МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБГАЛЕРЕЇ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ DJANGO»**

Виконав: студент 4 курсу, групи 6.1210-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Д.С. Жировой
(ініціали та прізвище)

Керівник професор кафедри комп'ютерних наук, професор, д.т.н. Чопоров С.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри комп'ютерних наук, доцент, д.т.н. Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Жировому Данилу Сергійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебгалереї з використанням фреймворку Django

керівник роботи Чопоров Сергій Вікторович, д.т.н., професор
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Моделювання та розробка вебгалереї.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	07.02.2024	
3.	Обробка методичних та теоретичних	13.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	11.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2024	

Студент _____ Д.С. Жировий
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.В. Чопоров
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебгалереї з використанням фреймворку Django»: 62 с., 27 рис., 2 табл., 20 джерел.

БАЗА ДАНИХ, ВЕБГАЛЕРЕЯ, ВЕБЗАСТОСУНОК, ДЖАНГО, КОНТРОЛЕР, КОМПОНЕНТ, МОДЕЛЬ, ПІТОН, ФРЕЙМВОРК.

Об’єкт дослідження – процес розробки вебгалереї.

Мета роботи: розробка вебгалереї.

Методи дослідження – методи об’єктно-орієнтованого програмування, методи програмної інженерії.

Предмет дослідження – розробка вебгалереї з використанням фреймворку Django.

У роботі наведено етапи розробки вебгалереї з використанням фреймворку Django, виявлено ключові проблеми та застосовано нові методи для збору та аналізу вимог до програмного забезпечення. Описано предметну область. Розроблено вебгалерею з використанням фреймворку Django, яка дозволяє користувачам завантажувати, зберігати та переглядати зображення в реальному часі. Зростання попиту на інтерактивні онлайн сервіси робить важливим розв’язок таких задач, як розгортання та підтримка вебгалерей. Тому, розробка вебгалереї з використанням фреймворку Django є актуальною задачею.

SUMMARY

Bachelor’s qualifying paper “Development of a Web Gallery Using the Django Framework”: 62 pages, 27 figures, 2 tables, 20 references.

DATABASE, WEB GALLERY, WEB APPLICATION, DJANGO, CONTROLLER, COMPONENT, MODEL, PYTHON, FRAMEWORK.

Object of research – the process of developing a web gallery.

Purpose of the work: development of a web gallery.

Research methods – object-oriented programming methods, software engineering methods.

Subject of research – development of a web gallery using the Django framework.

The work outlines the stages of developing a web gallery using the Django framework, identifies key issues, and applies new methods for gathering and analyzing software requirements. The domain area is described. A web gallery has been developed using the Django framework, which allows users to upload, store, and view images in real-time. The growing demand for interactive online services makes solving such tasks as deploying and maintaining web galleries important. Therefore, developing a web gallery using the Django framework is a relevant task.

ЗМІСТ

ВСТУП

Розвиток технологій значно трансформує різні аспекти нашого життя, включаючи способи комунікації та надання послуг через Інтернет. У сучасному цифровому світі важливо, щоб вебсайти пропонували ефективні та зручні інструменти для взаємодії та обміну інформацією. Сучасні споживачі очікують миттєвої взаємодії та швидких відповідей на свої запити. Це особливо актуально для бізнесів та організацій, які прагнуть забезпечити високий рівень обслуговування користувачів та покращити користувацький досвід на своїх вебресурсах.

Вебсайти сьогодні є ключовою складовою будь-якого бізнесу чи платформи, слугуючи основним засобом комунікації з клієнтами та відвідувачами. В умовах посиленої конкуренції компаніям необхідні рішення, які дозволяють реальну взаємодію з користувачами в режимі реального часу, відповідаючи на їхні запитання та надаючи необхідну допомогу. Використання інтерактивних вебзастосунків дозволяє досягти цього, забезпечуючи миттєве оновлення вмісту та передачу інформації.

Цей проєкт присвячений розробці вебгалереї з використанням потужного фреймворку Django. Метою проєкту є створення ефективного інструменту для швидкої та плавної взаємодії з відвідувачами вебсайтів. Django забезпечує високу продуктивність, гнучкість та інтерактивність, що значно покращує користувацький досвід, оскільки забезпечує швидку передачу повідомлень та оновлення контенту без необхідності перезавантаження сторінки.

В рамках цього проєкту будуть розроблені необхідні компоненти та інтерфейс для вебгалереї. Використання Django дозволить забезпечити швидке і ефективне передавання даних між користувачем і сервером. Крім того, будуть розглянуті можливості зберігання даних та інтеграції з базами даних для забезпечення зручного керування зображеннями та метаданими.

Використання Django забезпечить надійну серверну частину та інтуїтивно зрозумілий та інтерактивний інтерфейс.

Особливу увагу буде приділено питанням безпеки та захисту даних, оскільки вебгалереї часто включають обмін конфіденційною інформацією. Впровадження сучасних методів шифрування та аутентифікації допоможе забезпечити безпечне середовище для користувачів.

Цей проєкт має велике значення, оскільки він дозволяє покращити якість обслуговування користувачів та відвідувачів вебсайту, забезпечуючи швидку та ефективну комунікацію. Використання Django дозволяє створити інтерактивне середовище, яке задовольняє потреби користувачів та сприяє підвищенню їх задоволення від взаємодії з вебсайтом. Запровадження такого рішення також може сприяти підвищенню лояльності користувачів, збільшенню рівня їх задоволення та покращенню загального іміджу організації.

Таким чином, розробка вебгалереї на базі фреймворку Django є важливим кроком у напрямку створення ефективних та інноваційних вебзастосунків, що відповідають сучасним вимогам ринку та потребам користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЙ

1.1 Огляд існуючих рішень для вебгалерей

У сучасному цифровому світі вебгалереї стали важливим інструментом для зберігання, демонстрації та спільного використання зображень. Існує безліч рішень для створення та керування вебгалереями, кожне з яких має свої особливості, переваги та недоліки. Розглянемо найпоширеніші з них:

- WordPress з плагінами для галерей;
- Flickr;
- SmugMug;
- Zenfolio;
- Google Photos;
- Piwigo;
- Django-based Solutions.

WordPress є однією з найпопулярніших платформ для створення вебсайтів, що відзначається своєю гнучкістю та багатством плагінів (див. рис. 1.1). Для створення вебгалерей WordPress пропонує безліч плагінів, які надають різноманітні функції для організації, показу та управління зображеннями. Серед найвідоміших плагінів можна виділити NextGEN Gallery, Envira Gallery та FooGallery [3].

Розглянемо основні **переваги** WordPress:

- легкість використання: WordPress має інтуїтивно зрозумілий інтерфейс, який дозволяє легко встановлювати та налаштовувати плагіни для галерей навіть користувачам без технічних знань;
- велика кількість готових шаблонів і плагінів: існує безліч безкоштовних і платних плагінів для галерей, які можна налаштувати під свої потреби – це дозволяє швидко створювати привабливі та функціональні вебгалереї;

– спільнота підтримки: WordPress має велику та активну спільноту, яка надає підтримку через форуми, блоги та навчальні матеріали – це полегшує вирішення проблем та покращує досвід користування платформою;
– постійне оновлення: плагіни та сама платформа WordPress постійно оновлюються, що забезпечує безпеку та додавання нових функцій.

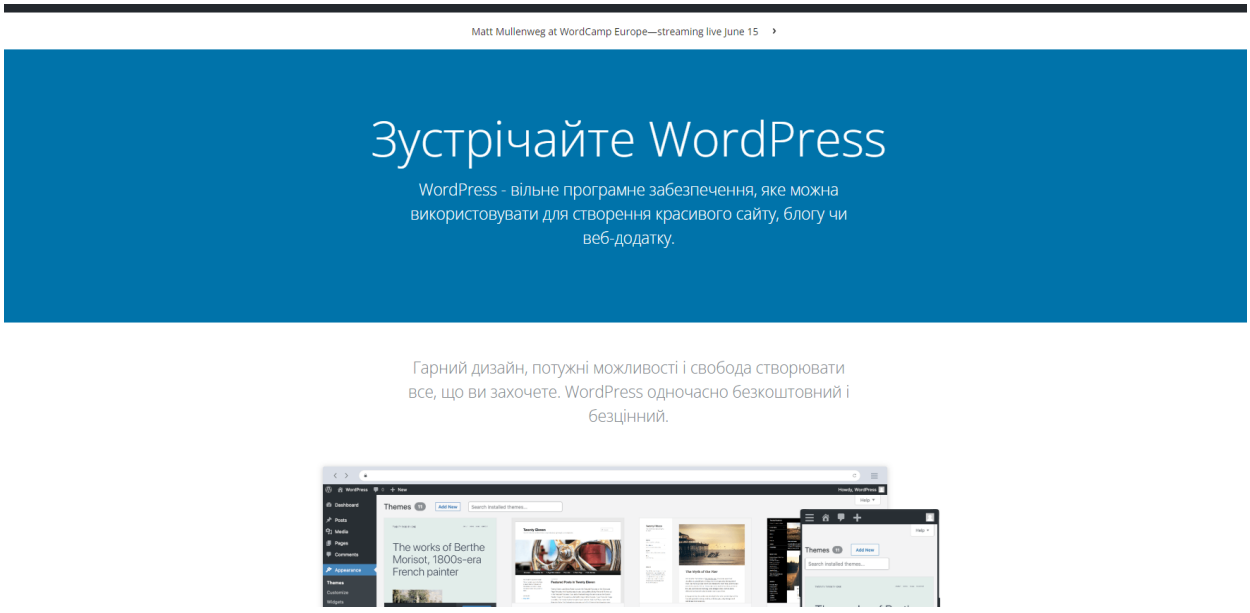

Рисунок 1.1 – WordPress [1]

Також давайте розглянемо основні **недоліки** WordPress:

– обмежена гнучкість у налаштуванні: хоча WordPress пропонує багато можливостей, деякі специфічні або дуже нестандартні вимоги можуть бути складними для реалізації – в такому випадку може знадобитися додаткове програмування або використання спеціалізованих плагінів;
– залежність від сторонніх плагінів: багато функцій галереї реалізовані за допомогою сторонніх плагінів, що може створити проблеми сумісності та залежності від оновлень цих плагінів;
– потенційні проблеми з продуктивністю: додавання великої кількості плагінів може негативно впливати на швидкість завантаження сайту

та загальну продуктивність – це особливо актуально для вебсайтів з великою кількістю зображень.

Використання WordPress з плагінами для створення вебгалерей є зручним та швидким способом отримати функціональний та привабливий результат. Це рішення підходить для користувачів різного рівня технічної підготовки, надаючи можливість швидко налаштувати та керувати вебгалереєю без необхідності глибоких знань програмування. Однак, для специфічних або масштабних проєктів можуть знадобитися додаткові налаштування та оптимізація.

Flickr є однією з найбільших та найпопулярніших платформ для зберігання, організації та обміну фотографіями. Вона пропонує користувачам можливість створювати альбоми та галереї зображень, а також ділитися ними з широкою спільнотою користувачів. Flickr забезпечує зручні інструменти для керування зображеннями та надає можливість налаштовувати конфіденційність для кожного альбому або зображення окремо (див. рис. 1.2).

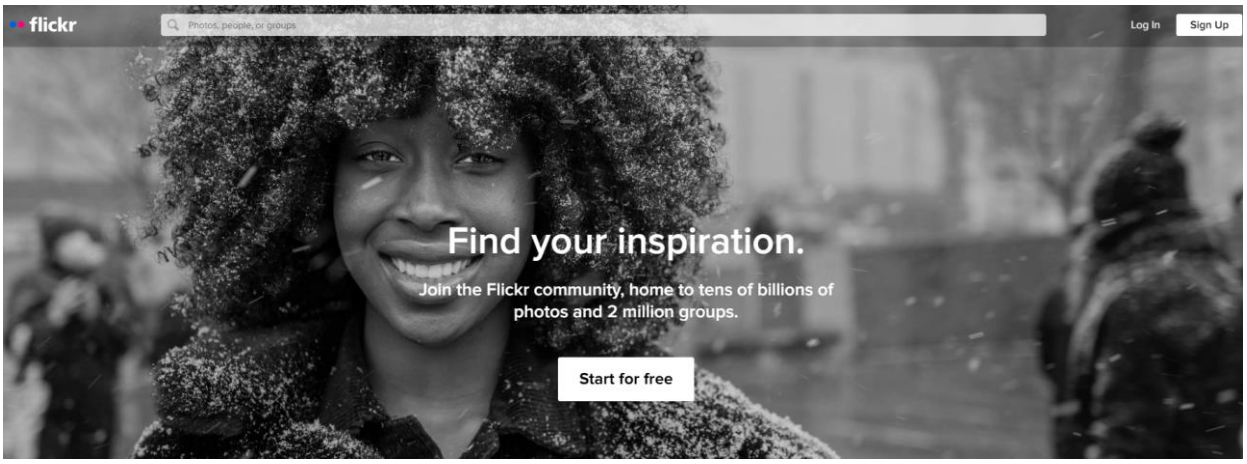

Рисунок 1.2 – Flickr [3]

Переваги:

– велика спільнота користувачів: Flickr має мільйони активних користувачів, що дозволяє легко ділитися фотографіями з великою аудиторією та знаходити нові цікаві зображення;
– безкоштовне використання: Flickr пропонує безкоштовний акаунт з

обмеженнями на кількість зображень та розмір завантажень, що дозволяє зберігати та обмінюватися фотографіями без додаткових витрат;

– організація та теги: гористувачі можуть створювати альбоми, додавати теги та описи до своїх зображень, що спрощує пошук та організацію фотографій;
– інтеграція з соціальними мережами: легке поширення зображень на інші платформи, такі як Facebook, Twitter, і Tumblr;
– мобільний доступ: доступ до Flickr можливий з мобільних пристроїв за допомогою офіційних додатків для iOS та Android.

Недоліки:

– обмеження на безкоштовний акаунт: безкоштовні користувачі мають обмеження на кількість завантажених фотографій (до 1000 зображень) та розмір кожного зображення (до 200 МБ);
– менша гнучкість у налаштуванні дизайну галереї: на відміну від власноруч створених сайтів, можливості налаштування вигляду галереї на Flickr є обмеженими;
– залежність від сервісу: використання стороннього сервісу означає залежність від його політики, умов користування та можливих змін в роботі платформи.

SmugMug – це платформа для зберігання, організації та спільного використання фотографій, орієнтована на професійних фотографів та фотоентузіастів. Запущена в 2002 році, SmugMug відома своїм акцентом на високу якість зображень та надійні можливості зберігання [6].

Функціональні можливості:

– зберігання фотографій та відео: SmugMug пропонує необмежене зберігання фотографій і відео, з підтримкою форматів високої роздільної здатності;
– організація контенту: користувачі можуть організовувати свої фотографії в галереї та альбоми, додавати теги, ключові слова та описи;

– налаштування приватності: SmugMug надає детальні налаштування приватності, дозволяючи користувачам контролювати доступ до своїх фотографій та галерей;
– редагування зображень: вбудовані інструменти редагування дозволяють користувачам покращувати свої фотографії безпосередньо на платформі;
– продаж фотографій: SmugMug пропонує можливості для продажу фотографій і друкованих копій, що робить його популярним серед професійних фотографів;
– інтеграція з іншими платформами: платформа підтримує інтеграцію з іншими сервісами та соціальними мережами, що дозволяє легко ділитися контентом.

Переваги:

– висока якість зображень: SmugMug зберігає фотографії у високій роздільній здатності, забезпечуючи відмінну якість зображень для перегляду та друку;
– потужні інструменти організації: можливості організації контенту дозволяють легко керувати великими колекціями фотографій;
– гнучкість дизайну галерей: SmugMug надає широкий вибір шаблонів та варіантів налаштування дизайну галерей, що дозволяє створювати унікальні та привабливі сторінки;
– підтримка клієнтів: SmugMug відома своєю високоякісною підтримкою клієнтів, що допомагає користувачам вирішувати будь-які питання або проблеми.

Недоліки:

– ціна: SmugMug є платною платформою, і її тарифи можуть бути високими для деяких користувачів, особливо для тих, хто не планує продавати свої фотографії.
– крута крива навчання: лля нових користувачів може знадобитися час, щоб освоїти всі функції та можливості платформи.

SmugMug є чудовим вибором для професійних фотографів та ентузіастів, які шукають платформу з високою якістю зображень, потужними інструментами організації та можливостями продажу фотографій (див. рис. 1.3).

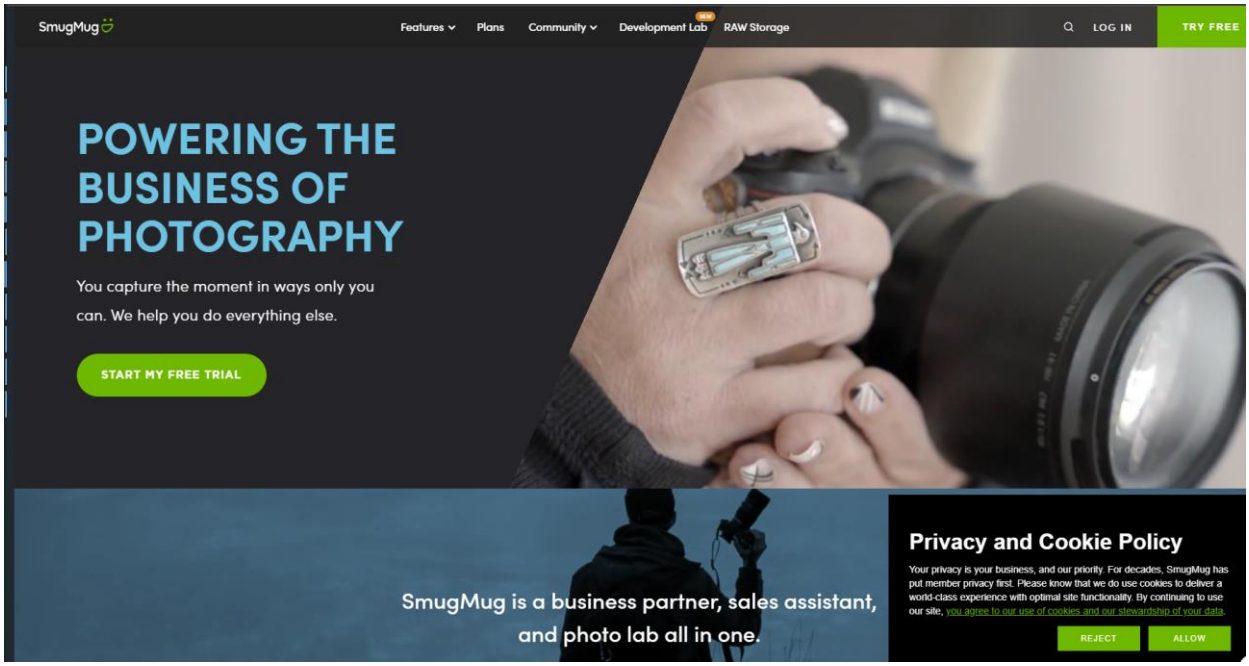

Рисунок 1.3 – SmugMug

Zenfolio – це платформа для зберігання, організації та продажу фотографій, створена спеціально для професійних фотографів та фотоентузіастів. Запущена у 2006 році, Zenfolio надає широкий спектр інструментів для управління фотографіями, створення портфоліо та продажу зображень [7].

Функціональні можливості:

– зберігання фотографій та відео: Zenfolio пропонує необмежене зберігання для фотографій і відео з підтримкою різних форматів і роздільних здатностей;

– організація контенту: користувачі можуть створювати галереї та альбоми, додавати теги, описи та ключові слова для легкого пошуку та навігації;

– інструменти редагування: вбудовані інструменти редагування

дозволяють покращувати зображення безпосередньо на платформі.

- продаж фотографій: Zenfolio надає можливості для продажу фотографій, друкованих копій та фототоварів, що особливо привабливо для професійних фотографів;
- налаштування приватності: платформа дозволяє користувачам налаштовувати рівень доступу до своїх галерей, забезпечуючи приватність або публічність за потреби;
- маркетингові інструменти: Zenfolio пропонує інструменти для маркетингу та SEO, що допомагає фотографам просувати свої роботи та залучати більше клієнтів;
- мобільні додатки: платформа підтримує мобільні додатки для зручного управління контентом та взаємодії з клієнтами на ходу.

Переваги:

- професійний вигляд портфоліо: Zenfolio надає широкий вибір шаблонів для створення привабливих та професійних портфоліо;
- гнучкість налаштувань: користувачі можуть налаштовувати дизайн та функціональність своїх галерей відповідно до своїх потреб і стилю;
- інтеграція з лабораторіями друку: платформа інтегрується з професійними лабораторіями друку, що дозволяє автоматизувати процес замовлення та доставки друкованих фотографій;
- підтримка клієнтів: Zenfolio надає високоякісну підтримку клієнтів, включаючи навчальні матеріали, вебінари та технічну допомогу.

Недоліки:

- ціна: як і у випадку з іншими професійними платформами, Zenfolio є платною послугою, що може бути недешевою для новачків або аматорів;
- складність інтерфейсу: деякі користувачі можуть знайти інтерфейс занадто складним для освоєння, особливо на початкових етапах використання.

Zenfolio є потужним інструментом для професійних фотографів, які шукають платформу для управління своїм портфоліо, продажу фотографій та покращення своєї онлайн-присутності. Завдяки широкому набору функцій, гнучким налаштуванням та інтеграціям з друкарнями, Zenfolio допомагає фотографам ефективно керувати своїм бізнесом та залучати нових клієнтів (див. рис. 1.4).

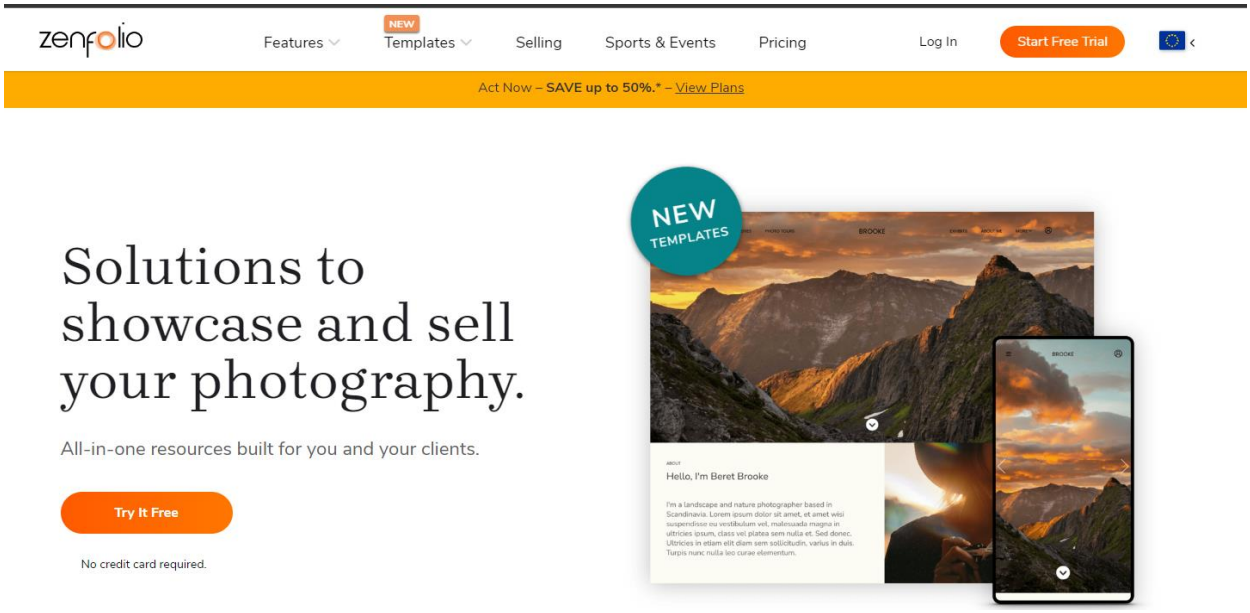

Рисунок 1.4 – Zenfolio [9]

Google Photos – це популярний сервіс для зберігання, організації та обміну фотографіями та відео, запущений Google у 2015 році (див. рис. 1.5). Він пропонує широкий спектр функцій, що роблять його зручним для використання як аматорами, так і професіоналами [5].

Функціональні можливості:

– необмежене зберігання: Google Photos пропонує безкоштовне необмежене зберігання фотографій і відео у високій якості (з певними обмеженнями щодо роздільної здатності та стиснення), а також платні плани для зберігання в оригінальній якості;

– автоматичне сортування та організація: сервіс автоматично сортує фотографії за датою, місцем і виявляє обличчя, об’єкти та сцени для легкого пошуку;

– редагування фотографій: вбудовані інструменти для редагування дозволяють покращувати фотографії, коригувати освітлення, кольори та додавати фільтри;
– створення альбомів та колажів: користувачі можуть створювати альбоми, а також автоматично генерувати колажі, анімації та фільми з фотографій;
– спільне використання та обмін: легкий обмін фотографіями та альбомами з іншими користувачами через посилання або соціальні мережі;
– автоматичне резервне копіювання: сервіс пропонує автоматичне резервне копіювання фотографій з мобільних пристроїв та комп’ютерів;
– інтелектуальний пошук: потужні алгоритми пошуку дозволяють знаходити фотографії за обличчями, предметами та місцями без необхідності тегування.

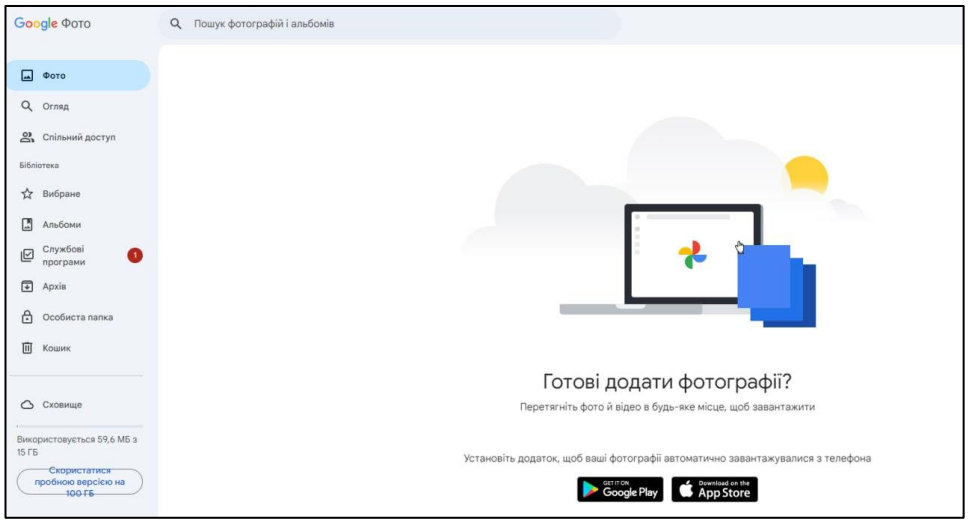

Рисунок 1.5 – Google Photos [6]

Переваги:

– інтеграція з іншими сервісами Google: Google Photos інтегрується з іншими сервісами Google, такими як Google Drive, Gmail і Google Assistant, що забезпечує зручність використання;

– доступність на різних платформах: сервіс доступний на Android, iOS і через вебінтерфейс, що дозволяє легко доступатися до фотографій з будь-якого пристрою;
– швидкість та надійність: завдяки інфраструктурі Google, сервіс працює швидко та надійно, з мінімальними затримками та високою доступністю;
– інтелектуальні функції: автоматичне розпізнавання облич, предметів і сцен, а також можливість створення автоматичних альбомів і колажів.

Недоліки:

– обмеження на безкоштовне зберігання: з листопада 2020 року Google обмежив безкоштовне зберігання до 15 ГБ спільного місця з Google Drive і Gmail, що може бути недостатнім для активних користувачів;
– політика конфіденційності: використання Google Photos може викликати занепокоєння щодо конфіденційності даних, оскільки Google збирає та аналізує фотографії для поліпшення своїх сервісів;
– обмежені можливості організації: хоча автоматичне сортування є корисним, користувачі можуть відчувати обмеження в ручному управлінні та організації фотографій.

Google Photos є потужним інструментом для зберігання, організації та редагування фотографій, який пропонує численні зручні функції. Він особливо підходить для користувачів, які вже активно користуються іншими сервісами Google, і шукають надійне рішення для управління своїми фотографіями [15].

Piwigo – це відкритий і безкоштовний вебдодаток для організації, управління та обміну фотографіями. Запущений у 2002 році, Piwigo пропонує багатий набір функцій і є популярним рішенням як для приватних осіб, так і для організацій, які потребують надійної системи управління зображеннями [5].

Функціональні можливості:

– організація фотографій: Piwigo дозволяє організовувати фотографії в категорії, альбоми, теги та колекції, забезпечуючи гнучкість у

структурі бібліотеки зображень;

- розширені можливості завантаження: можливість завантаження фотографій через вебінтерфейс, FTP, мобільні додатки або синхронізацію з іншими сервісами;
- користувацькі права та управління доступом: Piwigo надає потужні інструменти для налаштування прав доступу користувачів і груп, що дозволяє контролювати, хто може переглядати або редагувати контент;
- плагіни та розширення: Piwigo підтримує велику кількість плагінів, що дозволяють додавати нові функції та інтеграції, такі як підтримка відео, Google Maps, додаткові формати файлів тощо;
- шаблони та теми: користувачі можуть змінювати зовнішній вигляд своєї галереї за допомогою різних шаблонів та тем, що легко налаштовуються;
- локалізація: Piwigo підтримує численні мови, що робить його доступним для міжнародних користувачів;
- автоматичні оновлення: платформа регулярно оновлюється, забезпечуючи безпеку та нові функції для користувачів.

Переваги:

- безкоштовне та відкрите програмне забезпечення: Piwigo є повністю безкоштовним і відкритим, що дозволяє користувачам змінювати та налаштовувати його відповідно до своїх потреб;
- гнучкість та масштабованість: підходить для різних типів користувачів, від індивідуальних фотографів до великих організацій з тисячами зображень;
- спільнота та підтримка: активна спільнота користувачів і розробників надає підтримку, документацію та ресурси для нових користувачів;
- інтеграція з іншими сервісами: можливість інтеграції з популярними вебсервісами та додатками, такими як WordPress, Lightroom, і інші.

Недоліки:

– необхідність самостійного хостингу: Piwigo вимагає самостійного налаштування та управління сервером, що може бути складним для користувачів без технічних знань;
– обмеженість вбудованих функцій у порівнянні з комерційними рішеннями: деякі розширені функції можуть бути доступні лише через плагіни, що вимагає додаткових налаштувань;
– інтерфейс користувача: хоча Piwigo постійно оновлюється, його інтерфейс може здаватися застарілим або менш інтуїтивним порівняно з сучасними комерційними продуктами.

Piwigo є надійним та гнучким рішенням для управління фотографіями, особливо підходящим для користувачів, які шукають безкоштовну та налаштовану платформу з активною спільнотою підтримки. Його багатий набір функцій та можливість розширення роблять його популярним вибором для різних сценаріїв використання, від особистих фотоколекцій до професійних галерей (див. рис. 1.6) [18].

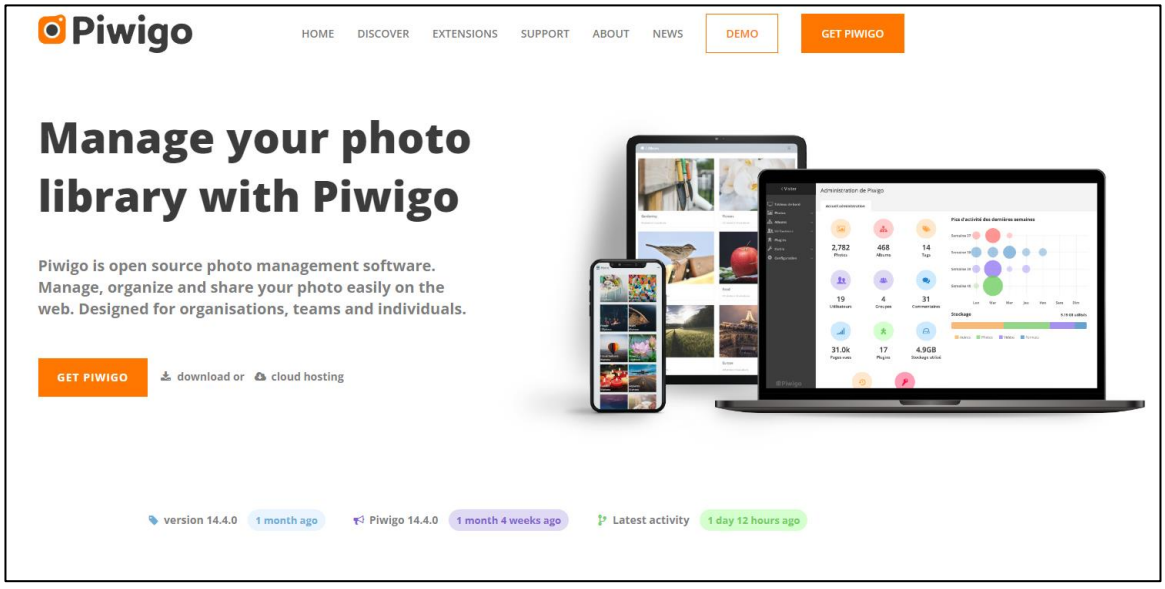

Рисунок 1.6 – Piwigo [19]

Django, як високорівневий фреймворк для веброзробки на Python, забезпечує потужну основу для створення вебгалерей. Існує кілька

популярних рішень для вебгалерей, розроблених за допомогою Django, які використовуються як приватними особами, так і організаціями для ефективного управління та обміну зображеннями [19].

Функціональні можливості:

– організація контенту: Django дозволяє створювати моделі для категоризації та організації зображень, включаючи альбоми, теги, колекції та інші структури;
– розширені можливості завантаження: підтримка різних методів завантаження файлів, включаючи вебінтерфейс, API, FTP та інтеграцію з хмарними сервісами;
– управління користувачами: вбудована система управління користувачами Django дозволяє легко створювати ролі та дозволи, забезпечуючи контроль доступу до контенту;
– розширення та інтеграції: Django має багату екосистему пакетів та бібліотек, які можна інтегрувати для додавання нових функцій, таких як підтримка відео, мапування зображень, обробка зображень та інше;
– шаблони та теми: використання Django шаблонів для створення кастомізованих інтерфейсів користувача, які можуть бути адаптовані під потреби конкретного проєкту;
– міжнародна підтримка: Django має вбудовану підтримку локалізації та інтернаціоналізації, що робить його зручним для створення багатомовних додатків.

Переваги:

– гнучкість та розширюваність: Django надає повну свободу у визначенні моделі даних, логіки додатка та інтерфейсу користувача, що дозволяє створювати високонастроювані рішення;
– вбудовані адміністративні інструменти: Django має потужний адміністративний інтерфейс, який автоматично генерується на основі моделей, забезпечуючи легке управління контентом;

– безпека та продуктивність: Django включає ряд вбудованих функцій для забезпечення безпеки та оптимізації продуктивності, таких як захист від CSRF, XSS, SQL ін’єкцій, кешування та інше;
– активна спільнота: Django має активну спільноту розробників та велику базу знань, що включає документацію, посібники, форуми та інші ресурси.

Недоліки:

– складність налаштування та підтримки: хоча Django пропонує багато можливостей, його налаштування та підтримка можуть вимагати значних технічних знань, особливо для великих проєктів;
– потенційно довший час розробки: через гнучкість та налаштованість, розробка на Django може зайняти більше часу у порівнянні з використанням спеціалізованих платформ або конструкторів;
– обмежена готова функціональність: Django є фреймворком, а не готовим рішенням, тому деякі функції, доступні у комерційних продуктах, потрібно буде розробляти самостійно.

Приклади Django-based рішень для вебгалерей:

– Mezzanine and Cartridge: Mezzanine – це потужна CMS на базі Django, яка може бути розширена для створення вебгалерей (Cartridge, розширення для Mezzanine, додає функціональність електронної комерції, що дозволяє створювати інтегровані галереї з можливістю продажу зображень);
– Wagtail: Wagtail – це сучасна, потужна CMS на базі Django, яка також може бути використана для створення галерей зображень (завдяки своїй гнучкості та можливості налаштування, Wagtail дозволяє легко організовувати контент та управляти медіафайлами);
– Django Photologue: Photologue – це популярне рішення для створення та управління фотогалереями на базі Django (включає функціонал для організації, завантаження та відображення зображень, підтримку галерей та слайдшоу);

– Django-Filer: Django-Filer – це бібліотека для управління файлами та зображеннями у проєктах на базі Django (вона забезпечує зручний інтерфейс для завантаження, організації та використання медіа контенту).

Django-based рішення надають високу гнучкість та потужність для створення вебгалерей, дозволяючи розробникам налаштовувати та розширювати функціональність відповідно до потреб проєкту. Хоча налаштування може вимагати технічних знань, результати часто виправдовують вкладені зусилля, забезпечуючи надійні та масштабовані системи управління зображеннями (див. рис. 1.7) [10].

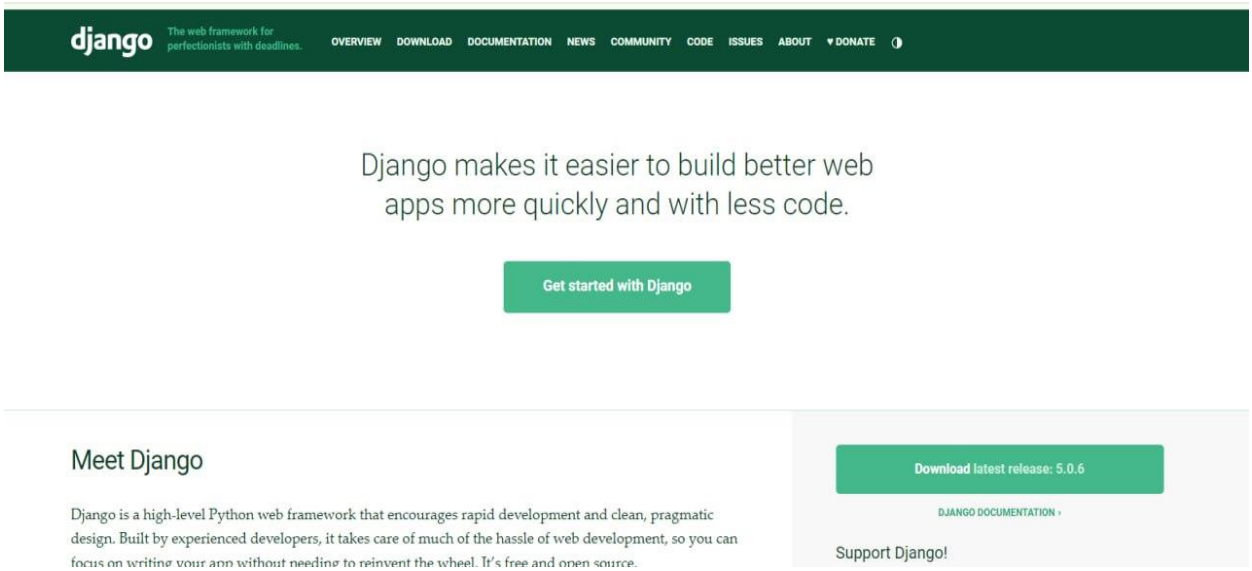

Рисунок 1.7 – Django

1.2 Порівняння фреймворків для розробки вебгалереї

На ринку існує кілька потужних фреймворків для розробки вебгалерей, кожен з яких має свої особливості і переваги. Розвиток технологій веброзробки відбувається досить швидко – сьогодні мало хто створює додатки і вебсайти без допомоги спеціалізованих інструментів, зокрема, фреймворків. Однак, крім основного вибору, що стосується мови програмування, часто існує

потреба в використанні ряду додаткових інструментів і, звичайно ж, фреймворка [10].

Django – це високорівневий безкоштовний вебфреймворк на базі Python, який був вперше представлений громадськості у 2005 році. Він відзначається високою продуктивністю команд завдяки готовим компонентам, які забезпечують більшість бекенд-розробки. Django спрощує створення бізнес-логіки і унікальних частин бекенда, а його код вважається безпечним і допомагає уникнути типових помилок [20].

Однією з найбільш відомих особливостей Django є його масштабованість, підтверджена численними популярними проєктами, що витримують значні навантаження користувачів. Django часто використовується для створення корпоративних систем, соціальних мереж, фінансових платформ та електронної комерції [9].

Ключові особливості Django включають:

– сумісність з усіма популярними операційними системами;
– швидкий процес розробки складних рішень за архітектурою MVT;
– вбудовані пакети шифрування для безпечної передачі даних;
– багатство бібліотек і інструментів для полегшення роботи розробників;
– велике спільнота розробників;
– SEO-приязність і доступність декораторів;
– масштабованість;
– деталізована документація;
– зручне взаємодія з базами даних без необхідності знання SQL;
– відсутність необхідності імпорту сторонніх бібліотек;
– можливість швидкої розробки багатомовних вебсайтів.

Це лише невеликий перелік особливостей Django, тому для остаточного вибору, чи підходить цей фреймворк саме для вашого проєкту, рекомендується проконсультуватися з вашою командою розробників [14].

Django став основою таких всесвітньо відомих проєктів, як:

– Instagram;
– NASA;
– Spotify;
– YouTube;
– DropBox;
– Pinterest**.**

Вперше світ дізнався про Laravel в 2011 році як про PHP-фреймворк із простим і елегантним синтаксисом, що мінімізує необхідність у програмуванні тривіальних частин проєкту і, таким чином, прискорює його розгортання. Laravel дозволяє використовувати PHP для розробки повного стеку (зокрема, через спільне використання з Livewire) та JavaScript (React або Vue для клієнтської частини, підключення до Laravel Inertia). Крім того, Laravel дозволяє будувати API для додатків, написаних на Next.js. Загалом, це передовий фреймворк для розробки бекенду, який має низький поріг входу і сприяє масштабуванню застосунків і вебсайтів завдяки великій бібліотеці пакетів і пропрієтарним технологіям Laravel Octane і Laravel Vapor [12].

Laravel є відмінним вибором для розробки масштабних рішень на рівні підприємства, що забезпечується, насамперед, повною сумісністю з AWS Lambda та наявністю достатньої кількості вбудованих пакетів і інструментів, які дозволяють динамічно підвищувати продуктивність додатків і вебсайтів [6].

Давайте, як і в попередньому випадку, розглянемо основні особливості цього фреймворка:

– інноваційна система шаблонів (шаблонізатор Blade), що прискорює процес розробки;
– підтримка архітектури MVC через вбудовані функції;
– відсутність необхідності знати SQL для встановлення зв’язків із базами даних;
– підтримка безлічі об’єктно-орієнтованих бібліотек і модульних функцій;

- вбудований інструмент Artisan для автоматизації стандартних завдань програмування;
- спрощене модульне тестування;
- спрощена реалізація систем авторизації;
- автоматичний захист від SQL-ін’єкцій, а також підробки міжсайтових запитів і сценаріїв;
- балансування навантажень;
- низький вхідний поріг і детальна документація.

Щоб зрозуміти, які саме особливості Laravel можуть бути корисними для вашого рішення, краще звернутися безпосередньо до вашої команди розробки для вибору оптимального технологічного стека [19].

Тепер давайте розглянемо порівняння Laravel і Django за найбільш важливими для бізнесу критеріями.

Популярність: Django і Laravel мають дуже чуйну і географічно розосереджену онлайн-спільноту, яка буде готова відповісти на будь-яке питання. Також наявність великої чисельності користувачів сприяє стабільному зростанню кількості бібліотек і загалом, постійному розвитку екосистеми цих фреймворків. Згідно з рейтингами GitHub, Django має 76.1 тисяч зірочок, водночас як Laravel – 76.3. Ця різниця не надто суттєва, а отже, і популярність обох рішень перебуває приблизно на однаковому рівні [7].

Продуктивність: Laravel виділяється серед сучасних фреймворків завдяки своїй продуктивності, яка часто недооцінюється порівняно з Django. Завдяки своїй архітектурі, яка оптимізована під високонавантажені завдання, Laravel демонструє чудові результати в управлінні ресурсоємними операціями. Його сучасний підхід до асинхронного програмування та підтримка різних технік кешування дають змогу масштабувати додатки без значних втрат у швидкості роботи;крім того, Laravel постійно оновлюється з урахуванням потреб розробників і кінцевих користувачів, що включає підтримку новітніх версій PHP, які пропонують поліпшення в продуктивності та безпеці. Також, завдяки гнучкості фреймворка, розробники можуть легко

інтегрувати сторонні компоненти, що ще більше підвищує продуктивність додатків [10].

Архітектура: Django заснований на MVT (Model-View-Template) архітектурі, що забезпечує підвищену швидкість розробки і простоту внесення модифікацій у програмний код. Що стосується Laravel, то в його основі лежить MVC (Model-View-Controller) архітектура, яка чудово підходить для масштабних програмних рішень завдяки своїй структурованості [16].

Масштабованість: для великих проєктів, які швидко зростають, Laravel є найкращим вибором, через свою гнучкість у масштабуванні та оптимізації. Завдяки інтеграції з різними системами кешування та підтримці балансування навантаження, Laravel дає змогу легко масштабувати додатки відповідно до потреб бізнесу. Крім того, його екосистема з численними пакетами та великою спільнотою сприяє швидкій адаптації до мінливих вимог ринку, що робить його ідеальним варіантом для комплексних проєктів, які потребують регулярного масштабування та оновлення.

Підтримувані бібліотеки: Django має дуже багатий набір різноманітних пропрієтарних бібліотек, а також декоратори, SEO-інструменти та підтримку інтеграції з безліччю сторонніх рішень (зокрема, ви можете інтегрувати проєкт на Django з окремими модулями, написаними на JavaScript). Що стосується Laravel, він має багато об’єктно-орієнтованих бібліотек і модульних функцій, що прискорюють процес розробки. Зрештою, оцінюючи цей параметр, слід орієнтуватися на вимоги саме вашого рішення та перевіряти наявність відповідних інструментів у конкретному фреймворку.

Безпека: що для Laravel, що для Django, безпека є пріоритетною особливістю. В обох фреймворках вона забезпечується через вбудовані функції захисту від SQL-ін’єкцій і клікджекінгу, а також підробки міжсайтових запитів і сценаріїв (тобто, XSS і CSRF атак).

Тестування: для спрощення процедури тестування у Django є вбудована можливість запуску автотестів. Що стосується Laravel, він підтримує модульне тестування, даючи змогу одночасно запускати відразу кілька тестів.

Вартість розробки: оскільки вартість розробки залежить насамперед від складності проєкту, оцінити цей параметр з погляду того чи іншого фреймворка можна за тим, як швидко розробники зможуть впоратися з однаковим завданням на кожному з фреймворків. І тут безперечним лідером є Laravel, завдяки його мінімалістичному синтаксису, вбудованому шаблонізатору і можливості одночасно запускати кілька тестів.

Швидкість розробки: якщо йдеться про стартапи та інші проєкти, що потребують якнайшвидшого запуску, варто звернути увагу саме на Laravel. Цей фреймворк забезпечує максимально чистий програмний код і, завдяки наявності великої кількості вбудованих функцій та інструментів автоматизації, здатний звести роботу розробників до мінімуму. Водночас Django також пропонує розробникам безліч готових компонентів і повністю укомплектовану екосистему, тому розробка з використанням цього фреймворка також відбувається досить швидко [17].

Ми вже зазначали вище, що вибір фреймворку – чи то Django, чи то Laravel – найкраще узгоджувати з вашою командою розробки. Водночас, ви можете самостійно отримати цінні інсайти, які дадуть вам правильний напрямок для вибору технологічного стека вашого майбутнього рішення (див. рис. 1.8).

Рисунок 1.8 – Django vs Laravel [11]

Отже, давайте розглянемо, які параметри потрібно враховувати насамперед, щоб зробити вибір, який згодом позитивно вплине на успіх вашого проєкту [13].

Специфіка вимог до проєкту: спочатку вам необхідно визначити функціональні та нефункціональні вимоги до проєкту. Далі з'ясуйте, чи існують специфічні вимоги до продуктивності, масштабованості, безпеки та інших важливих характеристик вашого програмного рішення.

Обмеження та спеціальні вимоги до використання: не менш важливо коректно визначити технічні та нетехнічні обмеження і вимоги щодо використання вашого програмного рішення. До них можна віднести бюджет, приблизну довжину життєвого циклу, а також наявні ресурси та доступність фахівців.

Різниця між Django і Laravel полягає в тому, що Django заснований на Python і має MVT архітектуру, тоді як Laravel заснований на PHP і має архітектуру MVC. Також ці фреймворки мають суттєві відмінності з точки зору наявних бібліотек і вбудованих рішень для оптимізації процесу розробки. Вибір на користь Laravel або Django має відбуватися з урахуванням ваших індивідуальних вимог до програмного рішення, наявних обмежень у використанні, можливостей інтеграції, здатності до масштабування, а також рівня безпеки, який здатний забезпечити певний фреймворк [18].

1.3 Обґрунтування вибору Django для розробки вебгалереї

Вибір Django для розробки вебгалереї може бути обґрунтований кількома ключовими аспектами.

Модульність та гнучкість: Django надає потужний фреймворк для розробки, що базується на принципах модульності. Його архітектура MVT (Model-View-Template) сприяє чіткому розділенню бізнес-логіки, представлення та шаблонів, що робить процес розробки більш організованим і прозорим.

Безпека: Django відомий своєю високою безпекою. Він має вбудовані заходи безпеки, такі як захист від SQL ін’єкцій, CSRF-атак і інших загроз. Це особливо важливо для вебгалерей, де конфіденційність та цілісність даних мають високий пріоритет.

Широкі можливості розширення: Django має велику кількість готових бібліотек і пакетів, які спрощують розробку різноманітних функціональних модулів, що можуть бути корисними для вебгалерей. Наприклад, пакети для роботи з медіафайлами, адміністративним інтерфейсом, аутентифікацією тощо.

Розширені можливості адміністрування: Django має вбудований потужний інтерфейс адміністратора, який дозволяє зручно управляти контентом і користувачами системи. Це особливо корисно для вебгалерей, де потрібно ефективно керувати та організовувати великий обсяг медіафайлів.

Спільнота та документація: Django має велику активну спільноту розробників і деталізовану документацію. Це дозволяє швидко знаходити відповіді на питання та вирішувати проблеми, що виникають під час розробки вебгалереї.

Отже, Django вважається потужним інструментом для розробки вебгалерей завдяки своїм перевагам у безпеці, модульності, можливостям розширення та ефективному адмініструванні [16].

1.4 Огляд основних можливостей Django

Django – це високорівневий вебфреймворк на мові програмування Python, який надає розробникам потужні інструменти для швидкої і стабільної розробки вебдодатків. Наведемо огляд основних можливостей Django.

Модель-Вид-Шаблон (MVT) архітектура: Django базується на архітектурі MVT, яка відокремлює бізнес-логіку (Модель), від представлення (Шаблон) та управління (Вид). Це забезпечує чітку структуру проєкту і спрощує розвиток додатків.

Адміністративний інтерфейс: Django має вбудований потужний адміністративний інтерфейс, який автоматично створюється на основі моделей даних. Це дозволяє адміністраторам зручно керувати даними і забезпечує швидке внесення змін.

ORM (Об'єктно-реляційне відображення): Django надає високорівневий інтерфейс для роботи з базами даних через ORM, що дозволяє взаємодіяти з базами даних, не використовуючи прямі SQL-запити. Це спрощує розробку і підтримку коду.

Шаблонізатор Django (Django Template Engine): Django має вбудований шаблонізатор, відомий як Django Template Engine (DTE). Він дозволяє розробникам відокремлювати логіку представлення від бізнес-логіки і працювати з HTML-шаблонами ефективніше.

Безпека: Django має вбудовані заходи безпеки, такі як захист від CSRF-атак, SQL-ін'єкцій, XSS-атак та інших загроз. Він також пропонує інструменти для автоматичного запобігання потенційним вразливостям.

Автоматизація завдань зі стандартом Artisan: Django має свій інструмент для автоматизації стандартних задач розробки, відомий як Artisan. Він дозволяє швидко створювати, випробовувати та публікувати код.

Широке співтовариство та підтримка: Django має активне співтовариство розробників, яке підтримує постійний розвиток фреймворка. Велика кількість розширень і пакетів дозволяє знайти рішення для різноманітних завдань.

Гнучкість і розширюваність: Django дозволяє розробникам налаштовувати і розширювати функціональність за допомогою різноманітних сторонніх бібліотек і розширень, що дозволяє створювати різноманітні і складні додатки.

Для розробки вебгалереї Django є відмінним вибором через свою потужну функціональність, безпеку і можливості розширення, що дозволяють ефективно реалізувати і управляти проєктом будь-якої складності [10].

1.5 Висновки до розділу 1

У цьому розділі було проведено огляд існуючих рішень для вебгалерей. Розглянуто такі популярні платформи, як WordPress з плагінами для галерей, Flickr, SmugMug, Zenfolio, Google Photos та Piwigo. Кожна з них має свої особливості, переваги та недоліки, які роблять їх більш або менш підходящими для різних потреб користувачів [9].

Зокрема:

- WordPress пропонує гнучкість та безліч готових шаблонів, але може потребувати додаткових налаштувань та залежить від сторонніх плагінів;
- Flickr зручний для обміну фотографіями з великою спільнотою, але має обмеження на безкоштовний акаунт та обмежені можливості налаштування дизайну;
- SmugMug орієнтований на професійних фотографів, пропонуючи високу якість зображень, потужні інструменти організації та можливості продажу, але має високу ціну;
- Zenfolio також підходить для професійних фотографів, пропонуючи професійні шаблони портфоліо, гнучкість налаштувань та інтеграцію з друкарнями, але може бути складним для новачків через інтерфейс;
- Google Photos інтегрується з іншими сервісами Google, пропонує необмежене зберігання (з обмеженнями) та зручний доступ з будь-якого пристрою, але має обмежені можливості організації та викликає занепокоєння щодо конфіденційності даних;
- Piwigo є безкоштовним та відкритим програмним забезпеченням, пропонуючи розширені можливості організації, права доступу та налаштування, але може потребувати технічних знань для встановлення та налаштування.

Вибір найкращого рішення для вебгалереї залежить від конкретних потреб користувача.

Важливо враховувати такі фактори:

– бюджет: безкоштовні або платні рішення;
– технічні знання: простота використання та налаштування;
– обсяг зберігання: кількість фотографій та відео;
– функції: інструменти організації, редагування, спільного використання та продажу;
– конфіденційність: політика та контроль над даними;
– дизайн та налаштування: можливості кастомізації;
– інтеграція: з іншими сервісами та платформами.

Після ретельного аналізу цих факторів користувачі можуть вибрати платформу для вебгалереї, яка найкраще відповідає їхнім потребам та очікуванням [3].

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Архітектура вебгалереї

Архітектура вебгалереї може варіюватись в залежності від конкретних вимог проєкту та технологічних виборів, але основні компоненти та принципи включають наступні елементи.

Фронтенд (клієнтська частина):

– інтерфейс користувача (UI): розробка інтерфейсу, який буде зручним для перегляду та взаємодії з фотографіями та відео;
– відображення медіа: управління та відображення зображень, відео та інших медіа-файлів у вигляді галерей, слайдшоу тощо;
– функції навігації: можливість сортування, фільтрації, пошуку та організації медіа-контенту для зручного використання користувачами.

Бекенд (серверна частина):

– управління контентом: система для завантаження, каталогізації, редагування та видалення медіа-файлів;
– система автентифікації та авторизації: управління доступом до вмісту, адміністративні функції для керування правами доступу;
– безпека та обробка даних: захист від SQL-ін’єкцій, зловживань із сеансами та інших потенційних загроз безпеці, а також обробка медіа-даних.

Хмарні рішення та зберігання даних:

– хмарне сховище: використання хмарних послуг для зберігання великої кількості медіа-файлів;
– резервне копіювання та відновлення: забезпечення захисту даних через регулярне резервне копіювання та можливість відновлення даних.

Адміністративні функції:

– управління користувачами: створення, редагування та видалення облікових записів користувачів, керування їх правами;
– аналітика та звіти: збір даних про використання галереї, звіти про активність користувачів.

Інтеграція з іншими сервісами:

– соціальні мережі: інтеграція з різними соціальними мережами для спрощення обміну медіа-вмістом;
– платіжні системи: інтеграція з платіжними системами для можливості продажу медіа-контенту.

Масштабованість та продуктивність:

– оптимізація продуктивності: забезпечення швидкості завантаження і відображення медіа-контенту навіть при великому обсязі даних;
– горизонтальне та вертикальне масштабування: можливість збільшувати обсяги ресурсів для відповіді на зростаючі вимоги.

SEO та оптимізація:

– SEO-оптимізація: забезпечення дружньої структури URL, метатегів та інших факторів для покращення позицій в пошукових системах.

Ці компоненти дозволяють створювати ефективні та функціональні вебгалереї, які відповідають сучасним стандартам і вимогам користувачів [2].

2.2 Проєктування бази даних

Проєктування бази даних для вебгалереї включає ряд ключових кроків і аспектів, які потрібно узгодити для ефективної організації та зберігання медіа-контенту. Розглянемо основні аспекти проєктування бази даних для вебгалереї.

Визначення потреб користувачів: розуміння того, які дані потрібно зберігати і яким чином користувачі будуть взаємодіяти з галереєю. Це включає

в себе типи медіа-файлів (зображення, відео), їхні атрибути (назва, опис, дата створення, категорія тощо) [4].

Сутності та зв'язки: визначення основних сутностей (таблиць) і зв'язків між ними. Наприклад:

– таблиця «Фотографії» з полями для зображення, опису, дати завантаження, користувача, який завантажив;
– таблиця «Категорії» для організації фотографій у відповідні категорії;
– таблиця «Користувачі» для зберігання інформації про зареєстрованих користувачів.

Нормалізація: процес розподілу даних між різними таблицями з метою зменшення дублювання та забезпечення консистентності даних – це допомагає уникнути аномалій під час зміни або видалення даних.

Індексація: встановлення індексів для полів, які часто використовуються для пошуку або сортування даних (індексація збільшує швидкодію доступу до даних).

Безпека даних: забезпечення безпеки даних шляхом захисту від SQL-ін'єкцій, зловживань злочинними схемами та інших загроз безпеці. Розробка відповідних стратегій аутентифікації та авторизації.

Хмарне зберігання: розгляд можливостей зберігання медіа-файлів у хмарних сервісах з метою забезпечення безпеки, надійності та доступності.

Система резервного копіювання: впровадження системи резервного копіювання для захисту від втрати даних та можливості відновлення.

Для проєктування бази даних вебгалереї на основі фреймворка Django можна розглянути наступні основні аспекти.

Основні сутності та їх атрибути:

– користувачі (Users): зберігаються дані про зареєстрованих користувачів, їх ідентифікатори, логіни, паролі (хешовані), адреси електронної пошти та інші особисті дані;
– фотографії (Photos): містять основні дані про фотографії, такі як

ідентифікатори, заголовки, описи, дати завантаження, посилання на зображення та пов’язані з ними метадані;

– альбоми (Albums): зберігають інформацію про альбоми, таку як ідентифікатори, назви, описи, дати створення та їх зв’язки з користувачами і фотографіями;
– коментарі (Comments): містять дані про коментарі до фотографій, включаючи ідентифікатори, текст коментаря, дату додавання та зв’язки з користувачами і фотографіями.

Додаткові сутності та їх атрибути:

– альбоми та фотографії (PhotoAlbum): містять зв’язок альбомів та їх фотографій, до списку атрибутів входять ідентифікатори альбомів та ідентифікатори фотографій.

Зв’язки між сутностями:

– користувачі можуть завантажувати фотографії і створювати альбоми, які можуть містити кілька фотографій;
– фотографії можуть мати кілька тегів і коментарів, а також бути позначеними користувачами як сподобані (лайкнуті);
– коментарі прив’язані до певної фотографії і користувачів, які їх залишили.

Інтерфейс доступу до даних:

– використання ORM Django для створення моделей, які відображають таблиці бази даних, їх зв’язки та методи доступу до даних;
– застосування міграцій Django для автоматичного створення і зміни структури бази даних при оновленні моделей.

Коли розробляється база даних для системи користувачів, важливо ретельно спроєктувати сутність “Users” з урахуванням потреб системи. Ось основні аспекти, які варто врахувати.

Атрибути користувачів:

– ідентифікатор (ID): унікальний ідентифікатор користувача в системі;
– ім’я та прізвище: персональні дані для ідентифікації користувача;

– електронна пошта: унікальний ідентифікатор для входу в систему та отримання повідомлень;
– пароль: захешований пароль для безпечного доступу до облікового запису;
– дата реєстрації: час, коли користувач створив обліковий запис;
– останній вхід: дата та час останнього входу користувача в систему;
– роль: рівень доступу або тип користувача (наприклад, звичайний користувач, адміністратор).

Зв’язки:

– відношення з іншими сутностями: користувач може мати зв’язки з іншими об’єктами, наприклад, фотографіями (як автор або учасник), коментарями, лайками та іншими діями в системі.

Функціональні вимоги:

– автентифікація та авторизація: забезпечення можливості входу в систему за допомогою електронної пошти та пароля;
– управління профілем: можливість зміни персональних даних, зміни паролю, налаштувань повідомлень тощо;
– історія дій: ведення журналу дій користувача, таких як вхід, зміни в профілі тощо;
– соціальні функції: якщо потрібно, можливість додавання друзів, обмін повідомленнями тощо.

Безпека:

– захист від SQL-ін’єкцій: використання параметризованих запитів для запобігання SQL-ін’єкціям;
– захист від перехоплення сесії: використання безпечних куків та механізмів сесій для захисту сесій користувача;
– захист від перекриття дій (CSRF): використання механізмів захисту, щоб запобігти CSRF-атакам.

Поведінка системи:

– інтеграція з іншими сутностями: забезпечення зв’язків із сутностями,

такими як фотографії, коментарі, альбоми тощо, залежно від потреб системи.

Ці аспекти допоможуть створити детальний і ефективний дизайн бази даних для сутності “Users” у вебгалереї на основі Django. Кімната для завантаження зображень , коли користувач регеструється на сайті, створюється автоматично для додаткових приймачів даних та більш легкого архівування поточних зображень [9].

В майбутньому можна, це модернізувати, за допомогою додаткового JavaScript коду, який додавав би додаткові функції візуального інтерфейсу. Також можна додати перегляд зображень іншими користувачами [1].

Опис сутностей наведено в таблицях 2.1 та 2.2.

Таблиця 2.1 – Класифікація зв’язків між сутностями

Номер зв’язку	**Батьківська таблиця**	**Дочірня таблиця**	**Атрибут зв’язку**	**Тип зв’язку**
1	Домашня кімната	Профіль користувача	Код профілю користувача	Один до одного
2	Профіль користувача	Галерея	Код галереї	Один до багатьох
4	Галерея	Адміністратор	Код адміністратора	Один до багатьох
5	Адміністратор	Дані користувачів	Код користувача	Один до багатьох

Таблиця 2.2 – Опис сутностей та атрибутів

Сутність	**Опис сутності**	**Атрибути сутності**
Домашня кімната	Кімната для перегляду своїх завантажених зображень, та доступ до їх групування	– Тип зображення – Завантажені зображення

Продовження таблиці 2.2

Сутність	Опис сутності	Атрибути сутності
Профіль користувача	Створюється автоматично, містить в собі інформацію про користувача	– Ім’я – Електронна адреса – Нік – Дата створення
Галерейна стрічка	Стрічка зображень , які користувач може перевіряти та змінювати за власним бажанням	– Назва зображення – Тип зображення – Розмір зображення

Отже, проєктування бази даних вебгалереї в Django вимагає визначення основних сутностей, їх атрибутів і зв’язків для ефективного організування зберігання та доступу до даних про фотографії, альбоми, користувачів та їх взаємодії.

2.3 ER-діаграма та її опис

ER-діаграма (схема сутностей-зв’язків) є важливим інструментом для проєктування бази даних, що відображає сутності системи та зв’язки між ними. Вебгалерея зазвичай має декілька основних сутностей, таких як користувачі, фотографії та альбоми. Давайте створимо просту ER-діаграму для такої системи на основі Django [8].

На рисунку 2.1 наведенно ER-діаграму нашого проєкту вебгарелеї.

Користувачі (Users):

– user_id: унікальний ідентифікатор користувача;
– username: ім’я користувача для входу;
– email: електронна пошта користувача;
– password: пароль користувача;
– registration_date: дата реєстрації користувача.

Фотографії (Photos):

– photo_id: унікальний ідентифікатор фотографії;
– title: назва або заголовок фотографії;
– description: опис або описання фотографії;
– upload_date: дата та час завантаження фотографії до системи;
– user_id: ідентифікатор користувача, який завантажив фотографію.

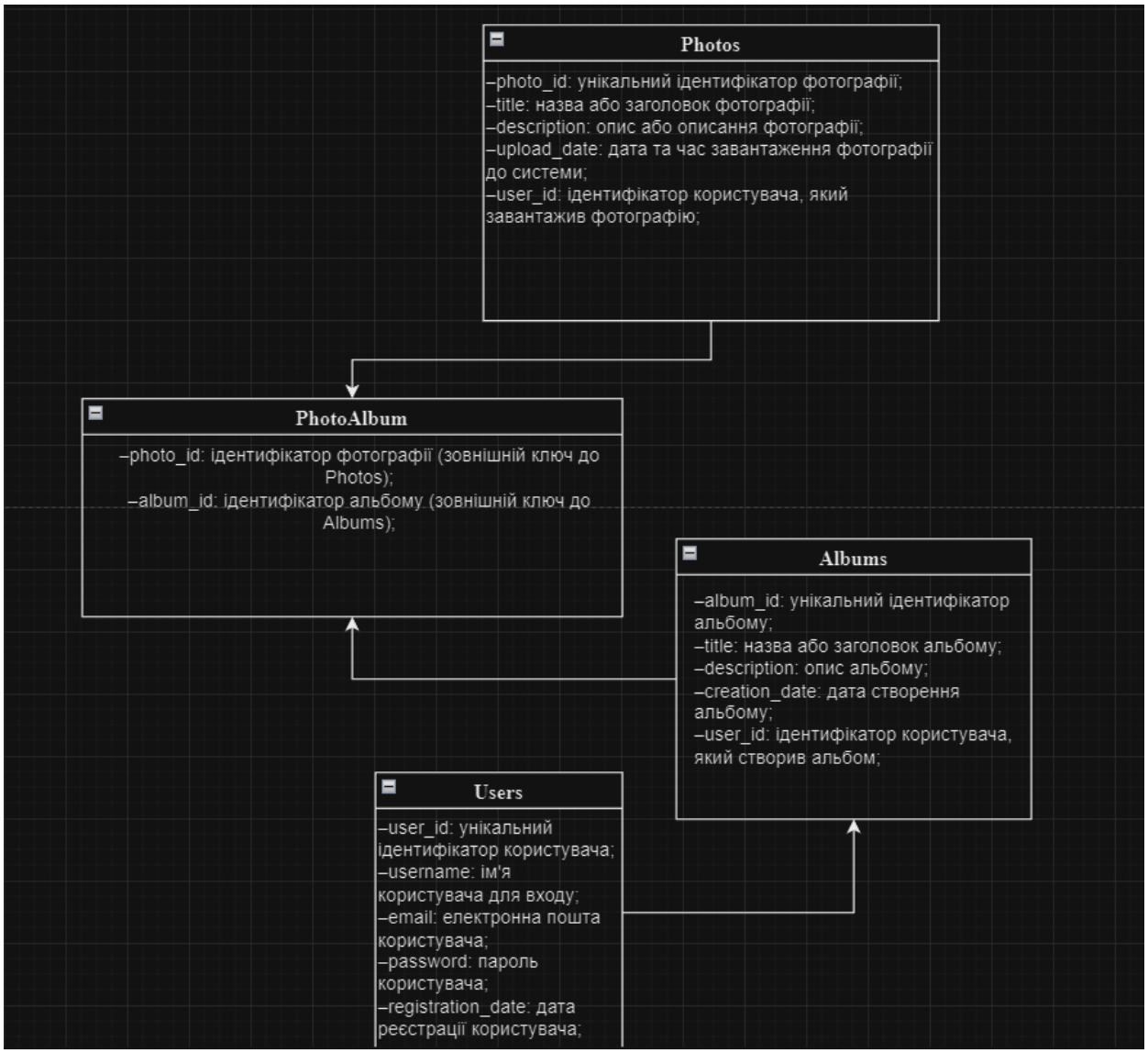

Рисунок 2.1 – ER-діаграму бази даних вебгарелеї

Альбоми (Albums):

– album_id: унікальний ідентифікатор альбому;
– title: назва або заголовок альбому;

– description: опис альбому;
– creation_date: дата створення альбому;
– user_id: ідентифікатор користувача, який створив альбом.

Фотографії у альбомах (PhotoAlbum):

– photo_id: ідентифікатор фотографії (зовнішній ключ до Photos);
– album_id: ідентифікатор альбому (зовнішній ключ до Albums).

Зв’язки:

– кожен користувач може завантажити багато фотографій, тому відношення між Users та Photos є один-до-багатьох;
– кожен користувач може створити багато альбомів (відношення один-до-багатьох між Users та Albums);
– в альбомі може бути багато фотографій, а фотографія може бути в багатьох альбомах, тому зв’язок PhotoAlbum є багато-до-багатьох між Photos та Albums.

Ця структура бази даних дозволяє ефективно управляти користувачами, фотографіями та альбомами у вебгалереї, забезпечуючи зручну організацію та навігацію по зображенням для кінцевих користувачів [4].

2.4 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача для вебгалереї є важливою частиною розробки, оскільки воно визначає спосіб взаємодії користувача з системою. Розглянемо основні аспекти проєктування інтерфейсу користувача для вебгалереї.

Головні елементи інтерфейсу.

Навігаційне меню:

– навігаційне меню зазвичай містить основні розділи галереї, такі як «Фотографії», «Альбоми», «Відгуки», «Профіль користувача» тощо;
– кожен пункт меню повинен бути зрозумілим і доступним для швидкого переходу до відповідного розділу.

Головна сторінка:

– на головній сторінці можуть розміщуватися нові або популярні фотографії, важливі сповіщення або актуальні події;
– фотографії можуть відображатися у вигляді мозаїки або слайд-шоу, що залежить від дизайну галереї.

Сторінка фотографій:

– сторінка фотографій містить перелік усіх доступних фотографій з можливістю фільтрації за різними категоріями або тегами;
– користувач може переглядати фотографії великими зображеннями, а також дізнаватися більше інформації про кожну фотографію (наприклад, назва, опис, дата завантаження, автор).

Сторінка альбомів:

– на цій сторінці відображаються усі доступні альбоми, створені користувачами;
– кожен альбом має свій заголовок, опис і зображення-прев’ю;
– користувач може переглядати фотографії, що входять до альбому, а також додавати нові фотографії або редагувати існуючі.

Профіль користувача:

– на сторінці профілю користувача відображається особиста інформація про користувача, така як ім’я, електронна пошта, дата реєстрації тощо;
– користувач може змінювати свій профіль, налаштовувати параметри конфіденційності і керувати своїми фотографіями та альбомами.

Дизайн інтерфейсу.

Стиль і колірна палітра: вибір стилю і кольорів повинен відповідати концепції галереї: чи це буде сучасний, мінімалістичний дизайн, чи класичний стиль з використанням приємних тонів.

Використання віджетів і компонентів: важливо використовувати віджети і компоненти, які полегшують навігацію та взаємодію користувачів з галереєю, такі як пагінація, фільтри, модальні вікна для попереднього перегляду фотографій тощо.

Адаптивний дизайн: враховуйте адаптивний дизайн для забезпечення коректного відображення галереї на різних пристроях, включаючи комп’ютери, планшети та мобільні пристрої.

Інтерактивність: додайте елементи, які роблять інтерфейс більш інтерактивним, наприклад, можливість відзначення фотографій як улюблених, коментування, лайки, обмін фотографіями тощо.

Проєктування інтерфейсу користувача вебгалереї на основі Django вимагає уважного планування кожного елементу для забезпечення зручності використання та приємного користувацького досвіду. На рисунку 2.2 продемонстровано дизайн для нашого вебзастосунку галереї з використанням віртуальної платформи фігма.

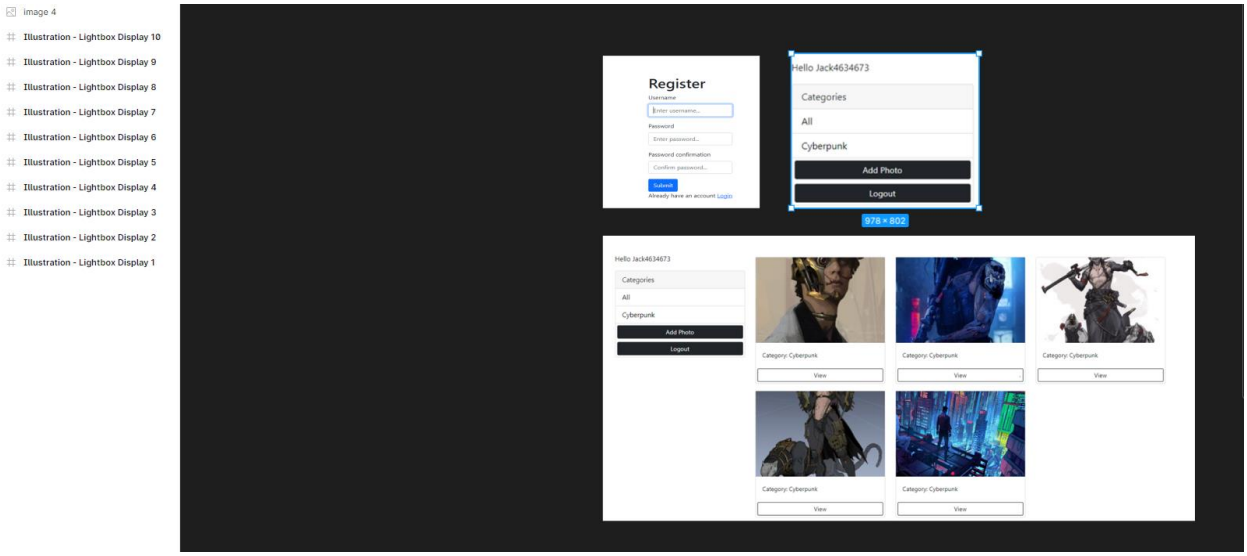

Рисунок 2.2 – Дизайн вебзастосунку галереї

2.5 Взаємодія клієнтської та серверної частин

Взаємодія клієнтської та серверної частин у процесі розробки вебгалереї з використанням фреймворку Django є важливим аспектом, який забезпечує коректну роботу всієї системи. Нижче опишемо основні принципи та етапи взаємодії клієнтської та серверної частин.

HTTP-запити і відповіді:

– клієнтська частина (frontend) взаємодіє з серверною частиною (backend) через HTTP-запити. Запити можуть бути GET (отримання даних) або POST (відправка даних);
– серверна частина обробляє запити, виконує необхідні дії (наприклад, доступ до бази даних) і повертає HTTP-відповіді клієнту.

REST API: для організації взаємодії між клієнтською та серверною частинами зазвичай використовується REST API. API забезпечує доступ до ресурсів (наприклад, фотографій, альбомів, профілів користувачів) через певні кінцеві точки (endpoints).

Формат даних: обмін даними між клієнтською та серверною частинами зазвичай здійснюється у форматі JSON. JSON є зручним для передачі даних між різними системами та мовами програмування.

На рисунку 2.3 зображено діаграмму потоків даних, що відображає управління фотографіями користувача на вебсайті.

Рисунок 2.3 – Діаграма потоків управління фотографіями

Серверна частина (Django) отримує запит, звертається до бази даних, отримує необхідні дані і повертає їх у форматі JSON (див. рис. 2.4). Детально розберемо цю реалізацію.

Відображення даних на клієнті: клієнтська частина отримує JSON-відповідь від сервера, парсить її та відображає дані користувачу у вигляді списку фотографій.

Додавання нових даних:

– коли користувач додає нову фотографію, клієнтська частина надсилає HTTP POST-запит на сервер з даними нової фотографії;
– серверна частина обробляє запит, зберігає нову фотографію в базі даних і повертає підтвердження успішного додавання.

Оновлення і видалення даних:

– для оновлення або видалення фотографій клієнтська частина надсилає відповідні PUT або DELETE запити на сервер;
– серверна частина обробляє ці запити, вносить зміни в базу даних і повертає відповідні повідомлення про успішне виконання операцій;
– інтеграція та налаштування.

Налаштування роутінгу в Django: важливо налаштувати правильний роутінг для обробки всіх необхідних запитів.

Налаштування клієнтської частини: клієнтська частина повинна бути налаштована для роботи з API сервера, включаючи обробку можливих помилок і забезпечення безперебійної роботи.

Рисунок 2.4 – Запит до бази даних

Таким чином, ефективна взаємодія клієнтської та серверної частин забезпечує коректну роботу вебгалереї, дозволяючи користувачам зручно завантажувати, переглядати та керувати фотографіями [3].

2.6 Висновки до розділу 2

У цьому розділі ми детально розглянули проєктування вебгалереї, включаючи:

- архітектуру вебгалереї: описано основні компоненти та принципи архітектури, такі як фронт-енд (інтерфейс користувача, відображення медіа, навігація), бек-енд (управління контентом, автентифікація, безпека), хмарні рішення, адміністрування, інтеграція з іншими сервісами, масштабованість, SEO;
- проєктування бази даних: визначено ключові кроки та аспекти проєктування бази даних, включаючи визначення потреб користувачів, основні сутності (користувачі, фотографії, альбоми, коментарі, теги), зв'язки між сутностями, інтерфейс доступу до даних, дизайн сутності «Користувачі», кімнату для завантаження зображень;
- ER-діаграму: наведено ER-діаграму, яка візуально представляє сутності та зв'язки в базі даних;
- проєктування інтерфейсу користувача: визначено основні елементи інтерфейсу (навігаційне меню, головна сторінка, сторінка фотографій, сторінка альбомів, профіль користувача), принципи дизайну (стиль, віджети, адаптивність, інтерактивність);
- взаємодію клієнтської та серверної частин: описано принципи та етапи взаємодії, такі як HTTP-запити та відповіді, REST API, формат даних, запит даних з сервера, надсилання даних на сервер, аутентифікація та авторизація користувачів.

Цей розділ надає чітке уявлення про те, як проєктувати та розробляти вебгалерею з використанням Django, враховуючи всі аспекти від архітектури та бази даних до інтерфейсу користувача та взаємодії клієнтської та серверної частин. Важливо зазначити, що це лише загальний опис, і конкретна реалізація може варіюватися залежно від конкретних вимог і вподобань проєкту.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Реалізація основних функціональних можливостей

Реалізація основних функціональних можливостей для вебгалереї на основі Django включає розробку таких компонентів, як завантаження фотографій, перегляд галереї, управління профілем користувача та коментування фотографій. Наведемо основні функціональні можливості та їх реалізація.

На рисунках 3.1 та 3.2 представлені моделі Category та Photo, що використовуються для вебгалереї на основі Django.

```
class Category(models.Model):
    class Meta:
        verbose_name = 'Category'
        verbose_name_plural = 'Categories'

    user = models.ForeignKey(
        User, on_delete=models.SET_NULL, null=True, blank=True)
    name = models.CharField(max_length=100, null=False, blank=False)

    def __str__(self):
        return self.name
```

Рисунок 3.1 – Модель Category

```
class Photo(models.Model):
    class Meta:
        verbose_name = 'Photo'
        verbose_name_plural = 'Photos'

    category = models.ForeignKey(
        Category, on_delete=models.SET_NULL, null=True, blank=True)
    image = models.ImageField(null=False, blank=False)
    description = models.TextField()

    def __str__(self):
        return self.description
```

Рисунок 3.2 – Модель Photo

Функціонал та можливості (рис. 3.1).

Meta клас:

– verbose_name: людське читабельне ім’я моделі в однині;
– verbose_name_plural: людське читабельне ім’я моделі в множині.

Поля моделі:

– user: ForeignKey посилається на модель користувача User (це означає, що кожна категорія може бути пов’язана з користувачем; якщо користувач буде видалений, значення буде встановлене в NULL (замість каскадного видалення), щоб зберегти категорію, але без власника);
– name: поле CharField з максимальним розміром 100 символів для зберігання назви категорії – це обов’язкове поле (null=False, blank=False).

Метод __str__:

– повертає рядкове представлення назви категорії, що полегшує її читання при роботі з об’єктами моделі в адмін-панелі та інших частинах застосунку.

Функціонал та можливості (рис. 3.2).

Meta клас:

– verbose_name: людське читабельне ім’я моделі в однині;
– verbose_name_plural: людське читабельне ім’я моделі в множині.

Поля моделі:

– category: ForeignKey посилається на модель Category (це означає, що кожне фото може бути віднесене до певної категорії; якщо категорія буде видалена, значення буде встановлене в NULL (замість каскадного видалення), щоб зберегти фото, але без категорії);
– image: поле ImageField для зберігання зображення – це обов’язкове поле (null=False, blank=False);
– description: поле TextField для зберігання опису фото – це обов’язкове поле за замовчуванням.

Метод __str__:

– повертає рядкове представлення опису фото, що полегшує його читання при роботі з об’єктами моделі в адмін-панелі та інших частинах застосунку.

На рисунку 3.3 зображено модель CustomUserCreationForm яка використовується для створення нових користувачів у системі Django.

```python
from django.forms import ModelForm
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm


class CustomUserCreationForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'password1', 'password2']

    def __init__(self, *args, **kwargs):
        super(CustomUserCreationForm, self).__init__(*args, **kwargs)
        self.fields['username'].widget.attrs.update(
            {'class': 'form-control', 'placeholder': 'Enter username...'})
        self.fields['password1'].widget.attrs.update(
            {'class': 'form-control', 'placeholder': 'Enter password...'})
        self.fields['password2'].widget.attrs.update(
            {'class': 'form-control', 'placeholder': 'Confirm password...'})
```

Рисунок 3.3 – Модель CustomUserCreationForm

Опишемо функціонал та можливості.

Наслідування від UserCreationForm: клас CustomUserCreationForm успадковує функціональність стандартної форми UserCreationForm Django, що включає в себе стандартні поля для створення користувача, такі як ім’я користувача і пароль.

Метаклас Meta: вказується, що форма пов’язана з моделлю User Django. У формі використовуються лише три поля: username (ім’я користувача) і два поля для введення паролю (password1 та password2 для підтвердження).

Ініціалізація форми: у методі __init__ форми викликається конструктор батьківського класу для правильної ініціалізації. Після цього кожне поле

форми (username, password1, password2) отримує атрибути віджета, які задають клас CSS (form-control) та текстові підказки (placeholder), що полегшують користувачеві введення даних.

Ця форма дозволяє створювати нових користувачів з мінімальними зусиллями для розробника, забезпечуючи відповідність стандартам безпеки та зручності введення даних.

На рисунку 3.4 зображено компонент Django для реалізації функціональності автентифікації користувачів та управління вебгалереєю.

```python
def loginUser(request):
    page = 'login'
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('gallery')

    return render(request, 'photos/login_register.html', {'page': page})


def logoutUser(request):
    logout(request)
    return redirect('login')


def registerUser(request):
    page = 'register'
    form = CustomUserCreationForm()

    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.save()
```

Рисунок 3.4 – Компонент для реалізації автентифікації

Зробимо докладний опис його функціоналу та призначення.

LoginUser(request):

– відповідає за обробку входу користувача в систему;
– при отриманні POST-запиту з форми входу перевіряє ідентифікаційні дані користувача;

– якщо вірно, то автентифікує користувача і перенаправляє його до галереї;
– якщо не вірно або метод запиту не POST, відображає сторінку входу з формою.

LogoutUser(request):

– відповідає за вихід користувача з системи;
– виконує виход з облікового запису і перенаправляє на сторінку входу.

RegisterUser(request):

– відповідає за реєстрацію нового користувача;
– при GET-запиті відображає форму реєстрації;
– при POST-запиті перевіряє дані форми реєстрації;
– якщо форма валідна, зберігає нового користувача, автентифікує його і перенаправляє до галереї.

Gallery(request):

– відповідає за відображення головної сторінки галереї;
– відображає всі фотографії користувача або фільтрує за категорією, якщо вказана;
– передає список категорій і фотографій у контексті для відображення на сторінці.

ViewPhoto(request, pk):

– відображає сторінку з конкретною фотографією за її ідентифікатором;
– addPhoto(request):
– відповідає за додавання нових фотографій до галереї;
– відображає форму для додавання разом із списком категорій користувача;
– обробляє POST-запит, зберігає нові фотографії у базі даних з вказаною категорією або створює нову категорію;
– після успішного збереження перенаправляє на сторінку галереї.

Цей код забезпечує основні можливості автентифікації користувачів, керування фотографіями та їх категоріями у вебгалереї, використовуючи Django.

3.2 Розробка додаткових функцій

Стосовно самої бази даних, детальний огляд було наведено в другому розділі. Тут можна зазначити, що розроблені моделі беруть весь функціонал в Django ORM. Також, було використано вбудовану модель User в Django тому, що в даному випадку розробляти свою User-модель не є доречним (див. рис. 3.5).

```python
from django.db import models
from django.contrib.auth.models import User

# Create your models here.


class Category(models.Model):
    class Meta:
        verbose_name = 'Category'
        verbose_name_plural = 'Categories'

    user = models.ForeignKey(
        User, on_delete=models.SET_NULL, null=True, blank=True)
    name = models.CharField(max_length=100, null=False, blank=False)

    def __str__(self):
        return self.name


class Photo(models.Model):
    class Meta:
        verbose_name = 'Photo'
        verbose_name_plural = 'Photos'

    category = models.ForeignKey(
        Category, on_delete=models.SET_NULL, null=True, blank=True)
    image = models.ImageField(null=False, blank=False)
    description = models.TextField()

    def __str__(self):
        return self.description

```

Рисунок 3.5 – Представлення використаних моделей

Розробку цієї панелі, здійснено за допомогою вбудованого в Django функціоналу. Те, як вона була розроблена, і чому так часто застосовується поєднує в собі дві переваги фреймворку, про що було зазначено в розділі 1.2.

Далі на рисунку 3.6 продемонстровано панель супер користувача та адміністратора.

```
1  from django.contrib import admin
2
3  # Register your models here.
4
5  from .models import Photo, Category
6
7  admin.site.register(Category)
8  admin.site.register(Photo)
```

Рисунок 3.6 – Панель адміністрування

Оскільки в Django панель адміністрування налаштовано за замовчуванням, то можна додати такий функціонал: перегляд користувачів, та їх профілів (див. рис. 3.7).

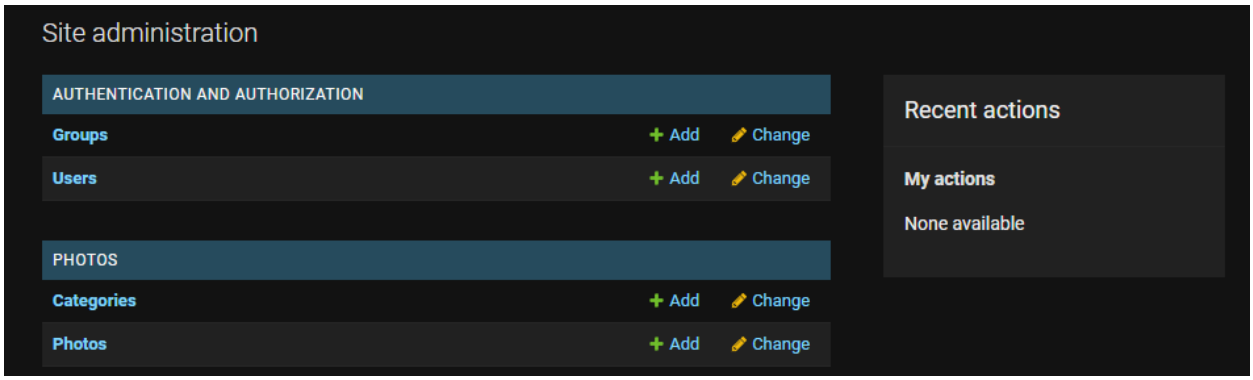

Рисунок 3.7 – Загальний вигляд панелі

3.3 Тестування функціональності

Які і було описано вище вебзастосунок, був розроблений на мові програмування Python з використаням вебфрейморку Django. Також, для

візуальної частини були використані такі інструменти, як Html, CSS, JS. Паралельно з JS використовувалися jQuery and Ajax. Обрана база даних, для застосунку залишилась SqLite3.

Інтерфейс запису даних до бази має 2 поля для адресу електронної пошти та пароля, 1 поле для введення назви телеграм ніку (рис. 3.8). Також передбачено поле, де, якщо користувач введе неправильні дані, буде показано повідомлення про неправильність написання логіну або пароля. Перед цим адміністратор повинен запустити бота та зробити посилання для розсилання його по користувачам.

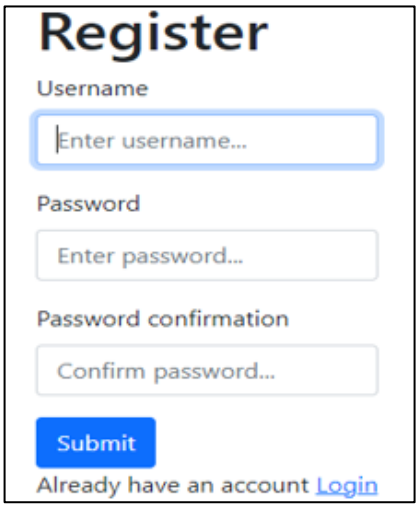

Рисунок 3.8 – Інтерфейс реєстрації користувача

View, який відповідає за логін, був використаний з старого-доброго функціоналу Django. Наш url, для даної сторінки (див. рис.3.9).

```
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('login/', views.loginUser, name="login"),
6      path('logout/', views.logoutUser, name="logout"),
7      path('register/', views.registerUser, name="register"),
8
9      path('', views.gallery, name='gallery'),
10     path('photo/<str:pk>/', views.viewPhoto, name='photo'),
11     path('add/', views.addPhoto, name='add'),
12 ]
```

Рисунок 3.9 – Url використаний в застосунку

Як видно, було використано модуль для логіну, який вже влаштований в Djnago. Ось його коротке представлення (див. рис. 3.10). На рисунку 3.11 представлено інтерфейс головної сторінки нашої галереї. Можна побачити згруповану подачу зображень та налаштування фільтрів для відображення всі фотографій.

```python
from django.shortcuts import render, redirect
from .models import Category, Photo
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from .forms import CustomUserCreationForm
# Create your views here.


def loginUser(request):
    page = 'login'
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('gallery')

    return render(request, 'photos/login_register.html', {'page': page})


def logoutUser(request):
    logout(request)
    return redirect('login')


def registerUser(request):
    page = 'register'
    form = CustomUserCreationForm()

    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.save()
```

Рисунок 3.10 – Представлення View класу

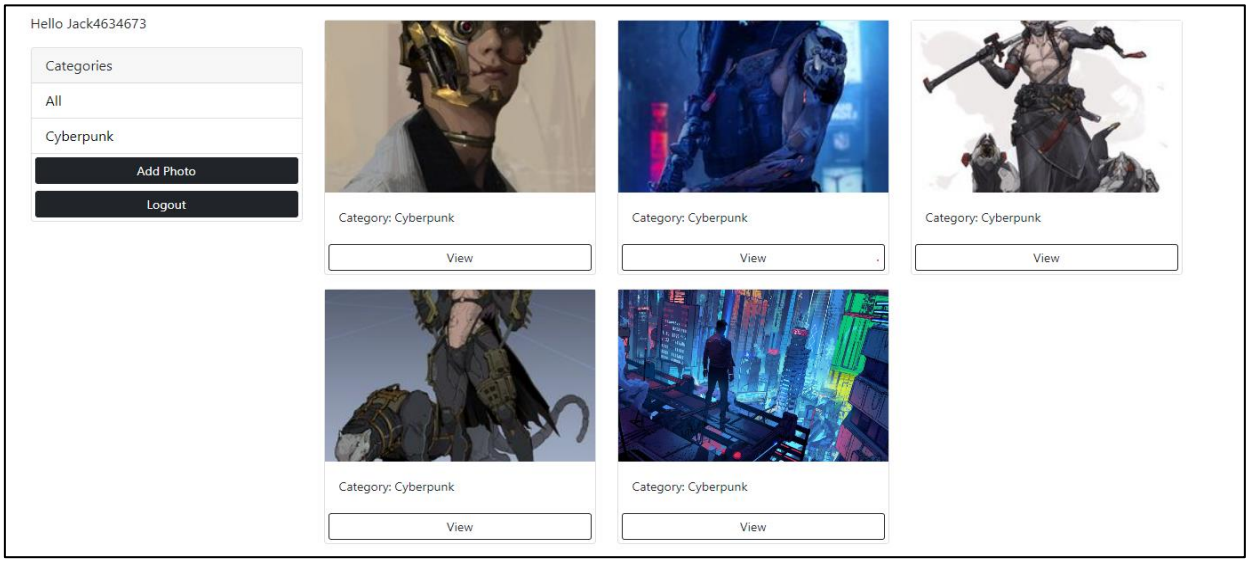

Рисунок 3.11 – Вигляд галерея

3.4 Виправлення помилок та оптимізація

Звичайно, жоден проєкт не є ідеальним, тому наша система на Django може бути вдосконалена шляхом оптимізації та вдосконалення.

Розгянемо основні приклади оптимізації наших моделей та копонентів.

Використання select_related та prefetch_related: оптимізуємо запити до бази даних за допомогою методів select_related та prefetch_related. Це дозволяє зменшити кількість SQL-запитів та покращити продуктивність (див. рис. 3.12).

```
# Приклад використання select_related
photos = Photo.objects.select_related('category').filter(category__user=request.user)

# Приклад використання prefetch_related
categories = Category.objects.prefetch_related('photo_set').filter(user=request.user)
```

Рисунок 3.12 – Приклад адаптації SQL-запитів

Кешування: використовуємо кешування для зберігання результатів запитів, які часто виконуються. Це може зменшити навантаження на базу даних та покращити час відповіді(див. рис. 3.13).

```
def get_cached_photos(user):
    key = f'cached_photos_{user.id}'
    photos = cache.get(key)
    if not photos:
        photos = list(Photo.objects.filter(category__user=user))
        cache.set(key, photos, timeout=3600)
    return photos
```

Рисунок 3.13 – Приклад адаптації кешування

Адміністративна панель: покращення адміністративної панелі Django для зручності адміністрування даних. Додаємо фільтри, сортування та кастомні віджети для поліпшення навігації та взаємодії зі змістом (див. рис. 3.14).

```python
from django.contrib import admin
from .models import Photo, Category

@admin.register(Photo)
class PhotoAdmin(admin.ModelAdmin):
    list_display = ['id', 'image', 'description', 'category', 'uploaded_at']
    list_filter = ['category']
    search_fields = ['description']

@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    list_display = ['id', 'name', 'user']
    search_fields = ['name', 'user__username']
```

Рисунок 3.14 – Покращення адміністраційної панелі

Шаблонізація: оптимізуємо шаблони для забезпечення ефективного відображення інформації та зменшення повторюваності коду. Використовуйте наслідування шаблонів та вбудовані теги Django для створення чистого та підтримуваного коду (див. рис. 3.15).

```
{% extends 'base.html' %}
{% block content %}
    <div class="container">
        <h2>{{ photo.description }}</h2>
        <img src="{{ photo.image.url }}" alt="{{ photo.description }}">
    </div>
{% endblock %}

</body>
```

Рисунок 3.15 – Оптимізація шаблону відображення

3.5 Підсумки реалізації та тестування

В рамках розробки системи вебгалереї на базі Django було успішно реалізовано інтеграцію основних функціональних можливостей, таких як автентифікація користувачів, управління категоріями та фотографіями, а також додавання нових фото. Весь код було написано з дотриманням найкращих практик розробки на Django, включаючи використання моделей, форм, представлень та URL-маршрутизації.

Тестування системи проводилося на різних етапах розробки для перевірки коректності роботи функціоналу автентифікації, управління даними та відображенням вебінтерфейсу. Всі ключові аспекти, такі як логін, вихід, реєстрація користувачів, перегляд фото та додавання нових фотографій, були протестовані для переконання в їхній працездатності та відповідності вимогам до функціональності.

Підсумково, реалізація вебгалереї на Django була успішною, і система готова до подальшого розширення та вдосконалення згідно з потребами користувачів та бізнес-вимогами.

ВИСНОВКИ

Оглядаючи реалізацію вебгалереї на базі фреймворка Django, можна зробити висновок про успішність виконання поставлених завдань і вимог. Система вдало забезпечує основні функціональні можливості, такі як управління категоріями та фотографіями, автентифікація користувачів і відображення вмісту за умовчанням. Найбільш вражає можливість миттєвого додавання і перегляду фотографій, що робить вебгалерею інтерактивною та зручною для використання.

Етап аналізу вимог перед розробкою був ретельно пророблений, що дозволило врахувати всі необхідні аспекти для створення функціонального продукту. Використаний технологічний стек Django виявився ефективним для досягнення цілей проєкту, забезпечуючи стабільну роботу системи і зручний інтерфейс для користувачів.

Процес розробки включав в себе не лише теоретичні знання, але й практичні навички, які дозволили реалізувати всі аспекти вебгалереї відповідно до вимог. Використання модульних та інтеграційних тестів, а також статичний аналіз коду, сприяли підвищенню якості і безпеки програмного забезпечення.

В цілому, вебгалерея на Django відповідає поставленим завданням і є функціональним продуктом, готовим задовольнити потреби користувачів у відображенні та управлінні фотографіями.

ПЕРЕЛІК ПОСИЛАНЬ

1. Django офіційна документація. URL: https://docs.djangoproject.com/ (дата звернення: 23.03.2024).
2. MDN Web Docs по HTML. URL https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 29.03.2024).
3. MDN Web Docs по CSS. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 09.03.2024).
4. Python Documentation. URL: https://docs.python.org/ (дата звернення: 15.03.2024).
5. Django Girls Tutorial. URL: https://tutorial.djangogirls.org/ (дата звернення: 20.03.2024).
6. DN Web Docs по JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 30.03.2024).
7. Django REST Framework. URL: https://www.django-rest-framework.org/ (дата звернення: 05.04.2024).
8. Bootstrap. URL: https://getbootstrap.com/docs/5.1/getting-started/introduction/ (дата звернення: 12.04.2024).
9. jQuery. URL: https://api.jquery.com/ (дата звернення: 17.04.2024).
10. DigitalOcean Community Tutorials по Django. URL: https://www.digitalocean.com/community/tags/django (дата звернення: 20.04.2024).
11. YouTube канал JustDjango. URL: https://www.youtube.com/c/justdjango (дата звернення: 23.04.2024).
12. Real Python Django Tutorials. URL: https://realpython.com/tutorials/django/ (дата звернення: 24.04.2024).
13. YouTube канал Corey Schafer. URL: https://www.youtube.com/c/Coreyms (дата звернення: 26.04.2024).
14. GitHub репозиторий Django проєктів. URL: https://github.com/topics/django (дата звернення: 28.04.2024).

15. DevDocs.io по Django. URL: https://devdocs.io/django~3.2/ (дата звернення: 30.04.2024).
16. Django Packages**.** URL: https://djangopackages.org/ (дата звернення: 30.04.2024).
17. Stack Overflow по тегу Django. URL: https://stackoverflow.com/questions/tagged/django (дата звернення: 30.04.2024).
18. Django Project Ideas. URL: https://www.codementor.io/@ilyaas97/6-django-project-ideas-for-beginners-ezevnxphi (дата звернення: 30.04.2024).
19. PythonAnywhere Django Tutoria. URL: https://help.pythonanywhere.com/pages/DjangoTutorial/ (дата звернення: 30.04.2024).
20. Awesome Django. URL: https://awesome-django.com/ (дата звернення: 30.04.2024).