Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка back-end частини сайту для музею УМСФ»

Виконав: студент групи К21-2

Спеціальність 122 Комп'ютерні науки

Галиновська А.О.
(прізвище та ініціали)

Керівник к.т.н., доц. Чупілко Т.А
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Університет митної справи та фінансів
(місце роботи)

Доцент кафедри кібербезпеки та інформаційних технологій
(посада)

к.т.н., доц. Савченко Ю.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Галиновська А. О. Розробка back-end частини сайту для музею УМСФ.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2025.

Пояснювальна записка: 70 с., 20 рисунків, 3 таблиці, 40 джерела.

У роботі реалізовано серверну частину веб-сайту для музею Університету митної справи та фінансів, що забезпечує збереження, адміністрування та подання інформаційного контенту. Проведено аналіз потреб музейної інформаційної системи та обґрунтовано вибір архітектури, технологій і методів реалізації. Розроблений програмний продукт базується на фреймворку Django та використовує шаблонізатор Jinja2 для генерації динамічних веб-сторінок, а також адміністративну панель Django Admin, розширену за допомогою Grappelli та редактора CKEditor.

Система дозволяє управляти експозиціями, публікаціями блогу та мультимедійними матеріалами; передбачено рольову модель доступу (адміністратор, куратор, гість). Забезпечено зручний інтерфейс керування контентом та реалізовано функціонал для редагування текстових описів, завантаження зображень і управлінням вмісту. Дані зберігаються у вбудованій базі даних SQLite, з можливістю подальшої міграції на PostgreSQL.

Ключові слова: веб-сайт, back-end, Django, музей, контент-менеджмент, адміністративна панель, CKEditor, Grappelli, Python.

ABSTRACT

Halynovska A. O. Development of the Back-End Part of the UMSF Museum Website.

Bachelor's Qualification Work for the degree of Bachelor in Specialty 122 "Computer Science". – University of Customs and Finance, Dnipro, 2025.

Explanatory note: 70 pages, 20 figures, 3 tables, 40 references.

This work presents the implementation of the server-side part of the website for the Museum of the University of Customs and Finance. The system ensures data storage, content administration, and delivery of informational materials. The project includes an analysis of the requirements for a museum information system and substantiates the choice of architecture, technologies, and development methods. The developed software product is based on the Django framework and uses the Jinja2 templating engine to generate dynamic web pages, along with the Django Admin panel enhanced by Grappelli and the CKEditor text editor.

The system enables management of exhibitions, blog posts, and multimedia content. A role-based access model (administrator, curator, guest) is implemented. A convenient content management interface is provided, supporting text editing, image uploads, and moderation. Data is stored in an embedded SQLite database, with the option for future migration to PostgreSQL.

Keywords: website, back-end, Django, museum, content management, admin panel, CKEditor, Grappelli, Python.

ЗМІСТ

АНОТАЦІЯ 1
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5
ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ 10
1.1 Опис предметної області та напрямків дослідження 10
1.2 Обґрунтування актуальності теми, формулювання завдання 11
1.3 Аналіз цифровізації музеїв і вимоги до системи 14
1.4 Визначення цільової аудиторії, об'єкта проєктування 22
1.5 Висновки до першого розділу 26
РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ, ЇХ АНАЛІЗ, ВИБІР МЕТОДІВ 28
2.1 Обґрунтування вибору засобів та методів розробки ПЗ 28
2.2 Порівняльний огляд CMS та фреймворків 32
2.3 Обґрунтування архітектури системи 38
2.4 Планування реалізації проєкту та організація команди 41
2.5 Висновки до другого розділу 46
РОЗДІЛ 3. РЕАЛІЗАЦІЯ РІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ 47
3.1 Опис структури Django-проєкту та моделей даних 47
3.2 Адміністративна панель та інструменти розробки 52
3.3 Клієнт-серверна архітектура та тестування сайту 56
3.4 Інструкція користувачу, розвиток проекту 59
3.5 Висновки до третього розділу 63
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

URL (Uniform Resource Locator) — уніфікований покажчик ресурсу; адреса, що визначає розташування ресурсу в Інтернеті;

БД — база даних;

ІС — інформаційна система;

ПЗ — програмне забезпечення;

CMS (Content Management System) — система керування контентом, що дозволяє створювати, змінювати та керувати цифровим вмістом;

WordPress — система керування контентом (CMS), що базується на PHP і MySQL, широко використовується для створення веб-сайтів і блогів;

Joomla — безкоштовна CMS з відкритим кодом, що дозволяє створювати як прості, так і складні веб-сайти;

Фронт-енд — частина веб-застосунку або сайту;

Бек-енд — серверна частина веб-застосунку, що відповідає за логіку;

Python — мова програмування високого рівня з простою та зрозумілою синтаксичною структурою;

Django — високорівневий веб-фреймворк для Python;

Flask — мікро-фреймворк для створення веб-застосунків на Python;

PostgreSQL — потужна об’єктно-реляційна система керування базами даних з відкритим кодом;

Agile — підхід до розробки ПЗ, орієнтований на ітеративну розробку, постійний зворотний зв’язок і адаптивне планування;

Scrum — гнучка методологія управління проєктами, яка передбачає роботу малими командами з чітким поділом ролей;

Jira — система для відстеження помилок, управління проєктами та завданнями, популярна в методології Agile;

Waterfall — каскадна модель розробки програмного забезпечення, що передбачає послідовне проходження етапів;

Kanban — метод управління завданнями, що візуалізує потік роботи за допомогою карток і колонок;

Drupal — гнучка CMS з відкритим кодом;

Pyramid — веб-фреймворк для Python, орієнтований на масштабовані застосунки з можливістю конфігурації;

FastAPI — сучасний веб-фреймворк для створення RESTful API на Python, що підтримує асинхронність;

Фреймворк — програмна платформа, яка полегшує розробку застосунків завдяки готовій структурі та інструментам;

Спринт — обмежений у часі етап розробки у методології Scrum;

DRY (Don't Repeat Yourself) — принцип програмування, який закликає уникати дублювання коду. ORM (Object-Relational Mapping) — технологія, що дозволяє працювати з базами даних через об'єкти мови програмування;

Branching — стратегія роботи з гілками в системах контролю версій, що дозволяє паралельну розробку;

.dot-графа — графічне представлення структур у форматі Graphviz, яке часто використовується для візуалізації зв’язків.

ВСТУП

Актуальність дослідження. У добу цифровізації та стрімкого розвитку веб-технологій важливого значення набуває створення гнучких, безпечних і масштабованих інформаційних систем. Музеї, як осередки культурної пам'яті, дедалі активніше інтегрують ІТ-рішення для збереження й популяризації експозицій. Особливу роль у цьому відіграють університетські музеї, що поєднують просвітницькі, наукові та навчальні функції. Вони потребують не просто вітринного сайту, а повноцінної інформаційної системи з можливістю керування контентом, поділу доступу та взаємодії з іншими освітніми платформами.

У межах даного дослідження було реалізовано та впроваджено функціонуючий веб-сайт музею Університету митної справи та фінансів (УМСФ), який працює в онлайн-режимі та демонструє ефективність розробленої серверної частини.

Мета роботи – розробити серверну частину веб-сайту музею, яка забезпечить зручне адміністрування контенту, розмежування прав доступу та інтеграцію з клієнтською частиною.

Для досягнення мети були поставлені такі завдання:

а) провести аналіз предметної області та цифрових музейних платформ;

б) обґрунтувати вибір технологій для серверної частини (Django, SQLite, Jinja2, CKEditor, Grappelli);

в) спроектувати структуру бекенду (моделі, URL-маршрути, контролери);

г) реалізувати адміністративну панель для керування контентом;

д) забезпечити автентифікацію, авторизацію та рольовий доступ;

е) забезпечити зв'язок із клієнтською частиною та шаблонізацію сторінок;

ж) провести тестування реалізованого функціоналу;

з) розробити інструкцію для користувача-адміністратора;

и) визначити перспективи подальшого розвитку системи.

Наукова новизна роботи полягає у розробці індивідуального бекенд-рішення для сайту музею УМСФ, що враховує потреби внутрішнього адміністрування, багаторівневого доступу, керування цифровими експозиціями, блогами та мультимедійними об’єктами без використання стандартних CMS.

Методи дослідження, використані у процесі виконання роботи, включають:

а) системний аналіз предметної області;

б) порівняльний аналіз сучасних технологій веб-розробки;

в) методи програмного проєктування та побудови інформаційних систем;

г) моделювання структури даних;

д) використання фреймворку Django;

е) тестування програмного продукту з метою перевірки відповідності функціональним і нефункціональним вимогам.

Об'єктом дослідження є веб-сайт музею УМСФ як інформаційна система.

Предметом дослідження є технології, методи та засоби реалізації серверної частини веб-системи на основі фреймворку Django.

Сучасні вимоги до музейного сайту передбачають не лише презентацію експонатів, а й створення цифрового простору для освітньої взаємодії, наукової комунікації та інтеграції в освітнє середовище університету.

Веб-сервіс повинен забезпечувати:

а) централізоване управління контентом;

б) поділ користувачів за ролями (адміністратор, куратор, гість);

в) мультимедійні можливості;

г) можливість подальшого розвитку (віртуальні тури, API-інтеграції, мобільна версія).

Практичне значення результатів полягає у створенні надійного, адаптивного інструменту для керування музейним сайтом, що не залежить від типових CMS-рішень. Розроблена система дозволяє розширювати функціональність відповідно до змін потреб установи, зберігаючи при цьому гнучкість і простоту адміністрування. Вона може бути впроваджена не лише в межах музею УМСФ, а й адаптована для інших культурно-освітніх організацій, які прагнуть реалізувати власну цифрову платформу.

Структурно робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 70 сторінку, включає 20 рисунків, 3 таблиці.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної області та напрямків дослідження

В основі дипломної роботи лежить предметна область розробки сайту для музеїв, зокрема тих, що поєднують наукові та освітні функції. Саме до таких належить музей УМСФ — структурний підрозділ закладу вищої освіти, що виконує культурно-просвітницьку, науково-дослідну й освітню місію. Актуальність обраної теми “Розробка back-end частини сайту музею УМСФ” визначається необхідністю інтеграції сучасних ІТ-рішень у сферу культури та освіти.

Зростаюче оцифрування документів, необхідність збереження культурного надбання та поширення доступу до нього в умовах обмежень фізичного відвідування актуалізують перехід музеїв до онлайн-формату.

Створення спеціалізованого веб-сайту з back-end частиною дозволяє централізовано керувати даними, впроваджувати рольову модель доступу до адміністративного інтерфейсу, а також адаптувати систему до змін в структурі музею або складі експозицій.

Розробка серверної частини передбачає ґрунтовне дослідження предметної області, зокрема структури музейної колекції, типів контенту (експонати, історичні довідки, наукові публікації), моделей користувачів (адміністратор, редактор, відвідувач, дослідник) і сценаріїв використання.

У межах дипломної роботи визначено такі ключові напрямки дослідження:

а) моделювання предметної області музейної системи засобами об’єктно-реляційного відображення;

б) реалізація контент-менеджменту через адміністративну панель Django Admin;

в) розробка функціональних модулів для керування експозиціями, публікаціями та мультимедійними об’єктами;

г) забезпечення коректної роботи системи з урахуванням безпеки зберігання даних і контролю доступу;

д) застосування принципів структурованого програмування та використання Github для контролю версій;

е) організація процесу розробки відповідно до сучасних підходів управління проєктами.

Впровадження зазначених підходів дає змогу створити функціональне, масштабоване та зручне у супроводі серверне рішення, яке відповідає потребам сучасної музейної справи.

1.2 Обґрунтування актуальності теми, формулювання завдання

У сучасному світі музеї вже давно перестали бути виключно фізичними просторами. Потреба на віддалений доступ до експозицій, колекцій, архівних матеріалів та інших елементів культурної спадщини зумовлено такими чинниками, як глобалізація, пандемічні обмеження, воєнні дії та активний розвиток дистанційної освіти [1].

Інтеграція музейного простору в цифрову екосистему відкриває нові можливості для комунікації та взаємодії з аудиторією — як місцевою, так і міжнародною. Це сприяє розширенню доступу до культурного контенту та залученню нових користувачів.

Ключовим компонентом інформаційної інфраструктури музею є серверна частина веб-ресурсу. Вона відповідає за обробку користувацьких запитів, управління вмістом, зберігання даних і забезпечення взаємодії між клієнтською частиною та базою даних. Ефективність її роботи прямо впливає на швидкодію, стабільність і безпеку всієї системи.

Для музею УМСФ створення власного веб-сайту з функціональною серверною частиною є особливо актуальним, оскільки це:

а) сприятиме збереженню та популяризації експонатів і культурної спадщини УМСФ;

б) дозволить налагодити зручну систему адміністрування контенту;

в) дозволить автоматизувати оновлення контенту, афіш подій;

г) забезпечить гнучке управління рівнями доступу до інформації для різних категорій користувачів (розмежування прав між адміністраторами, редакторами);

д) дозволить інтегрувати мультимедійний контент (зображення, відео, аудіо-екскурсії) відповідно до сучасних стандартів зручності та дизайну інтерфейсу (UX/UI);

е) створить технічну основу для подальшого розвитку функціоналу (онлайн-екскурсії, інтерактивні карти, віртуальна реальність тощо).

Застосування такого підходу дозволить укріпити роль музею як сучасного центру зберігання та трансляції культурної й освітньої інформації, що відповідає актуальним викликам суспільства та цифрової епохи.

Сам музей повинен стати цифровою візитівкою УМСФ, а створений веб-сервіс — дієвим інструментом для комунікації, збереження культурної спадщини та поширення знань серед широкої аудиторії.

Оцифрування музею — це не лише спосіб представити експозицію онлайн, а й інструмент для зручного керування контентом і взаємодії з аудиторією. Відповідно, було визначено конкретне завдання проєкту.

Основним завданням даної кваліфікаційної роботи є розробка серверної частини веб-сайту для музею УМСФ. Цей елемент повинен забезпечувати централізоване зберігання, обробку, адміністрування та подання цифрового музейного контенту.

Система має будуватися на сучасній платформі розробки, яка забезпечує зручну роботу з базою даних завдяки вбудованим засобам для опису й

керування структурами даних [2]. Завдяки модульному підходу архітектури стане можливим гнучке розширення функціоналу та ефективне масштабування проєкту відповідно до потреб.

Перед початком безпосередньої реалізації необхідно здійснити комплексний аналіз функціональних вимог до ІС музею. Таке дослідження передбачає вивчення основних сценаріїв взаємодії користувачів із платформою, визначення типів і обсягів даних, які підлягають обробці, а також специфіки роботи основних категорій користувачів.

На основі результатів аналізу формується логічна структура бекенд-системи [3]. Вона охоплює проєктування моделей даних, розробку контролерів (views), налаштування маршрутизації, а саме URL-шляхів і реалізацію прикладної бізнес-логіки (рис. 1.1).

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls'), name='index'),
    path('', views.index, name='index'),
    path('', include('exposition.urls')),
    path('gallery', views.gallery, name='gallery'),
    path('gallery/', include('gallery.urls')),
    path('expositions/', include('exposition.urls')),
    path('contacts', views.contacts, name='contacts'),
    path('history', views.history, name='history'),
    path('visitor', views.visitor, name='visitor'),
    path('grappelli/', include('grappelli.urls'))
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Рисунок 1.1 – Код URL-шляхів

Одним з ключових принципів при цьому є чітке розділення аспектів між компонентами, що сприяє зменшенню складності підтримки та полегшує подальше вдосконалення системи.

Далі відбувається створення функціональних модулів на базі фреймворку. Такі модулі мають забезпечувати основні функції системи, такі як збереження музейного контенту, категоризація та відображення користувачам.

Крім того, важливо впровадити надійні механізми авторизації та автентифікації користувачів, які забезпечують розмежування прав доступу відповідно до ролей. Контролювання доступу має гарантувати, що кожен користувач отримує доступ лише до тих ресурсів і функцій, які відповідають його ролі в системі.

А для забезпечення зручного адміністрування сайту передбачається впровадження інтуїтивно зрозумілих візуальних рішень, які розширюють стандартні можливості керування. Вони дадуть змогу працівникам музею самостійно редагувати та оновлювати вміст сторінок, працювати з текстами й зображеннями без потреби звертатися до розробників чи фахівців з технічної підтримки.

Завершальний етап розробки передбачає проведення всебічного тестування серверної частини інформаційної системи [4]. Особливий акцент буде зроблено на виявленні можливих помилок у роботі, технічних збоїв і вразливостей. Усі зафіксовані недоліки мають бути оперативно усунені до моменту впровадження системи в експлуатацію.

У підсумку, сформульоване завдання визначає чіткий вектор реалізації проєкту, сприяє раціональному плануванню робочого процесу та забезпечує передумови для досягнення бажаного функціонального результату.

1.3 Аналіз цифровізації музеїв і вимоги до системи

Традиційне уявлення про музей як про статичний фізичний простір із локальними виставками поступово трансформується: з'являється потреба в забезпеченні онлайн-доступу до експозицій, архівів та мультимедійного контенту.

Сьогодні все більше музеїв спрямовують свої зусилля на створення інтерактивного цифрового середовища, що дає змогу налагодити постійний зв’язок з аудиторією, незалежно від її географічного розташування.

Приклади реалізації цифровізації в музейній практиці:

а) музей історії Дніпра;

б) Дніпропетровський національний історичний музей;

в) музей «Машини часу».

Порівняльний аналіз у даному контексті був проведений для того, щоб визначити сильні та слабкі сторони існуючих цифрових музейних платформ, виявити загальні тенденції та стандарти а також сформувати чітке уявлення про ті функції, які обов'язково слід включити до розробки.

Музей історії Дніпра [5] є прикладом сучасного культурного простору, що вдало поєднує традиційні музейні формати з цифровими інструментами комунікації. Його експозиції охоплюють широке коло тематик, зокрема архітектурний розвиток міста, промислові здобутки.

Із точки зору цифровізації, музей демонструє низку прогресивних рішень. Офіційний веб-сайт вирізняється адаптивною верстkою, що забезпечує коректне відображення на пристроях різного типу — від настільних ПК до смартфонів. Серед доступного функціоналу — афіша подій, новинна стрічка, розділ електронної бібліотеки, інтеграція з соціальними мережами, система онлайн-бронювання квитків та англомовна версія для іноземних користувачів. Також активно використовуються мультимедійні елементи в самих експозиціях — відео, аудіо та інтерактивні панелі.

Втім, попри очевидні переваги, у функціоналі ресурсу наявні й певні недоліки. Зокрема, відсутність можливості здійснити віртуальний тур залами музею або переглянути 3D-моделі експонатів зменшує потенціал віддаленої участі для відвідувачів, які фізично не можуть потрапити до музею. Крім того, пошук у електронній бібліотеці реалізовано недостатньо гнучко: бракує інтелектуального пошукового механізму й фільтрів за категоріями, що ускладнює навігацію інформаційним масивом.

Дніпропетровський національний історичний музей [6] є одним з найстаріших в Україні та пропонує багаті колекції, що охоплюють історію

регіону від давнини до сучасності. Сайт містить розділи для наукових публікацій та освітніх програм.

Веб-сайт музею вирізняється тематичним розподілом матеріалів — окремі розділи присвячено освітнім ініціативам, науковим публікаціям та архівним джерелам. Ресурс адаптований під різні типи пристроїв, має англомовну версію, що відкриває доступ до інформації ширшій аудиторії. Підтримка RSS та зв'язок із соціальними платформами дозволяють оперативно поширювати новини та події серед зацікавлених користувачів.

Проте, незважаючи на наявність багатьох функціональних елементів, сайт має низку недоліків. Серед них — відсутність онлайн-сервісу придбання квитків або системи електронної каси, що створює незручності для відвідувачів. Крім того, навігація є дещо перевантаженою, а окремі сторінки — малозрозумілі через громіздкість текстового наповнення. Візуальна складова сайту в деяких розділах не оновлювалася, а тому створює враження застарілого інтерфейсу.

Технічний музей «Машини Часу» [7] присвячений технічному прогресу та ретро-автомобілям, пропонуючи відвідувачам занурення в атмосферу минулого століття.

Серед сучасних цифрових рішень закладу вирізняється впровадження інтерактивних квестів, доповнених елементами цифрової взаємодії. Активне ведення соціальних мереж дозволяє музею підтримувати постійну комунікацію з громадськістю, ділитися новинами й анонсами. Крім того, сайт надає можливість онлайн-реєстрації на події, що полегшує планування візиту. Присутня також англомовна версія ресурсу, що розширює його доступність для іноземних користувачів.

Попри ці переваги, цифрова присутність музею має низку обмежень. Зокрема, сайт не підтримує функцію віртуального перегляду залів або 3D-репрезентацій експонатів, що суттєво звужує дистанційні можливості ознайомлення з колекцією. Крім того, інформація про об'єкти подана

фрагментарно — відсутні докладні описи, історичний контекст і технічні характеристики. Немає інтелектуального пошуку, що ускладнює користування сайтом. Недостатньо продумана структура навігації та відсутність систематичних анонсів подій знижують інформативну цінність.

Отже, проведений порівняльний аналіз трьох музейних платформ виявив як сильні сторони кожного сайту: адаптивність, електронні архіви, онлайн-бронювання, інтерактивні квести, так і ключові недоліки: відсутність віртуальних турів, персоналізованого досвіду, зручної навігації та пошуку.

На основі отриманих результатів визначено, що при створенні нової цифрової музейної ІС для музею УМСФ необхідно врахувати наступні вимоги та тенденції:

а) наявність новин, анонсів та блогу — регулярне оновлення інформації про події музею, публікація новин і тематичних публікацій;

б) інтеграція з соціальними мережами — автоматичне поширення контенту, взаємодія з аудиторією через Facebook, Instagram тощо;

в) відсутність комерційної складової — сайт не включає платних функцій або продажу квитків відповідно до вимог замовника;

г) освітній блок експозицій — окремий розділ із навчальними матеріалами, поясненнями до експонатів і тематичними статтями;

д) віртуальні тури та 3D-відтворення експонатів (у перспективі) — заплановані функції, реалізація яких відкладена через високу вартість і складність;

е) розробка повноцінної адаптивної веб-платформи — забезпечення стабільної роботи сайту на мобільних пристроях.

У підсумку, сучасні напрями цифровізації музеїв демонструють активне прагнення до вдосконалення комунікації з відвідувачами, розширення доступу до культурних ресурсів та впровадження інноваційних технологій.

Проаналізовані приклади музейних платформ дозволили визначити ключові переваги та недоліки.

Ці спостереження лягли в основу формування вимог до майбутнього сайту музею [8] (рис. 1.2).

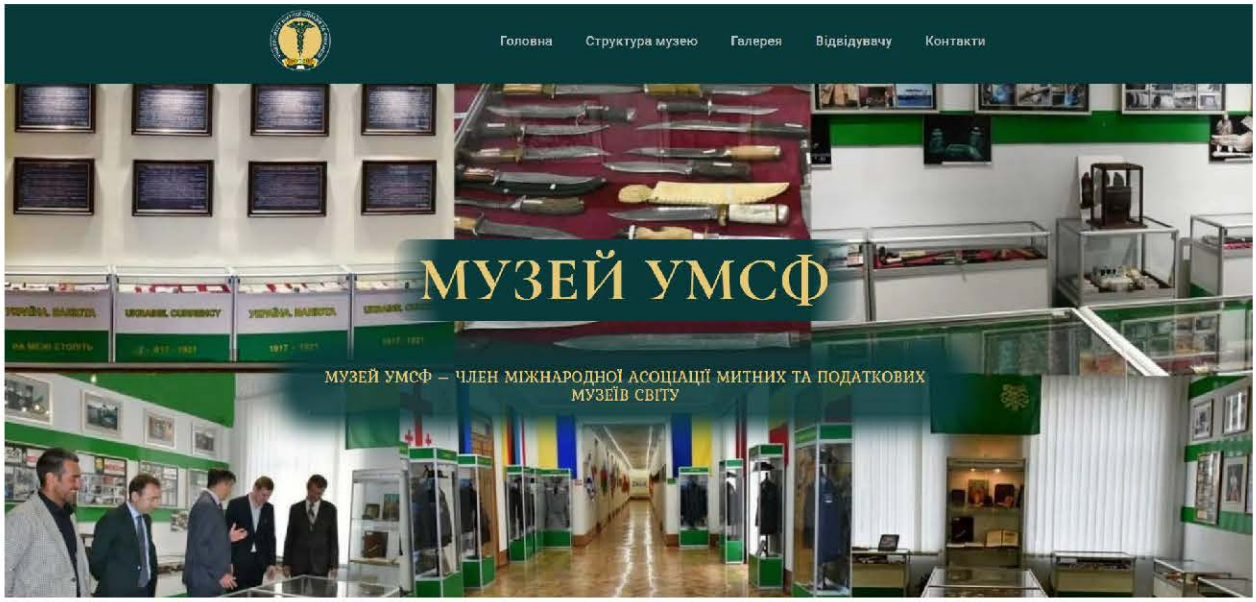

Рисунок 1.2 – Майбутній сайт музею

Зважаючи на отримані висновки, було сформовано загальну концепцію функціоналу майбутнього сайту музею, яка передбачає інтерактивність, зручне управління контентом, адаптивність для різних пристроїв, а також можливість інтеграції з мобільними застосунками та освітніми сервісами.

На етапі розробки ПЗ чітке формулювання вимог [9] є ключовим для успішного проєктування ІС. Без належного розподілу вимог на функціональні, нефункціональні та технічні існує ризик створити рішення, що не відповідатиме очікуванням користувачів і вимогам до якості.

Щоб мінімізувати подібні ризики, необхідно заздалегідь систематизувати вимоги, чітко розподіляючи їх за типами та пріоритетами.

Зазвичай вимоги поділяють на три основні категорії: функціональні, нефункціональні та технічні. Такий поділ дозволяє систематизувати очікування користувачів і замовників, а також забезпечити ефективну

комунікацію між розробниками, тестувальниками та іншими учасниками проєкту.

Функціональні вимоги описують, що саме повинна робити система: підтримувати авторизацію, керувати контентом, формувати HTML-сторінки тощо. Вони безпосередньо випливають з цілей і задач проєкту.

На основі поставленого завдання сформульовано наступні функціональні вимоги, як показано в табл. 1.1.

Таблиця 1.1

Функціональні вимоги

Функціональна вимога	Опис вимоги
Авторизація та автентифікація	Реалізація механізмів входу/виходу користувача, скидання паролю
Рольова модель доступу	Поділ прав на: адміністратора, редактора, користувача
Управління експозиціями, новинами, анонсами	Створення, редагування, публікація та видалення записів
Управління мультимедіа	Можливість додавати зображення до записів, організація галерей
Адміністративний інтерфейс	Зручна адмін-панель для керування контентом
Робота з базою даних	Збереження, пошук, фільтрація та зв'язки між об'єктами
Генерація динамічних сторінок	Подача контенту у вигляді HTML з використанням шаблонів

Нефункціональні вимоги представляють собою сукупність характеристик, що визначають якість функціонування інформаційної системи, а не перелік її конкретних дій чи поведінкових сценаріїв. Вони формулюють очікування до того, як саме повинні реалізовуватись функції системи з погляду її продуктивності, зручності використання, надійності та стійкості до зовнішніх загроз. Неврахування нефункціональних вимог може призвести до

збоїв, втрати даних або зниження загальної якості сервісу навіть за умови правильної реалізації функціональних компонентів.

До таких вимог зазвичай належать висока швидкодія, можливість масштабування, надійний захист даних, стабільна робота системи та зручність для користувачів, як показано в табл. 1.2.

Таблиця 1.2

Нефункціональні вимоги

Нефункціональна вимога	Вимога
Масштабованість	Можливість розширення функціоналу без перебудови архітектури
Продуктивність	Оптимізація запитів до бази даних
Надійність	Можливість відновлення після збоїв, коректна обробка винятків
Зручність підтримки	Документований код, модульність
Тестування	Наявність ручного тестування
Безпека	Захист від CSRF, SQL-ін'єкцій; хешування паролів

Технічні вимоги відіграють роль своєрідного каркасу для реалізації як функціональних, так і нефункціональних характеристик програмного забезпечення. Вони визначають основні параметри, яких слід дотримуватись під час проєктування, розробки та тестування системи. Такі вимоги забезпечують узгодженість між очікуваннями замовника, архітектурними рішеннями та кінцевою реалізацією продукту.

У процесі розробки веб-сайту музею Університету митної справи та фінансів було визначено низку ключових технічних вимог, що охоплюють архітектурні, інтерфейсні, безпекові та продуктивні аспекти. Їх формалізація дозволила чітко окреслити межі проектних рішень, запобігти ризикам невідповідності очікувань та забезпечити масштабованість системи в майбутньому.

Систематизований перелік технічних вимог до розроблюваної системи наведено в таблиці 1.3.

Таблиця 1.3

Технічні вимоги

Технічна вимога	Опис вимоги
Мова програмування	Сучасна мова програмування
Фреймворк	Обраний фреймворк повинен ефективно будувати багаторівневі веб-системи
База даних	Легка та ефективна БД
Архітектура	Модель архітектури повина розділяти логіку представлення, обробки даних та контролю
Інструменти	Використання сучасних середовищ розробки
ORM	Вбудована ORM для взаємодії з базою
Адмін-панель	Система повина містити інтуїтивно зрозумілу адміністративну панель
Розгортання	Локально, з можливістю міграції на сервер

Врахування всіх трьох категорій вимог, функціональних, нефункціональних та технічних, дає змогу створити програмне забезпечення, яке не лише виконує необхідні завдання, але й відповідає очікуванням користувачів за швидкодією, зручністю, надійністю та масштабованістю. Узгодження цих вимог сприяє досягненню балансу між технічними можливостями та бізнес-цілями проєкту.

Крім того, чітке та детальне документування вимог, а також їх подальший систематичний аналіз створюють міцну основу для ефективної комунікації між усіма учасниками розробки — замовниками, розробниками, тестувальниками та адміністраторами системи. Такий підхід сприяє єдиному розумінню цілей і задач проекту, забезпечуючи узгодженість дій та оперативне реагування на зміни.

Впроваджені заходи значно знижують ризики виникнення непорозумінь і помилок у процесі розробки, що в результаті підвищує якість кінцевого продукту. Це також дозволяє дотримуватися встановлених термінів виконання робіт та контролювати бюджет, забезпечуючи успішне і своєчасне завершення проєкту відповідно до запланованих параметрів.

1.4 Визначення цільової аудиторії, об'єкта проєктування

Правильне визначення цільової аудиторії є ключовим етапом у проектуванні будь-якої ІС, зокрема веб-сайту для музею вищого навчального закладу [10].

Від розуміння потреб, очікувань та поведінкових особливостей потенційних користувачів залежить ефективність взаємодії з цифровим продуктом, обґрунтованість функціональних рішень, а також загальний рівень задоволеності користувачів.

Цільову аудиторію веб-сайту музею УМСФ та фінансів можна умовно розділити на:

а) студенти;

б) викладачі та наукові працівники;

в) адміністрація музею та контент-редактори;

г) зовнішні користувачі (відвідувачі, абітурієнти, батьки).

Найчисельнішу частину аудиторії веб-ресурсу музею становлять здобувачі вищої освіти. Саме вони є найбільш активними користувачами сучасних цифрових платформ і щоденно звертаються до онлайн-сервісів для навчання, саморозвитку та дозвілля.

Молодь очікує швидкої навігації, мультимедійного наповнення та інтерактивних можливостей — таких як віртуальні екскурсії, фотогалереї, відеоматеріали чи оголошення про події. Важливу роль для цієї категорії відіграє адаптивність сайту до мобільних пристроїв, зручний інтерфейс та

яскравий візуальний супровід, що значною мірою визначають зацікавленість і тривалість взаємодії з платформою.

До наступної категорії належать викладацький склад та представники наукової спільноти. Для них важлива можливість отримати доступ до перевіреної, тематично впорядкованої інформації — цифрових копій експонатів, архівних описів, супровідних текстів. Такий контент часто використовується під час підготовки лекцій, семінарів або наукових публікацій.

Працівники музею та контент-менеджери є тими користувачами, які забезпечують постійне оновлення й наповнення ресурсу. Вони щоденно взаємодіють із системою для додавання матеріалів, публікації актуальних новин, підготовки описів до виставкових об'єктів, а також завантаження фото- чи відеофайлів. Тому для цієї групи особливо важливою є наявність зручного внутрішнього середовища для роботи з контентом.

До зовнішньої аудиторії належать особи, які не є частиною навчального процесу, але проявляють інтерес до музею — зокрема абітурієнти, члени родин студентів, мешканці регіону або випадкові відвідувачі. Такі користувачі розглядають цей сайт, як перше джерело інформації про університет і його культурне життя. Важливо, щоб платформа містила зрозумілу структуру, стислий огляд діяльності музею, контакти, карту розташування та фотоматеріали, які формують уявлення про атмосферу закладу.

Враховуючи, що серед користувачів сайту можуть бути не лише україномовні, доцільно було б реалізувати багатомовну підтримку, зокрема українську та англійську версії. Цей крок полегшить доступ до інформації для іноземної аудиторії і допоможе розширити охоплення.

Важливо також дотримуватись вимог доступності: забезпечити читабельність інтерфейсу, можливість збільшення шрифту, використання коректних кольорових контрастів і підтримку програм для озвучення вмісту.

Отже, ретельний аналіз потреб кожної групи користувачів дав змогу точно визначити вимоги до структури інтерфейсу, рівнів доступу, функціонального наповнення та здатності системи до подальшого розширення.

Що ж до розширення аудиторії після створення сайту, то важливу роль у цьому відіграють соціальні мережі. Через популярні платформи, такі як Instagram, Facebook чи Telegram, музей отримує можливість оперативно інформувати користувачів про нові експозиції, події та важливі новини.

Регулярні публікації, цікаві історії та візуальний контент допомагають підтримувати інтерес аудиторії, створюючи постійний зв'язок між музеєм і його відвідувачами.

Активна взаємодія з цільовою аудиторією не лише розширює коло відвідувачів, але й формує основу для подальшого розвитку цифрових сервісів музею. Саме на цій базі формується об'єкт проєктування.

Об'єктом проєктування є система адміністрування музейного веб-сайту. Вона забезпечує централізоване зберігання, адміністрування та подання цифрового музейного контенту. Даний об'єкт належить до класу інформаційних систем культурно-освітнього призначення, що слугують інструментом підтримки роботи освітніх установ.

Основним завданням таких систем є збереження, організація та презентація цифрового контенту у формі, зручній для широкого кола користувачів.

До основних характеристик проєкту відносяться його тип, цільове призначення, функціональні можливості, архітектурна структура та технічна реалізація.

За своїм типом сайт УМСФ належить до спеціалізованих веб-орієнтованих платформ контент-менеджменту, які забезпечують зберігання, редагування, категоризацію та подання цифрової інформації в інтерактивному форматі.

Роль і місце об'єкта в системі полягає в тому, що саме серверна частина є функціональним ядром веб-сайту музею. Вона виступає ключовим компонентом архітектури, що поєднує між собою всі інші елементи ІС. Через сервер реалізується централізоване управління контентом, а також відбувається обробка всіх запитів, які надходять від користувачів через клієнтський інтерфейс.

Головною метою створення серверної частини є цифровізація музейних експозицій та створення зручного інструменту для їх структурованого збереження й управління.

Це дозволяє виконати низку ключових завдань:

а) довгострокове збереження експонатів у цифровому вигляді;

б) відкритий доступ до культурної спадщини для широкого кола користувачів;

в) підвищення публічної присутності музею у навчальному та науковому середовищі.

Забезпечення довгострокового збереження музейних експонатів у цифровому форматі гарантує зберігати колекції без ризику їх пошкодження чи зникнення з часом, а саме уникнути фізичних пошкоджень чи виведення з ладу через природні процеси, такі як старіння матеріалів.

Відкритий доступ до музейних експонатів дає можливість знайомитися з ними не тільки для тих, хто фізично перебуває в музеї, а й для користувачів, що не мають змоги відвідати його особисто. Завдяки впровадженню передових цифрових рішень, музейний контент стає доступним у будь-який час доби, що дозволяє людям з різних куточків світу отримувати інформацію та насолоджуватися культурними цінностями без обмежень.

Третім важливим аспектом цифровізації музейної колекції є активне залучення музею до освітнього та наукового простору. Оцифровані експонати, доступні через веб-ресурс, стають не лише джерелом культурної інформації, а й важливим інструментом для навчального процесу.

Вони використовуються як ілюстративний матеріал у школах, вишах та наукових установах, де студенти можуть досліджувати предмети мистецтва, археології, етнографії та інших напрямів без необхідності фізичної присутності в музеї.

Таким чином, оцифрована система адміністрування музейного веб-ресурсу виконує роль не лише сховища цифрових матеріалів, а й стає потужним засобом комунікації між музеєм і широкою аудиторією. Завдяки цьому об'єкт проєктування набуває особливої значущості як ключовий інструмент у процесі цифрової модернізації установ культурно-освітнього призначення.

1.5 Висновки до першого розділу

У першому розділі дипломної роботи було детально розглянуто предметну область цифрових музейних систем і окреслено роль музею УМСФ як складової освітньої та культурної екосистеми університету. Показано, що впровадження цифрових технологій у музейну діяльність є не просто актуальним трендом, а нагальною потребою сучасності, що обумовлена глобальними викликами — пандемією COVID-19, військовими конфліктами, процесами глобалізації.

На підставі детального аналізу сформульовано основні функції та технічні вимоги до серверної частини веб-сайту музею. Переважна увага приділяється створенню надійної системи збереження, організації та впорядкування цифрового контенту різноманітних форматів — текстового, візуального та аудіовізуального, що забезпечує ефективну роботу з великими обсягами інформації.

Крім того, визначено потребу в інтегрованих інструментах адміністрування експозицій, які забезпечують зручне управління мультимедійними матеріалами. Важливою складовою системи є багаторівнева

модель доступу, що розмежовує права користувачів від адміністраторів до звичайних відвідувачів, дозволяючи гнучко контролювати операції редагування, перегляду та публікації контенту.

Особливим акцентом ставилося питання масштабованості розроблюваної системи, яке є визначальним для її подальшого розвитку. Проєкт має передбачати можливість легкого додавання нових функціональних модулів без необхідності глобальної перебудови існуючої архітектури.

В рамках дослідження здійснено глибокий огляд архітектурних рішень, безпекових стандартів і функціональних особливостей серверної частини, яка виступає центральним елементом цифрової музейної платформи. Її роль розглядається як інтегруючого механізму для управління всіма даними та процесами, зокрема роботою баз даних, логікою обробки запитів, взаємодією з клієнтською частиною сайту.

Додатково проведено комплексний аналіз сучасного стану цифрової трансформації музейних закладів. У цьому контексті виявлено як успішні приклади впровадження інноваційних інформаційних систем, так і типові проблеми.

Порівняльний аналіз ж дозволив не лише виокремити найкращі практики, але й уникнути поширених помилок у власній розробці, підвищуючи шанси на успіх проєкту.

Узагальнюючи викладене, можна стверджувати, що створення серверної частини веб-платформи музею УМСФ є не просто реалізацією технологічного проєкту, а стратегічним кроком у напрямку інтеграції закладу в цифрову культурну інфраструктуру України.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ, ЇХ АНАЛІЗ, ВИБІР МЕТОДІВ

2.1 Обґрунтування вибору засобів та методів розробки ПЗ

При розробці серверної частини веб-сайту музею було прийнято рішення використовувати мову програмування Python як основний інструмент. Вона була визнана однією із найпопулярніших і найзручніших для розробників мов програмування завдяки простоті синтаксису, широкій екосистемі бібліотек і можливостям швидкої розробки.

За даними аналітичного порталу (рис. 2.1) Stack Overflow Developer Survey 2024, Python входить до найпопулярніших мов серед розробників, а його застосування в веб-розробці зростає завдяки фреймворкам, таким як Django і Flask.

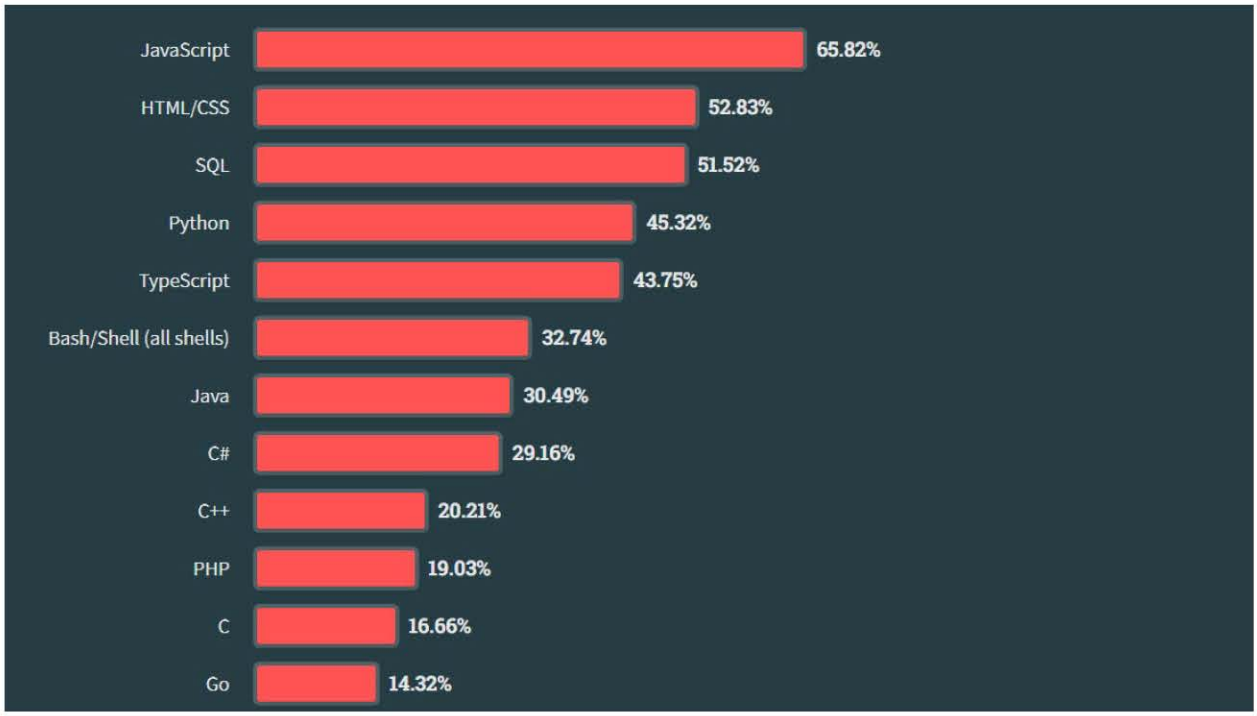

Рисунок 2.1 – Аналітика Stack Overflow 2024 [11]

Python підтримує парадигму об'єктно-орієнтованого програмування (ООП), що є однією з основних причин його популярності серед розробників.

ООП дозволяє організовувати код у вигляді об'єктів, які об'єднують дані та методи для роботи з цими даними в одному місці.

У порівнянні з іншими популярними мовами, такими як PHP, JavaScript або Java, Python [12] пропонує баланс між простотою та потужністю. Наприклад, PHP часто використовується в CMS (WordPress, Joomla), але має складнішу синтаксичну структуру і менш сучасний підхід до розробки.

JavaScript, у свою чергу, є незамінним інструментом для фронтенд-розробки веб-додатків, а також набирає популярності на серверній стороні завдяки Node.js. Однак, асинхронна архітектура Node.js вимагає ретельного планування, що додає додаткової складності розробникам. Java ж відома своєю високою стабільністю та надійністю, часто використовується у великих корпоративних рішеннях, але розробка на цій мові потребує значних часових і ресурсних витрат, що може бути недоцільним для швидкого створення прототипів чи невеликих проєктів.

Для рендерингу HTML-сторінок було обрано шаблонізатор Jinja2, який є потужним і гнучким інструментом, що дозволяє ефективно розділяти бізнес-логіку і представлення. Jinja2 підтримує розгалуження, цикли, наслідування шаблонів і фільтри, що робить його зручним для створення складних і динамічних інтерфейсів.

Щодо системи управління БД, на початковому етапі було застосовано SQLite [13] через її простоту використання і відсутність потреби в налаштуванні окремого серверу. Це значно спростило розгортання і тестування проєкту на локальній машині розробника.

Однак для виробничого середовища передбачено перехід на PostgreSQL [14] — сучасну, масштабовану і надійну реляційну систему, яка підтримує складні запити, транзакції, реплікацію і розширені типи даних.

Адміністрування сайту реалізовано через стандартну адміністративну панель Django Admin, яка є потужним інструментом для керування контентом

і користувачами. Для підвищення зручності роботи кураторів додані розширення — Grappelli та CKEditor.

Вибір даних технологій зумовлений їхньою стабільністю, поширеністю та активною підтримкою спільноти, що є критично важливим для довгострокового супроводу і розвитку проєкту.

Сучасні засоби створюють міцну основу для розробки веб-застосунку, однак цього недостатньо. Щоб досягти ефективного результату, важливо грамотно організувати процес — обрати методи, що визначають етапи роботи, командну взаємодію та впровадження змін.

Організація ефективного процесу розробки ПЗ вимагає ретельного вибору методології, яка максимально відповідає специфіці проєкту, доступним ресурсам. У сфері ІТ найбільш поширеними підходами є каскадна, спіральна модель та гнучкі методології, відомі під загальною назвою Agile.

Каскадна модель (Waterfall) передбачає поетапне, лінійне виконання усіх стадій проєкту — від збору вимог до впровадження й підтримки. Такий підхід доцільний для проєктів зі статичними вимогами.

Проте основним недоліком каскадної моделі є її жорстка структура, яка не дозволяє адаптуватися до змін на будь-якому етапі. Якщо в процесі розробки виникає потреба внести зміни або коригування, це може значно затримати проєкт і потребувати значних ресурсів для внесення змін у попередні етапи.

Таким чином, каскадна модель менш підходить для проєктів, де вимоги можуть змінюватися або виникають непередбачувані обставини, що вимагають гнучкої реакції на зміни.

Спіральна модель поєднує ідеї ітеративної розробки та оцінки ризиків, забезпечуючи поступове нарощування функціональності. Вона особливо доречна для проєктів із значною невизначеністю, де необхідне багаторазове уточнення.

Однак, спіральна модель має й недоліки. Одним із головних є її структурна складність, що ускладнює організацію процесу розробки та планування. Часто через численні етапи оцінки ризиків і багаторазові коригування результатів може виникнути затримка у досягненні фінальної мети. Крім того, вимагає більшої кількості часу та ресурсів на постійну перевірку і перегляд розроблених етапів.

На відміну від каскадної та спіральної моделей, Agile (гнучкі методології) передбачає високий рівень гнучкості та адаптивності до змін. В Agile процес розробки розбивається на короткі ітерації (спринти), кожен з яких завершується створенням робочої частини продукту.

Обираючи між традиційною каскадною моделлю, спіральною та Agile, вибір зупинили на останньому (рис. 2.2), оскільки він забезпечує більшу гнучкість у процесі розробки та швидке реагування на зміни вимог.

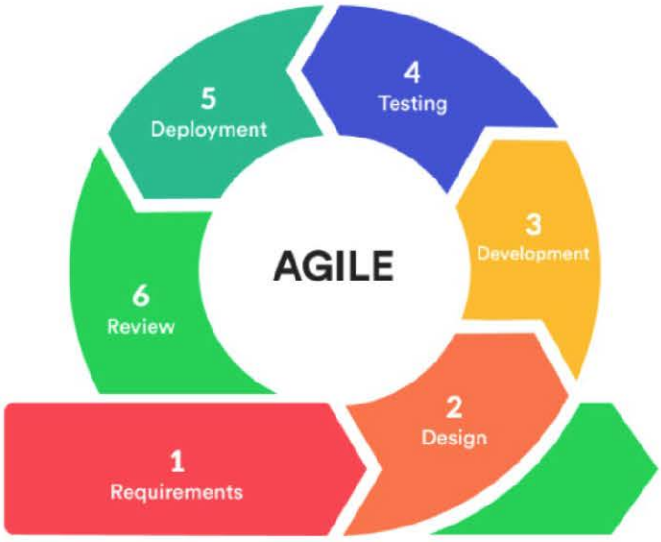

Рисунок 2.2 – Agile [15]

Ітеративність проявлялась через поетапну розробку з поступовим розширенням функціоналу. Кожен цикл завершувався створенням конкретного модуля (наприклад: авторизація, БД, адміністративна панель), а

наступним кроком ставало тестування з перевіркою відповідності всім необхідним вимогам.

Пріоритет на стороні користувача полягав у зосередженні уваги на зручності адміністрування сайту. Співробітники музею активно впливали на зміст і форму функціональності: зокрема, на основі їх побажань реалізовано підтримку текстового редактора CKEditor з візуальним інтерфейсом, що значно спростило наповнення контенту.

Постійна комунікація з кураторами й потенційними користувачами дала змогу оперативно оновлювати вимоги до інтерфейсу та функціональних можливостей.

Систему організації задач побудовано на основі Kanban-моделі, реалізованої в середовищі Jira. Інтерактивна дошка з поділом завдань забезпечила зручну навігацію. Такий підхід дозволив:

а) чітко візуалізувати стан проєкту, що стало важливим чинником під час одночасної реалізації кількох задач;

б) оптимізувати перемикання між завданнями без втрати продуктивності;

в) збалансувати навантаження на основі розуміння послідовності та взаємозалежності етапів;

г) мінімізації простоїв завдяки можливості гнучко перерозподіляти задачі між учасниками.

Використання підходу Agile у межах проєкту забезпечило ефективне управління змінами, які виникали в процесі реалізації. Такий підхід значно зменшив ризик технічних збоїв у фінальній версії проєкту.

2.2 Порівняльний огляд CMS та фреймворків

Під час проєктування та розробки серверної частини сайту було розглянуто низку сучасних інструментів для створення веб-сайтів, які умовно

можна поділити на дві категорії: готові системи управління контентом (CMS) та фреймворки для веб-розробки.

Готові CMS (рис. 2.3), такі як WordPress, Joomla, Drupal, — дозволяють швидко створювати веб-сайти з базовим функціоналом, не заглиблюючись у програмування. Вони мають вбудовані засоби керування контентом, шаблонні теми оформлення та модулі для поширених сценаріїв.

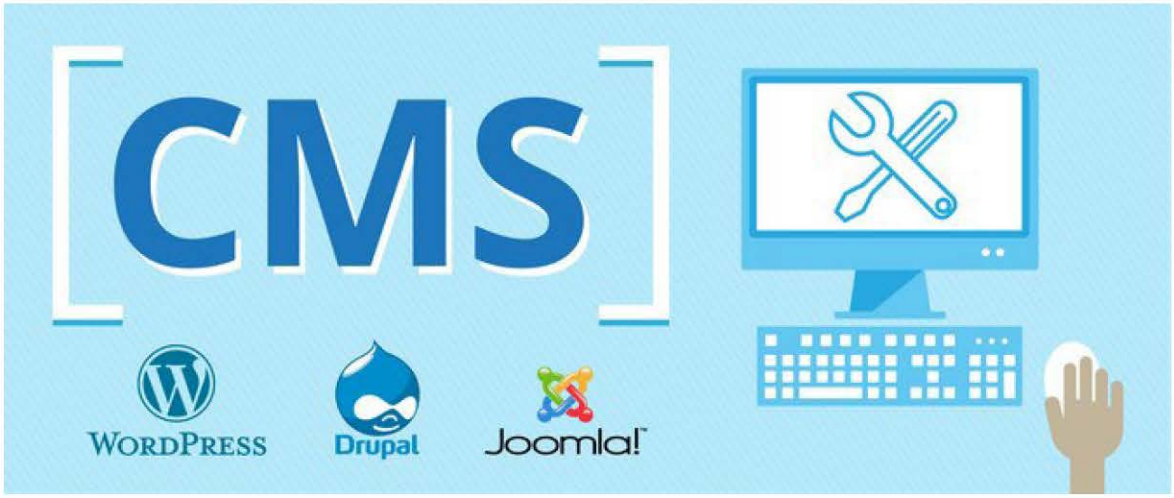

Рисунок 2.3 – WordPress, Joomla, Drupal [16]

WordPress побудований на мові програмування PHP та використовує в якості системи управління БД MySQL [17]. Він пропонує:

а) широку екосистему плагінів і тем (>60 000) для розширення функціоналу й зміни дизайну;

б) зручний інтерфейс редактора Gutenberg для створення контенту;

в) велику спільноту;

г) регулярні оновлення безпеки;

До обмежень WordPress можна віднести кілька важливих аспектів, які можуть вплинути на ефективність роботи сайту:

а) архітектура плагіну часто призводить до конфліктів версій і зависань сайту при значній кількості розширень;

б) базове ядро не призначене для складних типів контенту (наприклад, багаторівневі зв'язки між експонатами та галереями) без додаткового коду чи комерційних плагінів;

в) обмежена масштабованість, так як при зростанні числа запитів продуктивність падає через одночасне виконання великої кількості PHP-скриптів.

Joomla [18] теж побудована на PHP і MySQL, але вирізняється суворішою архітектурою MVC та розподілом системи на компоненти, модулі та плагіни. Вона призначена для більш структурованого підходу до керування вмістом.

До переваг Joomla можна віднести:

а) гнучку систему контролю доступу (ACL) із можливістю налаштування прав для різних груп користувачів;

б) вбудовану підтримка багатомовності без потреби у сторонніх розширеннях;

в) широкий вибір модулів для побудови портальних рішень, каталогів.

До обмежень Joomla можна віднести такі фактори:

а) складніший поріг входження для користувачів без технічної підготовки;

б) менша кількість якісних безкоштовних шаблонів і розширень у порівнянні з WordPress;

в) менш інтуїтивна адмін-панель, що може ускладнити роботу музейних кураторів без досвіду.

Drupal — потужна і гнучка CMS, орієнтована на створення складних структур контенту [19]. Drupal дозволяє формувати схему даних із підтримкою багатьох типів полів, зв'язків та ролей.

Переваги Drupal:

а) можливість створення глибоко структурованого контенту без написання коду;

б) високий рівень безпеки: використовується для урядових і корпоративних сайтів;

в) гнучка система ролей і дозволів, підтримка багатомовності та класифікацій.

Обмеження Drupal:

а) високий поріг входження: вимагає спеціалізованих знань навіть для базових налаштувань;

б) менше готових шаблонів і модулів для візуальної частини сайту;

в) більш тривалий цикл розробки порівняно з WordPress чи Joomla.

Однак для реалізації специфічної логіки, пов'язаної зі структурованим контентом, ролями користувачів і взаємозв'язаними даними, стандартні CMS часто виявляються обмеженими.

Для прикладу, інтерфейс користувача в CMS не завжди підтримує достатню гнучкість для налаштування специфічних умов доступу або реалізації складних взаємодій між елементами, що може бути критичним у контексті сайту музею з великою кількістю взаємопов'язаних даних.

Натомість веб-фреймворки, такі як Django, Flask або Pyramid, дозволяють створювати повністю індивідуально налаштовані рішення, які максимально відповідають вимогам проєкту. Вони надають засоби для моделювання бази даних, реалізації серверної логіки, шаблонного рендерингу та контролю доступу.

Загалом, хоча всі три CMS мають певні переваги, для створення функціонального та масштабованого рішення для сайту музею їх можливостей недостатньо.

Потреба у складній структурі даних, ролях користувачів, повного налаштування бізнес-логіки та мінімальній залежності від сторонніх плагінів зумовила доцільність вибору повнофункціонального веб-фреймворку.

Згідно з аналітичним оглядом “2025's Top 10 Python Web Frameworks Compared” (рис. 2.4), перші три місця серед Python-фреймворків для бекенд-розробки займають Django, FastAPI та Flask.

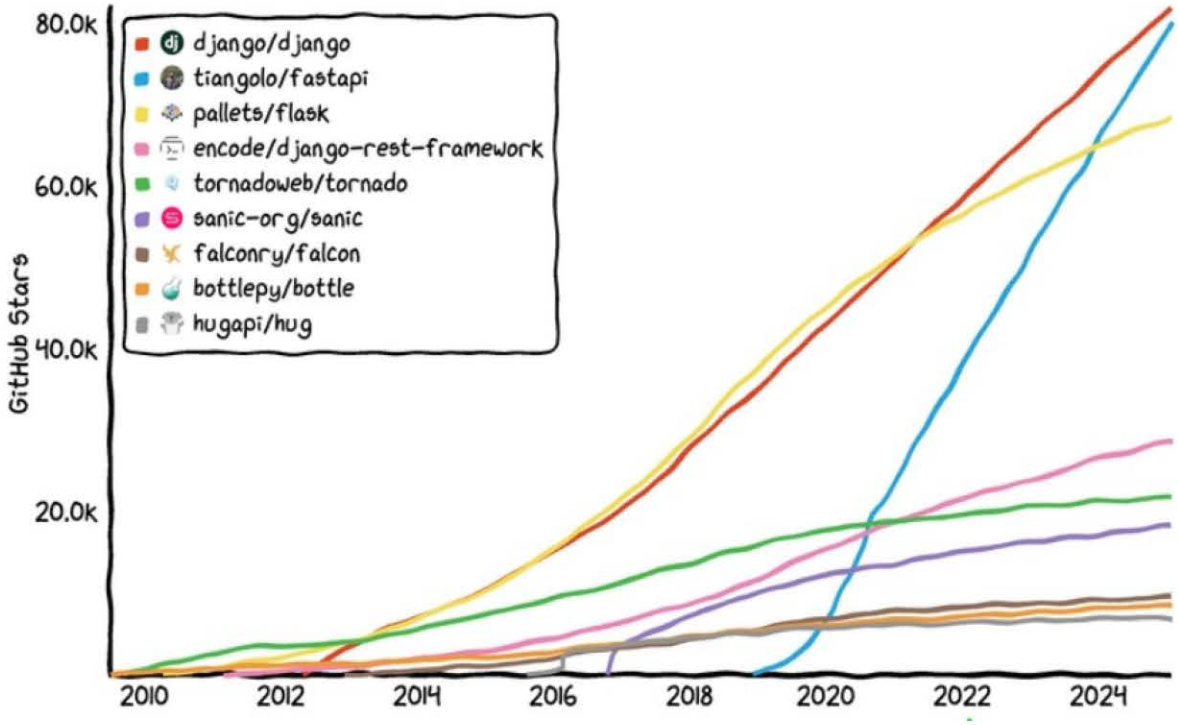

Рисунок 2.4 – Лідери серед Python-фреймворків [20]

Вони вважаються найбільш популярними серед розробників завдяки своїм технічним можливостям, активним спільнотам підтримки та широкому спектру застосування у веб-проєктах різного масштабу.

Django — це високорівневий веб-фреймворк на мові Python, що реалізує архітектуру MTV (Model–Template–View) та орієнтований на швидку розробку безпечних, масштабованих веб-застосунків [21].

До його переваг слід віднести:

а) вбудована ORM для зручної роботи з БД;

б) генерація адміністративної панелі «з коробки»;

в) захист від поширених атак (CSRF, XSS, SQL Injection);

г) повна система автентифікації, реєстрації, розмежування доступу;

д) підтримка міграцій, багатомовності.

Його обмеження [22]:

а) відносно «важкий» для простих або сайтів, де всього одна сторінка;

б) обмежена асинхронна підтримка у стандартній конфігурації.

FastAPI — це сучасний асинхронний веб-фреймворк для створення RESTful API на Python [23]. Його основною особливістю є підтримка типізації та автоматичне створення документації API.

Переваги:

а) висока продуктивність завдяки підтримці асинхронного програмування через async/await;

б) інтеграція з Pydantic для валідації даних, що допомагає забезпечити коректність і безпеку даних, які передаються через API;

в) автоматична генерація Swagger UI/OpenAPI документації, що значно спрощує інтеграцію з іншими системами та дозволяє тестувати API безпосередньо через браузер.

Обмеження:

а) відсутність вбудованої адмін-панелі, що ускладнює керування контентом без додаткових інструментів;

б) оскільки фреймворк орієнтований на створення API, він менш підходить для розробки класичних веб-сайтів з повноцінними інтерфейсами для користувачів;

в) незважаючи на високий рівень продуктивності, складність для новачків у розумінні всіх можливостей асинхронного програмування може бути перепоною для швидкого освоєння фреймворку.

Flask — це мікро-фреймворк Python, який надає базовий мінімум для створення веб-застосунків, надаючи розробнику повну свободу у виборі сторонніх рішень [24].

Переваги:

а) легкий та гнучкий, що дозволяє швидко створювати проєкти з мінімальними затратами часу на налаштування та конфігурацій;

б) добре підходить для невеликих застосунків та прототипів, дозволяючи швидко протестувати ідеї та концепти в реальному часі;

в) проста структура та швидкий старт, що робить Flask доступним для новачків, які тільки починають знайомитись з веб-розробкою.

Обмеження:

а) більшість функцій потрібно реалізовувати вручну, що може зайняти більше часу при створенні складних проєктів, оскільки розробник має контролювати всі аспекти роботи системи;

б) відсутність штатної адмін-панелі та системи міграцій, що додає додаткові складнощі при роботі з великими базами даних;

в) складно масштабувати проєкт при зростанні, оскільки Flask не надає вбудованих інструментів для обробки великих навантажень і автоматизації багатьох процесів.

Зважаючи на особливості проєкту — розробку інформаційного сайту музею з підтримкою шаблонів для сторінок, зручної адміністративної панелі для кураторів, різних ролей користувачів, складної структури експозицій та безпечної обробки запитів — найбільш доцільним було обрати саме Django.

На відміну від інших, він пропонує комплексне рішення «з коробки», яке поєднує в собі всі необхідні інструменти для створення інформаційного сайту музею — підтримку шаблонів сторінок, зручну адміністративну панель для кураторів, ролі користувачів.

2.3 Обґрунтування архітектури системи

Архітектура проєкту побудована на модульному підході, який реалізований через розподіл функціональності на окремі Django-додатки (apps).

Окремий додаток відповідає за відповідний сегмент системи. Такий підхід впливає на колективну розробку та полегшує роботу, оскільки дозволяє ефективно розподіляти завдання між учасниками команди. Кожен із розробників може працювати над окремим додатком (модулем), що мінімізує конфлікти під час об'єднання змін у Git-репозиторії.

У межах кожного додатку визначаються власні моделі, представлення (views), шаблони і маршрути (urls).

Таке розділення дозволяє:

а) дозволяє легко розширювати систему шляхом додавання нових модулів без ризику порушення існуючої логіки та коду;

б) гарантує ізоляцію даних і бізнес-логіки різних компонентів, що підвищує стабільність і безпеку системи;

в) сприяє більш ефективній командній роботі, оскільки кожен розробник може зосередитися на окремому додатку, що спрощує управління завданнями і зменшує конфлікти в коді.

Беручи до уваги зазначені принципи розподілу функціональності, серверна складова веб-сайту музею була побудована на основі архітектурної парадигми Model–Template–View (MTV).

Така архітектура забезпечує чітке розділення обов’язків між логікою бізнесу, представленням даних та їх обробкою.

Архітектурні елементи MTV-парадигми включають:

а) моделі (Model) — описують структуру бази даних (експозиції, категорії, зображення, користувачі, новини тощо);

б) представлення (View) — обробляють запити користувачів, формують відповідь і взаємодіють із моделями;

в) шаблони (Template) — відповідають за генерацію HTML-коду на основі динамічних даних.

Окрім архітектурної парадигми Model–Template–View (MTV), у веб-розробці часто використовуються й інші моделі, такі як Model–View–

Controller (MVC), Model–View–Presenter (MVP). MVC, що є популярною моделлю, схожа з MTV, проте у ній чіткіше розділені контролер та представлення, що іноді ускладнює розробку для невеликих проєктів. MVP більше орієнтована на відокремлення логіки представлення від бізнес-логіки, що корисно для складних інтерфейсів, але може бути надмірною для серверних додатків.

Архітектура MTV (рис. 2.5) ж найкраще підходить для проєкту серверної частини музею, оскільки вона забезпечує просте і зрозуміле розділення обов'язків між моделями даних, шаблонами для відображення та логікою обробки запитів.

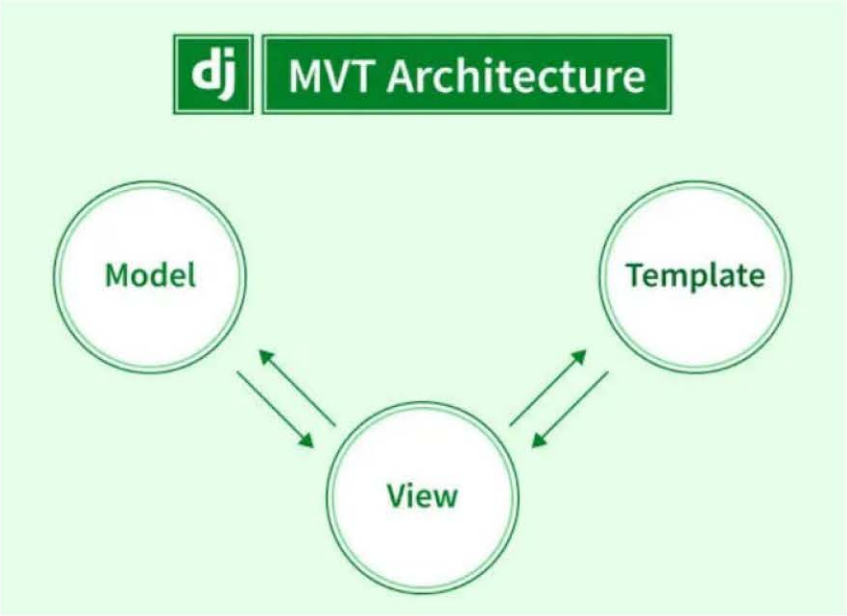

Рисунок 2.5 – Архітектура Model–Template–View [25]

Дані компоненти взаємодіють через Django ORM, що спрощує запити до бази даних і підтримує цілісність інформації. Всі дані розділені за логікою додатків, що дозволяє масштабувати проєкт без порушення загальної структури.

ORM (Object-Relational Mapping) — це технологія, яка забезпечує зв'язок між об'єктами мови програмування та реляційною базою даних [26]. Замість

того, щоб писати складні SQL-запити вручну, розробник працює з об'єктами, які автоматично пов’язані з відповідними таблицями в базі даних.

У Django реалізація ORM відбувається через модулі та моделі. Кожна модель визначається як клас Python у файлі models.py, і кожен клас відповідає таблиці в базі даних.

Наприклад, у класі Exposition описуються два поля (рис. 2.6):

а) title — текстове поле з обмеженням довжини до 200 символів;

б) description — текстове поле для зберігання детального опису.

```
class Exposition(models.Model):  8 usages
    title = models.CharField( *args: 'Заголовок запису', max_length=200)
    description = RichTextField('Текст запису')
    photo = models.ImageField( verbose_name: 'Зображення', upload_to='exposition_photos/')

    class Meta:
        verbose_name = 'Експозиція'
        verbose_name_plural = 'Експозиції'
```

Рисунок 2.6 – Код класу Exposition

Після визначення моделей фреймворк автоматично створює відповідні таблиці в базі даних під час виконання міграцій.

2.4 Планування реалізації проєкту та організація команди

Розробка здійснювалась на основі попередньо визначеного плану з урахуванням гнучкого підходу до управління розробкою. Весь процес було поділено на логічні етапи, що охоплювали повний життєвий цикл ПЗ — від аналізу вимог до тестування й підготовки до розгортання [27].

Основні етапи реалізації включали:

а) збір і формалізація вимог: вивчення потреб музейної системи, визначення функціональних модулів, узгодження логіки взаємодії з користувачами;

б) проєктування структури даних: побудова моделей об'єктів (експозиції, медіа-контент, публікації, користувачі) із застосуванням вбудованої ORM;

в) налаштування середовища розробки: створення робочого проєкту у середовищі PyCharm, ініціалізація локальної бази даних SQLite;

г) програмна реалізація: розробка представлень, маршрутизації, адміністративної панелі з підтримкою ролей та прав доступу, інтеграція CKEditor і Grappelli;

д) тестування та перевірка стабільності: ручне проходження ключових сценаріїв, обробка помилок, коригування логіки.

Організація командної взаємодії здійснювалась за допомогою онлайн-сервісу Jira (рис. 2.7), який надає широкі можливості для управління завданнями, моніторингу прогресу та контролю за дотриманням термінів виконання.

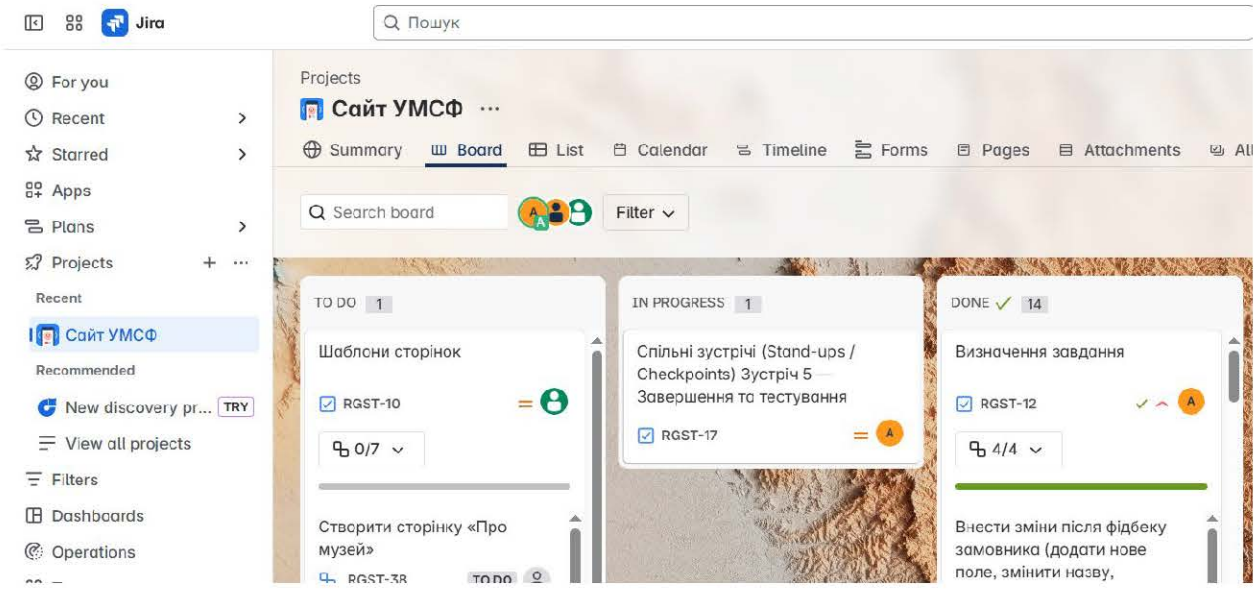

Рисунок 2.7 – Jira

Для координації роботи було створено Kanban-дошку, на якій задачі розподілялись за категоріями відповідно до їх поточного статусу: To Do (заплановані), In Progress (у роботі) та Done (виконані).

Kanban-дошка — це візуальний інструмент управління завданнями, який дозволяє організувати та контролювати робочий процес у реальному часі. Вона заснована на принципах методології Kanban [28], що передбачає поступовий і безперервний потік виконання задач без фіксованих спринтів (на відміну від Scrum). Його переваги використання:

а) прозорість усього процесу розробки;

б) швидке виявлення "вузьких місць";

в) покращення командної координації.

Scrum — це каркас для організації роботи в командах розробки ПЗ, та він передбачає ітеративну розробку шляхом розбиття роботи на короткі цикли, що називаються спринтами, зазвичай тривалістю від 1 до 4 тижнів [29]. До переваг Scrum зазвичай відносять:

а) часті релізи й можливість швидкої адаптації;

б) постійний зворотний зв'язок;

в) висока прозорість та залученість замовника;

г) безперервне вдосконалення процесу.

Так от, відповідно до логіки роботи Kanban-дошки, завдання переміщувались між відповідними стовпцями вручну в міру просування роботи: спочатку вони потрапляли в категорію To Do, де зберігалися всі заплановані, але ще не розпочаті задачі; після початку виконання завдання переносилося до стовпця In Progress, що свідчило про активну роботу над ним; по завершенні й перевірці результатів задачі переміщувалися до розділу Done.

Три основні учасники команди брали участь у процесі розробки ПЗ: фронтенд-розробник, бекенд-розробник, а також замовник.

Фронтенд-розробник займався створенням користувацького інтерфейсу, забезпечуючи комфортну та інтуїтивно зрозумілу взаємодію користувачів із сайтом. Він відповідав за дизайн, адаптивність та функціональність візуальної частини ресурсу на різних пристроях.

Бекенд-розробник відповідав за побудову серверної логіки, роботу з базою даних, обробку запитів і підтримку безпеки системи. Його завданням було гарантувати стабільність, надійність і ефективність роботи серверної частини.

Замовник виконував роль ініціатора проєкту, формулював вимоги, контролював відповідність кінцевого продукту поставленим цілям, а також затверджував дизайн та зміст.

Такий поділ обов'язків сприяв ефективній взаємодії в команді, забезпечував своєчасне внесення змін і високий рівень якості розробки.

Кожне завдання у Jira супроводжувалося коротким описом (рис. 2.8) суті роботи, позначеним терміном виконання, статусом і призначеним відповідальним.

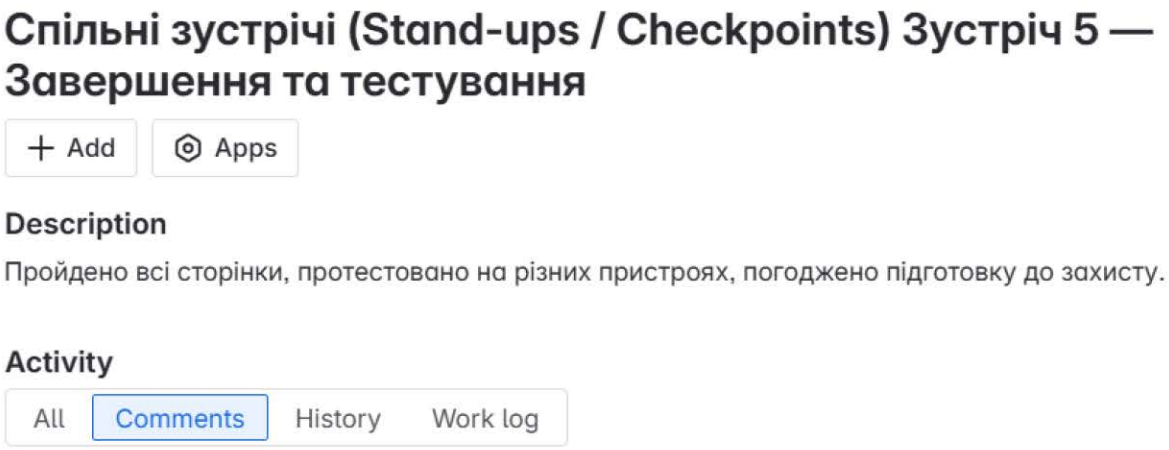

Рисунок 2.8 – Опис суті роботи у завданні

Запроваджена організація роботи сприяла підтримці високого рівня дисципліни в команді, дозволяла чітко розподіляти завдання між учасниками та забезпечувала гнучкість у процесі розробки, відповідно до методології Agile. Всі зміни у кодовій базі фіксувалися за допомогою системи контролю версій Git, що давало змогу зберігати повну історію розвитку проєкту,

відслідковувати внесені правки та при необхідності швидко повертатися до попередніх версій.

Віддалене співробітництво було організоване через платформу Github [30] (рис. 2.9).

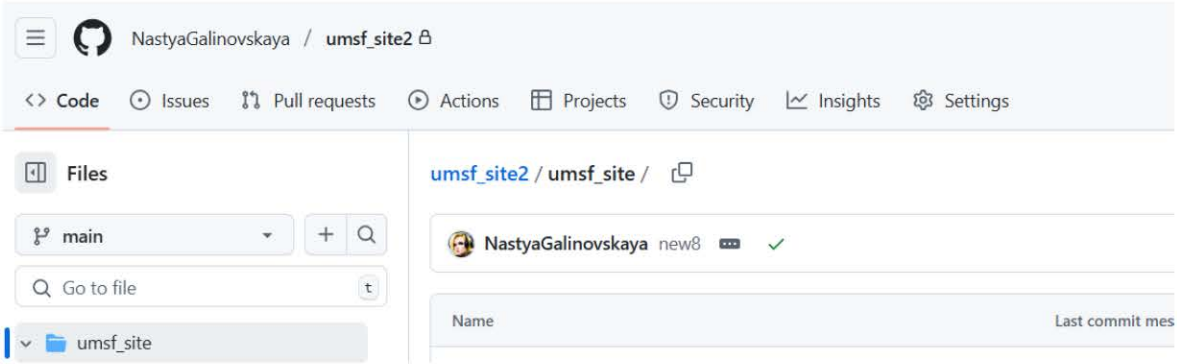

Рисунок 2.9 – Github

Репозиторій проєкту містив усі актуальні версії ПЗ, включаючи документацію, шаблони, конфігураційні файли тощо. Кожне оновлення супроводжувалося змістовними повідомленнями, що давало змогу швидко орієнтуватися у змінах, виявляти причини помилок або конфліктів.

Застосування багатогілкової структури (branching), тобто створення паралельних гілок коду в системі контролю версій, стало ключовим підходом до організації розробки. Такий підхід дозволив уникнути конфліктів у спільному коді, оскільки кожен розробник працював у межах окремої гілки, не впливаючи безпосередньо на основну версію проєкту.

Організація командної взаємодії в межах проєкту здійснювалася через використання запитів на злиття змін. Цей процес дозволяв іншим розробникам переглядати внесені зміни, перевіряти їх на відповідність вимогам, виявляти можливі логічні помилки, дублювання коду або відхилення від стилістичних норм. Система коментарів до коду забезпечувала двосторонній зворотний зв'язок – члени команди могли залишати зауваження або рекомендації безпосередньо в контексті конкретного фрагмента коду.

Загалом, впровадження чіткого механізму гілкування та узгодженого злиття коду дозволило налагодити злагоджену, безпечну та ефективну співпрацю, мінімізувати технічні ризики та забезпечити високу якість програмного забезпечення на всіх етапах його створення.

2.5 Висновки до другого розділу

У другому розділі дипломної роботи проведено огляд та аналіз існуючих інструментів для створення серверної частини веб-застосунків. Особливу увагу було приділено порівнянню CMS-систем (WordPress, Joomla) з фреймворками (Django, Flask, FastAPI), що дало змогу зробити наступні висновки:

а) CMS мають обмежені можливості масштабування, адаптації та безпеки, що не відповідає потребам музейної системи;

б) Django обрано як основний інструмент розробки завдяки його високому рівню інтеграції, наявності ORM, готового адміністративного інтерфейсу, підтримки авторизації та ефективної роботи з базами даних;

в) інструменти управління проєктом, такі як Jira та Git, дозволяють ефективно планувати та контролювати розробку;

г) чітке розділення логіки на рівні компонентів (моделі, представлення, шаблони) дозволяє забезпечити ізоляцію даних і функцій, що значно знижує ймовірність виникнення помилок.

д) обґрунтовано вибір архітектурного підходу до реалізації веб-додатка на базі клієнт-серверної моделі. Запропоновано структуру взаємодії компонентів: база даних — серверна логіка — клієнт.

У результаті розділу сформовано технічну основу для реалізації цільової системи та визначено оптимальні засоби, які дозволяють досягти цілей проекту з урахуванням безпеки, масштабованості та зручності в підтримці.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ РІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Опис структури Django-проєкту та моделей даних

Чітко організована структура дозволяє розробникам легко орієнтуватися в коді, швидко знаходити потрібні компоненти та уникати плутанини. Django-проєкт має базову структуру, яка включає кілька ключових компонентів [31]:

а) кореневу директорію проєкту umsf_site, яка містить основні конфігураційні файли, зокрема:

1) settings.py — налаштування бази даних, шляхів;
2) urls.py — конфігурація маршрутів на рівні всього проєкту;
3) wsgi.py / asgi.py — точки входу для запуску додатку через сервери;
4) __init__.py — файл ініціалізації пакету Python.

б) додатки (apps) або структурні модулі, кожен з яких відповідає за окрему функціональність системи:

1) expositions — управління експозиціями;
2) gallery — завантаження та відображення зображень;
3) blog — створення новин і анонсів.

в) окремі папки, що організовують ресурси:

1) media/ — папка для збереження файлів, таких як фотографії;
2) templates/ — папка з HTML-шаблонами;
3) static/ — папка для статичних файлів (CSS, JS, фото).

г) базу даних SQLite та файл manage.py;

д) папку venv, де знаходиться віртуальне середовище Python, яке ізолює залежності.

Кожен додаток містить окремі файли __init__.py, admin.py, apps.py, models.py, tests.py, views.py, urls.py, що забезпечують розділення різних обов'язків і зручність у розробці та підтримці коду. Винятком є додаток

expositions, який, окрім стандартного набору файлів як це видно з рис. 3.1, містить також файл context_processors.py.

```
1  from .models import Exposition
2
3  def global_expositions(request):
4      expositions = Exposition.objects.all()
5      return {'expositions': expositions}
6
```

Рисунок 3.1 – Вміст файлу context_processors.py

У Django контекстні процесори — це функції, які додають додаткові змінні у контекст шаблонів на всіх сторінках веб-сайту, без необхідності явно передавати їх у кожне подання [32].

Функція приймає HTTP-запит request, здійснює вибірку всіх записів моделі Exposition (експозиції) з бази даних і формує словник із ключем 'expositions'.

Завдяки цьому контекстний процесор автоматично додає отриманий словник у контекст усіх шаблонів, що забезпечує доступ до списку експозицій у будь-якому HTML-шаблоні проєкту без необхідності явно передавати їх із кожного окремого подання (view).

Файл settings.py у фреймворку Django виконує функцію центрального конфігураційного модуля проєкту, оскільки саме в ньому зосереджено ключові параметри системи: налаштування підключення до бази даних, шляхи до шаблонів і статичних ресурсів, підключення додатків через INSTALLED_APPS, конфігурації обробки медіа-файлів, а також параметри безпеки, зокрема SECRET_KEY, DEBUG, ALLOWED_HOSTS та інші.

Змінна SECRET_KEY (рис. 3.2) використовується для криптографічного підписування даних, зокрема сесій, токенів аутентифікації та інших внутрішніх механізмів фреймворку. DEBUG визначає режим роботи

застосунку: у разі встановлення значення True програма виводить детальні повідомлення про помилки. Зі свого боку, параметр ALLOWED_HOSTS визначає перелік доменних імен та IP-адрес, з яких дозволено звернення до сервера.

```
# SECURITY WARNING: keep the secret key used in production sec
SECRET_KEY = 'django-insecure-p8%j#0vclu=yh*xizeciy-z*gtdc7(ey

DEBUG = FALSE
ALLOWED_HOSTS = []

INSTALLED_APPS = [
```

Рисунок 3.2 – Файл settings.py

База db.sqlite3 є типовим рішенням під час розробки. Вбудована система SQLite зберігає дані в одному файлі, не потребує окремого сервера і зручна для локального тестування [33]. Однак SQLite має обмеження за продуктивністю і масштабованістю, тому для промислового використання часто рекомендують переносити базу на більш потужні системи, такі як PostgreSQL чи MySQL.

Окремий файл manage.py — це важливий скрипт у кореневій директорії, який служить для управління проектом із командного рядка. Він є точкою входу для виконання різних адміністративних та допоміжних команд, таких як запуск локального сервера розробки, застосування міграцій бази даних, створення нових додатків, тестування, збір статичних файлів тощо.

Після налаштування структури проєкту, включно з основними файлами на кшталт manage.py, наступним кроком є побудова внутрішньої логіки веб-застосунку. Центральною складовою цієї логіки є моделі даних, які визначають, як саме буде організовано, збережено та оброблено інформацію в базі даних.

Ретельно спроєктовані моделі даних є основою архітектури сайту, де кожна модель представляє окрему сутність предметної області. Всі моделі реалізовані у вигляді класів, що забезпечує ефективну організацію структури бази даних [34]. Основні принципи побудови моделей:

а) чітка відповідність предметній області;

б) об'єктно-реляційне відображення (ORM);

в) зв'язки між моделями;

г) інтеграція з адміністративною панеллю.

Усього в системі було реалізовано кілька основних моделей, які забезпечують повноцінне управління контентом.

Модель користувачів базується на стандартній обгортці AbstractUser, яку було розширено під специфіку проєкту. Основне призначення — автентифікація, авторизація та прив'язка прав доступу до контенту в системі.

Тоді як основу наповнення сайту складає модель експозицій, що зберігає структуровану інформацію про виставки музею, моделі новин, анонсів, каруселі та галереї зображень доповнюють її динамічним і візуальним контентом.

Структура даних уніфікована — більшість моделей мають спільні поля, зокрема:

а) title — заголовок об'єкта (експозиції, новини, зображення, анонсу тощо);

б) description — текстовий опис, часто з можливістю форматування (RichTextField);

в) img/photo — зображення, що візуалізує контент;

г) date — дата створення або додавання.

Моделі пов'язані між собою через ForeignKey або ManyToMany-зв'язки, що підтримує логічну цілісність і спрощує роботу з даними. Розширюваність модульної архітектури дає змогу легко додавати нові моделі або атрибути без порушення цілісності проєкту.

Після розробки моделей даних наступним важливим кроком є організація міграцій БД. Міграції — це механізм, що дозволяє автоматично застосовувати зміни у структурах таблиць бази даних відповідно до змін у моделях.

Процес міграції складається з двох основних команд. Команда makemigrations аналізує внесені зміни у визначення моделей і генерує відповідні файли міграцій — своєрідні інструкції або сценарії, які описують, які саме зміни слід виконати над структурою бази даних. А наступна migrate застосовує раніше створені файли міграцій безпосередньо до БД. В результаті структура таблиць оновлюється.

Візуалізацію моделей можна зробити за допомогою .dot-графа — спеціального формату файлів, який використовується для представлення структурованих графів (рис. 3.3).

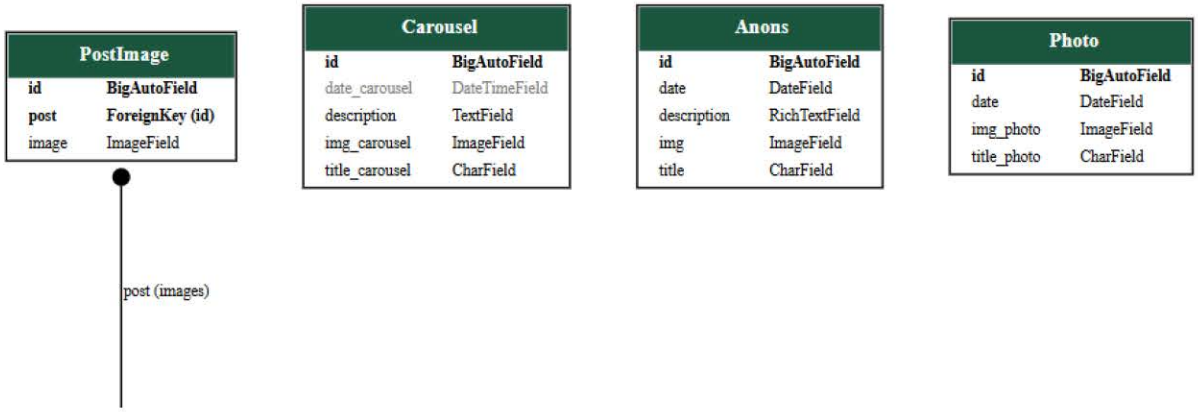

Рисунок 3.3 – Структура зв’язків моделей

Отримана схема дає змогу:

а) оцінити складність архітектури додатку;

б) побачити зв’язки між сутностями;

в) виявити ключові моделі.

На прикладі графа видно, що модель User, яка наслідує AbstractUser, є центральною у системі авторизації та управління правами. Вона включає поля

для ідентифікації користувачів, такі як email, is_staff, is_superuser та інші. Архітектурна прозорість, досягнута через моделювання та візуалізацію, суттєво полегшує супровід та підтримку системи.

3.2 Адміністративна панель та інструменти розробки

Адміністративна панель є важливою складовою системи, яка забезпечує централізоване управління її основними параметрами. Через цю панель адміністратор має можливість здійснювати налаштування системи відповідно до поточних потреб.

У межах проєкту використано Django Admin (рис. 3.4), адаптований під специфіку музею УМСФ, зручний для створення та редагування експозицій, новин, публікацій і контролю мультимедіа.

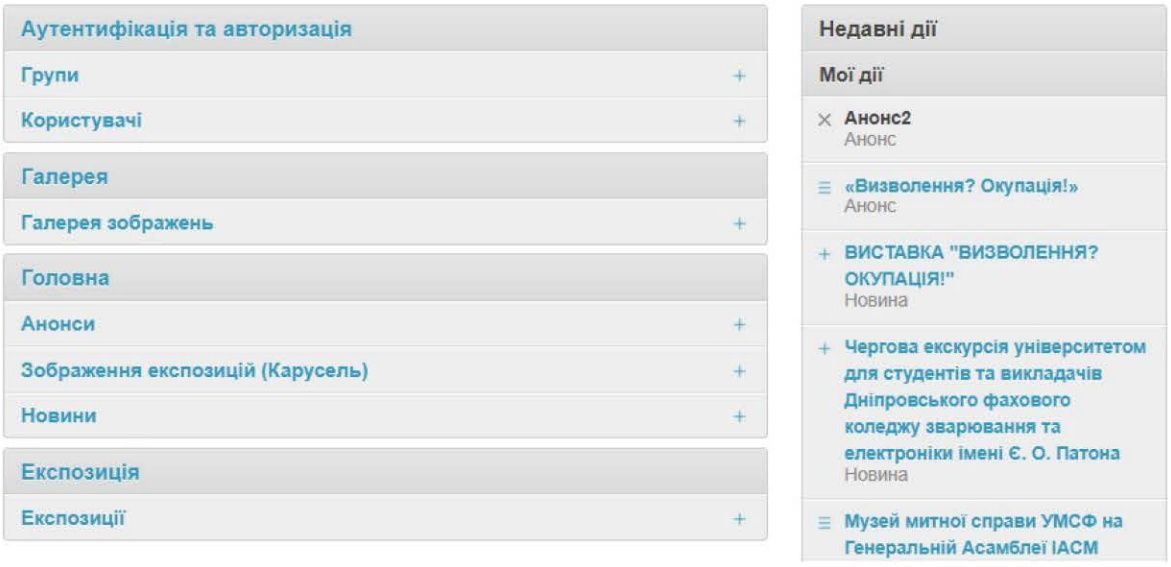

Рисунок 3.4 – Адміністративна панель

Особлива увага була приділена реалізації рольової моделі доступу. Система передбачає прав, які дозволяють розмежувати повноваження користувачів відповідно до їх функціональних обов’язків [35]:

а) адміністратор — має повний доступ до всіх можливостей панелі: управління контентом, користувачами, налаштуваннями системи.

б) редактор — обмежений доступом до інструментів створення та редагування матеріалів (текстів, зображень, галерей).

Таке розмежування дозволяє підвищити рівень безпеки системи, мінімізує ризик внесення несанкціонованих змін. Інтерфейс адміністратора розділений на кілька тематичних блоків, які дозволяють швидко орієнтуватися в системі керування сайтом.

Наприклад, у категорії «Головна» згруповані модулі (рис. 3.5), які відповідають за наповнення головної сторінки сайту. До цього блоку входять:

а) анонси — розділ для коротких повідомлень про майбутні заходи;

б) зображення експозицій (Карусель) — інструмент для формування динамічної галереї;

в) новини —публікації, що висвітлюють діяльність музею.

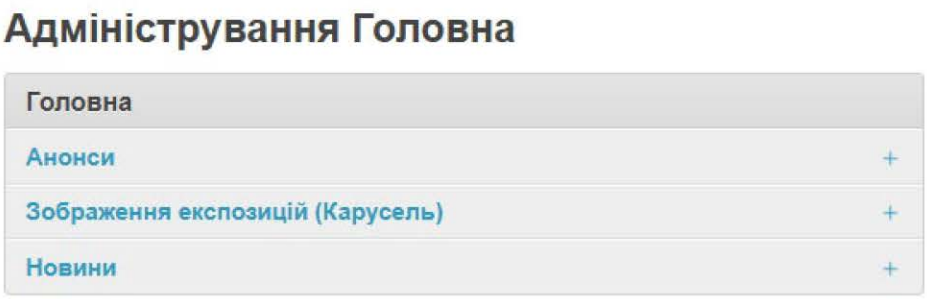

Рисунок 3.5 – Сторінка, що відповідає за головну сторінку

Завдяки продуманій структурі інтерфейсу, панель адміністратора виявилась зручною та легкою для навігації навіть для користувачів без технічної підготовки.

Основні розділи згруповані за логікою їхнього призначення, що створює чітку ієрархію та дозволяє швидко орієнтуватися серед великої кількості функцій.

Попри зручність адміністративної панелі для керування контентом та даними, розробка повнофункціонального веб-застосунку потребує гнучкіших і потужніших інструментів.

Важливою складовою веб-застосунку є система шаблонів, яка відповідає за формування HTML-сторінок на основі отриманих даних із бази. У цьому проєкті було застосовано шаблонізатор Jinja2, що відзначається високою продуктивністю та підтримкою принципу уникнення повторень коду (DRY — Don't Repeat Yourself).

1із підтримкою змінних, циклів, умов та логіки відображення [36]. Його переваги — шаблонне наслідування й включення елементів, що сприяє повторному використанню коду та мінімізує дублювання.

Особливістю Jinja2 є підтримка механізмів шаблонного наслідування та включення окремих елементів, що значно сприяє повторному використанню коду і зменшує дублювання, підвищуючи ефективність розробки та підтримки веб-додатку. На початку коду використовується тег {% load static %} (рис. 3.6), який активує можливість використовувати функцію {% static '...' %} для коректного підключення статичних файлів (зображень, CSS, JavaScript) у відповідності до конфігурації.

```
{% load static %}
  <html lang="uk">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="description"
      content="Музей університету митної справи та фінансів"
    />
    <meta
      name="keywords"
```

Рисунок 3.6 – Код layout.html

Базовий шаблон забезпечує єдину візуальну та функціональну основу для всіх сторінок сайту. Файл layout.html виступає в ролі такого шаблону в

проєкті УМСФ. Він містить основну структуру HTML-документу, включаючи секції <head> і <body>, а також підключення стилів, шрифтів, скриптів, заголовка сайту та навігаційного меню. Усі інші сторінки наслідують layout.html, розширюючи та замінюючи його окремі блоки, відповідно до потреб конкретної сторінки.

З метою спростити адміністрування сайту та оптимізувати роботу з контентом до проєкту було додано два зовнішні модулі — Grappelli і CKEditor.

CKEditor (рис. 3.7) є сучасним редактором із візуальним інтерфейсом, що дозволяє створювати й форматувати текст без необхідності володіння спеціальними знаннями у програмуванні [38].

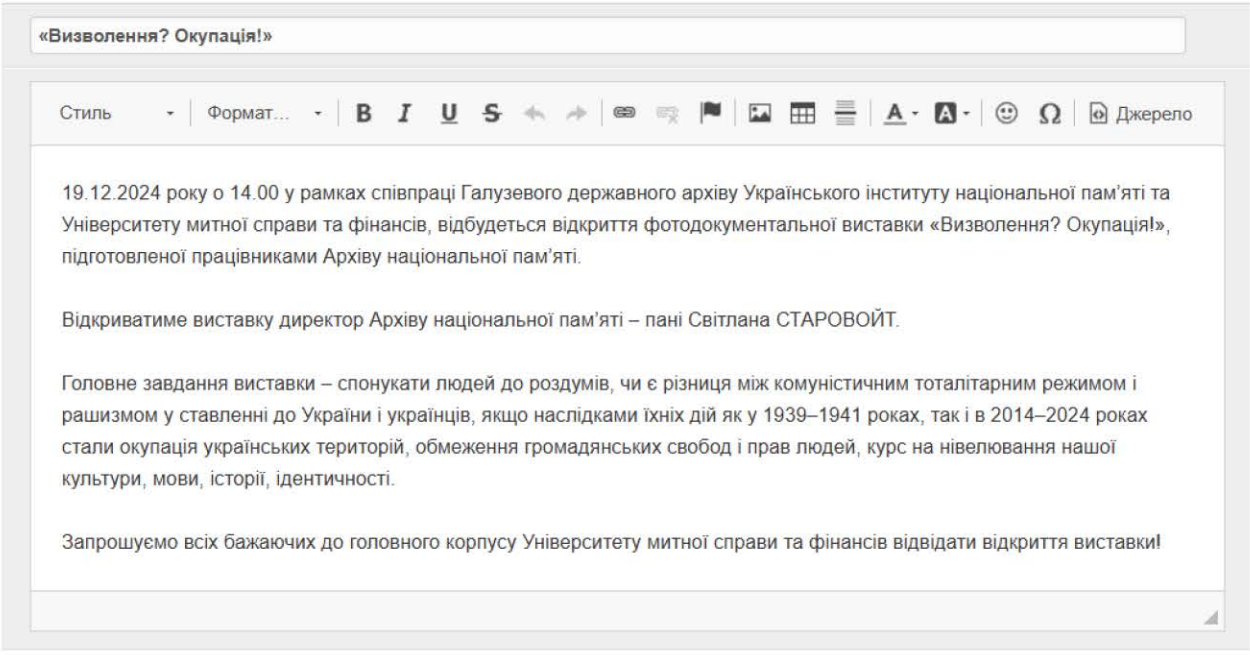

Рисунок 3.7 – Редагування анонсу

Grappelli ж є функціональним доповненням до базового адміністративного інтерфейсу Django, яке трансформує його у більш сучасне та зручне для користувача середовище керування. Оформлення отримує

оновлений дизайн із чіткою візуальною ієрархією, що сприяє легшій орієнтації в структурі панелі.

У тандемі Grappelli та CKEditor формують дружнє робоче середовище: перший робить навігацію простою та логічною, другий забезпечує зручне форматування матеріалів. Разом вони відповідають актуальним вимогам до систем керування веб-ресурсами та суттєво підвищують продуктивність редакційної команди.

3.3 Клієнт-серверна архітектура та тестування сайту

Сайт був реалізований за принципом клієнт-серверної архітектури, що передбачає чіткий поділ обов'язків між двома основними складовими системи: клієнтською та серверною частинами.

Архітектура типу клієнт–сервер (рис. 3.8) виступає фундаментальною концепцією побудови програмних систем, де функціональні обов'язки розподіляються між клієнтською стороною, яка надсилає запити, та серверною, що обробляє ці запити й надає відповідь.

Рисунок 3.8 – Клієнт-серверна архітектура [39]

Клієнтська частина — це те, що бачить і з чим взаємодіє користувач у своєму браузері. Вона включає в себе графічний інтерфейс, навігацію, форму введення, кнопки, а також усі елементи сторінки, які відображаються на

екрані. У даному проєкті клієнтська частина реалізована з використанням стандартних веб-технологій — HTML для структури сторінки, CSS для її візуального оформлення, та JavaScript для додання інтерактивності.

Сучасний фронтенд має ще одну важливу характеристику — динамічність. Це означає, що деякі частини сторінки можуть змінюватися без повного її перезавантаження, завдяки використанню JavaScript.

Серверна частина є невидимою для кінцевого користувача, але виконує критично важливі функції. У рамках цього проєкту вона реалізована з використанням фреймворку Django — потужного інструменту на мові програмування Python, який дозволяє створювати безпечні, структуровані та масштабовані веб-системи.

Саме серверна частина обробляє запити, що надходять з клієнта: аналізує їх, виконує необхідні обчислення, звертається до бази даних, проводить перевірку авторизації або автентифікації, застосовує бізнес-логіку.

Процес обміну даними між клієнтом і сервером реалізується через HTTP-запити.

Типовий сценарій виглядає наступним чином:

а) користувач через браузер звертається до певної URL-адреси сайту;

б) запит надходить до серверної частини, де Django визначає відповідний маршрут (url) і викликає відповідне представлення (view);

в) представлення обробляє запит: може здійснювати звернення до бази даних, перевірку форм, а також логіку авторизації чи фільтрації даних;

г) сервер формує відповідь, а саме динамічну HTML-сторінку з підключеними стилями й скриптами — та повертає її клієнту;

д) браузер користувача відображає отриману сторінку, з можливістю подальшої взаємодії.

Цикл запитів і відповідей може повторюватись багаторазово у межах однієї сесії користувача. Наприклад, при переході між сторінками, заповненні

форм, здійсненні пошуку або авторизації — кожна дія користувача ініціює запит до сервера, який формує релевантну відповідь.

Клієнт-серверна архітектура дозволяє досягти високої продуктивності, гнучкості та безпеки. У разі потреби обидві частини можуть масштабуватись незалежно — наприклад, можна змінити зовнішній вигляд сайту без впливу на бекенд, або модернізувати базу даних без зміни інтерфейсу.

Водночас для забезпечення стабільної роботи сайту недостатньо лише правильної архітектури — необхідним етапом є ретельне тестування, яке дозволяє виявити і виправити помилки до моменту запуску.

З метою перевірки надійності, коректності та зручності використання веб-сайту музею УМСФ було проведено поетапне тестування. Перевірка здійснювалась за кількома напрямами: функціональне, інтеграційне, модульне, інтерфейсне [40].

Функціональне тестування передбачало перевірку реалізованих функцій на відповідність їхньому призначенню. Було перевірено такі компоненти: завантаження зображень, перегляд контенту на сайті та коректність переходів між сторінками. Усі функції тестувалися вручну в контрольованому середовищі, згідно з розробленими сценаріями тестування.

Інтеграційне тестування оцінювало взаємодію між модулями бекенд — моделями, представленнями, маршрутами і шаблонами. Перевіряли передачу даних та обробку помилкових запитів, зокрема до неіснуючих сторінок. Результати підтвердили стабільність і цілісність роботи сервера.

Модульне тестування проводилося тестування окремих функцій і методів бекенд-коду ізольовано від решти системи.

Метою було переконатися в коректній роботі базових логічних блоків та алгоритмів. Результати модульного тестування забезпечили високий рівень довіри до стабільності системи при її подальшій розробці та вдосконаленні.

За перевірку зручності та коректності роботи адміністративного інтерфейсу відповідало інтерфейсне тестування. Воно включало оцінку логіки

навігації, доступності функцій створення і редагування. Отримані відгуки тестових користувачів дали змогу вдосконалити інтерфейс і підвищити ефективність адміністрування.

Окрему увагу було приділено тестуванню адміністративного інтерфейсу, що здійснювалося в рамках інтерфейсного тестування. Цей етап передбачав перевірку зручності використання інтерфейсу для адміністраторів, оцінку інтуїтивності навігації.

Під час тестування було виявлено окремі незначні недоліки, які своєчасно були усунуті в процесі налагодження та оптимізації.

Отже, веб-сайт повністю готовий до подальшого розгортання в продуктивному середовищі та використання кінцевими користувачами.

Проведене тестування охопило всі ключові аспекти функціонування системи — від перевірки окремих логічних блоків до оцінки зручності користування інтерфейсом.

3.4 Інструкція користувачу, розвиток проекту

Для того, щоб отримати доступ до системи керування контентом веб-сайту музею УМСФ, користувачу необхідно пройти певну послідовність кроків через адміністративну панель, реалізовану на базі Django.

Спершу слід відкрити будь-який зручний веб-браузер і в адресному рядку ввести точну URL-адресу, що веде до сторінки адміністративного входу.

Після завантаження цієї сторінки користувач побачить форму авторизації, яка вимагає введення імені користувача та пароля. Така процедура забезпечує захист системи, дозволяючи доступ лише уповноваженим особам. У випадку, якщо користувач вводить правильні облікові дані, логін та пароль (рис. 3.9), система здійснює перевірку автентичності на сервері. Після успішної авторизації відбувається автоматичне перенаправлення на головну сторінку адміністративного

інтерфейсу. Ця сторінка є центральною панеллю керування сайтом, звідки відкривається доступ до всіх зареєстрованих моделей, налаштувань та інструментів адміністрування.

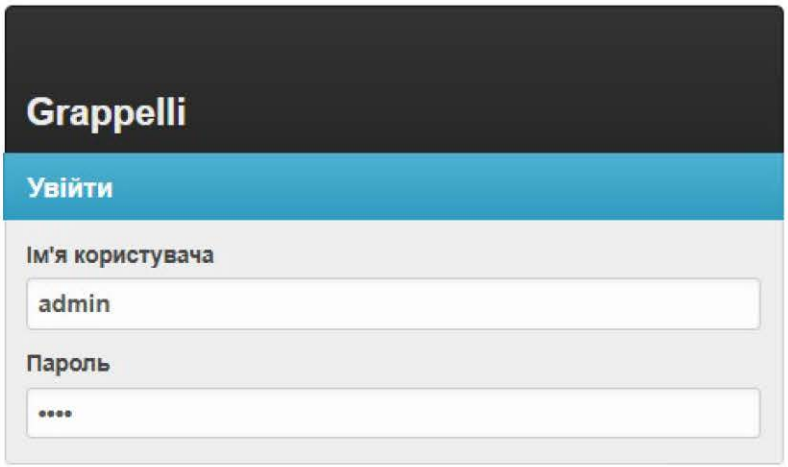

Рисунок 3.9 – Форма входу

Після авторизації користувач потрапляє на головну сторінку адміністративного інтерфейсу. Панель поділена на тематичні блоки:

а) аутентифікація та авторизація – управління обліковими записами та групами користувачів;

б) галерея – керування зображеннями в галереї;

в) головна – редагування анонсів, новин, зображень для головної сторінки;

г) експозиція – додавання та оновлення експонатів.

Кожна категорія оформлена у вигляді списку, що дає змогу зручно рухатися між модулями навіть у випадку великої кількості моделей. Завдяки цьому адміністратор може швидко зорієнтуватися та перейти до потрібної секції без надлишкових дій.

Праву частину інтерфейсу займає панель «Недавні дії», яка автоматично фіксує та відображає всі операції, здійснені користувачем протягом сесії: створення нових об’єктів, редагування наявних або їх видалення. Це особливо

корисно для відстеження активності в динамічному середовищі розробки або контент-менеджменту, а також для швидкого повернення до нещодавно змінених записів без потреби в повторному пошуку.

Для створення нового запису потрібно відкрити відповідний розділ, наприклад, «Експозиції» чи «Новини», і натиснути «Додати» ; після цього відкривається форма, у якій зазначаються назва, стислий або розгорнутий опис, дата публікації та, за потреби, додаються зображення чи інші пов'язані об'єкти.

Обов'язкові поля перевіряються вбудованими валідаторами, а правильність заповнення підтверджується повідомленням про успішне збереження після натискання «Зберегти».

За аналогічною схемою відбувається редагування вже наявного запису: необхідно вибрати потрібний об'єкт зі списку, відкрити його, внести корективи та підтвердити зміни.

Видалення здійснюється на сторінці самого об'єкта через кнопку «Видалити» з подальшим підтвердженням дії, після чого дані зникають зі списку та з бази.

Адміністратор також керує обліковими записами у розділі «Користувачі». Розділ керування обліковими записами дозволяє створювати нових користувачів, змінювати їм паролі, призначати або відкликати права доступу та включати до груп, що мають попередньо налаштовані набори дозволів.

Стандартно виділено щонайменше два рівні: супер-користувач із повним доступом до всіх функцій і редактор, який може оперувати контентом, але не має права змінювати налаштування безпеки чи системні параметри.

Після завершення роботи з адміністративною панеллю користувач натискає «Вийти» у верхньому правому куті екрана; це завершує сесію та унеможливлює несанкціонований доступ до системи з відкритої вкладки браузера.

Такий механізм гарантує конфіденційність даних і підтримує високий рівень безпеки під час повсякденного адміністрування веб-ресурсу музею.

Разом із тим, проект не є статичним — його функціональність має вдосконалюватися відповідно до нових викликів і потреб.

Подальший розвиток сайту передбачає розширення його функціональних можливостей, покращення зручності управління контентом та адаптацію до реальних потреб користувачів.

Насамперед постала проблема безмежного нарощування стрічки новин: щоразу з появою нової публікації головна сторінка видовжується, утруднюючи навігацію. Рекомендовано запровадити пагінацію або ж лімітований вивід останніх трьох–п’яти записів із посиланням «Переглянути всі новини». Подібне рішення забезпечує чистий візуальний простір і не перешкоджає індексації старих матеріалів у пошуку.

По-друге, було запропоновано додати можливість прикріплення кількох зображень до анонсів. Наразі форма створення анонсу дозволяє завантажити лише одне зображення, що обмежує якість візуального оформлення подій. Для реалізації цієї функції доцільно розширити модель «Анонс», додавши зв'язок із окремою моделлю зображень.

По-третє, необхідно впровадити функціонал редагування статичних інформаційних сторінок через адміністративну панель. Сьогодні такі розділи, як «Історія створення музею», «Відвідувачу» та «Контакти», оновлюються виключно за участю розробника. Пропонується створити окремий додаток, що відповідатиме за керування і редагуванням статичних сторінок сайту.

Четвертим напрямком удосконалення сайту є впровадження підтримки кількох мов. Беручи до уваги потенційний інтерес до музею з боку іноземних гостей, необхідно передбачити можливість перемикання мовної версії ресурсу. На першому етапі доцільним буде впровадження українсько-англійської локалізації з можливістю подальшого розширення.

П’ятою ідеєю є створення розділу «Віртуальні тури» або «Інтерактивна карта музею», що дозволить користувачам дистанційно ознайомитися з експозиціями та навігацією музею. Це не лише підвищить привабливість сайту, а й слугуватиме корисним інформаційним інструментом для планування візитів.

На завершення, доцільно реалізувати систему зворотного зв’язку або онлайн-форму для подання запитань, звернень або відгуків. Тоді відбудеться суттєве покращення взаємодії з відвідувачами, що дозволить швидше та ефективніше реагувати на їхні потреби, зауваження й коментарі.

Запропоновані удосконалення значно спростять взаємодію користувачів із сайтом, зробивши її зрозумілою та комфортною для всіх категорій відвідувачів — як для працівників музею, так і для широкої аудиторії. Оптимізація контенту, розширення мультимедійних можливостей та підтримка багатомовності зроблять сайт більш зручним у використанні.

3.5 Висновки до третього розділу

У третьому розділі було здійснено безпосередню реалізацію серверної частини веб-сайту для музею УМСФ. Було ретельно описано структуру проєкту, зокрема реалізацію ключових моделей даних, таких як експозиції, публікації, галереї та користувачі.

Зокрема, було створено й налагоджено моделі для збереження та обробки даних, що стосуються музейних експозицій, новинних публікацій, фотогалерей і облікових записів користувачів. У межах розробки також розглянуто логіку обробки запитів, налаштування маршрутизації між сторінками сайту та інтеграцію механізмів безпеки для захисту даних і доступу.

Досягнуто наступних практичних результатів:

а) розроблено та успішно протестовано адміністративну панель на базі Django Admin із розширеннями Grappelli та CKEditor, що значно покращують користувацький досвід управління контентом;

б) реалізовано шаблонізоване відображення даних за допомогою Jinja2, що підвищує гнучкість і динамічність інтерфейсу;

в) впроваджено рольову модель доступу з диференціацією прав, що забезпечує надійний контроль безпеки;

г) проведено комплексне тестування функціональності, зокрема перевірку маршрутів і процедур авторизації, що підтвердило стабільність роботи системи;

д) підготовлено детальну інструкцію для адміністратора з користування панеллю керування, що полегшує її освоєння.

Результатом проведеної роботи стала стабільну й ефективну серверну частину веб-платформи, що повністю відповідає сучасним вимогам веб-розробки. В основу системи покладено надійну архітектуру, яка дозволяє забезпечити безперебійну обробку запитів, захист користувацьких даних та гнучке керування контентом. Завдяки цьому сайт музею може служити технічною основою для подальшої цифрової трансформації його діяльності.

Розроблена система реалізує базовий функціонал, необхідний для ефективної роботи: від адміністрування виставкових матеріалів до управління мультимедійним контентом та користувацькими обліковими записами.

Окрім цього, у процесі розробки було окреслено низку перспектив для подальшого розвитку платформи. Зокрема, заплановано розширення можливостей адміністративної частини, впровадження інтерактивних інструментів для користувачів, а також інтеграцію з соціальними мережами та мобільними пристроями.

Такі кроки сприятимуть підвищенню зручності користування, кращій взаємодії з аудиторією та зміцненню цифрової присутності музею.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було створено повноцінну серверну частину веб-сайту музею УМСФ. Цей процес охоплював усі ключові етапи: від збору вимог та аналізу предметної області — до розробки, налаштування, тестування та документування системи. Кожен етап сприяв формуванню цілісного, зручного та надійного інформаційного рішення, адаптованого до потреб сучасного користувача.

У процесі виконання кваліфікаційної роботи були реалізовані такі спеціальні (фахові) компетентності згідно зі стандартом спеціальності 122 «Комп'ютерні науки»:

а) СК1, що передбачає дослідження моделей;

б) СК3 – побудову логічних висновків і розробку алгоритмів;

в) СК4 – використання методів математичного моделювання та чисельного розв'язування задач;

г) СК8 – проєктування і розробку програмного забезпечення із застосуванням сучасних парадигм програмування;

д) СК9 – реалізацію обчислювальних моделей клієнт-серверної архітектури з базами даних;

е) СК10 – управління життєвим циклом інформаційних систем відповідно до вимог замовника;

На початку роботи проведено глибокий аналіз сфери цифрової трансформації музеїв, зокрема таких, що виконують освітні та наукові функції. Увагу було зосереджено на специфіці діяльності музею УМСФ як частини освітньої установи, що зумовило особливі вимоги до веб-сайту.

Враховуючи сучасні виклики для закладів культури й освіти — обмежений фізичний доступ до експозицій, зростаючу потребу у цифровій взаємодії — обґрунтовано актуальність створення серверної частини сайту,

здатної ефективно обробляти запити, зберігати дані та забезпечувати стабільну роботу системи.

Поставлені завдання дали змогу чітко окреслити технічні параметри проєкту, визначити функціональні та нефункціональні характеристики системи, а також розробити її структуру відповідно до очікувань цільових груп. Було виділено чотири основні категорії користувачів: студенти, викладачі, музейна адміністрація та зовнішні відвідувачі.

Як об'єкт проєктування виступила система адміністрування музейного сайту, що забезпечує обробку запитів, взаємодію з базою даних і керування контентом. Проєкт розглядається як інформаційне рішення для культурно-освітніх установ, яке сприяє збереженню, організації та популяризації музейної спадщини.

Для управління процесом розробки було обрано методологію Agile. Технологічну основу склали мова програмування Python і веб-фреймворк Django, які забезпечили гнучкість і високу швидкість розробки.

Вибір на користь власної серверної частини, а не готової CMS, дозволив реалізувати специфічні функціональні потреби та уникнути обмежень стандартних рішень. Проєкт реалізовувався в умовах командної взаємодії. Для організації процесу розробки використовувалася система керування проєктами Jira.

Архітектура проєкту базувалася на принципі модульності та використанні шаблону Model–Template–View. Особливу увагу приділено адміністративній частині сайту — вона адаптована до потреб працівників музею, має чіткий розподіл ролей і підтримує зручне редагування контенту. Інтеграція інструментів Grappelli та CKEditor значно поліпшила інтерфейс керування й надала можливість працювати з мультимедійним контентом без спеціальних технічних знань.

Важливим технічним досягненням стало ефективне поєднання клієнтської та серверної логіки. Завдяки налаштованим маршрутам, передачі

даних через змінні оточення та продуманим шаблонам відображення вдалося забезпечити злагоджену роботу системи. Інтерфейс швидко реагує на дії користувача, оновлюючи вміст без перезавантаження сторінок.

На завершальному етапі виконано тестування функціональних модулів, зокрема системи авторизації, керування експозиціями, редагування сторінок та завантаження медіа. Тестування підтвердило стабільну роботу системи в умовах типової експлуатації.

На додаток до реалізації функціональних можливостей платформи, було підготовлено розгорнуту інструкцію для користувачів. Вона значно полегшує процес адміністрування ресурсу та забезпечує зручність його подальшого обслуговування й оновлення.

Серед перспектив розвитку системи — впровадження нових функцій, які враховують потреби користувачів і сучасні тенденції цифрової комунікації. Зокрема, до пропозицій покращення сайту можна віднести:

а) обмеження кількості новин на головній сторінці для покращення візуального сприйняття;

б) можливість прикріплення декількох зображень до публікацій;

в) редагування статичних сторінок через окремий додаток;

г) підтримка багатомовності (українська й англійська мови);

д) створення віртуального туру або інтерактивної карти музею;

е) впровадження онлайн-форми зворотного зв’язку.

Таким чином, у межах виконаної дипломної роботи вдалося виконати мету та реалізувати всі поставлені завдання — розробити функціональну серверну платформу для веб-сайту музею.

Отриманий результат має прикладне значення, оскільки створена система повністю відповідає потребам закладу в умовах цифровізації й забезпечує стабільну підтримку його інформаційної присутності в онлайн-просторі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Drotner K., Schroder K. Ch. Museum Communication and Social Media: The Connected Museum. – NY, 2013. – 214 p.

2. Основи будування сайтів / В. Манако, О. Войченко, Д. Манако, О. Данилова. – 2015. – 120 с.

3. Backend languages roadmap [Електронний ресурс]. – Режим доступу: https://roadmap.sh/backend/languages (дата звернення: 01.05.2025).

4. Мельник Р. А. Програмування веб-застосувань (фронт-енд та бекенд): навч. посіб. — Львів: Львівська політехніка, 2018. – 248 с.

5. Музей Історії Дніпра [Електронний ресурс]. – Режим доступу: https://midnipro.museum/ (дата звернення: 03.05.2025).

6. Офіційний сайт Музею Дніпра [Електронний ресурс]. – Режим доступу: https://www.museum.dp.ua/uk/ (дата звернення: 05.05.2025).

7. Технічний музей «Машини Часу» [Електронний ресурс]. – Режим доступу: https://mvr.org.ua/ (дата звернення: 06.05.2025).

8. Музей УМСФ [Електронний ресурс]. – Режим доступу: http://museum.umsf.dp.ua/ (дата звернення: 12.05.2025).

9. The web framework for perfectionists with deadlines [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com/ (дата звернення: 12.05.2025).

10. Цільова аудиторія [Електронний ресурс]. – Режим доступу: http://ukrainiandigital.com/tsilova-audytoriia/ (дата звернення: 13.05.2025).

11. Stack Overflow Developer Survey 2023 [Електронний ресурс]. – Режим доступу: https://survey.stackoverflow.co/2023/#most-popular-technologies-language-prof (дата звернення: 14.05.2025).

12. Python official site [Електронний ресурс]. – Режим доступу: https://www.python.org/ (дата звернення: 20.05.2025).

13. SQLite: офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/ (дата звернення: 21.05.2025).

14. PostgreSQL: офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/ (дата звернення: 15.05.2025).

15. Agile [Електронний ресурс]. – Режим доступу: https://indevlab.com/uk/blog-ua/shho-take-agile-rozrobka/ (дата звернення: 15.05.2025).

16. Безкоштовні CMS [Електронний ресурс] – Режим доступу: https://hyperhost.ua/info/ru/besplatnyie-cms-preimushhestva-i-nedostatk (дата звернення: 16.05.2025).

17. WordPress.com: офіційний сайт [Електронний ресурс]. – Режим доступу: https://wordpress.com/ (дата звернення: 16.05.2025).

18. Joomla: офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.joomla.org/ (дата звернення: 16.05.2025).

19. Drupal: офіційний сайт [Електронний ресурс]. – Режим доступу: https://www.drupal.org/ (дата звернення: 16.05.2025).

20. Топ-10 Python веб-фреймворків [Електронний ресурс] // Leapcell. – Режим доступу: https://dev.to/leapcell/top-10-python-web-frameworks-compared-3o82 (дата звернення: 16.05.2025).

21. Django official site [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com (дата звернення: 16.05.2025).

22. Django-configurations — [Електронний ресурс]. — Режим доступу: https://pypi.org/project/django-configurations/ (дата звернення: 17.05.2025).

23. FastAPI: офіційний сайт [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com/ (дата звернення: 17.05.2025).

24. Flask: офіційна документація [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/en/stable/ (дата звернення: 17.05.2025).

25. MTV у Django [Електронний ресурс] // Python in Plain English. – Режим доступу: https://python.plainenglish.io/creating-an-mtv-model-template-view-architecture-in-django-59c6502e15c8 (дата звернення: 17.05.2025).

26. Django ORM [Електронний ресурс]. – Режим доступу: https://tutorial.djangogirls.org/uk/django_orm/ (дата звернення: 17.05.2025).

27. Shaw B., Badhwar S. Web Development with Django. – 2021. – 826 с.

28. Канбан [Електронний ресурс] – Режим доступу: https://uaspectr.com/2021/01/26/shho-take-kanban/ (дата звернення: 17.05.2025).

29. Scrum Guide / Ken Schwaber, Jeff Sutherland. – 2020. – Електронний ресурс. – Режим доступу: https://scrumguides.org/ (дата звернення: 17.05.2025).

30. GitHub [Електронний ресурс]. – Режим доступу: https://github.com/ (дата звернення: 17.05.2025).

31. What is Django? [Електронний ресурс]. – Режим доступу: https://www.ibm.com/think/topics/django (дата звернення: 17.05.2025).

32. Vincent W. S. Django for Beginners: Build websites with Python and Django. – WelcomeToCode, 2020. – 221 с.

33. Siahaan V., Hasiholan S. R. SQLITE FOR DATA ANALYSIS AND VISUALIZATION WITH PYTHON. – Independently published, 2022. – 412 с.

34. Jaiswal S., Kumar R. Learning Django Web Development. – Packt Publishing, 2015. – 336 с.

35. Mele A. Django 4 By Example: Build powerful and reliable Python web applications from scratch. – Birmingham : Packt Publishing, 2022. – 766 с.

36. Jinja [Електронний ресурс] // Pallets Projects. – Режим доступу: https://jinja.palletsprojects.com/en/stable/ (дата звернення: 17.05.2025).

37. Django Grappelli [Електронний ресурс] – Режим доступу: https://django-grappelli.readthedocs.io/en/latest/ (дата звернення: 18.05.2025).

38. CKEditor: офіційна документація [Електронний ресурс] // CKEditor. – Режим доступу: https://ckeditor.com/docs/ (дата звернення: 18.05.2025).

39. Клієнт-серверна архітектура [Електронний ресурс] // DOU. – Режим доступу: https://dou.ua/forums/topic/44636/ (дата звернення: 18.05.2025).

40. Канер С., Фолк Дж., Нгуєн Г. Тестування програмного забезпечення. Фундаментальні принципи. – К. : BHV, 2008. – 320 с.