки запиту на головну сторінку
testPostControllerActionFind	PostController	Метод для обробки запиту на сторінку з знахідкою
testPostControllerActionLost	PostController	Метод для обробки запиту на сторінку з втратою
testFileControllerActionUpload	FileController	Метод для обробки запиту для завантаження файлу
testFileControllerActionDelete	FileController	Метод для обробки запиту для завантаження файлу
testCabinetControllerActionFind	CabinetController	Метод для обробки запиту для створення оголошення знахідки
testCabinetControllerActionLost	CabinetController	Метод для обробки запиту для створення оголошення втрати
testAuthControllerActionLogin	AuthController	Метод для обробки запиту для авторизації
testAuthControllerActionSignup	AuthController	Метод для обробки запиту для реєстрації

Таблиця 4.2 – Тестування сутностей

Назва тесту	Клас, який тестують	Опис тесту
testUserTokenCreate	UserToken	Створення токену
testUserTokenEdit	UserToken	Редагування токену
testUserValidatePassword	User	Валідація паролю
testUserSetPassword	User	Установити пароль
testUserGenerateAuthKey	User	Генерація аутентифікаційного ключа
testUserEdit	User	Редагування сутності користувача
testUserCreate	User	Створення сутності користувача
testFileCreate	File	Створення сутності файлу
testFileEdit	File	Редагування сутності файлу
testFileEdit	File	Видалення сутності файлу

Таблиця 4.3 – Тестування сервісів

testUserManageServiceEdit	UserManageService	Редагування користувача
testUserManageServiceRemove	UserManageService	Видалення користувача
testPostServiceCreate	PostService	Створення оголошення
testPostServiceEdit	PostService	Редагування оголошення
testPostServiceRemove	PostService	Видалення оголошення
testEmailServiceConfirm	EmailService	Підтвердження пошти
testAuthServiceAuth	AuthService	Авторизація користувача
testAuthServiceSignup	AuthService	Реєстрація користувача

Таблиця 4.4 – Тестування репозиторіїв

testUserRepositoryFind	UserRepository	Знаходження запису користувача
testUserRepositorySave	UserRepository	Збереження запису користувача
testUserRepositoryRemove	UserRepository	Видалення запису користувача
testUserRepositoryFindByEmail	UserRepository	Знаходження запису користувача по атрибуту пошти
testUserRepositoryFindAllUserIds	UserRepository	Знаходження всіх Id користувачів
testUserRepositoryGetBy	UserRepository	Отримання запису за певними параметрами
testPostRepositoryFind	PostRepository	Знаходження запису оголошення
testPostRepositoryFindAllFinds	PostRepository	Знаходження всіх записів знахідок за певними атрибутами
testPostRepositoryFindAllLosts	PostRepository	Знаходження всіх записів втрат за певними атрибутами
testPostRepositorySave	PostRepository	Збереження запису оголошення
testPostRepositoryRemove	PostRepository	Видалення запису оголошення

4.3 Аналіз результатів тестування Інтернет-платформи

Виконавши тестування для всіх модулів, які були описані в попередньому пункті розділу, були отримані такі результати (рисунок 4.7). Як видно на рисунку, всі 37 тестів були успішно виконані, а це означає, що помилок у функціонуванні розробленої системи не було виявлено.

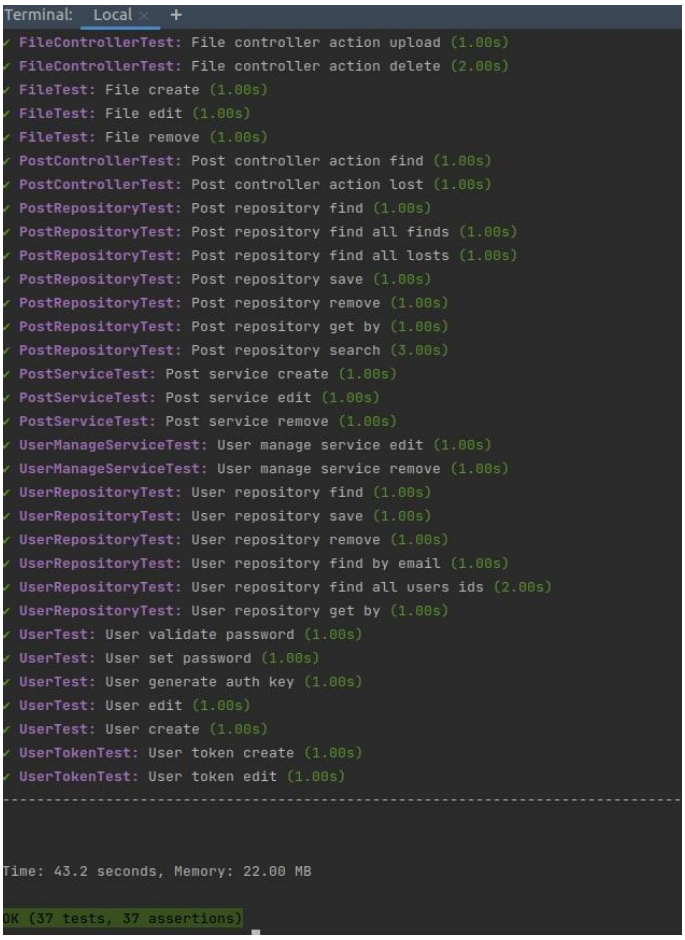


Рисунок 4.1 – Результати проведеного тестування

В цьому розділі було розглянуто інструменти для тестування та було проведено тестування системи. На основі результатів тестування було виявлено, що розроблена Інтернет-платформа працює коректно та відповідає поставленим функціональним вимогам.

ВИСНОВКИ

Темою дипломного проектування є Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт. При аналізі існуючого програмного забезпечення на ринку було виявлено, що проблема, яку вирішує тема дипломного проектування, вирішується у вигляді онлайн дошок або створення оголошень в соціальних мережах або месенджерах. На основі отриманих даних були визначені функціональні вимоги до розроблюваної Інтернет-платформи, розроблені діаграми варіантів використання та діаграми потоків даних в системі.

На етапі проектування була розглянута клієнт-серверна архітектура додатку, основи її роботи. Були проаналізовані стилі архітектур, такі як мікросервіси та моноліт, їхні переваги та недоліки. Для реалізації дипломного проекту було вирішено використовувати монолітний стиль архітектури, так як він простіший в проектуванні, легший для розвертання на інших машинах. У процесі проектування бази даних були визначені основні сутності системи, їх атрибути, та були нормалізовані їхні зв'язки. Для проектування клієнтської частини, був використаний метод прототипування, та були розроблені прототипи основних структурних одиниць сторінок.

Для того, щоб реалізувати дипломний проект, були розглянуті основні інструменти, які використовуються на ринку для розробки веб-додатків. Було вирішено використовувати мову програмування PHP та інструменти, які написані з її допомогою.

При реалізації дипломного проекту було розглянуто та використано методологію “чистої архітектури”, яка забезпечить написання такого коду, який потім можна без зайвих затрат часу та ресурсів доповнювати новим функціоналом та вносити зміни в існуючі бізнес-правила. Таким чином, проект був поділений на модулі, які мають строгі границі своєї відповідальності, що задовільняє принципи «чистої архітектури».

При тестуванні дипломного проектування були розроблені тест-кейси для

основних модулів системи. Було виконане модульне тестування в ході якого було визначено, що система коректно функціонує.

Отже, у результаті виконання дипломного проекту була реалізована Інтернет-платформа, яка допомагає користувачам повертати втрачене. Функціонал був реалізований у відповідності до поставлених вимог.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Глушенкова І. С. Методичні вказівки до самостійної роботи і практичних занять з дисципліни «Основи теорії систем та системний аналіз» / І. С. Глушенкова, Є. І. Кучеренко. – Х.: ХНАМХ, 2010. – 51 с.

2. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language / M. Fowler. – Publisher(s): Addison-Wesley Professional, 2004. – 175 p.

3. Alex Berson. Client/Server Architecture / Alex Berson. – Publisher(s): Bayside Book Sales, 1996. – 569 p.

4. Monolithic vs. Microservices Architecture [Online] / Web portal mlsdev.com. – Available: https://mlsdev.com/blog/128-microservices-vs-monoliths-how-to-understand-when-it-s-time-to-use-the-former-option.

5. Jesse Russel. ER-model / Jesse Russell, Ronald Cohn. – Publisher(s): Bookvika, 2012. – 122 p.

6. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design / Robert C. Martin – Publisher(s): Pearson, 2017. – 432 p.

7. Empowering App Development for Developers | Docker [Online] / Web portal Docker.com. – Available: https://www.docker.com.

8. Vagrant by HashiCorp [Online] / Web portal Vagrant.com. – Available: https://www.vagrantup.com.

9. PHP – Introduction [Online] / Web portal TutorialsPoint. – Available: https://www.tutorialspoint.com/php/php_introduction.htm.

10. Полное руководство по Yii 2.0 [Електронный ресурс] / Веб-портал Yii2. – Режим доступа: https://www.yiiframework.com/doc/guide/2.0/ru.

11. Современный учебник JavaScript [Електронный ресурс] / Веб-портал JAVASCRIPT.RU. – Режим доступа: https:// learn.javascript.ru.

12. Software Testing Overview [Online] / Web portal TutorialsPoint. – Available: https://www.tutorialspoint.com/software_engineering/software_testing _overview.htm

13. Codeception – PHP Testing framework [Online] / Web portal Codeception.com. – Available: https://codeception.com.

ДОДАТОК А

(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки Інтернет-платформи «Бюро знахідок» з підтримкою та використанням Google-карт

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Інтернет-платформа «Бюро знахідок» з використанням Google-карт.

2 Призначення розробки

Функціональне призначення: додавання, редагування, видалення, зберігання постів про втрату / знахідку, реєстрація/авторизація користувачів, спілкування з авторами постів.

Публікація постів на сайті відбувається модератором або адміністратором (так звана постмодерація).

Експлуатаційне призначення: доступ до платформи можна отримати з будь-якого браузера, з телефону або комп’ютера, ніяких додаткових налаштувань не потребує.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Сайт повинен забезпечувати наступні функції:

– функція авторизації та реєстрації;

– функція створення та перегляд власних постів;

– функція виходу з системи;

– функція редагування інформації в особистому кабінеті;

– функція вибору приблизного місця втрати / знахідки на google-карті;

– функція пошуку постів за місцем, назвою, часом;

– функції для адміністрування та модерації (CRUD);

– можливість зв’язку з авторами постів.

3.2 Вимоги до надійності

Система має виконувати наступні вимоги до надійності:

– підтримання захисту від несанкціонованого доступу;

– розмежування прав користувачів у системі;

– обробка помилок та надсилання повідомлень про них;

– валідація на стороні сервера для контролю інформації, що вводиться та виключення введення помилкових даних користувачем.

3.3 Умови експлуатації

Щоб користуватися сайтом, користувачу необхідний будь-який браузер. Абсолютно неважливо з якого пристрою користувач буде заходити на сайт, так як верстка адаптивна. Для керування системою можуть допускатися тільки навчені модератори та адміністратори.

3.4 Вимоги до інформаційної та програмної сумісності

Для створення клієнтської частини будуть використовуватися: мова розмітки HTML і таблиці стилів CSS, мова програмування Javascript.

Для створення серверної частини будуть використовуватися: мова програмування PHP, фреймворк Yii2, які дозволяють створювати

багатофункціональні веб-додатки та веб-сайти.

Для роботи з базою даних буде використовуватися мова SQL та СКБД MariaDB.

3.5 Спеціальні вимоги

Програма повинна мати зручний та інтуїтивно зрозумілий інтерфейс.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:
– текст програми з відповідними коментарями та поясненнями;
– опис програми – відомості про функціонування програми;
– технічне завдання;
– короткий посібник (довідкова інформація) користувачу.

5 Стадії та етапи розробки

Стадії та етапи розробки проекту Інтернет-платформа «Бюро знахідок» наведено в таблиці А.1

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.21 – 31.01.21	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання

Кінець таблиці А.1

1	2	3
Ескізний проект 01.02.21 – 14.02.21	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури системи, що буде розроблюватися
Технічний проект 15.02.21 – 28.02.21	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми; остаточне визначення конфігурації технічних засобів
Робочий проект 01.03.21 – 10.04.21	Розробка програмного забезпечення	Реалізація програмного забезпечення; відладка; проведення попереднього тестування
Розробка програмної документації 11.04.21 – 20.04.21	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 21.04.21 – 30.04.21	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і передача програмного забезпечення; навчання персоналу використовуванню програмного забезпечення; внесення коректувань в програмне забезпечення і документацію

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування системи.

Після закінчення розробки системи повинні бути проведене тестування на захист від некоректного введення.

ДОДАТОК Б

(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Б.1 Програмний код контролерів

Б.1.1 Клієнтська частина

Клас SiteController

```
<?php
namespace app\controllers;
use app\models\repositories\PostRepository;
use Yii;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\web\Response;
use yii\filters\VerbFilter;
class SiteController extends Controller
{
    private $repository;
    public function __construct(
        $id,
        $module,
        PostRepository $repository,
        $config = []
    ) {
        parent::__construct($id, $module, $config);
        $this->repository = $repository;
    }
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'only' => ['logout'],
                'rules' => [
                    [
                        'actions' => ['logout'],
                        'allow' => true,
                        'roles' => ['@'],
                    ],
                ],
            ],
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'logout' => ['post'],
                ],
            ],
        ];
    }
    public function actions()
    {
        return [
            'error' => [
                'class' => 'yii\web\ErrorAction',
            ],
            'captcha' => [
                'class' => 'yii\captcha\CaptchaAction',
                'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
            ],
```

```
        ];
    }
    public function actionIndex()
    {
        $dataProviderFinds = $this->repository->findAllFinds();
        $dataProviderLosts = $this->repository->findAllLosts();
        return $this->render(
            'index',
            [
                'dataProviderFinds' => $dataProviderFinds,
                'dataProviderLosts' => $dataProviderLosts,
            ]
        );
    }
}
```

Клас PostController

```
<?php
namespace app\controllers;
use app\helpers\UuidGenerator;
use app\models\forms\post\PostForm;
use app\models\services\AuthService;
use app\models\services\PostService;
use Yii;
class PostController extends \yii\web\Controller
{
    private $service;
    public function __construct(
        $id,
        $module,
        PostService $postService,
        $config = []
    ) {
        parent::__construct($id, $module, $config);
        $this->service = $postService;
    }
    public function actionFind()
    {
        if (Yii::$app->user->isGuest) {
            return $this->goHome();
        }
        $id    = UuidGenerator::generate();
        $model = new PostForm();

        if ($model->load(Yii::$app->request->post()) && $model->validate()) {
            try {
                $this->service->create($model);

                return $this->redirect('/cabinet');
            } catch (\RuntimeException $e) {
                Yii::$app->errorHandler->logException($e);

                Yii::$app->session->setFlash('error', $e->getMessage());
            }
        }
        return $this->render('forms/find', [
            'model' => $model,
            'id' => $id,
```

```
        ]);
    }
    public function actionLost()
    {
        if (Yii::$app->user->isGuest) {
            return $this->goHome();
        }

        return $this->render('forms/lost', [
            'model' => $model,
        ]);
    }
    public function getViewPath()
    {
        return '@app/views/cabinet';
    }
}
```

Клас FileController

```
<?php
namespace app\controllers;
use app\models\forms\post\PostForm;
use Yii;
use yii\helpers\FileHelper;
use yii\helpers\Json;
use yii\web\UploadedFile;
class FileController extends \yii\web\Controller
{
    public function actionUpload()
    {
        $model = new PostForm();
        $imageFile = UploadedFile::getInstance($model, 'test');
        $directory = Yii::getAlias('@app/web/img/temp') . DIRECTORY_SEPARATOR . Yii::$app->session->id .
DIRECTORY_SEPARATOR;
        if (!is_dir($directory)) {
            FileHelper::createDirectory($directory);
        }
        if ($imageFile) {
            $uid = uniqid(time(), true);
            $fileName = $uid . '.' . $imageFile->extension;
            $filePath = $directory . $fileName;
            $imageFile->saveAs($filePath);
        }

        return '';
    }
    public function actionDelete($name)
    {
        $directory = Yii::getAlias('@app/web/img/temp') . DIRECTORY_SEPARATOR . Yii::$app->session->id;
        if (is_file($directory . DIRECTORY_SEPARATOR . $name)) {
            unlink($directory . DIRECTORY_SEPARATOR . $name);
        }
        $files = FileHelper::findFiles($directory);
        $output = [];
        foreach ($files as $file) {
            $fileName = basename($file);
            $path = '/img/temp/' . Yii::$app->session->id . DIRECTORY_SEPARATOR . $fileName;
            $output['files'][] = [
```

```
            'name' => $fileName,
            'size' => filesize($file),
            'url' => $path,
            'thumbnailUrl' => $path,
            'deleteUrl' => 'image-delete?name=' . $fileName,
            'deleteType' => 'POST',
        ];
    }
    return Json::encode($output);
  }
}
```

Клас AuthControler

```
<?php
namespace app\controllers;
use app\models\forms\user\LoginForm;
use app\models\forms\user\SignupForm;
use app\models\services\AuthService;
use Yii;
use yii\base\BaseObject;
use yii\web\Response;
class AuthController extends \yii\web\Controller
{
  private $service;

  public function __construct(
    $id,
    $module,
    AuthService $authService,
    $config = []
  ) {
    parent::__construct($id, $module, $config);
    $this->service = $authService;
  }
  public function actionLogin()
  {
    if (!Yii::$app->user->isGuest) {
      return $this->goHome();
    }
    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->validate()) {
      try {
        $this->service->auth($model);
            return $this->goBack();
      } catch (\DomainException $e) {
        Yii::$app->errorHandler->logException($e);
        Yii::$app->session->setFlash('error', $e->getMessage());
      }
    }
    $model->password = '';
    return $this->render('login', [
      'model' => $model,
    ]);
  }
  public function actionSignup()
  {
    if (!Yii::$app->user->isGuest) {
      return $this->goHome();
```

```
        }
        $model = new SignupForm();
        if ($model->load(Yii::$app->request->post()) && $model->validate()) {
            try {
                $this->service->signup($model);

                return $this->goBack();
            } catch (\RuntimeException $e) {
                Yii::$app->errorHandler->logException($e);

                Yii::$app->session->setFlash('error', $e->getMessage());
            }
        }
        $model->password = '';
        return $this->render('register', [
            'model' => $model,
        ]);
    }
    public function actionLogout()
    {
        Yii::$app->user->logout();

        return $this->goHome();
    }
}
```

Б.1.2 Адміністративна частина

Клас AdminController

```
<?php
namespace app\controllers\backend;
use yii\filters\VerbFilter;
use yii\web\Controller;
class AdminController extends Controller
{
    public $layout = '@app/views/backend/layouts/main.php';
    public function behaviors()
    {
        return [
            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'logout' => ['post'],
                ],
            ],
        ];
    }
    public function actionIndex()
    {
        return $this->render('index');
    }
    public function getViewPath()
    {
        return '@app/views/backend';
    }
}
```

Клас PostController

```
<?php
namespace app\controllers\backend;
use app\models\forms\backend\PostSearch;
use app\models\forms\backend\PostUpdate;
use app\models\repositories\PostRepository;
use Yii;
use yii\web\NotFoundHttpException;
class PostController extends \yii\web\Controller
{
    public $layout = '@app/views/backend/layouts/main.php';
    private $postRepository;
    public function __construct(
        $id,
        $module,
        PostRepository $postRepository,
        $config = []
    ) {
        parent::__construct($id, $module, $config);
        $this->postRepository = $postRepository;
    }
    public function actionIndex()
    {
        $searchModel = new PostSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    public function actionView(string $id)
    {
        try {
            $model = $this->postRepository->find($id);
        } catch (NotFoundHttpException $e) {
            Yii::$app->errorHandler->logException($e);
            Yii::$app->session->setFlash('viewError', $e->getMessage());
            return $this->redirect(Yii::$app->request->referrer);
        }
        return $this->render('view', [
            'model' => $model,
        ]);
    }
    public function actionUpdate($id)
    {
        $user = $this->postRepository->find($id);
        $model = new PostUpdate($user);

        if ($model->load(Yii::$app->request->post()) && $model->validate()) {
            try {
                $this->service->edit($user->id, $model);
            } catch (\DomainException $e) {
                Yii::$app->errorHandler->logException($e);
                Yii::$app->session->setFlash('error', $e->getMessage());
            }
            return $this->redirect(['view', 'id' => $user->id]);
        }
        return $this->render('update', [
```

```
            'model' => $model,
        ]);
    }
    public function actionDelete($id)
    {
        try {
            $this->service->remove($id);
        } catch (NotFoundHttpException $e) {
            Yii::$app->errorHandler->logException($e);
            Yii::$app->session->setFlash('viewError', $e->getMessage());
            return $this->redirect(Yii::$app->request->referrer);
        }
        return $this->redirect(['admin/users']);
    }
    public function getViewPath()
    {
        return "@app/views/backend/post";
    }
}
```

Клас UserController

```
<?php
namespace app\controllers\backend;
use app\models\forms\backend\UserSearch;
use app\models\forms\backend\UserUpdate;
use app\models\repositories\UserRepository;
use app\models\services\UserManageService;
use Yii;
use yii\web\NotFoundHttpException;
class UserController extends \yii\web\Controller
{
    public $layout = '@app/views/backend/layouts/main.php';
    private $usersRepository;
    private $service;
    public function __construct(
        $id,
        $module,
        UserRepository $usersRepository,
        UserManageService $userManageService,
        $config = []
    ) {
        parent::__construct($id, $module, $config);
        $this->usersRepository = $usersRepository;
        $this->service = $userManageService;
    }
    public function actionIndex()
    {
        $searchModel = new UserSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    public function actionView(string $id)
    {
        try {
```

```
            $model = $this->usersRepository->find($id);
        } catch (NotFoundHttpException $e) {
            Yii::$app->errorHandler->logException($e);
            Yii::$app->session->setFlash('viewError', $e->getMessage());
            return $this->redirect(Yii::$app->request->referrer);
        }
        return $this->render('view', [
            'model' => $model,
        ]);
    }
    public function actionUpdate($id)
    {
        $user = $this->usersRepository->find($id);
        $updateForm = new UserUpdate($user);

        if ($updateForm->load(Yii::$app->request->post()) && $updateForm->validate()) {
            try {
                $this->service->edit($user->id, $updateForm);
            } catch (\DomainException $e) {
                Yii::$app->errorHandler->logException($e);
                Yii::$app->session->setFlash('error', $e->getMessage());
            }
            return $this->redirect(['view', 'id' => $user->id]);
        }
        return $this->render('update', [
            'updateForm' => $updateForm,
        ]);
    }
    public function actionDelete($id)
    {
        try {
            $this->service->remove($id);
        } catch (NotFoundHttpException $e) {
            Yii::$app->errorHandler->logException($e);
            Yii::$app->session->setFlash('viewError', $e->getMessage());
            return $this->redirect(Yii::$app->request->referrer);
        }
        return $this->redirect(['admin/users']);
    }
    public function getViewPath()
    {
        return "@app/views/backend/user";
    }
}
```

Б.2 Програмний код моделей-сутностей

Клас Post

```
<?php
namespace app\models\entities;
use app\helpers\UuidGenerator;
use yii\behaviors\TimestampBehavior;
use yii\db\Expression;
use yii\helpers\ArrayHelper;
/**
 * This is the model class for table "{{%posts}}".
 *
 * @property string $id
 * @property string $title
```

```
 * @property string $text
 * @property int $date
 * @property int $type
 * @property int $reward
 * @property int $status
 * @property string $coordinates
 * @property int $created_at
 * @property int $updated_at
 */
class Post extends \yii\db\ActiveRecord
{
    public const STATUS_WAIT = 0;
    public const STATUS_ACTIVE = 1;
    public const TYPE_FIND = 10;
    public const TYPE_LOST = 11;
    public static function tableName(): string
    {
        return '{{%posts}}';
    }
    public function behaviors(): array
    {
        return [
            [
                'class' => TimestampBehavior::class,
                'createdAtAttribute' => 'created_at',
                'updatedAtAttribute' => 'updated_at',
                'value' => new Expression('CURRENT_TIMESTAMP()'),
            ],
        ];
    }
    public function getPhotos()
    {
        return self::hasMany(File::class, ['entity_id' => 'id']);
    }
    public function getPhoto()
    {
        return self::hasOne(File::class, ['entity_id' => 'id']);
    }
    public function getStatusName()
    {
        return ArrayHelper::getValue(self::getStatusesArray(), $this->status);
    }
    public static function getStatusesArray(): array
    {
        return [
            self::STATUS_WAIT => 'На модерації',
            self::STATUS_ACTIVE => 'Активний',
        ];
    }
    public function getTypeName()
    {
        return ArrayHelper::getValue(self::getTypesArray(), $this->type);
    }
    public static function getTypesArray(): array
    {
        return [
            self::TYPE_FIND => 'Знахідка',
            self::TYPE_LOST => 'Втрата',
        ];
    }
    public static function create(
        $title,
```

```
        $text,
        $date,
        $type,
        $reward,
        $status,
        $coordinates
    ): Post {
        $post = new self();
        $post->id = UuidGenerator::generate();
        $post->title = $title;
        $post->text = $text;
        $post->date = $date;
        $post->type = $type;
        $post->reward = $reward;
        $post->status = $status;
        $post->coordinates = $coordinates;

        return $post;
    }
    public static function edit(
        $title,
        $text,
        $date,
        $type,
        $reward,
        $status,
        $coordinates
    ): void {
        $this->id = UuidGenerator::generate();
        $this->title = $title;
        $this->text = $text;
        $this->date = $date;
        $this->type = $type;
        $this->reward = $reward;
        $this->status = $status;
        $this->coordinates = $coordinates;
    }
}
```

Клас UserToken

```
<?php
namespace app\models\user;
use app\helpers\UuidGenerator;
use Yii;
use yii\behaviors\TimestampBehavior;
use yii\db\ActiveQuery;
use yii\db\ActiveRecord;
/**
 * This is the model class for table "{{%user_token}}".
 *
 * @property integer $id
 * @property string $type
 * @property string $token
 * @property integer $expire_at
 * @property integer $created_at
 * @property integer $updated_at
 * @property integer $status
 * @property integer $user_id
```

```
 *
 * @property UserToken $userToken
 */
class UserToken extends ActiveRecord
{
    public const STATUS_ACTIVE  = 1;
    public const STATUS_DELETED = 0;
    public const TYPE_ACTIVATION     = 'activation';
    public const TYPE_PASSWORD_RESET = 'password_reset';
    public const TYPE_LOGIN_PASS     = 'login_pass';
    public const LOGIN_URL = 'user/signin/login-by-pass';
    protected const TOKEN_LENGTH = 40;
    public function __toString()
    {
        return $this->token;
    }
    public function behaviors(): array
    {
        return [TimestampBehavior::class];
    }
    public static function tableName(): string
    {
        return '{{%user_token}}';
    }
    public function rules(): array
    {
        return [
            [
                [
                    'user_id',
                    'type',
                    'token',
                ], 'required',
            ],
            [
                ['expire_at'],
                'integer',
            ],
            [
                ['type'], 'string', 'max' => 255
            ],
            [
                ['token'], 'string', 'max' => self::TOKEN_LENGTH
            ],
        ];
    }
    public function getUser(): ActiveQuery
    {
        return $this->hasOne(User::class, ['id' => 'user_id']);
    }
    public static function create(string $type, int $duration = null): self
    {
        $userToken = new self();

        $userToken->id        = UuidGenerator::generate();
        $userToken->token     = Yii::$app->security->generateRandomString(self::TOKEN_LENGTH);
        $userToken->type      = $type;
        $userToken->expire_at = (true === (bool)$duration) ? (time() + $duration) : null;
        $userToken->status    = self::STATUS_ACTIVE;

        return $userToken;
    }
```

```
    public static function edit(string $type, int $duration = null): void
    {

        $this->id       = UuidGenerator::generate();
        $this->token    = Yii::$app->security->generateRandomString(self::TOKEN_LENGTH);
        $this->type     = $type;
        $this->expire_at = (true === (bool)$duration) ? (time() + $duration) : null;
        $this->status   = self::STATUS_ACTIVE;
    }

}
```

Клас UserIdentity

```
<?php
namespace app\models\user;
use yii\web\IdentityInterface;
class UserIdentity implements IdentityInterface
{
    private $user;
    public function __construct(User $user)
    {
        $this->user = $user;
    }
    public static function findIdentity($id): ?UserIdentity
    {
        //set identity
        $user = User::findOne(['id' => $id, 'status' => User::STATUS_ACTIVE]);
        return $user ? new self($user) : null;
    }
    public function getId()
    {
        return $this->user->id;
    }
     public function getEmail()
    {
          return $this->user->email;
    }
    public function getAuthKey(): string
    {
        return $this->user->auth_key;
    }
    public function validateAuthKey($authKey): bool
    {
        return $this->getAuthKey() === $authKey;
    }
    public static function findIdentityByAccessToken($token, $type = null)
    {
        // TODO: Implement findIdentityByAccessToken() method.
    }
    public function __get($name)
    {
        return $this->user->$name;
    }
    public function __call($methodName, $args)
    {
        return $this->user->$methodName($args);
    }
}
```

Б.3 Програмний код сервісів

Клас UserManageService

```
<?php
namespace app\models\services;
use app\models\forms\backend\UserUpdate;
use app\models\repositories\UserRepository;
class UserManageService
{
    private $repository;
    public function __construct(UserRepository $repository)
    {
        $this->repository = $repository;
    }
    public function edit($id, UserUpdate $form): void
    {
        $user = $this->repository->find($id);
        $user->edit(
            $form->username,
            $form->email,
            $form->status
        );
        $this->repository->save($user);
    }
    public function remove($id): void
    {
        $user = $this->repository->find($id);
        $this->repository->remove($user);
    }
}
```

Клас PostService

```
<?php
namespace app\models\services;
use app\models\entities\Post;
use app\models\forms\post\PostForm;
use app\models\repositories\PostRepository;
class PostService
```

```
{
    private $repository;
    public function __construct(PostRepository $repository)
    {
        $this->repository = $repository;
    }
    public function create(PostForm $form): void
    {
        $post = Post::create(
            $form->title,
            $form->text,
            $form->date,
            $form->test,
            $form->type,
            $form->reward,
            $form->coordinates
        );

        $transaction = \Yii::$app->db->beginTransaction();
        try {
            $this->repository->save($post);
            $transaction->commit();
        } catch (\Exception $ex) {
            $transaction->rollBack();
            throw $ex;
        }
    }
    public function edit(string $id, PostForm $form): void
    {
        $post = $this->repository->find($id);
        $post->edit(
            $form->title,
            $form->text,
            $form->date,
            $form->test,
            $form->type,
            $form->reward,
            $form->coordinates
        );
        $transaction = \Yii::$app->db->beginTransaction();
        try {
            $this->repository->save($post);
```

```
            $transaction->commit();
        } catch (\Exception $ex) {
            $transaction->rollBack();
            throw $ex;
        }
    }
    public function remove($id): void
    {
        $transaction = \Yii::$app->db->beginTransaction();
        try {
            $this->repository->remove($id);
            $transaction->commit();
        } catch (\Exception $ex) {
            $transaction->rollBack();
            throw $ex;
        }
    }
}
```

Б.4 Програмний код репозиторіїв

Клас PostRepository

```
<?php
namespace app\models\repositories;
use app\models\entities\Post;
use app\models\forms\post\PostSearch;
use yii\data\ActiveDataProvider;
use yii\data\Pagination;
class PostRepository
{
    public function find($id): ?Post
    {
        return Post::findOne($id);
    }
    public function findAll()
    {
        return Post::findAll(['status' => Post::STATUS_ACTIVE]);
    }
    public function findAllFinds()
    {
        return Post::find()->andWhere(
                [
                    'status' => Post::STATUS_ACTIVE,
                    'type'   => Post::TYPE_FIND,
                ]
            )
            ->orderBy('updated_at')
            ->limit(8)
            ->all();
```

```
    }
    public function findAllLosts()
    {
        return Post::findAll(
            [
                'status' => Post::STATUS_ACTIVE,
                'type'   => Post::TYPE_LOST,
            ]
        );
    }
    public function save(Post $user): void
    {
        if (!$user->save()) {
            throw new \RuntimeException('Saving error.');
        }
    }
    public function remove($id): void
    {
        $user = $this->find($id);
        if (!$user->delete()) {
            throw new \RuntimeException('Removing error.');
        }
    }
    private function getBy(array $condition)
    {
        $user = Post::find()->andWhere($condition)->limit(1)->one();

        if (null === $user) {
            throw new \DomainException('User not found.');
        }

        return $user;
    }
    public function search(PostSearch $form)
    {
        $pagination = new Pagination([
            'pageSize' => 6,
            'pageSizeParam' => false
        ]);

        $query = Post::find()
            ->where(['user_id' => \Yii::$app->user->id])
            ->with('user');

        return new ActiveDataProvider([
            'query' => $query,
            'pagination' => $pagination,
        ]);
    }
}
```

Клас UserRepository

```
<?php
namespace app\models\repositories;
use app\models\user\User;
class UserRepository
```

```
{
    public function find($id): ?User
    {
        return User::findOne($id);
    }
    public function save(User $user): void
    {
        if (!$user->save()) {
            throw new \RuntimeException('Saving error.');
        }
    }
    public function remove($id): void
    {
        $user = $this->find($id);
        if (!$user->delete()) {
            throw new \RuntimeException('Removing error.');
        }
    }
    public function findByEmail($value): ?User
    {
        return User::findOne(['email' => $value]);
    }
    public function findAllUserIds(): array
    {
        return User::find()->select('id')->asArray()->column();
    }
    private function getBy(array $condition)
    {
        $user = User::find()->andWhere($condition)->limit(1)->one();
        if (null === $user) {
            throw new \DomainException('User not found.');
        }
        return $user;
    }
}
```

Клас UserTokenRepository

```
<?php
namespace app\models\repositories;
use app\models\user\UserToken;
class UserTokenRepository
```

```
{
    public function find($id): ?UserToken
    {
        return UserToken::findOne($id);
    }
    public function save(UserToken $userToken): void
    {
        if (!$userToken->save()) {
            throw new \RuntimeException('Saving error.');
        }
    }
    public function remove($id): void
    {
        $userToken = $this->find($id);
        if (!$userToken->delete()) {
            throw new \RuntimeException('Removing error.');
        }
    }
    private function getBy(array $condition)
    {
        $userToken = UserToken::find()->andWhere($condition)->limit(1)->one();

        if (null === $userToken) {
            throw new \DomainException('UserToken not found.');
        }

        return $userToken;
    }
}
```

Б.5 Програмний код форм

Клас LoginForm

```
<?php
namespace app\models\forms\user;
class LoginForm extends \yii\base\Model
{
    public $email;
    public $password;
    public $rememberMe;
    public function rules(): array
    {
        return [
            [['email', 'password'], 'required'],
            ['rememberMe', 'boolean'],
        ];
    }
}
```

Клас SignUpForm

```
<?php
namespace app\models\forms\user;
use app\models\user\User;
use yii\base\Model;
```

```
class SignupForm extends Model
{
    public $username;
    public $email;
    public $phone;
    public $password;
    public $passwordRepeat;
    public $agreement;
    public function rules(): array
    {
        return [
            ['username', 'filter', 'filter' => 'trim'],
            [['username', 'agreement'], 'required'],
            ['username', 'match', 'pattern' => '#^[\w_-]+$#i'],
            ['username', 'unique', 'targetClass' => User::class, 'message' => 'This username is already taken'],
            ['username', 'string', 'min' => 4, 'max' => 255],

            ['email', 'filter', 'filter' => 'trim'],
            ['email', 'required'],
            ['email', 'email'],
            ['email', 'unique', 'targetClass' => User::class, 'message' => 'This male is already taken'],

            ['password', 'required'],
            ['password', 'string', 'min' => 8],

            ['passwordRepeat', 'compare', 'compareAttribute' => 'password'],
            ['passwordRepeat', 'required'],
        ];
    }
}
```

Клас PostEditForm

```
<?php
namespace app\models\forms\post;
use app\models\entities\Post;
use yii\base\Model;
class PostEditForm extends \yii\base\Model
{
    public $id;
    public $title;
    public $text;
    public $date;
    public $test;
    public $type;
    public $status;
    public $reward;
    public $coordinates;

    public function rules(): array
    {
        return [
            [['id', 'title', 'text'], 'string'],
            [['reward', 'type', 'status'], 'integer'],
            [['date'], 'datetime', 'format' => 'yyyy-MM-dd'],
            [['coordinates'], 'safe'],
        ];
    }
    public function __construct(Post $post, $config = [])
```

```
    {
        parent::__construct($config);
        $this->title = $post->title;
        $this->text = $post->text;
        $this->date = $post->date;
        $this->type = $post->type;
        $this->status = $post->status;
        $this->reward = $post->reward;
        $this->coordinates = $post->coordinates;
    }
}
```

Клас PostForm

```
<?php
namespace app\models\forms\post;
use app\models\entities\Post;
use yii\base\Model;
class PostForm extends Model
{
    public $id;
    public $title;
    public $text;
    public $date;
    public $test;
    public $type;
    public $status;
    public $reward;
    public $coordinates;
    public function rules(): array
    {
        return [
            [['id', 'title', 'text'], 'string'],
            [['reward', 'type', 'status'], 'integer'],
            [['date'], 'datetime', 'format' => 'yyyy-MM-dd'],
            [['coordinates'], 'safe'],
        ];
    }
}
```

Клас PostSearch

```
<?php
namespace app\models\forms\post;
use app\models\entities\Post;
use yii\base\BaseObject;
use yii\data\ActiveDataProvider;
class PostSearch extends \yii\base\Model
{
    public $title;
    public $type;
    public $status;
    public function rules(): array
    {
        return [
            [['title'], 'string'],
```

```
            [['type', 'status'], 'integer'],
        ];
    }
    public function search(array $params): ActiveDataProvider
    {
        $query = Post::find();

        // add conditions that should always apply here

        $dataProvider = new ActiveDataProvider([
            'query' => $query,
        ]);

        $this->load($params);

        if (!$this->validate()) {
            // uncomment the following line if you do not want to return any records when validation fails
            // $query->where('0=1');
            return $dataProvider;
        }

        $query->andFilterWhere(['like', 'title', $this->title]);

        $query->andFilterWhere(
            [
                'type', $this->type,
                'status', $this->status,
            ]
        );

        return $dataProvider;
    }
}
```

Клас UserSearch

```
<?php
namespace app\models\forms\backend;
use app\models\user\User;
use yii\data\ActiveDataProvider;
class UserSearch extends \yii\base\Model
{
    public $id;
    public $username;
    public $email;
    public $status;
    public $created_at;
    public $updated_at;

    public function rules(): array
    {
        return [
            [['id', 'status', 'created_at', 'updated_at'], 'integer'],
            [['username', 'email'], 'safe'],
        ];
    }
    public function search(array $params): ActiveDataProvider
    {
        $query = User::find();
```

```
        // add conditions that should always apply here

        $dataProvider = new ActiveDataProvider([
            'query' => $query,
        ]);

        $this->load($params);

        if (!$this->validate()) {
            // uncomment the following line if you do not want to return any records when validation fails
            // $query->where('0=1');
            return $dataProvider;
        }

        // grid filtering conditions
        $query->andFilterWhere([
            'id' => $this->id,
            'status' => $this->status,
        ]);

        $query->andFilterWhere(['like', 'username', $this->username])
            ->andFilterWhere(['like', 'email', $this->email]);

        return $dataProvider;
    }
}
```

Клас UserUpdate

```
<?php
namespace app\models\forms\backend;
use app\models\user\User;
class UserUpdate extends \yii\base\Model
{
    public $username;
    public $email;
    public $status;

    public function __construct(User $user, $config = [])
    {
        parent::__construct($config);
        $this->username = $user->username;
        $this->email = $user->email;
        $this->status = $user->status;
    }

    public function rules(): array
    {
        return [
            [['username', 'email'], 'filter', 'filter' => 'trim'],
            [['username', 'email'], 'required'],
            [['email'], 'email'],
            [['username', 'email'], 'string', 'max' => 255],
            [['status'], 'safe'],
        ];
    }
}
```

Б.6 Програмний код допоміжних функцій

Клас UuidGenerator

```
<?php
namespace app\helpers;
use Ramsey\Uuid\Uuid;
class UuidGenerator
{
   public static function generate() : string
   {
      return Uuid::uuid4()->toString();
   }
}
```

ДОДАТОК В

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Кафедра інженерії програмного забезпечення

Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт

Виконав:
Студент 4 курсу, група ІПЗ-17-1
Стьопич Владислав

Керівник:
Доцент, кандидат технічних наук
Радельчук Г. І.

Актуальність теми

На сьогоднішній день існує така проблема: користувачі соціальних мереж та різноманітних месенджерів часто бачать такі типи постів, як: «Шукаю (річ, домашнього улюбленця, людину, документ)» або «Знайшов ..., поверну». Оскільки ці пости розміщуються у стрічці, то вони мають таку властивість як губитися. Буде достатньо, щоб загубитися десь в стрічці звичайного користувача з кількістю підписників більше 100. А отже шанс того, що втрачена/знайдена річ повернеться до власника, зменшується.

Саме тому, потрібно створити спеціальну платформу, де буде зберігатися такий тип даних. Особливо зручною є електронна бібліотека, реалізована у вигляді веб-сайту, оскільки для роботи з нею не потрібно встановлювати додаткового програмного забезпечення (ПЗ); потрібні лише браузер і доступ до мережі Інтернет.

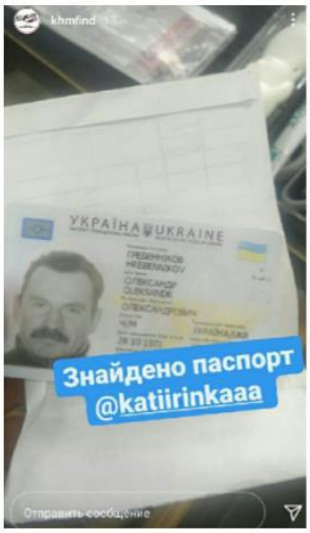
Метою дипломного проектування є створення Інтернет–платформи, яка призначена для допомоги громадянам України у віднаходженні втрачених речей.

Завдання проектування:

1. провести аналіз предметної області та виявити її особливості;
2. провести аналіз ринку на наявність готового вирішення поставленої проблеми дипломного проекту;
3. обрати стиль серверної архітектури розроблюваної Інтернет-платформи;
4. обрати інструменти та технології для розробки, які зможуть задовільнити поставлені вимоги;
5. реалізувати серверну та клієнтську частини Інтернет-платформи;
6. написати інструкцію користування Інтернет-платформою;
7. розписати про процес запуску Інтернет-платформи на хостингу;
8. провести модульне тестування системи на наявність недоліків та відповідність поставленим вимогам.

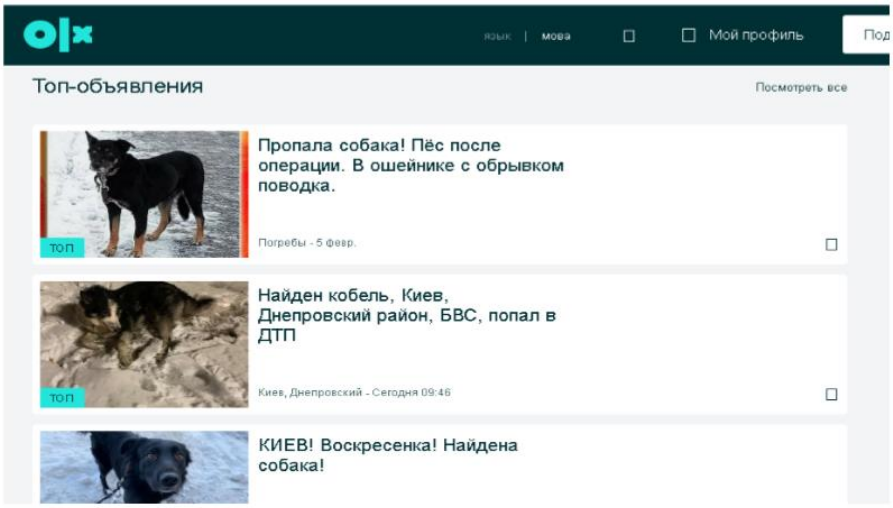
Аналіз наявного програмно-технічного забезпечення

Месенджери / соціальні мережі

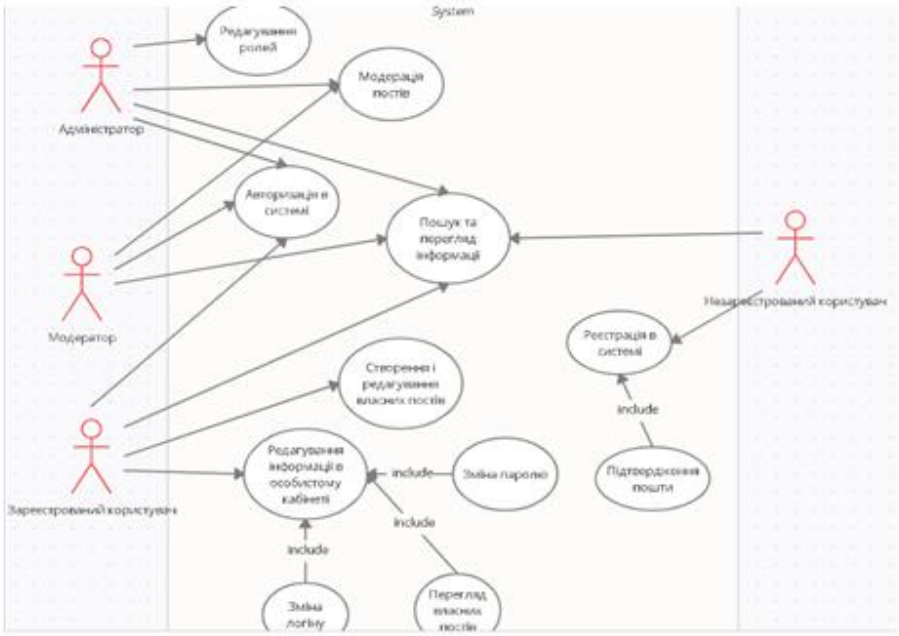


Аналіз наявного програмно – технічного забезпечення

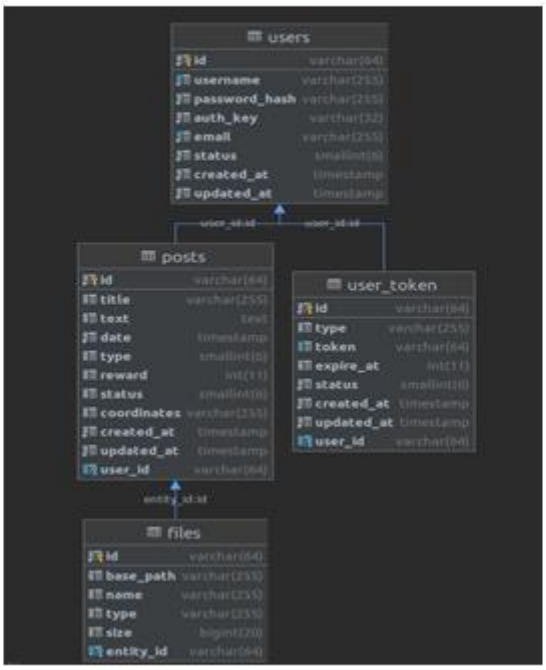
Онлайн дошки оголошень



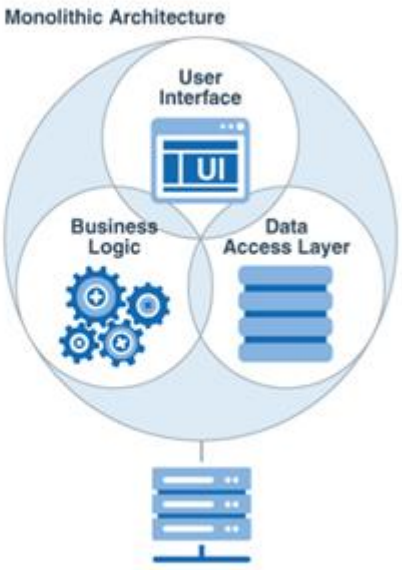
Діаграма варіантів використання

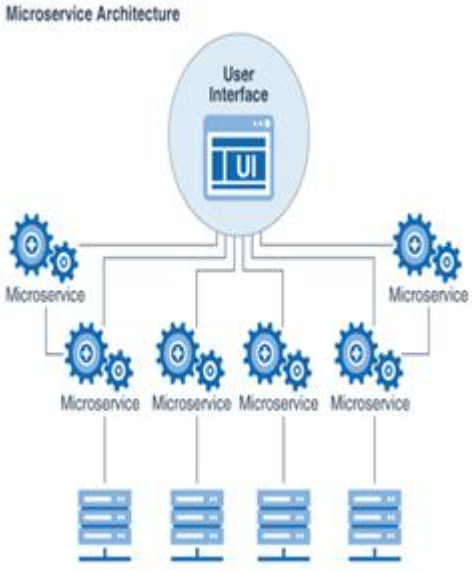


Модель бази даних



Вибір архітектури розроблюваного програмного забезпечення





Інструменти та технології для реалізації

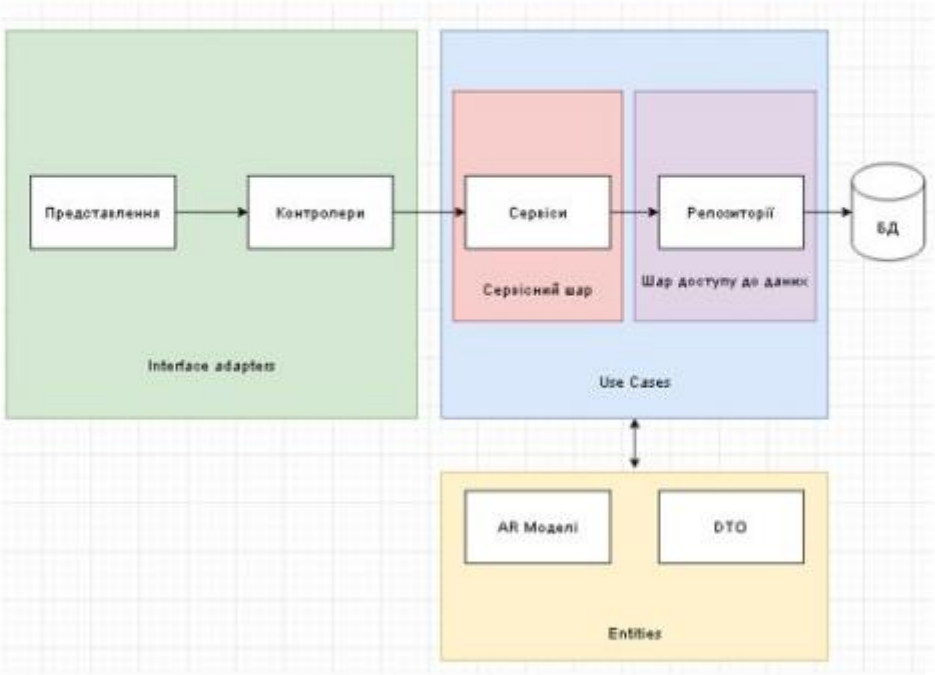
yiiframework

MariaDB

NGINX

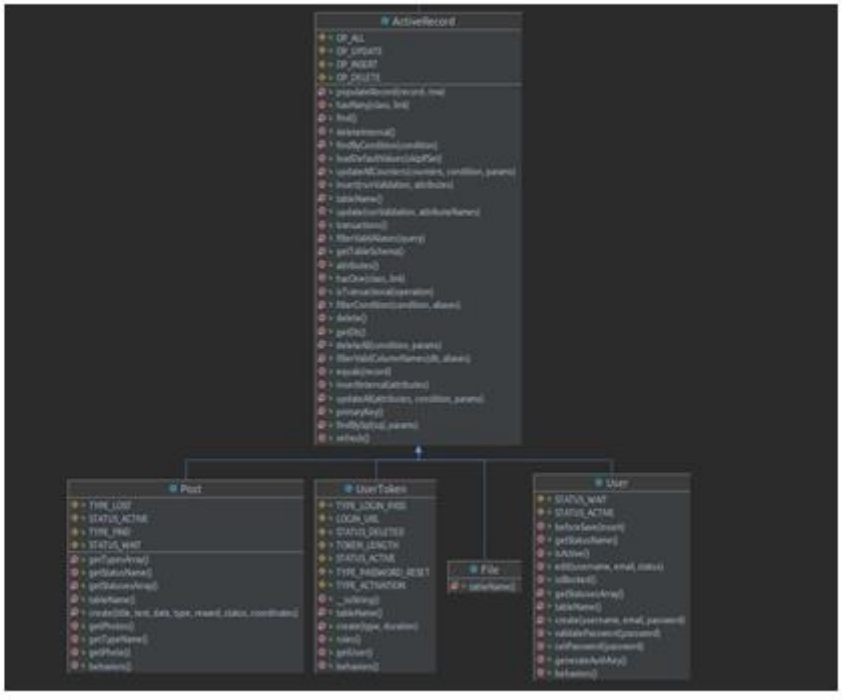
php

Реалізація архітектури Інтернет-платформи



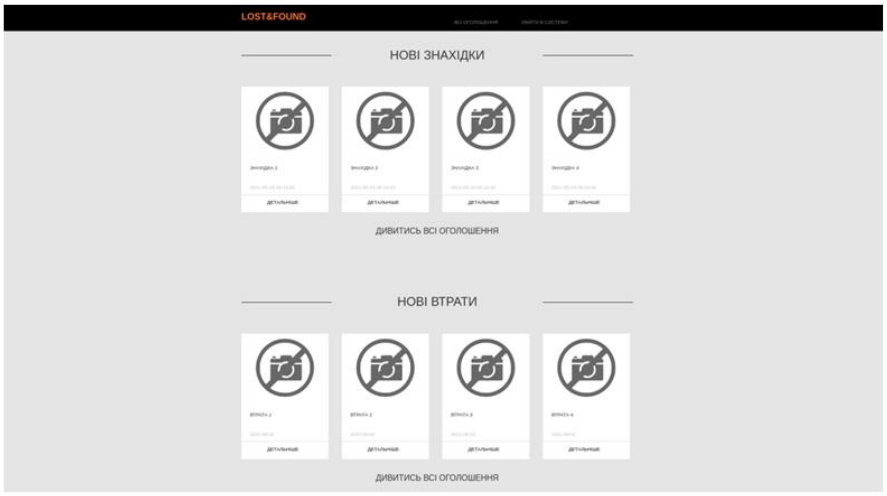
Сервісний шар та шар доступу до даних

AR моделі



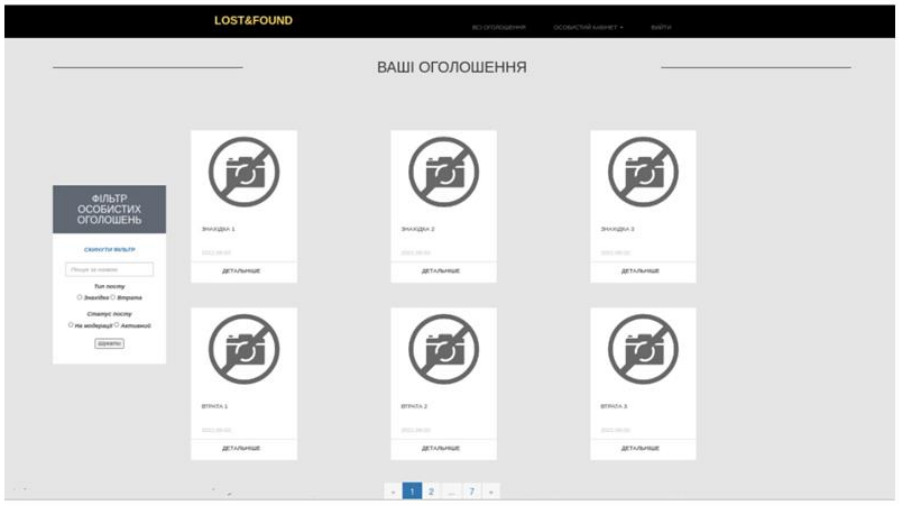
Веб-інтерфейс клієнта

Головна сторінка



Веб-інтерфейс клієнта

Особистий кабінет користувача



Веб-інтерфейс клієнта

Адміністративна панель

Висновок

Темою дипломного проектування є Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт. При аналізі існуючого програмного забезпечення на ринку було виявлено, що проблема, яку вирішує тема дипломного проектування, вирішується у вигляді онлайн дошок або створення оголошень в соціальних мережах або месенджерах.

На етапі проектування була розглянута клієнт-серверна архітектура додатку, основи її роботи. Були проаналізовані стилі архітектур, такі як мікросервіси та моноліт, їхні переваги та недоліки.

Для того, щоб реалізувати дипломний проект, були розглянуті основні інструменти, які використовуються на ринку для розробки веб-додатків.

При тестуванні дипломного проектування були розроблені тест-кейси для основних модулів системи. Було виконане модульне тестування в ході якого було визначено, що система коректно функціонує.

Отже, у результаті виконання дипломного проекту була реалізована Інтернет-платформа, яка допомагає користувачам повертати втрачене.

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Стьопича В .В.
Прізвище, ініціали

факультет ПКТС, 4 курс, група ІПЗ-17-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

01.06.2021р.
дата

підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 3.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 22%**

	Документ		Сумарний збіг по Базі Даних	
ID: 92376 Назва: Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт Додано в БД: 2021-06-06 Автора: В. В. Стьопич Керівники: Г. І. Радельчук Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	96629	1008	5566 (6%)	93 (9%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1008196324

Дата перевірки:
06.06.2021 14:51:06 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2021 15:04:31 EEST

ID користувача:
100005589

Назва документа: **Стьопич Диплом Бюро знахідок**

Кількість сторінок: **107** Кількість слів: **14285** Кількість символів: **118438** Розмір файлу: **3.25 MB** ID файлу: **1008272511**

17.7%
Схожість

Найбільша схожість: **6.88%** з джерелом з Бібліотеки (ID файлу: **1008272523**)

10.4% Джерела з Інтернету	444	Сторінка 109
8.89% Джерела з Бібліотеки	70	Сторінка 113

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	3

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНИЙ ПРОЕКТ
освітнього ступеня «Бакалавр»

Дипломник Стьопич Владислав Валерійович

Тема Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломного проекту:

Кількість листів креслень 0 ; кількість сторінок записки 105

1.Короткий зміст пояснювальної записки та прийнятих рішень У дипломному проекті проведений аналіз предметної області та проблеми, які мають бути вирішені. Також проведений аналіз наявного програмного забезпечення на предмет достатності їх функціоналу для вирішення поставлених задач. На основі отриманих даних були розроблені функціональні вимоги до Інтернет-платформи. Також розглянуті деталі реалізації Інтернет-платформи, проведено її тестування, за результатами якого можна вважати, що платформа працює коректно.

2. Висновок про відповідність проекту поставленому завданню Дипломний проект освітнього ступеня «бакалавр» в основному відповідає поставленому завданню як в теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовується актуальність обраної теми дипломного проектування, формуються мета і завдання проектування, описується практична значимість дипломного проекту. У першому розділі виконано аналіз предметної області, охарактеризовані існуючі рішення для вирішення поставленої задачі, викладено опис функціональних вимог до розроблюваної Інтернет-платформи. У наступних розділах розкриваються деталі проектування Інтернет-платформи, моделювання бази даних та системи в цілому, а також описана програмна реалізація системи та тестування

4. Позитивні сторони проекту Тема дипломного проекту є актуальною, оскільки люди дійсно часто гублять свої речі. До позитивних сторін реалізації Інтернет-платформи можна віднести те, що при розробці системи використовувалася низка новітніх технологій .

5. Негативні сторони проекту До негативних сторін проекту можна віднести веб-інтерфейс. Він є занадто простим на сьогодні. Окрім того, на реалізованій Інтернет-платформі відсутня інформація про суть та діяльність Інтернет-платформи. Щодо зручності використання, то на сторінці з усіма оголошеннями не вистачає фільтру пошуку за місцезнаходженням.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконане відповідно до теми дипломного проекту. Графічне представлення виконано на задовільному рівні. Пояснювальна записка відповідає стандартам оформлення.

7. Відгук про дипломний проект в цілому В цілому дипломний проект заслуговує задовільної оцінки. Матеріал пояснювальної записки структурований та розписаний на рівні, що дозволяє зрозуміти викладений матеріал у рамках тематики дипломного проекту. Графічний матеріал підкріплений інформацією та дозволяє наочно побачити результати виконання дипломного проекту.

8. Інші зауваження

9. Оцінка дипломного проекту Дипломний проект заслуговує оцінки «задовільно»

РЕЦЕНЗЕНТ Мартинюк Валерій Володимирович, доктор технічних наук, професор, зав. кафедри автоматизації, комп'ютерно-інтегрованих технологій і телекомунікацій (АКІТ І ТК)

"01" червня 2021 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Інтернет-платформа «Бюро знахідок» з підтримкою та використанням Google-карт»

Автор: Стьопич Владислав Валерійович

Спеціальність: 121 – Інженерія програмного забезпечення

Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»

Науковий керівник: Радельчук Галина Іванівна, канд. техн. наук, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломного проекту системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання на проектування, відомість документів), у структурі ЗМІСТУ, написах в рамках, назвах розділів/підрозділів тощо) та в назвах переліку джерел посилання;

2) в якості запозичень системою зафіксовано деякі послідовності вихідного коду, які є стандартними мовними конструкціями, спільними для значної кількості задач, і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) найбільшу схожість виявлено з одним документом: вона становить 6,88% в частині загальноприйнятої термінології;

4) усі запозичення є фрагментарними або мають належним чином оформленні посилання.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 17.7% і адресується до 444 першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ Г. І. Радельчук

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк