

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти (освітній ступінь))

на тему ВЕБПЛАТФОРМА ФОТОКОНТЕНТУ ІЗ ЗАСТОСУВАННЯММ
ФРЕЙМВОРКІВ YII, KONANA ТА SYMPHONY

Виконав: студент 2 курсу, групи КНТ-622м
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Спеціалізовані комп'ютерні системи»

ПРЕСНОВ Д.О.

(ПРИЗВИЩЕ та ініціали)

Керівник КИРИЧЕК Г. Г

(ПРИЗВИЩЕ та ініціали)

Рецензент ЄФИМЕНКО М.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Спеціалізовані комп'ютерні системи
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.

« » 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТА
ПРЕСНОВ Денис Олегович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Вебплатформа фотоконтенту із застосуванням фреймворків Yii, Kohana та Symfony

керівник проекту (роботи) КИРИЧЕК Галина Григорівна к. т. н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «24» жовтня 2023 року № 400

2. Строк подання студентом проекту (роботи) 10 грудня 2023 року

3. Вихідні дані до проекту (роботи) система призначена для хостингу фотопорталу, що включає в себе web-платформа з динамічними сторінками та базою даних. Максимальна кількість одночасно підключених клієнтів – 200. Вимоги до програмного забезпечення: сервер Apache, інтерпретатор PHP з фреймворками, СКБД MySQL.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- 1) Аналіз сучасних засобів побудови web-платформ;
- 2) Створення web-платформи;
- 3) Впровадження бази даних web-платформи;
- 4) Дослідження розробленої системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	КИРИЧЕК Г. Г., к.т.н., доцент		
нормо контроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 02.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	12.10.2023 р.	
2	Аналіз засобів побудови web-платформ	14.10.2023 р.	
3	Проектування структури сайту	17.10.2023 р.	
4	Проектування дизайну сайту	20.10.2023 р.	
5	Створення web-платформи на основі фреймворка Yii	12.11.2023 р.	
6	Проектування бази даних	20.11.2023 р.	
7	Розгортання бази даних	25.11.2023 р.	
8	Вибір апаратного устаткування	27.11.2023 р.	
9	Налаштування програмного забезпечення	30.11.2023 р.	
10	Обґрунтування системи хостингу	01.12.2023 р.	
11	Розробка засобів щодо забезпечення захисту інформації	05.12.2023 р.	
12	Оформлення ПЗ	08.10.2023 р.	

Студент(ка)



(підпис)

Денис ПРЕСНОВ

(Ім'я ПРИЗВИЩЕ)

Керівник проекту (роботи)



(підпис)

Галина КИРИЧЕК

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 83 с., 37 рис., 1 табл., 30 лістингів, 21 джерел.

APACHE, MYSQL, PHP, CMS, WEB-ПЛАТФОРМА, ФРЕЙМВОРК, ХОСТИНГ.

Об'єкт дослідження – є комп'ютерна система web-платформи фотографа.

Предмет дослідження – основні можливості фреймворку Yii для реалізації дизайну та інтерфейсу web-платформи фотопорталу.

Метою магістерської роботи є дослідження комп'ютерної системи web-платформи фотографа з використанням новітніх технічних і програмних засобів. Метою цієї системи є розміщення web-платформи порталу фотографа в Інтернеті.

Дипломна робота містить детальний аналіз та обґрунтування засобів впровадження системи. Проект заснований на виборі апаратного та програмного забезпечення. Досліджена і впроваджений web-платформа, що містить базу даних. Для дослідження використовувалася платформа Yii, движок СУБД MySQL.

Розглянуто основні можливості фреймворку Yii для втілення дизайну та інтерфейсу web-платформи фотопорталу. Розроблено концептуальну модель та структурну схему бази даних. Web-платформа розміщений за допомогою web-сервера Apache.

ABSTRACT

PP: 83 p., 37 figures, 1 table, 30 listings, 21 sources.

APACHE, MYSQL, PHP, CMS, WEB PLATFORM, FRAMEWORK, HOSTING.

The object of research is the computer system of the photographer's web platform.

The subject of the study is the main capabilities of the Yii framework for implementing the design and interface of the photo portal web platform.

The purpose of the master's thesis is to study the computer system of the photographer's web platform using the latest hardware and software. The purpose of this system is to place the web platform of the photographer's portal on the Internet.

The thesis contains a detailed analysis and justification of the means of implementing the system. The project is based on the choice of hardware and software. A web platform containing a database has been researched and implemented. The Yii platform and the MyISAM database engine were used for the study.

The main capabilities of the Yii framework for implementing the design and interface of the photo portal web platform are considered. A conceptual model and a structural diagram of the database have been developed. The web platform is hosted using the Apache web server.

ЗМІСТ

Вступ.....	8
1 Аналіз сучасних засобів побудови web-платформ	9
1.1 CMS	9
1.1.1 CMS Wordpress	9
1.1.2 CMS Joomla.....	10
1.1.3 CMS Drupal	10
1.1.4 Вибір CMS	11
1.2 Огляд найновішого Web-фреймворку	12
1.2.1 Фреймворк Yii	13
1.2.2 Фреймворк Kohana	15
1.2.3 Фреймворк Symfony	15
1.2.4 Вибір фреймворка	16
1.3 Огляд новітньої архітектури СУБД.....	17
1.3.1 СУБД MySQL	19
1.3.2 Підсистема зберігання даних СУБД MySQL	23
1.4 Висновки по розділу	25
2 Створення web-платформи	26
2.1 Компоненти web-платформи	26
2.2 Аутентифікація та авторизація	28
2.3 Створення інтерфейсу.....	33
2.3.1 Головна сторінка	33
2.3.2 Сторінка реєстрації.....	37
2.3.3 Сторінка входу.....	38
2.3.4 Сторінка зворотного зв'язку.....	40
2.3.5 Головна сторінка користувача	41
2.3.6 Сторінка групового редагування	43
2.3.7 Сторінка для додавання зображень	44

2.3.8 Сторінка перегляду зображень за категоріями	46
2.3.9 Сторінки для перегляду зображень і коментування до них	47
2.3.10 Сторінка оновлення даних користувача	50
2.3.11 Додаткові можливості ролі адміністратора	51
2.4 Висновки по розділу	54
3 Впровадження бази даних web-платформи	54
3.1 Проектування сутностей	54
3.2 Визначення відносин між сутностями	64
3.3 Висновки по розділу	68
4 Дослідження розробленої системи	68
4.1 Серверне обладнання web-платформи	68
4.1.1 Вибір конфігурації сервера	68
4.1.2 Вибір джерела безперебійного живлення	69
4.1.3 Вибір системи хостингу	70
4.2 Конфігурування web-сервера Apache	70
4.3 Впровадження бази даних	73
4.4 Система захисту даних	75
4.5 Дослідження web-платформи	77
4.6 Висновки по розділу	79
Висновки	81
Перелік джерел посилання	82

ВСТУП

Метою магістерської роботи є дослідження комп'ютерної системи для web-платформи фотографів з використанням сучасних технічних і програмних засобів. Для досягнення цієї мети необхідно вирішити наступні завдання:

- проаналізувати новітні програмні засоби для web-розробки;
- створити web-платформу системи;
- впровадити базу даних web-платформи;
- встановити та налаштувати програмне забезпечення;
- забезпечити безпеку і надійність системи;
- провести дослідження розробленої системи.

Основна частина дослідження складається з введення, 4 розділів, висновків і списку джерел.

У першому розділі проводиться аналіз сучасних засобів побудови web-платформи, включаючи CMS та фреймворки, що обґрунтовує вибір фреймворку Yii для системи.

У другому розділі створення web-платформи за допомогою Yii та php. Функції php з описом для створення основних розділів сайту, таких як дизайн та інтерфейс користувача.

У третьому розділі визначається концепція бази даних, формується її структурна схема, та представлені скрипти наповнення бази даних контентом.

У четвертому розділі визначається апаратна конфігурація web-платформи, здійснюється встановлення та налаштування програмних компонентів системи, впроваджуються заходи забезпечення безпеки системи, та проводиться аналіз web-платформи.

Наукова новизна роботи полягає в отриманні кількісних оцінок продуктивності підсистем зберігання даних СУБД, які можуть реалізовувати web-платформи.

Практична цінність даної роботи полягає в підвищенні ефективності взаємодії фотографа за рахунок застосування розробленої системи.

1 АНАЛІЗ СУЧАСНИХ ЗАСОБІВ ПОБУДОВИ WEB-ПЛАТФОРМ

1.1 CMS

Система управління контентом - це програмний пакет, який дозволяє автоматизувати процес управління як всім сайтом, так і об'єктами всередині сайту: шаблонами виведення даних, структурою, вмістом, користувачами, правами доступу та додатковими послугами, якщо це можливо. Наприклад, списки розсилки, статистика, пошукові запити та способи взаємодії з користувачами.

CMS розділяє сайт на два елементи: дизайн (загальний вигляд сайту, окремі сторінки та конкретні блоки інформації) і контент (зміст). Дизайн сайту включається в шаблон і змінюється частіше, ніж зміст.

Системи управління контентом вирішують два основні завдання. З точки зору користувача, це інструменти, що дають змогу публікувати новини, розміщувати нові сторінки на сайті та виконувати інші операції з контентом за допомогою зручного інтерфейсу. При цьому користувачі можуть не знати технології, на основі якої створюється web-платформа, але їм необхідно розуміти, як працює web-платформа. З погляду розробника, це інструмент, який прискорює створення складних сайтів, будує рішення з готових блоків і дає змогу в певних межах змінювати логіку роботи та дизайн. Налаштування, обслуговування, очищення, генерація статистики, створення нового контенту все це виконується на серверній частині авторизованим персоналом. Серверний доступ можна отримати за URL, відмінний від URL-адреси, доступної самій web-платформі.

Одними з найпопулярніших CMS, що дозволяють створювати функціональні і легко керовані сайти без серйозних економічних витрат, є Drupal, Joomla і WordPress.

1.1.1 CMS Wordpress

Дуже проста установка, зручне управління і "заточка" пошукової системи- ось ті фактори, які зробили Wordpress популярним движком. Зазвичай ця платформа використовується для блогів, але також успішно використовується у

web-проектах та для інших цілей. Wordpress сам по собі не дуже функціональний, тому для розширення функціональності використовуються плагіни. Мова програмування PHP і база даних MySQL використовуються для створення і реалізації плагінів для Wordpress. Основні можливості:

- підтримка web-стандартів (xhtml, css), технологій rss, atom, trackback і pingback;
- багаторівневе схвалення змін (модерація опублікованих матеріалів);
- планувальник стану матеріалів (розклад вмісту);
- настроювані форми зворотного зв'язку;
- диференціація доступу (розподіл прав);
- підтримка захищених протоколів при роботі з системами (ssl);
- підтримка плагінів і тим, що дозволяють легко змінювати як зовнішній вигляд, так і спосіб виведення даних;
- пошук по сайту.

1.1.2 CMS Joomla

Joomla-одна з найпопулярніших портальних CMS у світі. Популярність Joomla заснована, перш за все, на величезній кількості шаблонів і розширень, які дозволяють реалізувати на цій CMS практично все, що завгодно – при цьому, не вдаючись до допомоги PHP-програмістів.

Joomla-це набір сценаріїв, написаних мовою програмування PHP за допомогою MySQL. У web-додатку є 2 частини розробки та виконання коду: серверна та клієнтська. Клієнтська сторона включає HTML, CSS і Javascript, в той час як серверна сторона включає ASP, JAVA, PHP і.т.д. Щоб використовувати серверну частину, Joomla (як і будь – яка інша CMS) необхідно встановити локальний сервер-DENWER, по суті, має свій власний хостинг для клієнта. Робота з цією CMS трохи складніше, ніж з Wordpress, але можливості системи ширше, ніж у Joomla. Його можна рекомендувати для великих навчальних закладів.

1.1.3 CMS Drupal

Drupal - це система управління контентом (CMS), написана на мові PHP, яка використовує як сховище даних реляційну базу даних (підтримуються MySQL,

PostgreSQL та ін.). Drupal є вільним програмним забезпеченням під ліцензією GPL і розробляється по всьому світу, розробляється ентузіастами. Це високомодульна платформа з відкритим вихідним кодом, призначена для спільного управління вмістом сайту багатьма користувачами. Вона легко розширювана, відповідає більшості стандартів, а код простий, зрозумілий і невеликий за обсягом. Система Drupal встановлюється у версії, що забезпечує всю базову функціональність. Функціональність спливаючих підказок реалізується шляхом встановлення додаткових модулів. Система Drupal спроектована таким чином, щоб її можна було налаштовувати відповідно до потреб користувача, але без втручання в ядро системи, і конфігурувати без додавання модулів або зміни коду ядра. Це дозволяє ефективно відокремити механізм управління контентом від подання останнього користувачеві.

Технологія Drupal може використовуватися для інтернет-платформи, особистого, відомчого або корпоративного web-платформи, інтернет-магазину, каталогу мережеских ресурсів, онлайн-газети, інтернет-дистанційного навчання. Спеціальна група безпеки намагається захистити drupal від усіх нових загроз, своєчасно оновлюючи систему безпеки. Постійно розширюється мережеве співтовариство користувачів, адміністраторів сайтів, дизайнерів і web-розробників докладає всіх зусиль для поліпшення програмного забезпечення Drupal [2].

Одним з основних недоліків Drupal є погане використання властивостей об'єктів PHP:API Drupal насправді не використовує функції ООП, доступні в PHP. Розробники стверджують, що це погана реалізація ООП мовою. Об'єктна модель Drupal доступна, але це дещо незвична форма для PHP. Оскільки активне використання бази даних Drupal, може бути непосильною для ресурсів, пропонованих дешевим тарифним планом хостинг-провайдера.

1.1.4 Вибір CMS

Для цього система управління сайтом повинна бути універсальною, що, як правило, пов'язано не тільки з надмірністю коду, більш складною архітектурою, складністю вирішення конкретних завдань, але і з жорсткими вимогами до дизайну сайту, які не дозволяють творчо підходити до процесу створення дизайну.

Кожна розглянута CMS-система відрізняється унікальним дизайном і гнучкістю (Drupal), зручністю роботи (Joomla), легким і зручним інтерфейсом (Wordpress). Wordpress спочатку являє собою блог CMS, на ньому дуже складно створювати сайти іншого типу. Joomla простіше в освоєнні і управлінні, ніж Drupal, але в той же час Drupal володіє більшою гнучкістю [1].

Вивчивши всі недоліки і переваги цих CMS, можна зробити висновок, що Drupal є найбільш перспективною системою управління контентом, але в той же час до її істотних недоліків відносяться: система перевантажена складними настройками, складними системними ядрами, жорсткими вимогами до шаблонів проектування та несумісністю різних версії системи.

У результаті проведеного аналізу було ухвалено рішення відмовитися від використання наявної CMS для розроблення цієї системи. Це зробило б її найбільш підходящою і найзручнішою для виконання поставленого завдання, а також полегшило б розробку з погляду функціональних розширень. CMS багатомодульна і має величезну кількість конфігурацій, не пов'язаних безпосередньо зі змістом сайту, що ускладнює роботу розробника і ускладнює складність системи.

Система буде включати в себе гнучку систему шаблонів, простий інтерфейс, зручний для читання шлях до сторінки, інструменти пошукової оптимізації, розмежування прав доступу, багатомовність. У системі буде строго відокремлює користувачів від редакторів і відвідувачів сайту, але це пов'язано з підвищеними вимогами до безпеки.

1.2 Огляд найновішого Web-фреймворку

Фреймворк - це структура програмного комплексу в інформаційній системі та програмне забезпечення, яке полегшує розробку та інтеграцію різних компонентів великого програмного проекту. На відміну від бібліотеки, яка об'єднує набір схожих функціональних процедур, фреймворк містить набір бібліотек

різного призначення. Фреймворковий підхід можна виразити як підхід до побудови програм, при якому структура програми будується з двох частин. Перша фіксована частина - це каркас, який не змінюється від конфігурації до конфігурації, а друга - змінна частина (або точка розширення).

Фреймворк PHP останнім часом набирає популярності, і він надає базову платформу для розробки web-додатків, іншими словами, базову структуру додатків. Фреймворк PHP дозволяє заощадити багато часу, знизити навантаження на процес розробки, позбутися від проблем з повторюваним кодом, ефективно використовувати ресурси сервера і швидко створювати додатки.

Жодного великого нестандартного проекту (наприклад twitter.com). Не робіть цього за допомогою готової CMS. У той же час створювати web-додатки, підтримувати і модернізувати їх без використання фреймворку PHP буде набагато складніше. Всі оригінальні проекти розробляються на фреймворку.

На даний момент існує безліч web-фреймворків, але розглянемо, який з них найбільш підходить для вирішення поставленого завдання.

1.2.1 Фреймворк Yii

Yii - це високоефективний компонентний PHP-фреймворк для швидкого розроблення великомасштабних web-додатків. Він повною мірою використовує концепцію повторного використання коду і дає змогу значно прискорити процес web-розробки. Yii - є універсальною платформою web-програмування і може бути використаний для розробки практично будь-яких web-додатків. Завдяки своїй легкості та наявності розвинених засобів кешування Yii можна використовувати для застосунків із високою відвідуваністю, таких як портали, форуми, системи керування контентом (CMS), системи електронної комерції тощо.

Yii використовує шаблон дизайну Model-View-Controller (MVC, Model-View-Controller), який широко використовується у web-програмуванні.

MVC призначена для відокремлення бізнес-логіки від призначеного для користувача інтерфейсу, що дає змогу розробникам легко змінювати одні частини додатка, не зачіпаючи інші. В архітектурі MVC модель надає дані та правила

бізнес-логіки. подання відповідає за користувацький інтерфейс (текст, поля введення тощо), а контролер забезпечує взаємодію між моделлю та поданням.

Крім того, Yii використовує фронт-контролер, який називається `application`, який інкапсулює контекст обробки запитів. Додаток збирає інформацію про запит і передає її відповідному контролеру для подальшої обробки. Діаграма (рис. 1.1) показує структуру програми Yii [3].

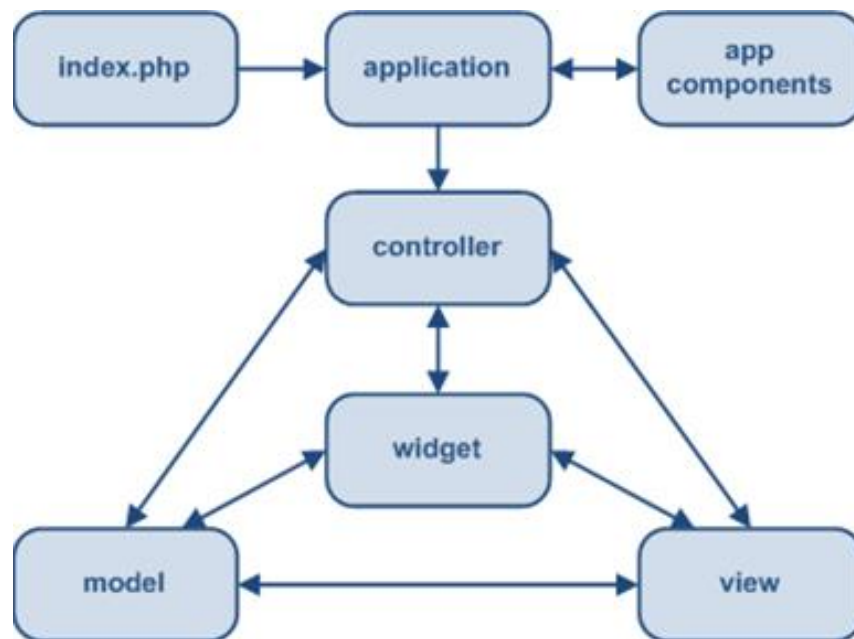


Рисунок 1.1 – Структура програми Yii

Yii надає розробникам об'єкти доступу до даних (DAO), потужний інструмент для роботи з базами даних. Yii DAO-це доповнення до розширення PHP Data Objects (PDO), яке дозволяє маніпулювати різними СУБД через єдиний інтерфейс. Програми, розроблені за допомогою DAO, можуть легко переходити з однієї СУБД на іншу без необхідності редагувати програмний код, відповідальний за доступ до даних.

Конструктор запитів Yii забезпечує об'єктно-орієнтований спосіб побудови запитів SQL та зменшує ризик ін'єкції SQL.

Yii ActiveRecord (AR) реалізує перероблений підхід об'єктно-реляційної проекції (ORM) для подальшого спрощення операцій з базою даних.

Представляючи таблицю бази даних як клас і рядок таблиці як екземпляр класу, Yii AR позбавляє необхідності написання SQL-виразів, пов'язану з операціями CRUD (створення, читання, оновлення, видалення).

У той час як функції бази даних, вбудовані в Yii, підходять практично для всіх завдань, пов'язаних з роботою з базою даних, використання інші бібліотеки для роботи з базою даних. Yii спочатку був розроблений для того, щоб дозволити розробникам працювати зі сторонніми бібліотеками.

1.2.2 Фреймворк Kohana

Kohana - це HMVC PHP5 web-фреймворк, який надає інструменти для розробки web-додатків. Завдяки своїй каскадній файловій системі, численним вбудованим утилітам та послідовним API, він ідеально підходить для більшості завдань, а його висока швидкість виконання та широкі можливості профілювання дозволяють прискорити розробку та використовувати менш потужні ресурси для вмісту кінцевого продукту.

Особливості цього фреймворку:

- висока швидкість виконання;
- використання всіх можливостей php5;
- ряд вбудованих інструментів, включаючи ORM та Database;
- невеликі системні вимоги;
- інструменти профілювання та налагодження;
- підтримка UTF-8 і I18N.

Kohana поширюється за ліцензією BSD і дає комерційним проектам право використовувати її на законних підставах [4].

1.2.3 Фреймворк Symfony

Symfony-це безкоштовний фреймворк, написаний на PHP5 і використовує шаблон Model-View-Controller (MVC). Symfony забезпечує швидку розробку web-додатків і управління ними, полегшуючи вирішення повсякденних завдань web-програмістів. Це працює лише з PHP5.

Особливість:

- незалежність бази даних (підтримуються MySQL, PostgreSQL, MS-SQL, SQLite, Oracle);
- використовується ORM (Propel + Creole);
- подання складаються з шаблонів і макетів, які можуть бути створені дизайнерами без знання symfony;
- підтримка автоматичної перевірки форм;
- автоматичне очищення вводу користувача (щоб уникнути ін'єкцій SQL та атак XSS);
- робота з Ajax, інструментами для посторінкового виведення списків, їх фільтрації та сортування;
- автоматизація створення керуючої частини.

Одним із способів прискорити роботу web-програми є використання кешу, який можна використовувати як на стороні сервера, так і на стороні клієнта.

Symfony безкоштовна і публікується за ліцензією MIT, спонсорованою французькою компанією Sensio [5].

1.2.4 Вибір фреймворка

Основними критеріями вибору PHP-фреймворку були зручність і простота вивчення, відповідність вимогам створюваної системи, безпека. Можна зробити висновок, що Yii є прийнятним варіантом. Фреймворк має безліч вбудованих функцій, які відрізняють його від інших фреймворків і роблять розробку з його допомогою дуже швидкою і захоплюючою. Окремо варто відзначити рівень повної підтримки ООП, генератора коду і захисту (всі стандартні класи Yii володіють високим рівнем безпеки, який може повністю захистити ваш сайт від Sql-Inj, XSS, CSRF та інших атак).

Переваги Yii перед іншими фреймворками полягають в ефективності, широкій функціональності і високоякісній документації. Спочатку Yii був розроблений дуже ретельно, щоб відповідати всім вимогам при розробці серйозних web-додатків. Yii не є побічним продуктом проекту або набором сторонніх рішень. Це результат великого досвіду автора в розробці web-додатків і вивчення найбільш популярних web-фреймворків і додатків.

1.3 Огляд новітньої архітектури СУБД

Взаємодія між web-платформою та базою даних здійснюється за допомогою СУБД, яка розшифровує запит та виконує операцію, використовуючи інформацію з бази даних [6].

Існують такі типи баз даних:

- ієрархічні;
- реляційні;
- об'єктно-орієнтовані;
- гібридні.

Ієрархічні бази даних засновані на деревоподібній структурі зберігання інформації. У цьому сенсі ієрархічна база даних дуже схожа на файлову систему комп'ютера.

У реляційній базі даних дані збираються в таблиці, які складаються зі стовпців і рядків і знаходяться в точці перетину, де розташовані комірки. Запит до такої бази даних повертає таблицю, яка може бути повторно використана при наступному запиті. Дані в деяких таблицях, як правило, пов'язані з даними в інших таблицях, тому вони і називаються «реляційними».

В об'єктно-орієнтованій базі даних дані зберігаються у вигляді об'єктів. Корисно використовувати об'єктно-орієнтоване програмування для роботи з об'єктно-орієнтованими базами даних. Але на сьогоднішній день такі бази даних ще не досягли популярності реляційних баз даних, оскільки їх продуктивність все ще значно поступається.

Гібридна СУБД поєднує в собі можливості реляційних та об'єктно-орієнтованих баз даних.

У web-додатках зазвичай використовується реляційна база даних. Реляційна модель бази даних представляє дані у вигляді таблиць, розділених на рядки та стовпці, причому дані розміщуються на перетині рядків і стовпців.

Простіше кажучи, функції реляційної бази даних можна описати наступним чином:

- дані зберігаються в таблиці, що складається зі стовпців і рядків;
- на перетині кожного стовпця і рядка знаходиться рівно одне значення;
- кожен стовпець має своє ім'я, воно діє як ім'я, і всі значення в одному стовпці мають однаковий тип;
- на відміну від рядків, які розміщуються в будь-якому порядку, стовпці розміщуються в певному порядку, який визначається при створенні таблиці. у таблиці не може бути більше одного рядка, але у вас повинен бути принаймні один стовпець;
- запити до бази даних повертають результати у вигляді таблиць, які також діють як об'єкти запиту.

Можливість доступу до бази даних через Інтернет істотно змінила стратегію доступу до даних у зв'язку з тим, що:

- кількість користувачів не тільки значно зміниться, але й може змінитися без попередження;
- фізично неможливо ідентифікувати користувача, і він може підключитися до іншої програми, не закриваючи програму, але це не гарантує належного відключення від програми;
- реалізує доступ до бази даних з боку клієнта посередник - це спеціальний web-сервер, що забезпечує функціональність призначених для користувача додатків, що виконують прямі запити до даних і повертають відповіді у вигляді HTML-документів.

Доступ бази даних на боці сервера здійснюється стандартизованими засобами. Підтримка діалогових форм на рівні гіпертекстового документа. Кожен документ пов'язаний з URL-адресою, доступ до якої здійснюється за допомогою об'єктно-орієнтованого протоколу HTTP. Можливість запуску серверної програми, що взаємодіє через стандартний CGI-інтерфейс або через інтерфейс web-сервера.

Доступ до бази даних здійснюється з боку клієнта через HTTP-запит. Рядок запиту складається з трьох частин тексту, розділених пробілами. Перша частина

визначає методи (команди), які застосовуються до ресурсів сервера. Найпоширенішим методом є GET, який вимагає від сервера надсилання копії ресурсу клієнту. Друга частина визначає ім'я цільового ресурсу, тобто URL-адресу, що складається з протоколу та доменного імені сервера. Третя частина визначає версію протоколу, що використовується клієнтом (наприклад, HTTP/1.0).

Запит клієнта оформляється в заголовку, відправляється на web-сервер, який обробляє запит, а потім відправляється в СУБД. СУБД витягує запитувані дані з бази даних (або вносить зміни до бази даних) і повертає їх серверу, який генерує та надсилає відповідь клієнта (рис. 1.2) [7].

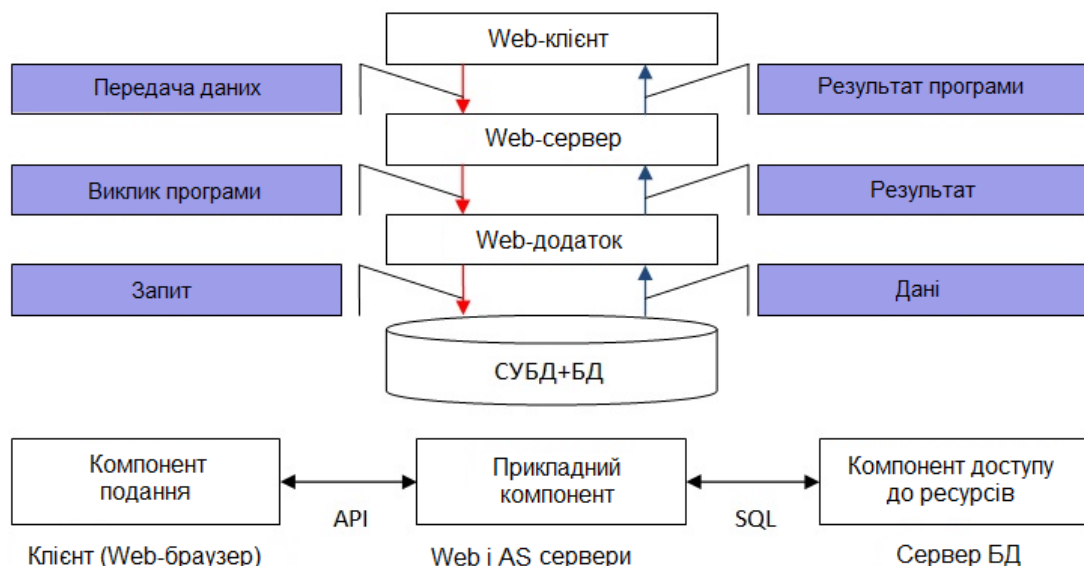


Рисунок 1.2 – Схема взаємодії між клієнтом, web-сервером і СУБД

Таким чином, найважливішим компонентом web-платформи є СУБД. Найбільш поширеною СУБД є MySQL.

1.3.1 СУБД MySQL

Архітектура MySQL дуже відрізняється від інших серверів баз даних і є ефективною для широкого спектру завдань; MySQL не є панацеєю, але вона достатньо гнучка, щоб бездоганно працювати навіть у дуже вимогливих середовищах, таких як web-додатки. MySQL також можна використовувати у

вбудованих додатках, сховищах даних, програмному забезпеченні для індексації та розповсюдження контенту, надійних системах резервного копіювання та системах обробки транзакцій в режимі онлайн (OLTP).

Логічна архітектура MySQL показана на рисунку 1.3 [8]. Верхній рівень містить служби, які не є специфічними для MySQL. Ці служби необхідні для більшості мережеских клієнт-серверних інструментів і серверів. Ці служби забезпечують підтримку з'єднань, ідентифікацію, безпеку тощо.

Другий рівень містить основні частини MySQL Intelligence. До них відносяться розбір запитів, оптимізація, кешування і всі вбудовані функції (операції з датою і часом, математичні функції, шифрування і т.д.). Цей рівень реалізує незалежні від підсистеми зберігання функції, такі як збережені процедури, тригери і представлення.

Третій рівень містить підсистему зберігання даних, яка відповідає за зберігання і пошук усіх даних, що зберігаються в MySQL. Як і різні файлові системи GNU/Linux, кожна підсистема зберігання даних має свої переваги та недоліки. Взаємодія із сервером здійснюється за допомогою API підсистеми зберігання. Цей інтерфейс приховує відмінності між підсистемами зберігання і практично прозорий на рівні запитів, цей інтерфейс містить десятки низькорівневих функцій, що виконують такі операції, як "почати транзакцію" або "отримати рядок із цим первинним ключем". Підсистема зберігання не розбирає, не взаємодіє і просто відповідає на запити сервера.

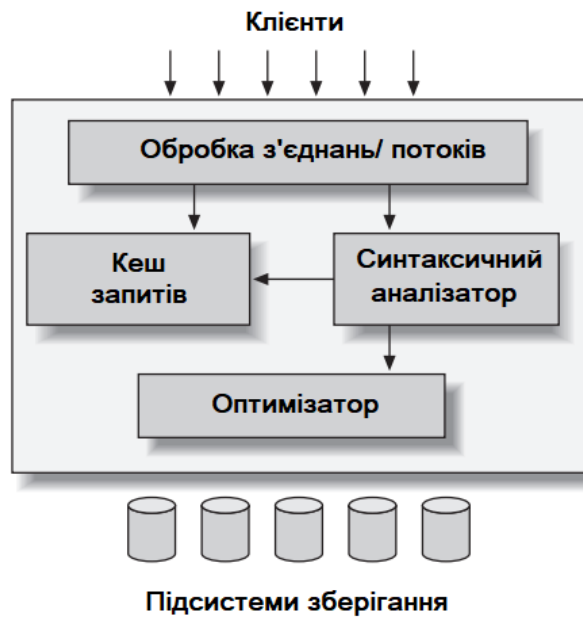


Рисунок 1.3-Логічна архітектура MySQL

Для кожного клієнтського підключення в серверному процесі призначається окремий потік. Запити на це з'єднання виконуються в рамках цього потоку, який виконується одним ядром або процесором. Сервер кешує потік, так що йому не потрібно створювати або знищувати потік для кожного нового підключення. Коли клієнт (додаток) підключається до сервера MySQL, сервер повинен його ідентифікувати. Ідентифікація базується на імені користувача, адресі хоста, на якому було встановлено з'єднання, та паролі. Крім того, при підключенні за допомогою протоколу Secure Sockets Layer (SSL) також можна використовувати сертифікат [9]. Після підключення клієнта сервер перевіряє необхідні дозволи для кожного запиту [9].

У будь-який момент можна зіткнутися з ситуаціями, коли кільком запитам потрібно змінити дані одночасно, що може призвести до завдань з управління конфліктуючим доступом. MySQL повинен вирішити цю проблему на двох рівнях: сервер і підсистема зберігання даних.

Одним із способів поліпшити управління блокуваннями спільних ресурсів є підвищення вибіркової блокувань. Замість того щоб блокувати весь ресурс

цілком, можна заблокувати тільки ті частини, які необхідно змінити. Рекомендується блокувати лише змінні дані. Мінімізуючи кількість ресурсів, які блокуються в даний момент часу, дозволяє виконувати паралельні операції на одному об'єкті, якщо ці операції не конфліктують між собою.

Ключова проблема полягає в тому, що блокування споживають ресурси. Всі операції, з яких вони складаються – отримання блокування, перевірка, чи вільна блокування, зняття блокування і т.д. Самі по собі викликають накладні витрати. Якщо система витрачає занадто багато часу на керування блокуванням, замість того, щоб витрачати час на збереження та вилучення даних, продуктивність знизиться. Стратегія блокування-це компроміс між накладними витратами на реалізацію блокування та безпекою даних, і цей компроміс має помітний вплив на продуктивність. Блокування таблиць на рівні рядків часто поєднуються з багатьма складними способами забезпечення чудової продуктивності через велику кількість блокувань. У цій частині СУБД MySQL пропонує таку можливість. Підсистеми зберігання даних можуть реалізовувати власні політики блокування та рівні деталізації блокування. Управління блокуваннями є дуже важливим рішенням при проектуванні підсистеми зберігання даних. Фіксація деталізації на певному рівні може підвищити продуктивність в деяких випадках, але ця підсистема не дуже підходить для інших цілей. MySQL пропонує кілька підсистем зберігання, тому вам не потрібно створювати єдине рішення на всі випадки життя.

СУБД MySQL працює з транзакціями, які надають такі властивості:

- атомарність. Транзакція повинна діяти як єдина, неподільна одиниця роботи, щоб вся транзакція була виконана або скасована. Якщо транзакція є атомарною, то немає нічого кращого, ніж частково виконана транзакція: все або нічого;
- несуперечність. База даних завжди повинна переходити з одного несуперечливого стану в інший. Якщо транзакція не здійснена, зміни в цій транзакції не відображаються в базі даних;
- ізолюваність. Результати транзакції зазвичай не видно до тих пір, поки вони не будуть завершені;

– довговічність. Після фіксації зміни, внесені під час транзакції, є постійними. Це означає, що зміни повинні бути записані таким чином, щоб дані не були втрачені в разі збою системи. Деякі стратегії довговічності дають більш надійну гарантію безпеки, ніж інші, і жодна з них не є абсолютно надійною.

1.3.2 Підсистема зберігання даних СУБД MySQL

Найбільш важливим елементом СУБД є підсистема зберігання даних, вибір якої залежить як від загальної продуктивності системи, так і від її надійності. Найпоширенішими підсистемами зберігання даних є InnoDB та MyISAM.

Механізм MyISAM-це вдосконалена версія оригінального механізму зберігання ISAM, який обробляв таблиці, не пов'язані з транзакціями. Кожна таблиця MyISAM зберігається на диску у вигляді трьох файлів, їх назва починається з імені таблиці, а розширення вказує на тип файлу. У файлі «.frm» зберігається опис таблиці, а дані містяться у файлі з розширенням «.MYD» (MYData), а розширення файлу індексу «.MYI» (MyIndex). Таблиця MyISAM не залежить від платформи. Табличні файли можуть бути перенесені на будь-який комп'ютер, незалежно від архітектури і встановленої операційної системи. Як правило, тип MyISAM обмежений граничним розміром файлу файлової системи операційної системи.

InnoDB-це транзакційний (ACID-сумісний) механізм зберігання даних MySQL, який забезпечує фіксацію транзакцій, відкат і відновлення даних після збою. Таблиці InnoDB можуть використовувати блокування на рівні рядків, а оператори SELECT використовують послідовний метод читання без блокування (як в Oracle). Ці функції покращують паралельність та підвищують продуктивність у багатокористувацькому режимі. У InnoDB блокування цих таблиць на рівні рядків займає мало місця, тому розширення блокування не потрібно. Таблиці InnoDB підтримують обмеження зовнішнього ключа. Запити SQL дають вам свободу поєднувати таблиці InnoDB з таблицями різних типів MySQL навіть в одному запиті.

Механізм зберігання даних InnoDB був розроблений для забезпечення максимальної продуктивності при обробці великих обсягів даних. З точки зору

продуктивності процесора, це не можна порівняти з іншими моделями реляційних баз даних зі сховищем на диску.

Повністю інтегрований у сервер Mysql, движок InnoDB підтримує власний пул буферів для кешування даних та індексів в основній пам'яті. InnoDB зберігає таблиці та індекси в просторі пам'яті, який може складатися з декількох файлів (або розділів диска низького рівня). Це різниця між механізмами InnoDB та MyISAM, де кожна таблиця зберігається в окремому файлі.

Якщо ви порівнюєте нетранзакційні таблиці (MyISAM) з транзакційними таблицями (InnoDB), ви зможете виявити деякі переваги останніх:

- більш безпечний. У разі збою MySQL або серйозної апаратної неполадки ви все одно можете відновити дані з резервних копій і журналів транзакцій або за допомогою автоматичного відновлення;
- можна приєднатися до кількох операторів і прийняти їх за допомогою одного оператора COMMIT (якщо режим автоматичного фіксації вимкнено);
- можна скасувати внесені зміни, запустивши оператор ROLLBACK (якщо режим автоматичного фіксації вимкнено);
- якщо під час оновлення виникає помилка, ви можете відновити всі внесені зміни (не можна скасувати зміни, внесені для таблиць без транзакцій);
- механізми зберігання транзакцій можуть краще обробляти паралельність у таблицях, де дані оновлюються одночасно під час читання.

У той же час таблиця без транзакцій має перевагу, яке якраз і пояснюється відсутністю транзакцій:

- більш висока робоча швидкість;
- нижчі вимоги до дискового простору;
- для виконання оновлень потрібно менше пам'яті.

В одній і тій же інструкції можна поєднувати таблиці з транзакціями та таблиці без транзакцій. Це дозволяє використовувати переваги обох типів таблиць. Однак, якщо для транзакції вимкнено режим автоматичної фіксації, зміни в таблиці без транзакції негайно виправляються без можливості збою.

В експериментальній частині роботи протестуємо обидві підсистеми зберігання даних. Визначаємо, яка з них покаже більшу продуктивність щодо нашої web-платформи.

1.4 Висновки по розділу

В результаті аналізу новітніх технологій побудови web-платформ ми формулюємо наступні висновки:

- жодна з сучасних CMS не дозволяє створювати код для гнучкого сайту фотопорталу, саме тому при розробці цієї системи було прийнято рішення відмовитися від існуючої CMS;
- для написання системи ми використовуємо фреймворк Yii, який є більш ефективним, ніж інші фреймворки, має широкий спектр функцій та високоякісну документацію;
- web-платформа включає реляційну базу даних SQL та базу даних MySQL;
- підсистема зберігання даних СУБД визначається після дослідження InnoDB і MyISAM.

Таким чином, проект web-платформи включає розробку web-платформи на платформі Yii, проектування та впровадження баз даних SQL, вибір конфігурації серверного обладнання та інструментів web-хостингу.

2 СТВОРЕННЯ WEB-ПЛАТФОРМИ

Сайт надає користувачам наступні можливості:

- можливість зареєструватися в системі, а потім увійти в свій обліковий запис;
- перегляд завантажених фотографій за категоріями;
- дивіться найпопулярніші фотографії;
- дивіться випадкові фотографії;
- завантажувати власні зображення та переглядати їх, а також групове редагування;
- коментарі до інших зображень у системі;
- рейтинг зображень.

2.1 Компоненти web-платформи

При створенні web-платформи за допомогою фреймворку Yii виділяються наступні компоненти:

а) компоненти (components):

- 1) PhpAuthManager.php - менеджер аутентифікації;
- 2) UserIdentity.php - клас ідентифікації з необхідною логікою аутентифікації;
- 3) WallManager.php - віджет групового редагування зображень;
- 4) WebUser.php - клас, який розширює логіку користувача CWebUser.

б) контролери (controllers):

- 1) MessageController.php – контролер для маніпулювання коментарями користувачів;

2) PhotoController.php – контролер для роботи із зображенням користувача;

3) SiteController.php – базовий контролер.

в) моделі (models):

1) ContactForm.php – модель для розширення CFormModel, форма зворотного зв'язку;

2) LoginForm.php – модель, що розширює CFormModel, форму введення;

3) RegForm.php – модель, що розширює CFormModel, форму реєстрації користувача;

4) Message.php – модель повідомлення, що розширює CActiveRecord;

5) Photo.php – модель зображення, що розширює CActiveRecord;

6) Region.php – розширює регіональну модель (яка визначає місцезнаходження) CActiveRecord;

7) User.php – системна користувацька модель, що розширює CActiveRecord;

8) UserRole.php – рольова модель системного користувача, що розширює CActiveRecord.

г) модулі:

1) Admin – Специфічні функції для системних адміністраторів:

– Контролер (CategoryController.php – контролер управління категоріями зображень; ConfigController.php – контролер управління конфігурацією системи; DefaultController.php – контролер базового модуля; UserController.php – контролер управління користувачами);

– Модель (Category.php – модель категорії зображень з розширенням CActiveRecord; Config.php – модель конфігурації системи для розширення CActiveRecord);

2) User – функції для користувачів системи:

– Контролер (DefaultController.php – контролер базового модуля).

2.2 Аутентифікація та авторизація

Сторінки, до яких можуть отримати доступ лише деякі користувачі, вимагають автентифікації та авторизації. Аутентифікація - це перевірка того, чи є хтось тим, ким ви себе називаєте. Зазвичай потрібно ввести ім'я користувача та пароль. Перевірка того, чи може автентифікований користувач виконати певну дію (зазвичай її називають ресурсом). У більшості випадків це визначається шляхом перевірки того, чи призначена користувачеві певна роль, яка має доступ до ресурсу.

У Yii була створена відповідна структура аутентифікації та авторизації (auth). Центральним компонентом структури аутентифікації є заздалегідь визначений компонент програми «user» - об'єкт, що реалізує інтерфейс `IWebUser`, який містить певну інформацію про поточного користувача. Ви можете використовувати `Yii::app()->user` для доступу до нього з будь-якої точки вашої програми.

Цей компонент дозволяє використовувати `CWebUser::isGuest` для перевірки автентичності користувача. Ви можете увійти або вийти. Корисно використовувати `CWebUser::checkAccess` для перевірки прав на конкретну дію. Ви також можете імпортувати унікальні ідентифікатори та інші постійні дані користувача.

Щоб додати підтримку різних методів аутентифікації, Yii має відповідний клас ідентифікаторів. `Identity` - клас реалізований і містить необхідну логіку перевірки автентичності. Такий клас повинен реалізувати інтерфейс `IUserIdentity`. Коли ви створюєте програму, вам потрібно розширити клас `CUserIdentity`, який є базовим класом, який реалізує перевірку імені користувача та пароля. Основне завдання при створенні класу ідентифікаторів-реалізувати метод `IUserIdentity::authenticate`. Цей метод використовується для визначення базового алгоритму аутентифікації. Цей клас також може містити додаткову інформацію про користувача, необхідну в процесі роботи з сеансом. Вміст файлу `UserIdentity.php` вказаний у (лістингу 2.1.)

Лістинг 2.1 – Зміст файлу UserIdentity.php

```

class UserIdentity extends CUserIdentity
{
    protected $_id;
    // цей метод викликається один раз під час автентифікації
    користувача.
    public function authenticate() {
        $user = User::model()-
        >find('LOWER(login)=?', array(strtolower($this->username)));
        if(($user===null) or (md5($this->password)!==$user->password))
        {
            $this->errorCode = self::ERROR_USERNAME_INVALID;
        }
    else
    {
        // Використовується id замість імені користувача як ідентифікатор.
        // Обов'язково потрібно перевизначити метод getId
        $this->_id = $user->id;
        $this->username = $user->name;
        $this->errorCode = self::ERROR_NONE;
        $user->lastEnterDate = date("y-m-d H:m:s");
        $user->ip = Yii::app()->request->userHostAddress;
        $user->save();
    }

    return !$this->errorCode;
}

    public function getId()
    {
        return $this->_id;
    }
}

```

Вся інформація, що зберігається в стані (при виклику `CBaseUserIdentity::setState`), передається `CWebUser`, який зберігається в постійному сховищі, такому як сеанс. Ви можете отримати доступ до цієї інформації як властивість `CWebUser`. Отже, успадковуючи `CWebUser`, ви можете зберігати дані користувача, для кожного користувача (лістинг 2.2).

Лістинг 2.2 – Файл WebUser.php

```

class WebUser extends CWebUser
{
    private $_model = null;
    function getRole()
    {
        if($user = $this->getModel())
        {
            return $user->role->roleCode;
        }
    }
}

```

```

}}
    private function getModel()
    {
    if (!$this->isGuest && $this->_model === null)
    {
        $this->_model = User::model()->find('id=:id',
array(':id'=>$this->id));
    }
    return $this->_model;
}}

```

Для безпечного зберігання паролів в базі даних потрібні додаткові запобіжні заходи. Зловмисник, якому належить таблиця користувачів (або її резервна копія), може відновити пароль. Зокрема, потрібно використовувати постійні хеш-функції, пошук яких може зайняти багато часу для зловмисника. Система, що розробляється, використовує алгоритм хешування md5.

Можна реалізувати введення-виведення, використовуючи створений клас `UserIdentity` для створення відповідних методів на базовому контролері (`Site`). Частина, відповідальна за введення і виведення даних контролера, показана в лістингу 2.3.

Лістинг 2.3 - частина контролера, відповідальна за введення і виведення даних.

```

public function actionLogin()
{
    $model=new LoginForm;
    if(isset($_POST['ajax']) && $_POST['ajax']=='login-form')
    {
        echo CActiveForm::validate($model);
        Yii::app()->end();
    }
    if(isset($_POST['LoginForm']))
    {
        $model->attributes=$_POST['LoginForm'];
        if($model->validate() && $model->login())
        {
            if(Yii::app()->user->checkAccess('admin'))
            {
                $this->redirect(array('admin/default/index'));
            }
            else if(Yii::app()->user->checkAccess('userL1'))
            {
                $this->redirect(array('user/default/index'));
            }
        }
    }
}

```

```

}}
}
$this->render('login',array('model'=>$model));
}
public function actionLogout()
{
    Yii::app()->user->logout();
    $this->redirect(Yii::app()->homeUrl);
}

```

Управління доступом на основі ролей (RBAC) - це простий і потужний спосіб централізованого управління доступом. У Yii ієрархічний RBAC реалізований за допомогою компонента `authManager`. Налаштування компонента `authManager` в глобальному файлі конфігурації:

```

'authManager' => array(
    'class' => 'PhpAuthManager',
    'defaultRoles' => array('guest'), //роль по умовчанию
),

```

Менеджер авторизації вказаний в лістингу 2.4.

Лістинг 2.4 - Менеджера авторизації `PhpAuthManager.php`

```

class PhpAuthManager extends CPhpAuthManager
{
    public function init()
    {
        if($this->authFile===null)
        {
            $this->authFile=Yii::getPathOfAlias('application.config.auth').'.php';
        }
        parent::init();
        if(!Yii::app()->user->isGuest)
        {
            // Пов'язує роль, зазначену в базі даних, з ідентифікатором користувача,
            // повертається UserIdentity.getId().
            $this->assign(Yii::app()->user->role, Yii::app()->user->id);
        }
    }
}

```

Основною концепцією RBAC Yii є елемент авторизації. Елемент авторизації-це дозвіл на виконання дій (створення нових записів в блозі, управління користувачами). Залежно від структури та призначення елементи затвердження можна розділити на операції, завдання та ролі. Ролі складаються із завдань.

Завдання складаються з операцій. Операція-дозвіл на будь-яку дію. Елемент авторизації однозначно ідентифікується по його унікальному імені. Елемент авторизації може бути пов'язаний з бізнес-правилом - РНР-кодом, який використовується при перевірці доступу. Користувач може отримати доступ до елемента, лише якщо код повертає значення true.

Використовуючи елемент авторизації для побудови ієрархії авторизації. Після побудови ієрархії авторизації ми можемо призначити ролі користувачам у своєму додатку звідти. Користувач отримує всі права на ту роль, яка йому призначена. Файл ієрархії ролей наведено в лістингу 2.5.

Лістинг 2.5 - Файл ієрархії ролей auth.php

```
return array(
    'guest' => array(
        'type' => CAuthItem::TYPE_ROLE,
        'description' => 'Гість',
        'bizRule' => null,
        'data' => null
    ),
    'userL1' => array(
        'type' => CAuthItem::TYPE_ROLE,
        'description' => 'Користувач першого рівня',
        'children' => array(
            'guest', // успадкування від гостя
        ),
        'bizRule' => null,
        'data' => null
    ),
    'userL2' => array(
        'type' => CAuthItem::TYPE_ROLE,
        'description' => 'Користувач другого рівня',
        'children' => array(
            'userL1',
        ),
        'bizRule' => null,
        'data' => null
    ),
    'userL3' => array(
        'type' => CAuthItem::TYPE_ROLE,
        'description' => 'Користувач третього рівня',
        'children' => array(
            'userL2',
        ),
        'bizRule' => null,
        'data' => null
    ),
),
```

```

        'admin' => array(
            'type' => CAuthItem::TYPE_ROLE,
            'description' => 'Адміністратор',
            'children' => array(
                'userL3',
            ),
            'bizRule' => null,
            'data' => null
        ),
    );

```

2.3 Створення інтерфейсу

2.3.1 Головна сторінка

Якщо не вказати точний контролер, який буде обробляти базовий запит при доступі до сервера, запит буде оброблений Site-контролером та представлений у вигляді файлу SiteController.php. При запуску контролер виконує відповідну дію. Зазвичай для цього потрібно створити правильну модель і відобразити бажаний вид. У найпростішому випадку дії - цей метод у класі контролера, ім'я якого починається з action.

Контролер має дію за замовчуванням, яка виконується, якщо користувач не вказує дію під час запиту. За замовчуванням ця дія називається index, і коли вона виконується, відображається подання. Дія index контролера SiteController описано в лістингу 2.6.

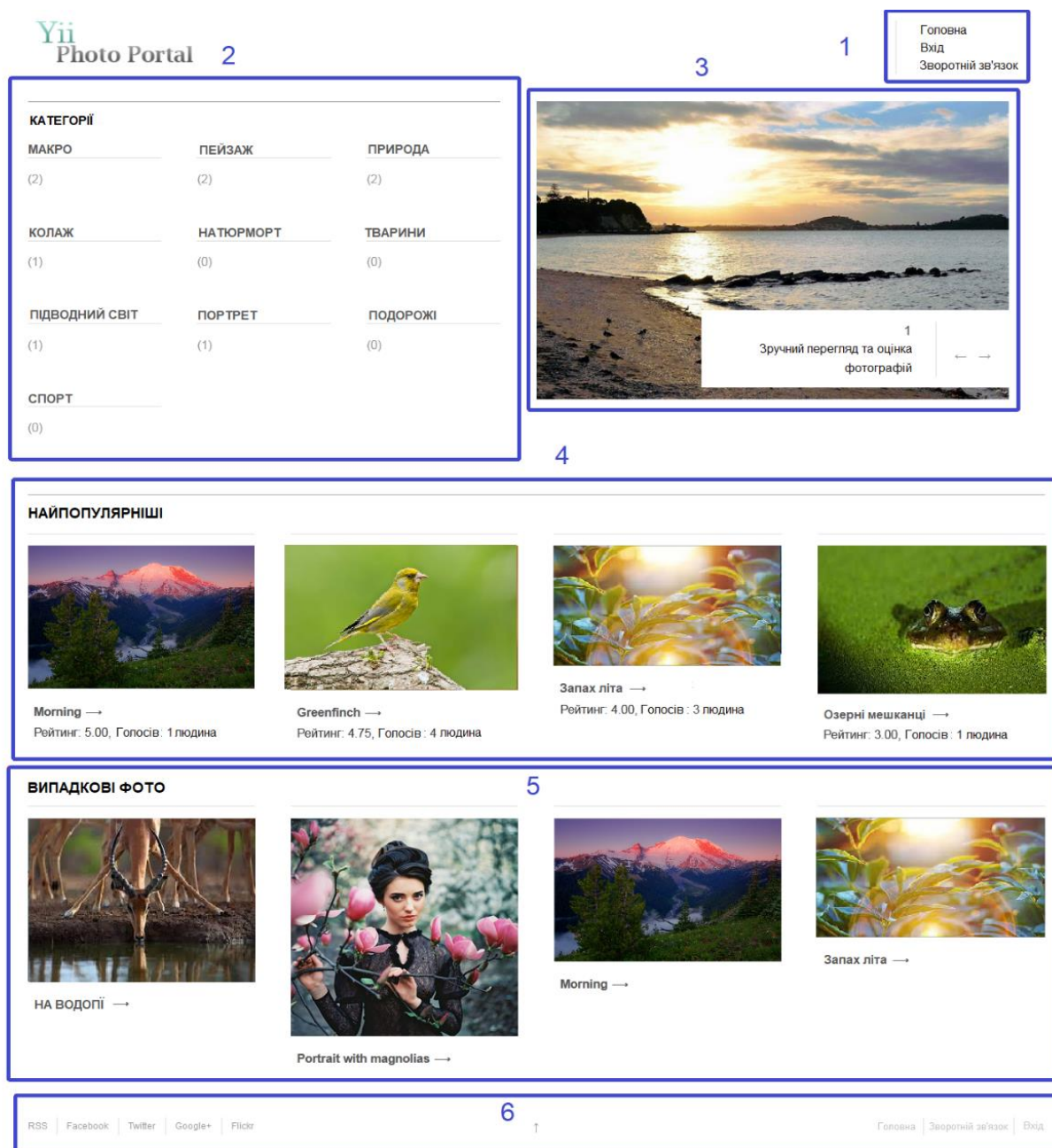
Лістинг 2.6 – Дії index контролера SiteController:

```

public function actionIndex()
{
    $criteria=new CDbCriteria();
    $criteria->alias = "c";
    $criteria->select = 'c.name, COUNT( c.name ) AS count';
    $criteria->group = 'c.name';
    $criteria->order = 'count DESC';
    $this->render('index', array('cathData' =>
    Category::model()->with('photos')->findAll($criteria)));
}

```

Під час перегляду подання виконується надсилання масиву об'єктів категорії зображень типу CActiveRecord, які відображаються на головній сторінці. Після того, як браузер відобразить вигляд головної сторінки, вона буде мати вигляд, як показано на рисунку 2.1.



1 - Головне меню; 2 - Список категорій; 3 - Статичний слайдер; 4 - Блок популярних зображень; 5 - Блок випадкових зображень; 6 - Нижній колонтитул сторінки.

Рисунок 2.1 – Індексна сторінка сайту

Головне меню відображає інформацію відповідно до ролі користувача, виконує такі перевірки, відображає саме меню і виводить на екран `main.php` (лістинг 2.7).

Лістинг 2.7 – Головне відображення `main.php`

```
$this->widget('zii.widgets.CMenu',array(
    'items'=>array(
        array('label'=>'Головна', 'url'=>array('/')),
        //перевірка, чи пройшов користувач автентифікацію
        array('label'=>'Вхід', 'url'=>array('/site/login'),
            'visible'=>Yii::app()->user->isGuest),
        array('label'=>'Зворотній зв'язок', 'url'=>array('/site/contact'),
            'visible'=>Yii::app()->user->isGuest),
        array('label'=>'Мої фото', 'url'=>array('/user/default/index'),
            'visible'=>!Yii::app()->user->isGuest),
        array('label'=>'Користувачі', 'url'=>array('#'),
            'visible'=>Yii::app()->user->checkAccess('admin'), 'items'=>array(
                //перевірка, чи є у користувача роль адміністратора
                array('label'=>'Перелік', 'url'=>array('/admin/user/index'),
                    'visible'=>Yii::app()->user->checkAccess('admin')),
                array('label'=>'Створення користувача',
                    'url'=>array('/admin/user/create'), 'visible'=>Yii::app()->user->checkAccess('admin')),
                array('label'=>'Керування користувачами',
                    'url'=>array('/admin/user/admin'), 'visible'=>Yii::app()->user->checkAccess('admin')),
            )),
        array('label'=>'Вихід ('.Yii::app()->user->name.'),'
            'url'=>array('/site/logout'), 'visible'=>!Yii::app()->user->isGuest)
    ),
));
```

`$cathData` використовується для відображення списку категорій на головній сторінці, а змінна `$content` відображає динамічний вміст з інших подань у головному поданні. Вихідні дані списку категорій представлені у лістингу 2.8.

Лістинг 2.8 - Виведення списку категорій

```
<?php foreach ($cathData as $cath) { ?>
<article class="grid col-one-third mq3-col-full">
<h5><a href="<?php echo $this->
createAbsoluteUrl('/photo/viewByCat', array('id'=>$cath->id));
?>"><?php echo $cath->name; ?></a></h5>
<aside>(<?php echo $cath->photosCount; ?>)</aside>
</article>
<?php } ?>
```

Також на головній сторінці є статичний блок-слайдер, що містить візуальну інформацію про роботу системи. Код показаний в лістингу 2.9.

Лістинг 2.9 - Статичний блок-слайдер що містить інформацією про роботу системи

```
<div class="slider grid col-one-half mq2-col-full">
  <div class="flexslider">
    <div class="slides">
      <div class="slide">
        <figure>
          
          <figcaption>
            <div>
              <h5>1</h5>
              <p>Зручний перегляд та оцінка фотографій </p>
            </div>
          </figcaption>
        </figure>
      </div>
      <!-- ... ще блоки slide -->
    </div>
  </div>
</div>
```

У блоках "Найпопулярніші" і "Випадкове фото" відображаються дані, отримані після виконання запиту до бази даних. У першому випадку отримується список зображень, відсортованих у порядку зменшення середніх оцінок, а в другому випадку отримується випадкове зображення з усієї таблиці. Ці блоки є на кожній сторінці, тому запит виконується кожного разу, коли відкривається нова. Відповідно до цього логічно помістити цей код в контролер Controller в якості його методу (лістинг 2.10, 2.11).

Лістинг 2.10 – Метод отримання випадкових зображень

```
public function getRandomPhoto() {
    $criteria=new CDbCriteria();
    $criteria->limit = 4;
    $criteria->select = '*, rand() as rand';
    $criteria->order = 'rand';
    return Photo::model()->findAll($criteria);}

```

Лістинг 2.11 – Метод отримання найпопулярніших зображень

```
public function getTopRatedPhoto() {
$criteria=new CDbCriteria();
$criteria->limit = 4;
$criteria->order = 'rating DESC';
return Photo::model()->findAll($criteria);}
```

Також на кожній сторінці сайту є нижній колонтитул, ліва сторона якого являє собою статичну посилання, а права - посилання, сформовану за тим же принципом, що і головне меню.

2.3.2 Сторінка реєстрації

Коли ви реєструєте користувача, відображається спеціальне представлення контролера SiteController, і робота виконується над об'єктом, описаним у моделі LoginForm.php :

```
<h1>Створення користувача </h1>
<?php echo $this->renderPartial('_form', array('model'=>$model)); ?>
```

Потім це відображення намагається створити відображення з файлу _form.php. Передавши до нього масив \$model (лістинг 2.12).

Лістинг 2.12 – Вміст _form.php:

```
<div class="form">
<?php $form=$this->beginWidget('bootstrap.widgets.TbActiveForm',
array(
'id'=>'config-form',
'enableAjaxValidation'=>false,
)); ?>

<?php echo $form->errorSummary($model); ?>
<div class="row">
<?php echo $form->labelEx($model,'login'); ?>
<?php echo $form
>textField($model,'login',
array('size'=>45,'maxlength'=>45)); ?>
<?php echo $form->error($model,'login'); ?>
</div>
<div class="row">
<?php echo $form->labelEx($model,'password'); ?>
<?php echo $form->passwordField($model,
'password',array('size'=>45,'maxlength'=>32, 'value'=>''));?>
<?php echo $form->error($model,'password'); ?>
</div>
<!-- інші поля введення -->
```

```

<div class="row buttons">
    <?php
        $this->widget('bootstrap.widgets.TbButton',
        array('buttonType'=>'submit', 'type'=>'primary', 'label'=>$model-
        >isNewRecord ? 'Створити' : 'Зберегти')); ?>
    </div>
<?php $this->endWidget(); ?>

```

Зовнішній вигляд сторінки реєстрації показано на рис. 2.2.

РЕЄСТРАЦІЯ

Логін *

Пароль *

Повтор пароля *

Ім'я

Email

Телефон

Регіон *

Додаткова інформація

Рисунок 2.2 - зовнішній вигляд сторінки реєстрації

2.3.3 Сторінка входу

Алгоритм сторінки авторизації описаний в підрозділі 2.1.2. Для цього ми надаємо метод `login()`, який викликається контролером `SiteController` при вході в систему (список 2.13).

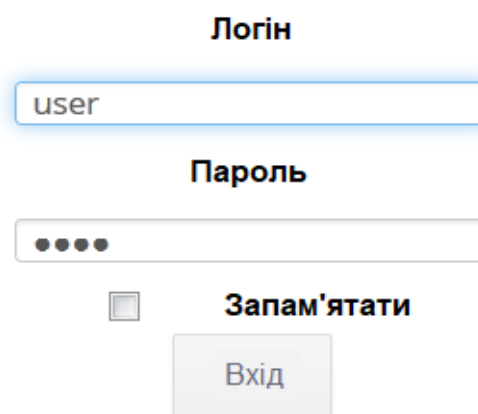
Лістинг 2.13 – Код методу login()

```
public function login()
{
    if($this->_identity === null)
    {
        $this->_identity = new UserIdentity($this->login,$this->password);
        $this->_identity->authenticate();
    }

    if($this->_identity->errorCode===UserIdentity::ERROR_NONE)
    {
        $duration=$this->rememberMe ? 3600*24*30 : 0; // 30 days
        Yii::app()->user->login($this->_identity,$duration);
        return true;
    }

    else
        return false;
}
```

Задаючи другий параметр методу login () з класу CWebUser, задається кількість секунд, протягом яких користувач може залишатися в системі. За замовчуванням 0 означає, що він буде підтверджений, поки користувач не закриє браузер. Якщо значення більше 0, використовується вхід на основі файлів cookie. Представлення сторінки показано на рис. 2.3.



Логін

user

Пароль

••••

☐
Запам'ятати

Вхід

Рисунок 2.3 – Вигляд сторінки входу користувача на сайт

2.3.4 Сторінка зворотного зв'язку

Сторінка зворотного зв'язку містить форму для надсилання запитань та пропозицій, а також конкретне поле підтвердження користувача для використання цього поля в `ContactForm.php` моделі у методі опису правил перевірки `rules()` вимагає, щоб був передан спеціально сформований масив виду:

```
array('verifyCode', 'captcha',  
'allowEmpty'=>!CCaptcha::checkRequirements())
```

Після додавання правила необхідно вивести на дисплей код, показаний в лістингу 2.14. Результати показані на рис. 2.4.

Лістинг 2.14 – Код, що міститься у відображенні

```
<?php if(CCaptcha::checkRequirements()): ?>  
<div class="row"><?php echo $form->label($model, 'verifyCode'); ?>  
<div><?php $this->widget('CCaptcha'); ?>  
<?php echo $form->textField($model, 'verifyCode',  
array('class'=>'input-small')); ?>  
</div>  
<?php echo $form->error($model, 'verifyCode', array('class'=>'text-  
error')); ?></div>  
<?php endif; ?>
```

Зворотній зв'язок

Ім'я

Email

Тема

Текст

Код перевірки

boyoe

Отримати новий код

Відправити

Рисунок 2.4 – Форма зворотнього зв'язку

2.3.5 Головна сторінка користувача

Після успішного підтвердження користувач перейде на сторінку для перегляду завантаженого зображення, але якщо воно не існує, з'явиться відповідне повідомлення і буде запропоновано завантажити перше зображення.

Сторінка користувача містить злегка змінене головне меню для роботи з функціями сайту, які доступні тільки зареєстрованим користувачам: меню керування зображеннями, витягує дані з масиву `$horizontalMenu`, визначеного в `Controller.php`. Якщо цей масив заповнений на будь-якому відображенні (він не порожній), з'явиться меню.

Він також відображає повзунок на сторінці користувача, який автоматично прокручує завантажені користувачем зображення. Для його реалізації використовується код відображення, показаний в лістингу 2.15.

Лістинг 2.15-Код слайдера із зображенням

```
if (!empty($photoData)) { ?>
<h3>Перегляд</h3>
<div class="flexslider">
<div class="slides">
<?php foreach ($photoData as $photo) {?>
<div class="slide">
<figure>

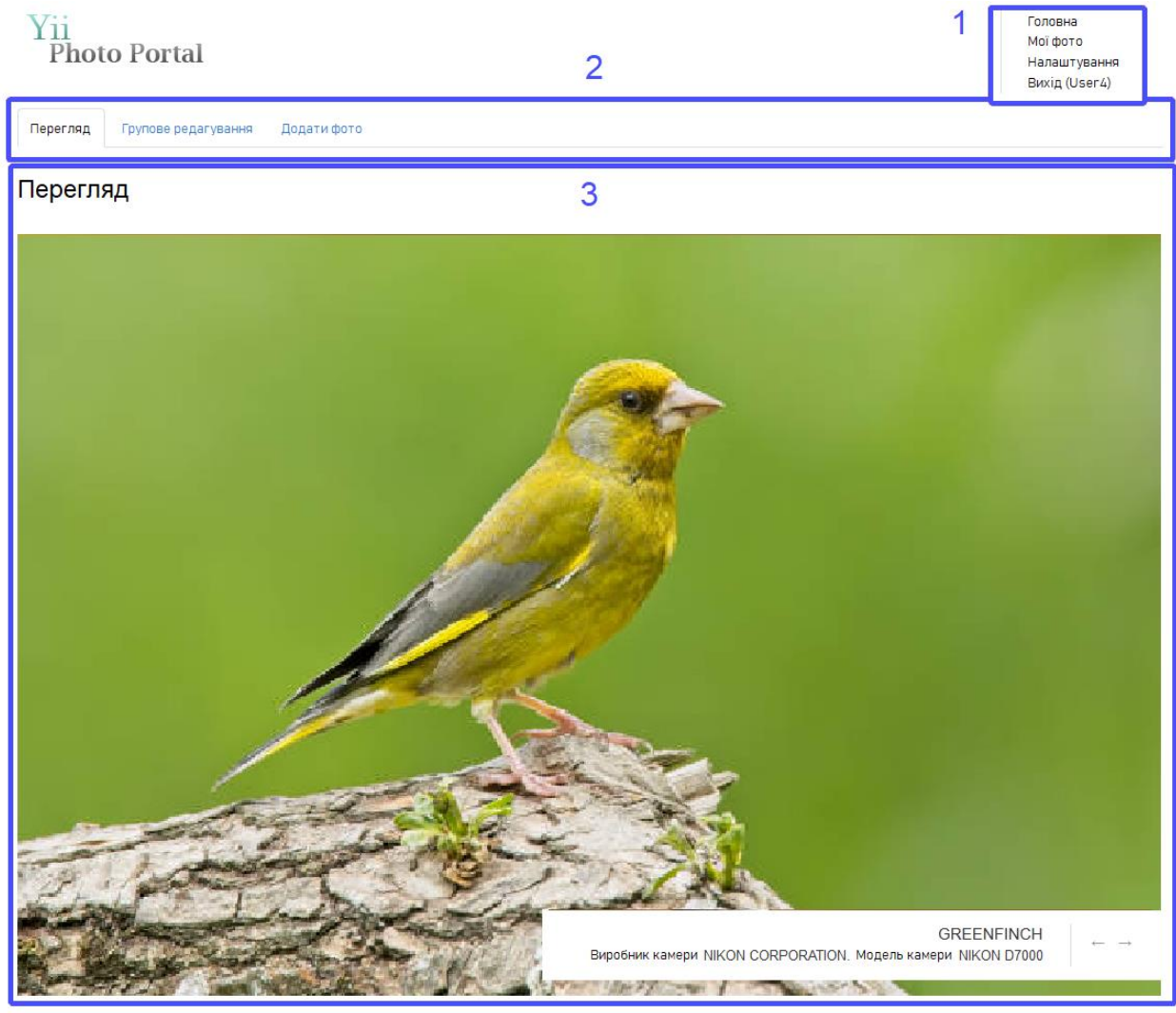
<figcaption>
<div>
<h5><a href="<?php echo $this->createAbsoluteUrl('/photo/view',
array('id'=>$photo->id)); ?>"><?php echo $photo->name; ?></a></h5>
<p><?php echo $photo->description; ?></p>
</div></figcaption></figure></div>
<?php } ?></div></div>
<?php } else { ?>
<h3 class="pagination-centered"> Ви не додали жодне зображення
</h3><?php } ?>
```

З наведеного вище лістингу можна бачити, що дані зображення для відображення слайдера беруться з масиву даних `$photoData`, відправленого контролером `DefaultController` модуля `user`. Цей масив формується за допомогою методу, показаного в лістингу 2.16.

Лістинг 2.16 - Метод формування масиву даних для слайдера

```
public function actionIndex(){
$criteria=new CDbCriteria();
$criteria->condition='userId=:userId';
$criteria->params=array(':userId'=>Yii::app()->user->getId());
$this->render('index', array('photoData' => Photo::model()-
>findAll($criteria))); }
```

Головна сторінка користувача наведена на рис. 2.5



1 - Головне меню після входу в систему; 2 - Меню керування зображеннями;

3 - Користувацький слайдер зображень

Рисунок 2.5 – Головна сторінка користувача

2.3.6 Сторінка групового редагування

При рендерингу сторінки групового редагування використовуються спеціально створені відображення і віджет WallManager.php. Якщо користувач хоче перейти на сторінку редагування групи, виконується метод `actionGroupEdit()` контролера `DefaultController`:

```
public function actionGroupEdit()
{
    $criteria=new CDbCriteria();
    $criteria->condition='userId=:userId';
```

```
$criteria->params=array(':userId'=>Yii::app()->user->getId());
$this->render('groupEdit', array('photoData' => Photo::model()-
>findAll($criteria)));
}
```

Метод виконує рендеринг відображення `groupEdit`, в якому виконується виведення віджету `WallManager`:

```
$this->widget('WallManager', array(
    'gallery' => $photoData,
    'controllerRoute' => '/photo', //route to gallery controller));
```

Віджет рендерит своє відображення `_galleryManager.php`, тобто передається масив `$photoData`, що містить дані про зображення користувача. Можна виконувати групові операції із зображенням, використовуючи функціональність створеного віджета. Сторінка редагування групових фотографій показана на рис. 2.6.

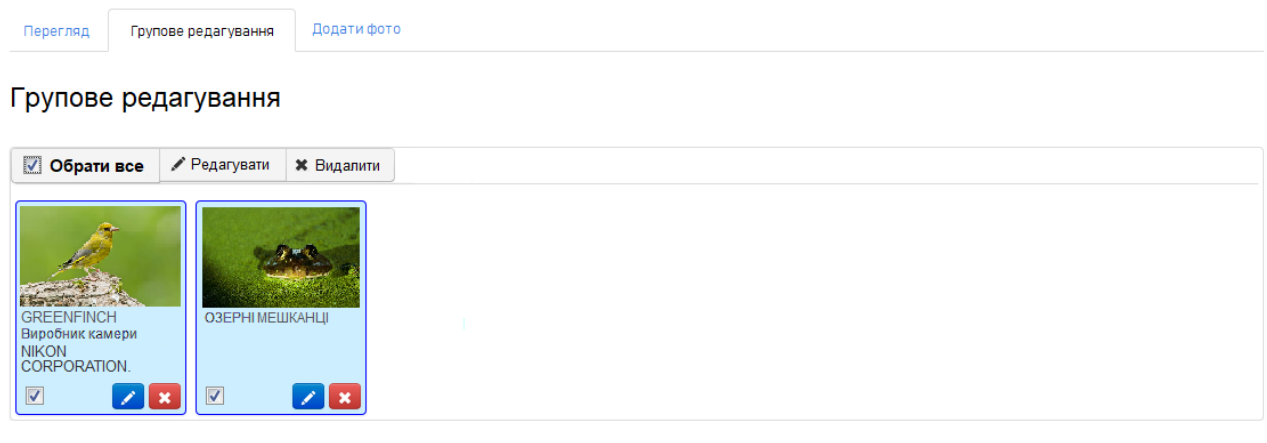


Рисунок 2.6 – Сторінка редагування групових фотографій

2.3.7 Сторінка для додавання зображень

При додаванні нового зображення до системи (рис. 2.7) використовується метод `actionCreate()` контролера `PhotoController`, який оперує моделями типу `Photo`. Код методу наведено у лістингу 2.17.

[Перегляд](#)
[Групове редагування](#)
[Додати фото](#)

ДОДАВАННЯ НОВОГО ФОТО

Категорія *

Зображення *

Назва *

Опис

Рисунок 2.7 – Сторінка додавання нового зображення до системи

Лістинг 2.17 – Код методу додавання нового зображення

```

public function actionCreate()
{
    $model=new Photo;
    if(isset($_POST['Photo']))
    {
        $model->attributes=$_POST['Photo'];
        $this->tempImage=CUploadedFile::getInstance($model,'path');
        if(!empty($this->tempImage)) {
            $model->path = md5(time()) . $model->galleryExt;
            if($model->save()){
                if (!file_exists(Yii::app()->image->params['directory'] .
                    Yii::app()->user->getId())){

                    mkdir(Yii::app()->image->params['directory'] . Yii::app()->user-
                        >getId());
                }
                $this->image = new Image($this->tempImage->tempName);
                $this->image->resize(1600, 1200, Image::AUTO);
                $this->image->save(Yii::app()->image->params['directory'] .
                    Yii::app()->user->getId().DIRECTORY_SEPARATOR. $model->path);
                $this->redirect(array('view','id'=>$model->id));
            }
            $this->render('create',array(
                'model'=>$model,
            ));
        }
    }
}

```

Також можна перевизначити метод `Save()` в моделі `Photo`, щоб можна було задати додаткові дані, які користувач не вводить, наприклад, дату зміни або створення запису в базі даних.

```
public function save($runValidation = true, $attributes = null)
{if ($this->getIsNewRecord()) {
$this->creationDate = new CDbExpression('NOW()');}
$this->lastEditingDate = new CDbExpression('NOW()');
$this->userId = Yii::app()->user->getId();
return parent::save($runValidation, $attributes); }
```

Після успішного збереження запису про нове зображення в базі даних завантажений файл перетворюється і поміщається в спеціальну директорію на сайті.

2.3.8 Сторінка перегляду зображень за категоріями

Для відображення списку зображень за обраною категорією (рис. 2.8) використовується компонент `yii-TbGridView` і передається об'єкт типу `CActiveDataProvider`, сформований в контролері (лістинг 2.18).

Лістинг 2.18 - Код для відображення списку зображень за обраною категорією

```
public function actionViewByCat($id) {
    $dataProvider = new CActiveDataProvider('Photo', array(
        'criteria' => array(
            'condition' => 'categoryId = ' . $id,
        ),
        'pagination' => array(
            'pageSize' => 30,
        ),
    ));
    $this->render('category', array(
        'dataProvider' => $dataProvider,
    ));
}

$this->widget('bootstrap.widgets.TbGridView', array(
    'dataProvider' => $dataProvider,
    'columns' => array(
        array('name' => 'path',
            'type' => 'image',
            'value' => 'Yii::app()->params["imagesPath"].$data->userId."/".$data->path',

            'headerHtmlOptions'=>array('width'=>'300px'),
        ),
        'description',
        // інші стовпці
    ),
));
```

Елемент 1—2 з 2			
Зображення	Назва	Опис	Рейтинг
	Запах літа		4.00
	НА ВОДОПОЇ	Модель камери: FinePix S200EXR	0.00

Рисунок 2.8 – Сторінка перегляду зображень за категоріями

2.3.9 Сторінки для перегляду зображень і коментування до них

При переході на відображення зображення (рис. 2.9), його зовнішній вигляд формується з використанням коду, показаного в лістингу 2.19.

Лістинг 2.19 – Код формування перегляду зображення

```
<div style="float: left;">
<figure>
<a href="<?php echo Yii::app()->params['imagesPath'].$model-
>userId."/". $model->path; ?>">
path; ?>" alt="">
<span class="zoom"></span>
</a>
</figure>
</div>
<div id="comments">
<?php if(count ($model->messages)>=1): ?>
<h3>
<?php echo count($model->messages) . ' коментаря(єв)'; ?>
</h3>
<?php $this->renderPartial('_messages',array(
'post'=>$model,
'comments'=>$model->messages,
)); ?>
<?php endif; ?>
<a href="<?php echo $this->createUrl('/message/create',
array('photoId'=>$model->id)); ?>">Додати свій коментар </a>
```

```

</div>
<div>
<?php foreach($comments as $comment) {?>
<article>

<?php echo $comment->user->name; ?> : <?php echo $comment->message;
?>
</article>
<?php } ?></div>

```

ОЗЕРНІ МЕШКАНЦІ 1



2 коментаря

User: гарна
User2: чудові очі
[Додати свій коментар](#)

Користувач	User 4
Категорія	Природа
Назва	Озерні мешканці
Опис	
Дата створення	2023-09-11 20:12:10
Рейтинг	3.00
Проголосувало	1

1 - Блок відображення зображення; 2 - Блок коментарів; 3 - Блок інформації про зображення

Рисунок 2.9 – Сторінка перегляду зображення

Для відображення супровідної інформації про зображення був використаний спеціальний Yii компонент CDetailView, де були перераховані всі вихідні поля моделі Photo (лістинг 2.20).

Лістинг 2.20 - Висновок супровідної інформації про зображення

```
<div style="float: right">
    <?php $this->widget('zii.widgets.CDetailView', array(
        'data'=>$model, 'attributes'=>array(
            array(
                'name'=>'userId',
                'type'=>'raw',
                'value'=>$model->user->name,
            ),
            array(
                'name'=>'categoryId',
                'type'=>'raw',
                'value'=>$model->category->name,
            ),
            'name', 'description', 'creationDate', 'rating', 'voted',
        ),
    )); ?></div>
```

Коли користувач додає коментар (рис. 2.10), якщо до роботи підключений контролер MessageController, дія actionCreate() створює коментар для вказаного зображення. Метод actionCreate() приймає ідентифікатор зображення, в якому створюється повідомлення, тому він передається через URI, а саме повідомлення стає тілом запиту POST.

ДОДАТИ КОМЕНТАР

Повідомлення *

Рисунок 2.10 – Форма додавання коментарю

2.3.10 Сторінка оновлення даних користувача

Для оновлення даних користувача (рис.2.11) використовує контролер `UserController` та його дію `actionUpdate()` яке за переданим ідентифікатором рендерит відповідне відображення (листинг 2.21).

Лістинг 2.21- Код рендеринга для відображення даних користувача

```
public function actionUpdate($id)
{
    $model=$this->loadModel($id);
    if(isset($_POST['User']))
    {
        $model->attributes=$_POST['User'];
        if($model->save())
        $this->redirect(array('view','id'=>$model->id));
    }
    $this->render('update',array(
        'model'=>$model,
    ));
}
```

Поле моделі передається в об'єкті `$model` моделі користувацького типу. Цей об'єкт надалі використовується на відображенні.

ОБНОВЛЕННЯ КОРИСТУВАЧА

Логін *

Пароль *

Повтор пароля *

Ім'я

Email

Телефон

Region *

Додаткова інформація

Створити

Рисунок 2.11 – Форма оновлення даних користувача

2.3.11 Додаткові можливості ролі адміністратора

Після перегляду ролі користувача ролі адміністратора призначаються додаткові функції. Механізм правил контролю доступу використовується як механізм захисту для доступу до дій контролера лише для певних ролей і визначається індивідуально для кожного контролера. У лістингу 2.22 показано приклад такого правила для CategoryController.

Лістинг 2.22 – Приклад правил для контролера керування категоріями

```
public function accessRules()
{
    return array(
        array('allow',
            'actions'=>array('index', 'view'),
            'users'=>array('*'),
        ),
    ),
```

```

        array('allow',
            'actions'=>array('create','update', 'admin','delete'),
            'users'=>array('admin'),
        ),
        array('deny',
            'users'=>array('*'),
        ),
    );
}

```

З наведеного вище коду ви можете бачити, що дії index та view цього контролера доступні всім користувачам, а операції create, update, admin та delete доступні лише користувачам із роллю "admin". Жоден користувач не може отримати доступ до всіх інших дій. Механізм працює. Доступ до всього, що явно заборонено - заборонено.

Коли адміністратор створює користувача (рис. 2.12), йому доступні додаткові поля – роль користувача, це можливо за рахунок коду, використовуюваного на відображенні для перевірки поточної ролі користувача, лістинг 2.23.

Лістинг 2.23 – Приклад відображення списку ролей користувача

```

<?php if(Yii::app()->user->checkAccess('admin')) { ?>
    <div class="row">
        <?php echo $form->labelEx($model,'roleId'); ?>
        <?php echo $form->dropDownList($model,'roleId',
            CHtml::listData(UserRole::model()->findAll(),
                'id', 'roleName'),
            array('prompt' => 'Выбрать из списка', )); ?>
        <?php echo $form->error($model,'roleId'); ?>
    </div>
    <?php } ?>

```

При рендерингу відображень керування користувачами (рис. 2.13) та керування налаштуваннями (рис. 2.14) використано компонент TbGridView, а також TbHeroUnit для рендерингу відображення пошукової форми при натисканні на відповідне посилання.

СТВОРЕННЯ КОРИСТУВАЧА

Логін *

user

Пароль *

••••

Повтор пароля *

Ім'я

Email

Телефон

Роль *

Выбрать из списка

Регіон *

Выбрать из списка

Додаткова інформація

Створити

Рисунок 2.12 – Форма створення користувача адміністратором

КЕРУВАННЯ КОРИСТУВАЧАМИ

Можна ввести порівняння (<, <=, >, >=, <> або =) на початку кожного із шуканого значення, щоб вказати, як порівняння має бути зроблено.

[Розширений пошук](#)

Елемент 1—6 з 6.

Логін	Ім'я	Роль	Регіон	Email	Телефон	
admin	Адміністратор	Адміністратор	Запорізька область	admin@admin.com	111111111111	
user	User	Користувач рівень 1	Миколаївська область	user@user.com	222222222222	
user2	User 2	Користувач рівень 1	Київська область	user2@user2.com	222222222222	
user3	User 3	Користувач рівень 1	Херсонська область	user3@user3.com	222222222222	
user4	User 4	Користувач рівень 1	Львівська область	user4@user4.com	222222222222	
user5	User 5	Користувач рівень 1	Хмельницька область	user5@user5.com	222222222222	

Рисунок 2.13 – Форма керування користувачами

УПРАВЛІННЯ НАЛАШТУВАННЯМИ

Можна ввести порівняння (<, <=, >, >=, <> або =) на початку кожного із шуканого значення, щоб вказати, як порівняння має бути зроблено.

[Розширений пошук](#)

Елемент 1—1 з 1.

Код	Назва	Значення	
adminEmails	Поштові скриньки для повідомлень	admin@admin.com	  

Рисунок 2.14 – Форма керування налаштуваннями

2.4 Висновки по розділу

Цей розділ займається розробкою web-платформи, який включає розробку функцій, графічний інтерфейс користувача та основні форми взаємодії користувача з сайтом. Використовується функціональність фреймворку Yii. Існує лістинги функціональних частин з описами і скріншотами відображення відповідних форм і web-сторінок.

3 ВПРОВАДЖЕННЯ БАЗИ ДАНИХ WEB-ПЛАТФОРМИ

3.1 Проектування сутностей

З урахуванням виділених вимог при проектуванні функціоналу сайту формуються наступні об'єкти бази даних: категорія (category), фотографія (photo), налаштування (settings), користувач (user), регіон (region), роль (user_role), повідомлення (message). Розглянемо кожен сутність.

Сутності ролей (user_role) призначені для зберігання списку користувацьких ролей, які визначають доступні користувацькі набори функцій (рис. 3.1).

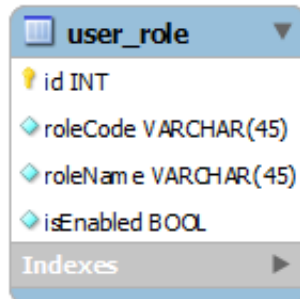


Рисунок 3.1 – Структура сутності Ролі

Сутність містить наступні поля:

- id - ідентифікатор ролі;
- roleCode - код ролі, який перевіряється для присвоєння певних привілеїв користувачеві;
- roleName - короткий опис, що характеризує назву ролі і відповідний їй набір дозволів;
- IsEnabled - ознака доступності ролі, яку користувач може використовувати.

Лістинг 3.1 – показує фрагмент SQL-коду, який створює цю сутність і вводить дані.

```
CREATE TABLE IF NOT EXISTS `user_role` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `roleCode` varchar(45) NOT NULL,
  `roleName` varchar(45) NOT NULL,
  `isEnabled` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  UNIQUE KEY `ID_UNIQUE` (`id`),
  UNIQUE KEY `levelCode_UNIQUE` (`roleCode`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;
INSERT INTO `user_role` (`id`, `roleCode`, `roleName`, `isEnabled`) VALUES
```

```
(1, 'admin', 'Адміністратор', 1), (2, 'user1', 'Користувач рівень 1', 1), (3, 'userL2', 'Користувач рівень 2', 0), (4, 'userL3', 'Користувач рівень 3', 0);
```

Сутність регіони (region) призначена для зберігання списку регіонів України, і один з них може бути обраний користувачем при реєстрації в системі(рис. 3.2).

Сутність містить наступні поля:

- id – ідентифікатор регіону;
- name – назва регіону.

Лістинг 3.2 –показує фрагмент SQL-коду, який описує цю сутність та вводить дані.

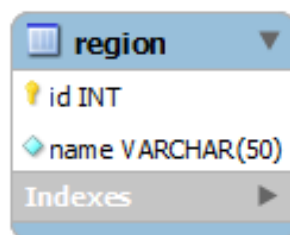


Рисунок 3.2 – Сутність Регіонів

Лістинг 3.2 – Фрагмент SQL-коду для сутностей регіонів.

```
CREATE TABLE IF NOT EXISTS `region` (`id` int(10) unsigned NOT NULL
AUTO_INCREMENT, `name` varchar(50) NOT NULL, PRIMARY KEY (`id`))
ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=26 ;
INSERT INTO `region` (`id`, `name`) VALUES
(1, 'АР Крим'), (2, 'Вінницька область'), (3, 'Волинська область'),
(4, 'Дніпропетровська область'), (5, 'Донецька область'), (6,
'Житомирська область'),
(7, 'Закарпатська область'), (8, 'Запорізька область'), (9, 'Івано-
Франківська область'), (10, 'Київська область'), (11, 'Кіровоградська
область'), (12, 'Луганська область'), (13, 'Львівська область'), (14,
'Миколаївська область'), (15, 'Одеська область'), (16, 'Полтавська
область'), (17, 'Рівненська область'), (18, 'Сумська область'), (19,
'Тернопільська область'), (20, 'Харківська область'), (21, 'Херсонська
область'), (22, 'Хмельницька область'), (23, 'Черкаська область'),
(24, 'Чернігівська область'), (25, 'Чернівецька область');
```

Сутність категорії (category) призначена для зберігання списку категорій фотографій (рис. 3.3). Потім ці категорії з'являться на головній сторінці сайту, але це один з основних факторів групування зображень на сайті.

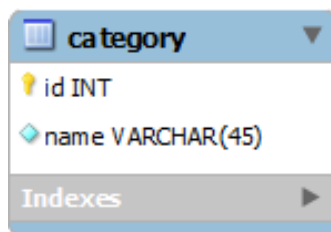


Рисунок 3.3 – Сутність Категорії

Сутність містить наступні поля:

- id – ідентифікатор регіону;
- name – назва регіону.

Лістинг 3.3 показує фрагмент SQL-коду, який регулює цю сутність і вводить дані.

Лістинг 3.3 - Фрагмент SQL коду сутності категорії

```
CREATE TABLE IF NOT EXISTS `category` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=12 ;
INSERT INTO `category` (`id`, `name`) VALUES
(2, 'Колаж'), (3, 'Макро'), (4, 'Натюрморт'), (5, 'Тварини'),
(6, 'Пейзаж'), (7, 'Підводний світ'), (8, 'Портрет'),
(9, 'Природа'), (10, 'Подорожі'), (11, 'Спорт');
```

Сутність конфігурації (config) призначений для зберігання системних налаштувань, констант, що задають параметри відображення інформації на сайті (рис. 3.4). Збереження таких налаштувань у базі даних дає вам можливість гнучко змінювати зовнішній вигляд сторінки, поведінку всього сайту, змінюючи деякі значення в базі даних, а не замінюючи параметри різних php-скриптів.

Сутність включає наступні поля:

- id - ідентифікатор параметра;
- code - код параметра, який використовується для перевірки наявності параметра і отримання значення;
- name - ім'я параметра;
- value - значення параметра, дані, які використовуються при виборі конкретного параметра.

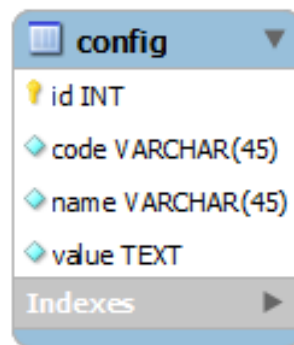


Рисунок 3.4 – Сутність Налаштування

Лістинг 3.4 – показує фрагмент SQL-коду, який описує цю сутність та вводить дані.

```
CREATE TABLE IF NOT EXISTS `config` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `code` varchar(45) NOT NULL,
  `name` varchar(45) NOT NULL,
  `value` text NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `configCode_UNIQUE` (`code`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;
INSERT INTO `config` (`id`, `code`, `name`, `value`)
VALUES (1, 'adminEmails', 'Поштові скриньки для сповіщень',
'admin@admin.com');
```

Сутність користувачі (users) являє собою таблицю для зберігання даних про користувачів у системі (рис. 3.5). Ця таблиця містить основні поля, які ідентифікують і авторизують користувачів у системі, відображають дані про зміни в їхніх облікових записах і дають змогу блокувати їх за IP. Під час створення

користувача він вказує регіон і призначає стандартну роль. Користувач також вводить персональні дані, логін і пароль, які зберігаються в хешованому вигляді та підвищують рівень безпеки системи.

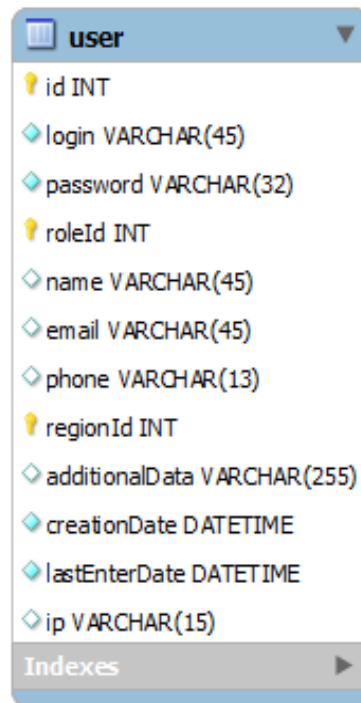


Рисунок 3.5 – Сутність Користувачі

Поля сутності:

- id - ідентифікатор користувача;
- login - логін користувача для входу в систему;
- password - хешований результат пароля користувача;
- roleId - зовнішній ключ. Ідентифікатор ролі, яку використовує користувач;
- name - ім'я користувача;
- email - адреса електронної пошти користувача;
- phone - номер телефону користувача;
- regionId - зовнішній ключ. Ідентифікатор регіону, до якого належить користувач;

- additionalData - додаткова інформація про користувача;
- creationDate - дата створення запису;
- lastEnterDate - дата останньої успішної аутентифікації в системі;
- ip - IP-адрес користувача;

Фрагмент SQL-коду, що описує цю сутність і заповнює її, наведено в лістингу 3.5.

Лістинг 3.5 – Фрагмент SQL-коду для сутності користувачі

```
CREATE TABLE IF NOT EXISTS `user` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `login` varchar(45) NOT NULL, `password` varchar(32) NOT NULL,
  `roleId` int(10) unsigned NOT NULL,
  `name` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  `phone` varchar(13) DEFAULT NULL,
  `regionId` int(10) unsigned NOT NULL,
  `additionalData` varchar(255) DEFAULT NULL,
  `creationDate` datetime NOT NULL,
  `lastEnterDate` datetime NOT NULL,
  `ip` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`id`,`roleId`,`regionId`),
  UNIQUE KEY `ID_UNIQUE` (`id`),
  UNIQUE KEY `login_UNIQUE` (`login`),
  UNIQUE KEY `email_UNIQUE` (`email`),
  KEY `fk_user_user_role_idx` (`roleId`),
  KEY `fk_user_region1_idx` (`regionId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=8;
INSERT INTO `user` (`id`, `login`, `password`, `roleId`, `name`,
`email`, `phone`, `regionId`, `additionalData`, `creationDate`,
`lastEnterDate`, `ip`) VALUES
(1, 'admin', '21232f297a57a5a743894a0e4a801fc3', 1, 'Адміністратор ',
'admin@admin.com', '111111111111', 1, NULL, '2023-07-29 21:04:51',
'2023-07-29 21:04:51', '127.0.0.1'),
(2, 'user', 'ee11cbb19052e40b07aac0ca060c23ee', 2, 'User',
'user@user.com ', '22222222222', 1, NULL, '2023-08-12 22:04:47',
'2023-08-12 22:04:47', '127.0.0.1'),
(3, 'user2', '7e58d63b60197ceb55a1c487989a3720', 2, 'User 2',
'user2@user2.com ', '22222222222', 2, NULL, '2023-08-26 12:21:17',
'2023-08-26 12:21:17', '127.0.0.1'),
(4, 'user3', '92877af70a45fd6a2ed7fe81e1236b78', 2, 'User 3',
'user3@user3.com ', '22222222222', 3, NULL, '2023-09-04 20:21:59',
'2023-09-04 20:21:59', '127.0.0.1'),
(5, 'user4', '3f02ebe3d7929b091e3d8ccfde2f3bc6', 2, 'User 4',
'user4@user4.com ', '22222222222', 4, NULL, '2023-09-02 20:22:42',
'2023-09-02 20:22:42', '127.0.0.1'),
```

```
(6, 'user5', '0a791842f52a0acfb3a783378c066b8', 2, 'User 5',
'user5@user5.com ', '2222222222', 5, NULL, '2023-08-11 20:22:40',
'2023-08-11 20:22:40', '127.0.0.1');
```

Сутність фотографії (photo) призначена для зберігання всіх фотографій у системі, самих фотофайлів і даних про те, які користувачі в які категорії завантажили фотографії (рис. 3.6).

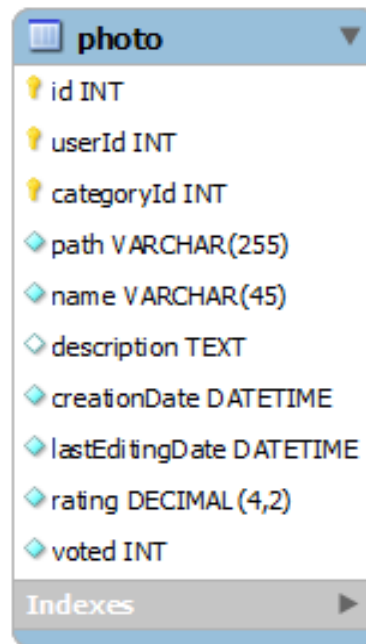


Рисунок 3.6 – Сутність Фото

Сутність включає наступні поля:

- id - ідентифікатор фотографії;
- userid - зовнішній ключ. Ідентифікатор користувача, якому належить дане зображення;
- categoryid - зовнішній ключ. Ідентифікатор категорії, до якої належить дане зображення; зовнішній ключ.
- path - шлях та ім'я файлу до зображення;
- name - ім'я зображення;
- description - опис зображення;
- creationdate - дата створення;

- lasteditingdate - дата останнього редагування;
- рейтинг - середній рейтинг;
- voted - кількість користувачів, які поставили оцінку.

У лістингу 3.6 показано фрагмент SQL-коду, який описує цю сутність і заповнює її даними.

Лістинг 3.6 – Фрагмент SQL коду сутності фото

```
CREATE TABLE IF NOT EXISTS `photo` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `userId` int(10) unsigned NOT NULL,
  `categoryId` int(10) unsigned NOT NULL,
  `path` varchar(255) NOT NULL,
  `name` varchar(45) NOT NULL,
  `description` text,
  `creationDate` datetime NOT NULL,
  `lastEditingDate` datetime NOT NULL,
  `rating` decimal(4,2) unsigned NOT NULL DEFAULT '0.00',
  `voted` int(10) unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`,`userId`,`categoryId`),
  KEY `fk_photo_user1_idx` (`userId`),
  KEY `fk_photo_category1_idx` (`categoryId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=49 ;
INSERT INTO `photo` (`id`, `userId`, `categoryId`, `path`, `name`,
`description`, `creationDate`, `lastEditingDate`, `rating`, `voted`)
VALUES
(44, 4, 6, '884ab571ee19ceca2247ed157ce84c95.jpg', '****', '', '2023-
05-02 20:35:30', '2023-05-02 20:35:30', '0.00', 0),
(45, 4, 7, '8bae77ad6a34a4bac8b4dfed4e04e3a7.jpg', 'How much is the
fish?', 'Виробник камери NIKON CORPORATION', '2023-05-02 20:36:54',
'2023-05-02 20:36:54', '0.00', 0),
(42, 3, 2, 'dc660646075738ccd011a64284de243f.jpg', 'Дорога у мрії',
'', '2023-08-02 20:31:43', '2023-08-02 20:31:43', '0.00', 0),
(43, 3, 6, '57069d4bf98f22ddf86a8bc4ea3b658c.jpg', 'Morning',
'Виробник камери Canon. Модель камери Canon EOS 5D.', '2023-09-02
20:32:51', '2023-09-02 20:32:51', '5.00', 1),
(40, 3, 3, '0506f94a02d3dbc8c8d3d28e9461d09e.jpg', 'Запах літа', '',
'2023-05-02 20:29:19', '2023-05-02 20:29:19', '4.00', 3),
(41, 3, 3, '4ec55c02c9735d83ee76819b7a3ede63.jpg', 'НА ВОДОПОЇ',
'Модель камери FinePix S200EXR', '2023-05-02 20:30:41', '2023-05-02
20:30:41', '0.00', 0),
(46, 5, 8, '73d1681a832a8cd4bbfd274ed384eff9.jpg', 'Portrait with
magnolias', '', '2023-04-01 20:38:01', '2023-04-01 20:38:01', '0.00',
0),
(47, 6, 9, '4926e0a0623764724fa74fa01b10f4ec.jpg', 'Greenfinch',
'Виробник камери NIKON CORPORATION. Модель камери NIKON D7000', '2023-
05-02 20:39:33', '2023-05-02 20:39:33', '4.75', 4),
```

```
(48, 6, 9, '98d7955777ec4937f407a65cde1d73da.jpg', 'Озерні  
мешканці', '', '2023-12-02 20:40:13', '2023-12-02 20:40:13', '3.00',  
1);
```

Сутності повідомлення (message) - рис. 3.7. Кожне зображення користувача не тільки оцінюється, а й коментується іншими користувачами системи. Для цього використовується таблиця повідомлень, яка пов'язує користувачів і прокоментовані ними зображення.

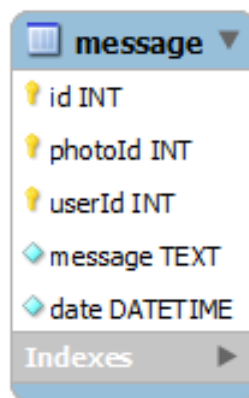


Рисунок 3.7 – Сутність Повідомлення

Сутність включає наступні поля:

- id - ідентифікатор повідомлення;
- photoId - зовнішній ключ. Ідентифікатор коментованого зображення;
- userId - зовнішній ключ. Ідентифікатор користувача, який залишив коментар;
- message - текст повідомлення;
- date - дата відправлення повідомлення.

Фрагмент SQL-коду, що описує цю сутність і заповнює її даними, наведено в лістингу 3.7.

Лістинг 3.7 – Фрагмент SQL коду сутності повідомлення

```
CREATE TABLE IF NOT EXISTS `message` (  
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,  
  `photoId` int(10) unsigned NOT NULL,
```

```

`userId` int(10) unsigned NOT NULL,
`message` text NOT NULL, `date` datetime NOT NULL,
PRIMARY KEY (`id`,`photoId`,`userId`),
UNIQUE KEY `id_UNIQUE` (`id`),
KEY `fk_message_photo1_idx` (`photoId`),
KEY `fk_message_user1_idx` (`userId`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 AUTO_INCREMENT=4;
INSERT INTO `message` (`id`, `photoId`, `userId`, `message`, `date`)
VALUES
(1, 48, 3, 'гарна', '2023-07-11 16:42:10'),
(2, 48, 4, 'чудові очі', '2023-09-01 21:13:07'),
(3, 42, 4, 'відмінне фото!!!', '2023-05-22 09:44:08');

```

3.2 Визначення відносин між сутностями

Установимо зв'язки між сутностями. Зв'язок роль (user_role) - користувач (user). Одне-ко-многим. Кожна роль може належати багатьом користувачам, а кожен користувач може мати тільки одну роль (рис. 3.8).

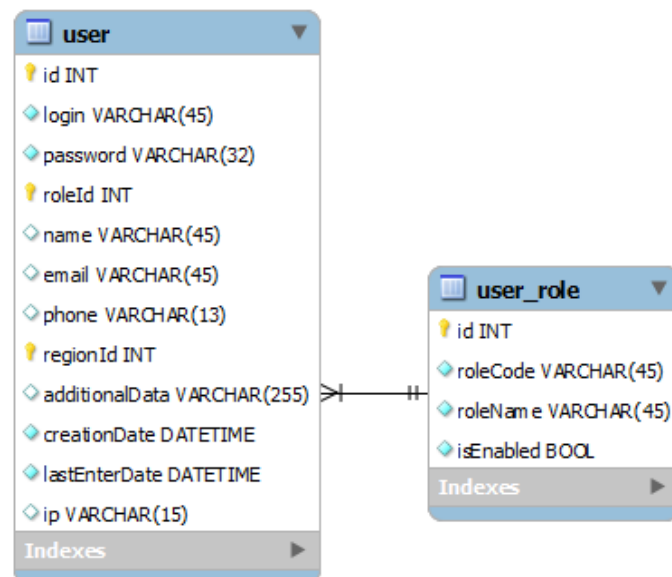


Рисунок 3.8 – Зв'язок Ролі (user_role) - Користувачі (user)

Зв'язок регіон (region) - користувач (user). Один-ко-многим. Кожен регіон може належати кільком користувачам, а кожен користувач може мати тільки один регіон (рис. 3.9).

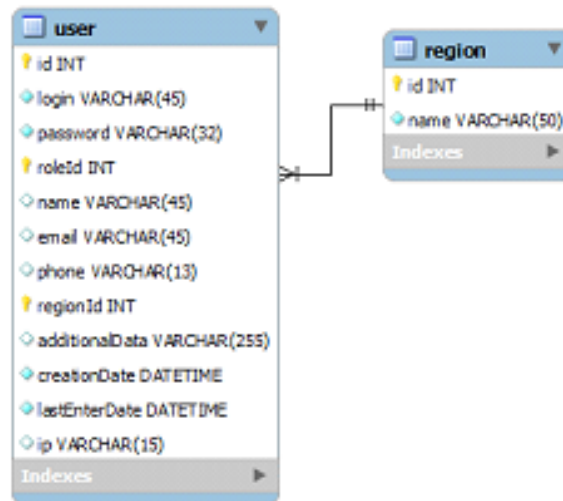


Рисунок 3.9 – Зв'язок Регіони (region) - Користувачі (user)

Посилання категорія (category) - фотографія (photo). Один-ко-многим. Кожна категорія може належати кільком фотографіям, а кожна фотографія може належати лише до однієї категорії (рис. 3.10).

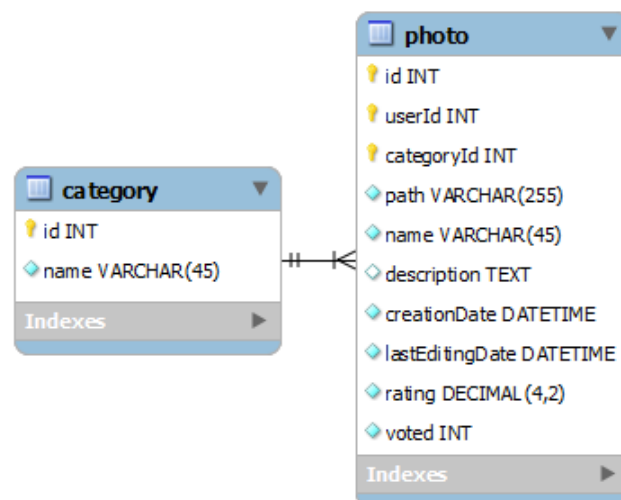


Рисунок 3.10 – Зв'язок Категорії (category) - Фотографії (photo)

Зв'язок користувач (user) - фотографія (photo). Один-ко-многим. Кожен користувач може створити безліч фотографій, і до кожної фотографії може бути додано тільки один користувач (рис. 3.11).

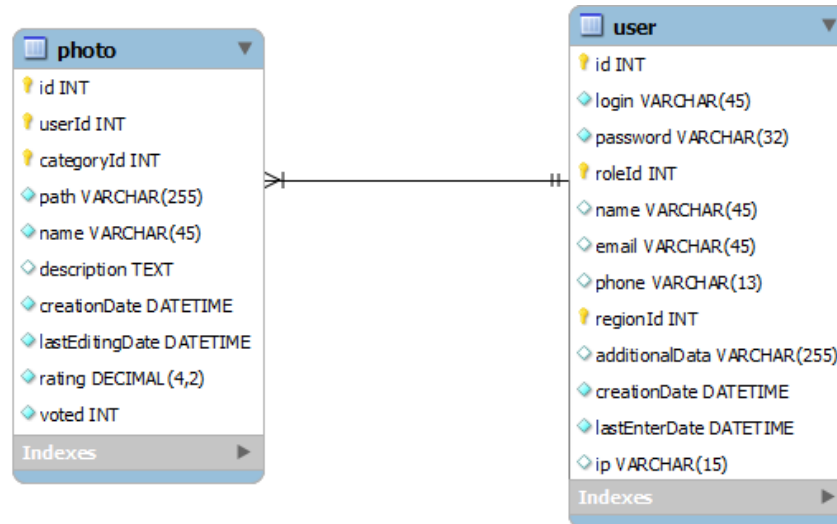


Рисунок 3.11 – Зв'язок Користувачі (user) - Фотографії (photo)

Зв'язок користувачі (user) - повідомлення (message). Один-ко-многим. Кожен користувач може створити безліч повідомлень, при цьому кожне повідомлення може бути додано тільки одним користувачем (рис. 3.12).

Зв'язок фотографії (photo) - повідомлення (message). Один-ко-многим. Кожна фотографія може містити деяку кількість повідомлень, і кожне повідомлення може бути опубліковано тільки для однієї фотографії (рис. 3.13).

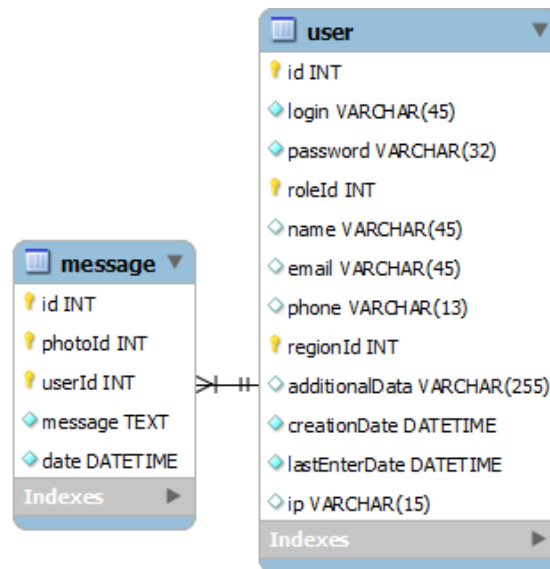


Рисунок 3.12 – Зв'язок Користувачі (user) - Повідомлення (message)

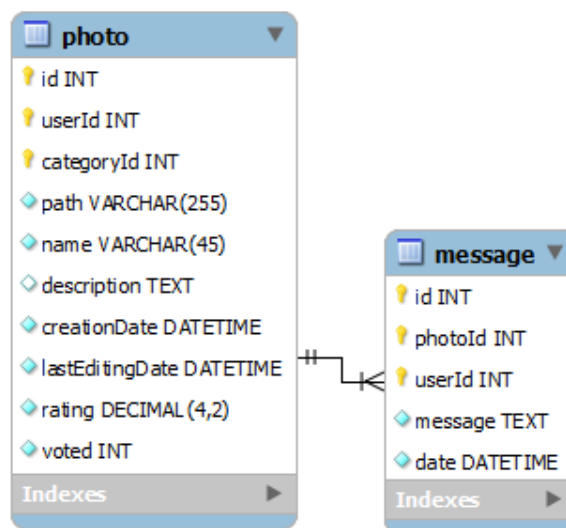


Рисунок 3.13 – Зв'язок Фото (photo) - Повідомлення (message)

Файлова структура движка MyISAM розробленої бази даних показана на рис. 3.15. Розроблена база даних підключена до сайту за допомогою функцій мови php та розміщена на сервері web-платформи.

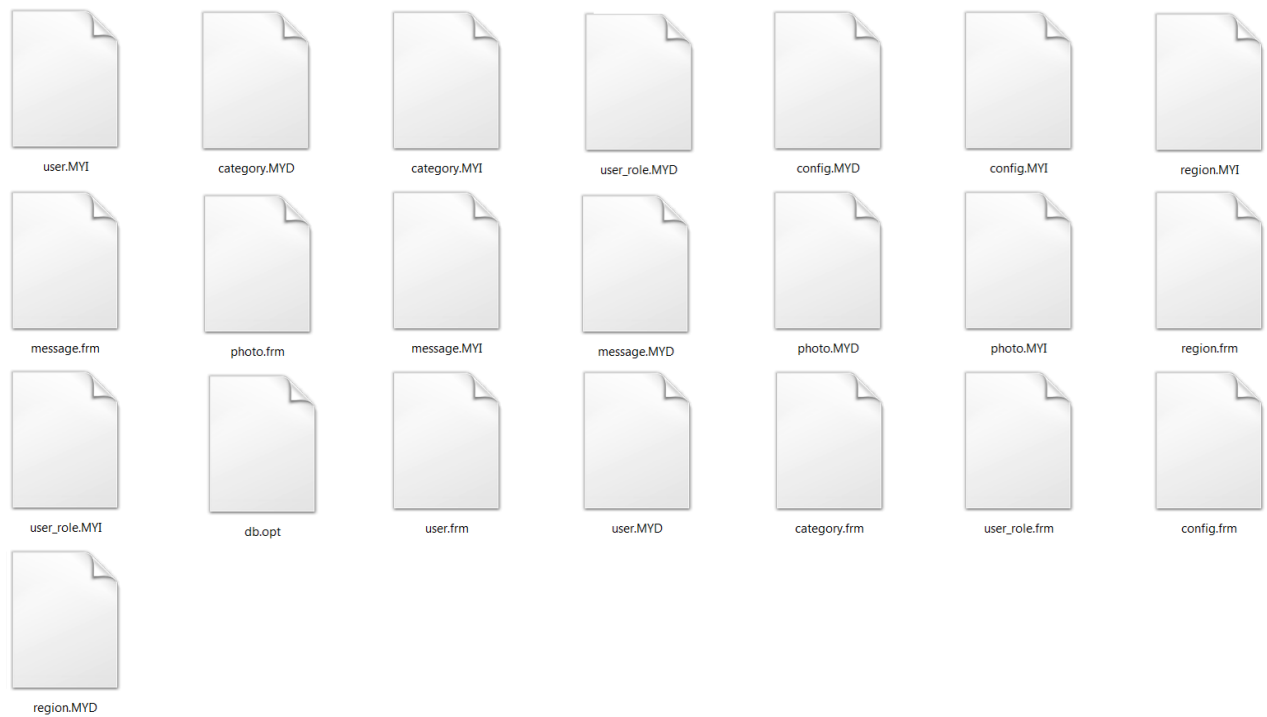


Рисунок 3.15 – Файлова структура движка MyISAM БД

3.3 Висновки по розділу

У цьому розділі створюються основні сутності бази даних web-платформи: категорії, фотографії, конфігурації, користувачі, регіони, ролі, повідомлення та їхні обґрунтовані поля, а також визначаються основні зв'язки між сутностями.

4 ДОСЛІДЖЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Серверне обладнання web-платформи

4.1.1 Вибір конфігурації сервера

Використання платформ Intel за критерієм співвідношення ціна/якість. У цьому випадку використовуються рішення з чотирнадцятьма ядрами процесорами.

Це дає змогу серверу одночасно обробляти більшу кількість потоків і, відповідно, обслуговувати понад 200 клієнтів з одночасним доступом до бази даних.

У табл. 4.1 наведено огляд апаратних компонентів сервера.

Таблиця 4.1 - Апаратні компоненти сервера web-платформи

Компонент	Опис
Материнська плата	Gigabyte B760
Процесор	Intel Core i5-13600K BOX
Відеоадаптер	Інтегрований
Оперативна пам'ять	16 Гб DDR4-5200 RAM
Дисковий накопичувач	Відмовостійкий HGST Ultrastar 7k6000 4 Tb
Блок живлення	850 Вт
Мережевий адаптер	Інтегровані адаптери 2 x Gigabit Ethernet
Монітор	"23" Samsung S23A300N LED, 1920x1080

4.1.2 Вибір джерела безперебійного живлення

Для забезпечення цілодобової безперебійної роботи серверів мають бути передбачені промислові системи безперебійного живлення, що виконують такі функції:

- забезпечення живлення обладнання за відсутності зовнішнього джерела живлення;
- захист від імпульсних перешкод від зовнішніх джерел живлення;
- підтримання напруги живлення в межах номінальних значень протягом не менше 2 годин;
- управління мережею за допомогою спеціального програмного забезпечення.

APC Smart-UPS RT 3000VA RM 230V використовується як промислове джерело безперебійного живлення. Він може забезпечити 2,1 кВт потужності протягом 2 годин при вхідній напрузі 45-65 Гц, 155-290 В і температурі 0-40 0°C[10].

4.1.3 Вибір системи хостингу

Для розміщення web-платформи до сервера повинен бути підключений зовнішній мережевий канал. Пропускної здатності, яку забезпечує технологія Fast Ethernet, достатньо для задоволення вимоги 100 Мбіт/с.

В якості провайдера ми використовуємо ресурси компанії "Воля" в Запоріжжі. Згідно з умовами хостингу, які надає ця компанія, швидкість прийому українського трафіку становить 100 Мбіт/с, іноземних каналів - 15 Мбіт/с. Вихідний трафік - 100 Мбіт/с. Виділення статичної IP-адреси та реєстрація в базі даних DNS.

4.2 Конфігурування web-сервера Apache

Виконаємо встановлення та налаштування web-сервера Apache

HTTP Server. Розпаковуємо архів дистрибутива до каталогу C:\Apache2. Відкриваємо в текстовому редакторі файл "C:\Apache2\conf\httpd.conf", який є основним файлом конфігурації сервера Apache.

Щоб завантажити модуль mod_rewrite, знайдіть і закоментуйте наступний рядок: "LoadModule rewrite_module modules/mod_rewrite.so". Додайте рядок: LoadModule php5_module "C:/php/php5apache2_2.dll". Додайте наступний рядок для визначення каталогу, що містить конфігураційний файл PHP: PHPIniDir "C:/php". Знайдіть і закоментуйте рядок: "ServerName www.example.com:80". Відредагуйте цей рядок, щоб встановити початкове ім'я сервера: "ServerName localhost:80". Знайдіть наступний рядок: "DocumentRoot "c:/Apache2/htdocs". Розпізнайте кореневий каталог адміністрування сайту: "DocumentRoot "C:/apache". Знайдіть наступний блок:

```
"<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
```

```
</Directory>"
```

Замінюємо на наступний:

```
"<Directory />
    Options Includes Indexes FollowSymLinks
    AllowOverride All
    Allow from all
</Directory>"
```

Якщо видалити вихідний блок управління каталогом, то він матиме такий вигляд:

```
"<Directory "c:/Apache2/htdocs">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>"
```

Знаходимо блок:

```
"<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>"
```

Замінюємо його на:

```
"<IfModule dir_module>
    DirectoryIndex index.html index.htm index.shtml index.php
</IfModule>"
```

Знаходимо рядок:

```
"ErrorLog "logs/error.log"
```

Замінюємо на наступний:

```
"ErrorLog "C:/apache/error.log"
```

Знаходимо рядок:

```
"CustomLog "logs/access.log" common"
```

Замінюємо на:

```
"CustomLog "C:/apache/access.log" common"
```

Для того щоб SSI працював (був увімкнений на стороні сервера), необхідно знайти і відкоментувати наступний рядок у блоці<IfModule mime_module>:

```
"AddType text/html .shtml
AddOutputFilter INCLUDES .shtml"
```

Додаємо нижче, у той же блок <IfModule mime_module>, два рядки:

```
"AddType application/x-httpd-php .php"
```

```
AddType application/x-httpd-php-source .phps"
```

Розкоментуємо рядки:

```
"Include conf/extra/httpd-mpm.conf
Include conf/extra/httpd-autoindex.conf
Include conf/extra/httpd-vhosts.conf
Include conf/extra/httpd-manual.conf
Include conf/extra/httpd-default.conf"
```

Зберігаємо зміни та закриваємо файл "httpd.conf".

Відкриваємо файл "C:\Apache2\conf\extra\httpd-vhosts.conf" і робимо такі зміни: видаляємо існуючий блок прикладу віртуального хоста вставляємо лише наступний блок:

```
"NameVirtualHost *:80
<VirtualHost *:80>
    DocumentRoot "C:/apache/localhost/www"
    ServerName localhost
    ErrorLog "C:/apache/localhost/error.log"
    CustomLog "C:/apache/localhost/access.log" common
</VirtualHost>"
```

Зберігаємо зміни та закриваємо файл "httpd-vhosts.conf". Тепер, коли роботу з конфігураційним файлом завершено, наступним кроком буде встановлення служби Apache. Для цього запускаємо командний рядок і вводимо в ньому "C:\Apache2\bin\httpd.exe -k install".

Далі слід створити структуру каталогів віртуального хоста. У корені диска необхідно створити каталог "apache". Цей каталог містить віртуальний хост (домен) користувача, який орендує сервер центру оброблення даних, глобальний файл журналу помилок "error.log" (автоматично створюється програмою під час першого запуску), глобальний файл доступу "access.log" (автоматично створюється програмою), включно з порожньою папкою "localhost" у каталозі "apache" та папкою "www" у ній.

Якщо необхідно встановити власні віртуальні хости, відкрийте файл "httpd-vhosts.conf" і створіть у ньому такий блок:

```
"<VirtualHost *:80>
    # Папка, у якій буде корінь хоста.
    DocumentRoot "C:/apache/fotoportal.zp.ua/www"
    # Домен, яким звертатися до віртуального хосту."
```

```

ServerName fotoportal.zp.ua
# Аліас (додаткове ім'я) домену.
ServerAlias fotoportal
# Файл, у якому записуватимуться помилки.
ErrorLog "C:/apache/test.ru/error.log"
# Файл журналу доступу до хоста.
CustomLog "C:/apache/test.ru/access.log" common
</VirtualHost>"

```

Далі створіть папку "fotoportal.zp.ua" в каталозі "apache" і папку "www" всередині неї. Потім запустіть сервер Apache, запустивши командний рядок:

"C:\Apache2\bin\httpd.exe -k start". Відкрийте браузер і введіть в адресному рядку "fotoportal.zp.ua" або "www.fotoportal.zp.ua".

4.3 Впровадження бази даних

В якості СУБД використовується MySQL. Встановлюємо MySQL 5.7.X. Далі вибираємо "Detailed configuration" і позначте пункти, показані на рисунку 4.1. Вибираємо "Server machine" для типу вузла (для web-сервера). Слід вибрати пункт "Multifunctional Database" оскільки він використовує як транзакційні таблиці, такі як InnoDB, так і швидку нетранзакційну таблицю MyISAM". Для нашої роботи потрібен MyISAM (обґрунтування див. у розділі 1), але залежно від обраного типу бази даних за потреби ми можемо перейти на InnoDB.

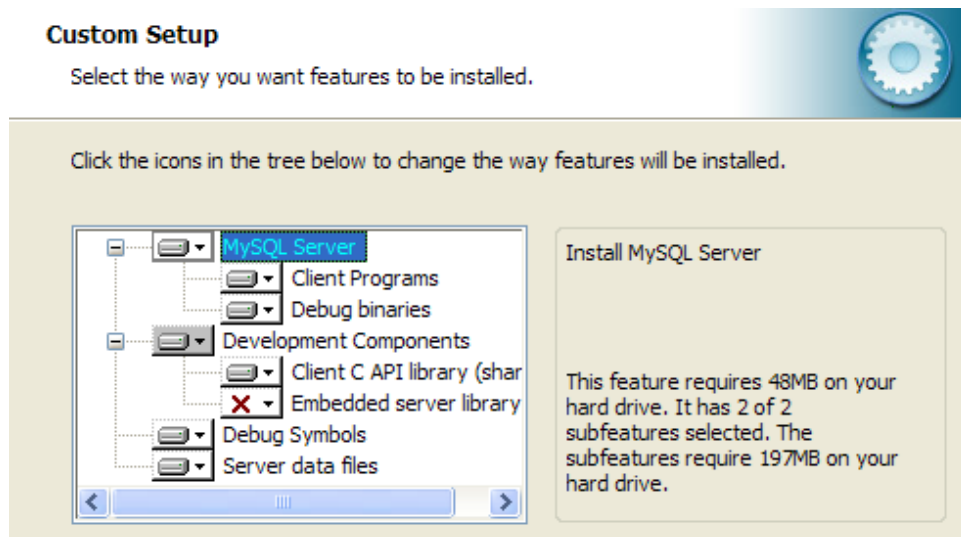


Рисунок 4.1 – Встановлювані компоненти MySQL

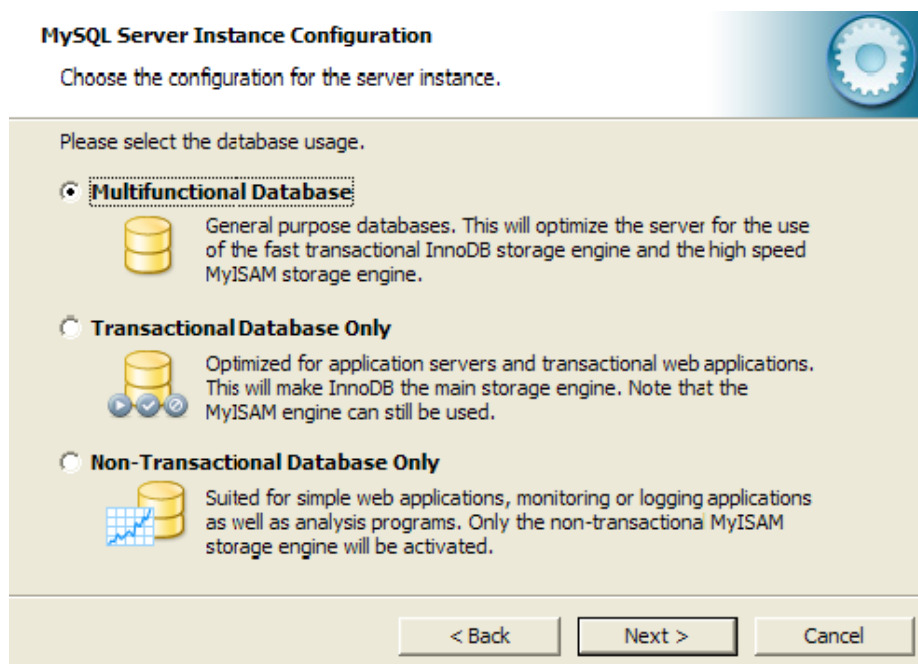


Рисунок 4.2 – Конфігурація БД

Далі слід вибрати розділ і каталог, у якому зберігатимуться таблиці; вказати максимальну кількість одночасних підключень до сервера MySQL – 250. Увімкнути підтримку TCP/IP-з'єднань, встановивши прапорець "Enable TCP/IP Networking", і вибрати порт для підключення. Стандартний порт для серверів MySQL – 3306. Встановимо прапорець "Enable Strict Mode", щоб встановити режим

суворої відповідності стандарту SQL. Як кодування вибираємо "cp1251". Використовувати кириличне кодування Windows (cp1251) для таблиць встановимо прапорець "Install As Windows Service" для запуску сервера як служби. У випадаючому списку під ним заходимо ім'я служби. Встановлюємо прапорець "Include Bin Directory in Windows PATH" - це призведе до появи каталогу "bin" у командному рядку. Встановлюємо пароль для користувача "root" (не менше 12 символів). Відредагуємо додатковий конфігураційний файл "my.ini":

- після "port=3306" в [client] додаємо рядок, що ідентифікує директорію, де знаходиться файл опису кодувань:
 - character-sets-dir="C:/Program Files/MySQL/MySQL Server 5.7/share/charsets";
- в [mysqld] додаємо рядок після "port=3306";
- character-sets-dir="C:/Program Files/MySQL/MySQL Server 5.7/share/charsets";
- init-connect="set names cp1251". Далі зробимо MyIsam таблицею без транзакцій за замовчуванням, замінивши рядок default-storage-engine=INNODB на default-storage-engine=MYISAM.

Для дослідження продуктивності MySQL необхідно відкрити СУБД на сервері:

```
> mysql -u root -p
> Presnov
```

Далі створюється база даних і налаштовуються зв'язки, як описано в розділі 3.

4.4 Система захисту даних

Системи захисту даних містять комплексний захист від несанкціонованого доступу, підслуховування і крадіжки даних, а також антивірусний захист.

Захист від несанкціонованого фізичного доступу передбачає виділення окремого приміщення для web-сервера системи з метою контролю доступу та запобігання проникненню третіх осіб.

Антивірусне програмне забезпечення ESET Internet Security [11] використовується для захисту від вірусів і хакерських атак. До складу пакета входять антивірусний модуль, міжмережевий екран, антиспам і антишпигунський фільтр. Усі компоненти, що відповідають за захист від конкретних загроз, згруповані в меню "Захист". Модуль "Антишпигун" доповнено засобами захисту від несанкціонованого передавання конфіденційних даних. Файловий Антивірус заснований на трьох методах захисту: сигнатурах, евристиці та поведінці. Найресурсомісткішим є евристичний аналіз, за якого програму спочатку запускають у спеціальному захищеному середовищі, після чого їй дозволяється працювати у звичайних умовах. Таким чином, можна проаналізувати всі дії програми і заздалегідь зробити висновки про потенційні небезпеки з високою ймовірністю.

Евристичні аналізатори націлені на виявлення нових шкідливих програм, які не присутні в сигнатурах антивірусів. Такий аналіз може давати помилкові спрацьовування, однак він підвищує рівень безпеки мережі. Хоча евристичний аналіз особливо ефективний для захисту від поліморфних вірусів, які діють шляхом генерації програмного коду, процедура генерації коду не повинна бути постійною і змінюється при кожному новому зараженні.

Найважливішим модулем захисту є web-антивірус, який перевіряє вхідний і вихідний трафік під час взаємодії користувача з Інтернетом. Обмін трафіком між сервером та Інтернетом шифрується за допомогою алгоритму WPA-2/PSK, який на сьогоднішній день є найбільш безпечним типом шифрування. Шифрування самого трафіку здійснюється таким чином:

- генерація випадкових чисел, звичайних і асиметричних ключів;
- хешування даних;
- шифрування та дешифрування даних за допомогою симетричних та асиметричних ключів і перевірка хешу.

Політика безпеки паролів облікових записів: мінімальна довжина пароля становить 12 символів і містить щонайменше одну цифру та розділові знаки. Змінювати пароль кожні два місяці.

Для резервного копіювання бази даних використовується програма Handy Backup Server. Один модуль програми встановлюється на web-сервер, а другий - на сторонній комп'ютер, де регулярно проводиться резервне копіювання даних.

4.5 Дослідження web-платформи

Дослідження web-платформи є невід'ємним етапом її життєвого циклу. Під час дослідження експериментально визначається, на якому движку СУБД (підсистемі зберігання даних) найкраще працює розроблений web-платформи фотографа. Для цього буде створено базу даних на движках MyISAM і innodb відповідно до розділа 3 і виміряно об'єктивну міру продуктивності системи - час виконання 10, 100 і 1000 запитів до бази даних за такими критеріями:

- вибірка за первинним ключем;
- вибірка за назвою фотографії;
- додати фотографію.

Результати дослідження системи за критерієм вибірки по первинному ключу подано на рис. 4.5.

Результати дослідження системи на основі вибірки за назвою фотографії наведено на рис. 4.6.

Результати дослідження системи за критерієм додавання фотографії наведено на рис. 4.7.

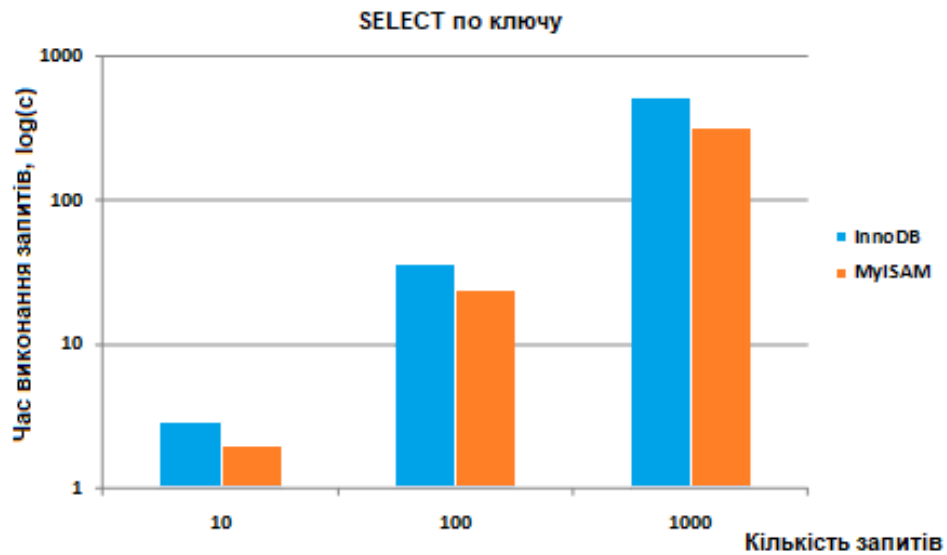


Рисунок 4.5 - Результати дослідження системи в логарифмічній шкалі за критерієм вибірки по первинному ключу

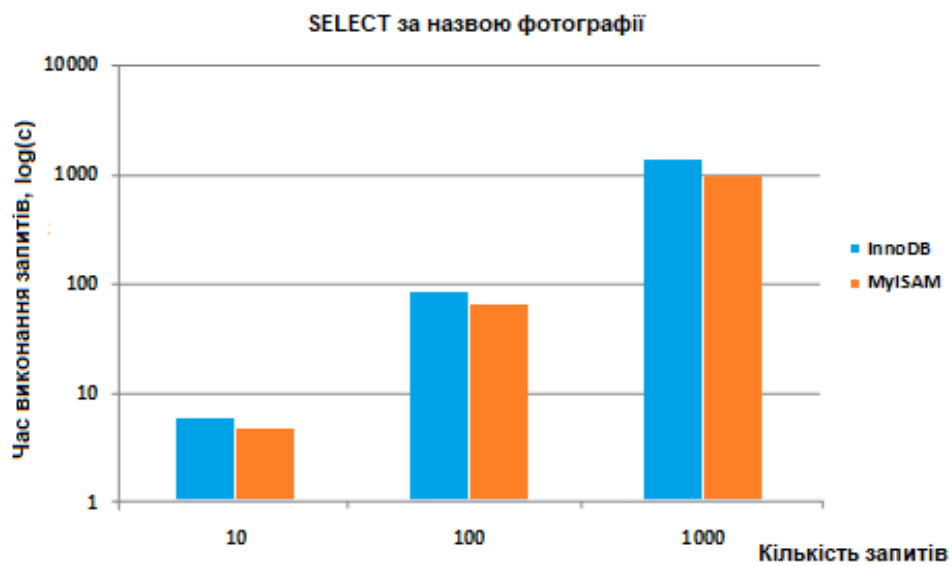


Рисунок 4.6 - Результати дослідження системи в логарифмічній шкалі за критерієм вибірки за назвою фотографій

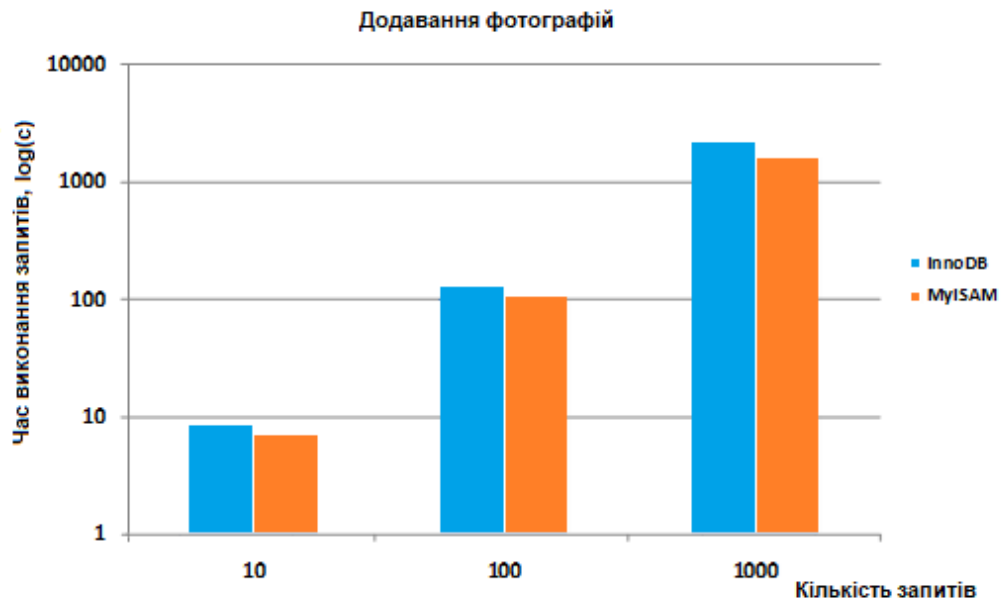


Рисунок 4.7 - Результати дослідження системи в логарифмічній шкалі за критерієм додавання фотографій

Як видно з рис. 4.5-4.7, продуктивність движок MyISAM для web-платформи фотографа значно вища, ніж у InnoDB. 56% за критерієм вибору первинного ключа, 44% за критерієм вибору за назвою фотографії та 42% за критерієм додавання фотографії. Водночас дії з базою даних фотоportалу не передбачають виконання транзакцій. Тому для розробленої системи рекомендується використовувати движок MyISAM.

4.6 Висновки по розділу

У цьому розділі було розглянуто такі питання:

- конфігурування апаратної частини web-платформи;
- встановлення та налаштування програмних компонентів системи, таких як Apache, MySQL та інтерпретатори PHP;

- для забезпечення безпеки роботи системи було впроваджено засоби захисту на базі програмних продуктів ESET Internet Security і Handy Backup Server;
- проведено дослідження web-платформи, за результатами якого рекомендовано використання движка СУБД MyISAM.

ВИСНОВКИ

У цій дипломній роботі за допомогою сучасних технологій і програмного забезпечення було досліджено комп'ютерну систему web-платформи фотографа. Було отримано такі результати:

- проаналізовано сучасні програмні засоби для розробки web-платформ, обґрунтовано вибір фреймворка Yii;
- обґрунтовано апаратну конфігурацію системи;
- web-платформа системи розроблений за допомогою інструментів php на фреймворку Yii;
- створили базу даних для сайту за допомогою mySQL;
- хостинг було обрано на базі запорізького провайдера "Воля": швидкість прийому українського трафіку - 100 Мбіт/с, іноземних каналів - 15 Мбіт/с;
- програмне забезпечення встановлено та налаштовано;
- ESET Internet Security та Handy Backup Server забезпечують безпеку та надійність системи;
- проведено дослідження системи та отримано кількісні оцінки продуктивності движка СУБД.

Web-платформа для фотографів дозволяє користувачам публікувати свої фотографії, а також коментувати та оцінювати творчість інших учасників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Учасники проектів Вікімедіа. Система керування вмістом – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Система_керування_вмістом.
2. Drupal - Універсальна система керування сайтом, збудована на модулях. Browser check, please wait ... URL: <https://astwellsoft.com/uk/blog/cms/drupal.html>.
3. Повний посібник (v2): Про Yii. Українська спільнота Yii Framework. URL: <https://yiiframework.com.ua/uk/doc/guide/2/>.
4. Учасники проектів Вікімедіа. Kohana – вікіпедія. Вікіпедія. URL: <https://uk.wikipedia.org/wiki/Kohana>.
5. Symfony, High Performance PHP Framework for Web Development. URL: <https://symfony.com/doc/current/index.html>
6. PHP + MySQL. URL: <https://sites.znu.edu.ua/webprog/lect/1222.ukr.html>.
7. СУБД (Системи управління базами даних) - що це, види та функції. Highload.today - медиа для разработчиков. URL: <https://highload.today/uk/subd-yaki-buvayut-yak-vibrati/>.
8. Feasel K. Optimizing SQL server backup performance. Berkeley, CA : Apress, 2021. URL: <https://doi.org/10.1007/978-1-4842-7194-0>
9. SQL Підручник. HTML CSS JavaScript PHP SQL. W3Schools українською. URL: <https://w3schoolsua.github.io/sql/index.html#gsc.tab=0>.
10. Блок безперебійного живлення APC. Режим доступу: URL <https://www.apc.com/ua/uk/product/SURT3000XLIX554/apc-smartups-rt-3000va-230v/>
11. ESET - офіційний сайт. антивірусні програми ісет в Україні. ESET. URL: <https://www.eset.com/ua/>
12. Mitchell L. J. PHP web services: apis for the modern web. O'Reilly Media, 2013. 118 p.
13. Ii C. R. P. Mastering yii. Packt Publishing, Limited, 2016.

14. EIRZNTU - Electronic Institutional Repository of "Zaporizhzhia Polytechnic" National University: Home. URL: http://eir.zntu.edu.ua/bitstream/123456789/2046/1/Fedoronchak_Methodica_instruction_i.pdf.

15. Структура CMS системи | Вісник Національного технічного університету «ХПІ». Серія: нові рішення у сучасних технологіях. Записки з романо-германської філології. URL: <http://rgnotes.onu.edu.ua/index.php/2079-5459/article/view/2413-4295.2020.04.12>.

16. SumDU Repository: Home. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/32215/1/WEB_Programming.pdf.

17. Киричек Г.Г. Автоматизоване тестування web-платформ з використанням Java та Selenium / Г.Г. Киричек, М.Ю. Тягунова, А.І. Курач // Інформаційні технології та суспільство. – 2022. – Вип.3. №1. – С.31-37.

18. Zakir A. Implementasi teknologi framework yii pada aplikasi berbasis web. InfoTekJar (jurnal nasional informatika dan teknologi jaringan). 2017. Vol. 2, no. 1. P. 45–48.

19. Lipskyi D. O. Development of a web application testing automation platform. Bulletin of taras shevchenko national university of kyiv. series: physics and mathematics. 2023. No. 1. P. 89–93.

20. MySQL and mSQL. Computers & mathematics with applications. 1999. Vol. 38, no. 11-12. P. 289.

21. CMS–системи управління контентом: навчальний посібник / О.В. Геніатуліна. – Миколаїв: Вид-во НГТУ, 2015, с. 50 – 55.