Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук

(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)

(рівень вищої освіти)

на тему: "Розроблення вебплатформи для управління контентом сайту навчального курсу EMTC на базі Django CMS"

Виконав: студент 4 курсу групи КН-41
спеціальності:

122 "Комп'ютерні науки"

(шифр і назва напряму підготовки, спеціальності)

Гнидин Б.Б.

(прізвище та ініціали)

Керівник: Пірко І.Б.

(прізвище та ініціали)

Керівник: Сінкевич О.В.

(прізвище та ініціали)

Рецензент: Часковський О.Г.

(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 "Комп'ютерні науки"

(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувачка кафедри КН
Борецька І. Б.
"10" червня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Гнидину Богдану Богдановичу

(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення вебплатформи для управління контентом сайту навчального курсу EMTC на базі Django CMS

керівник роботи *Пірко Ігор Богданович, кандида фізико-математичних наук, доцент кафедри комп'ютерних наук*,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Сінкевич Олексій Вячеславович, доктор філософії,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "15" листопада 2024 року № С-882

2. Термін подання студентом роботи 10 червня 2025 р.

3. Вихідні дані до роботи Розробити систему управління контентом про академічну мобільність з використанням Django CMS та сучасних вебтехнологій. Реалізувати можливість створення та управління контентом про подорожі, включаючи фотогалереї та описи подорожей.

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

Стан проблемної області

Інформаційне та математичне забезпечення

Програмне та технічне забезпечення

Висновки

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

Підготовка матеріалів до доповіді

6. Дата видачі завдання 18 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	18.11.24 – 20.12.24	Виконано
2	Налаштування БД	21.12.24 – 25.01.25	Виконано
3	розроблення серверної частини	26.01.25 – 17.02.25	Виконано
4	Реалізація реєстрації та авторизації адміністраторів.	18.02.25 – 03.03.25	Виконано
5	Реалізація клієнтської частини	04.03.25 – 08.04.25	Виконано
6	Тестування додатку	09.04.25 – 03.05.25	Виконано
7	Виправлення помилок	04.05.24 – 14.05.25	Виконано
8	Оформлення пояснювальної записки	15.05.25– 09.06.25	Виконано

Студент _____ Гнидин Б.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Пірко І.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Сінкевич О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка до бакалаврської роботи містить 60 сторінок, 3 таблиці, 15 ілюстрацій, 2 додатки, 25 використаних джерел.

У роботі було розроблено вебплатформу для управління контентом про академічну мобільність з використанням Django CMS, що дозволила створити гнучку та потужну систему управління контентом з адаптивним дизайном. У роботі реалізовано механізми завантаження фотографій, створення галерей, версіонування контенту та організації описів подорожей.

Результати цієї роботи можуть бути впроваджені дирекцією університету для ефективного керування контентом вебплатформи.

Ключові слова: Django CMS, Системи управління контентом, Академічна мобільність, Версіонування, Фотогалереї, Python, Адаптивний дизайн, Веб-розроблення

ABSTRACT

The explanatory note to the bachelor's thesis contains 60 pages, 3 tables, 15 illustrations, 2 appendices, and 25 references.

The thesis presents the development of a web platform for managing content related to academic mobility using Django CMS, which enabled the creation of a flexible and powerful content management system with a responsive design.
The work implements mechanisms for uploading photos, creating galleries, content versioning, and organizing travel descriptions.

The results of this work can be used by the university administration to efficiently manage the content of the web platform.

Keywords**:** Django CMS, Content Management Systems, Academic Mobility, Versioning, Photo Galleries, Python, Responsive Design, Web Development.

ТЕХНІЧНЕ ЗАВДАННЯ

Мета проекту: Розробити вебплатформу для управління контентом про академічну мобільність з використанням Django CMS та сучасних вебтехнологій.

Функціональні вимоги:

- Можливість створення та редагування контенту про подорожі та обміни.
- Можливість завантаження фотографій та створення фотогалерей.
- Збір та аналіз даних про академічну мобільність.
- розроблення механізму версіонування контенту для відстеження змін.
- Відображення адаптивного інтерфейсу для зручного перегляду матеріалів.

Технічні вимоги:

1. Клієнтська частина:

- Вебплатформа розроблена з використанням Django Templates та Bootstrap для створення інтерфейсу.
- Вебсайт повинен бути респонсивним і підтримувати різні роздільні здатності та браузери.
- Використання системи шаблонів Django для навігації між сторінками.
- Інтерфейс повинен бути зручним і інтуїтивно зрозумілим для адміністраторів та користувачів.

2. Серверна частина:

- розроблення сервера з використанням Django.
- Використання фреймворку Django CMS для керування вмісту та маршрутизації.
- Реалізація системи для автентифікації, авторизації, завантаження контенту, організації галерей та версіонування матеріалів.
- Забезпечення захисту від несанкціонованого доступу (авторизація, обмеження доступу до адміністративних ресурсів).

3. База даних:

- Використання SQLite для розробки та PostgreSQL для виробничого середовища.
- Створення моделей для управління контентом та користувачами.
- Використання індексів для покращення роботи запитів.

4. Управління контентом:

- розроблення механізмів версіонування та модерації контенту.
- Реалізація системи для організації та категоризації матеріалів про академічну мобільність.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

Backend(“бекенд”) – серверна частина;

Frontend(“фронтетд”) – клієнтська частина;

БД – база даних;

CMS - Content Management System(Система Керування Вмістом);

HTML - HyperText Markup Language(Мова Розмітки Гіпертексту);

CSS - Cascading Style Sheets(Каскадні Таблиці Стилів);

API - Application Programming Interface(Інтерфейс Програмування Застосунків);

UML - Unified Modeling Language(Уніфікована Мова Моделювання);

SQL - Structured Query Language(Мова Структурованих Запитів);

WYSIWYG - What You See Is What You Get(Що Бачиш, Те й Отримаєш) ;

ORM - Object-Relational Mapping(Об’єктно-Орієнтоване Відображення);

MVC - Model-View-Controller(Модель-Вигляд-Контролер);

MVT - Model-View-Template(Модель-Вигляд-Шаблон);

URL - Uniform Resource Locator(Уніфікований Локатор Ресурсів);

HTTP - Hypertext Transfer Protocol(Протокол Передачі Гіпертексту);

HTTPS - Hypertext Transfer Protocol Secure(Безпечний Протокол Передачі Гіпертексту);

ВСТУП

В епоху інформаційного надлишку, коли доступний контент про академічну мобільність досягає значних розмірів, проблема ефективного управління та представлення матеріалів про подорожі та обміни стає все більш гострою. Адміністратори та викладачі витрачають багато часу на організацію та публікацію контенту, що значно знижує ефективність роботи. Тому розроблення потужної системи управління контентом є вкрай актуальним завданням.

Ці системи можуть значно скоротити час на створення та публікацію матеріалів та значно покращити якість представлення інформації. Завдяки зручним інструментам управління контентом адміністратори можуть ефективно організовувати матеріали про академічну мобільність, які відповідають потребам користувачів, і забезпечити максимальну доступність та корисність інформації.

У цьому дослідженні я вивчив взаємозв'язок між системами управління контентом та потребами користувачів у контексті інформаційних технологій, особливо вебплатформ для представлення матеріалів про академічну мобільність.

Об'єктом дослідження є покращення вебплатформи для управління контентом про академічну мобільність. Предметом дослідження є процеси та методи створення вебплатформ для управління контентом про академічну мобільність з використанням сучасних вебтехнологій та систем управління контентом.

Метою цієї роботи є розроблення вебплатформи, яка забезпечує ефективне управління контентом про академічну мобільність та подорожі. На основі сучасних технологій веброзробки та Django CMS. Ця система відрізняється високим рівнем функціональності, швидкістю роботи і зручністю використання.

Завдання роботи:

- Аналіз існуючих систем управління контентом: дослідження та порівняння різних методів та підходів, що використовуються в цій галузі;
- розроблення архітектури вебплатформи: розроблення архітектури вебплатформи за допомогою Django CMS та сучасних вебтехнологій забезпечує зручний інтерфейс і ефективну організацію даних;
- Впровадження системи управління фотогалереями та матеріалами:

розробити механізм створення, редагування та зберігання контенту про академічну мобільність, включаючи фотографії та описи подорожей;

- Тестування та покращення роботи: ретельне тестування вебплатформи, щоб оцінити її ефективність та зручність використання, а також покращення роботи продуктивність системи.

Практична цінність цієї роботи полягає в тому, що розроблена вебплатформа може стати основою для створення повноцінного порталу про академічну мобільність, що значно покращує доступність інформації, скорочує час, витрачений на пошук та публікацію потрібного контенту, і підвищує якість представлення матеріалів. Крім того, розроблення такої платформи сприяє поширенню знань про новітні технології веброзробки, що використовуються в системах управління контентом.

Таким чином, розроблення системи управління контентом для академічної мобільності з використанням Django CMS є важливим кроком на шляху до поліпшення взаємодії користувачів з інформаційними ресурсами і підвищенню ефективності управління та споживання освітнього контенту.

РОЗДІЛ 1 СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Аналіз проблемної області

У сучасному цифровому світі, де навчальні заклади щодня стикаються з необхідністю управління величезними обсягами інформації про академічну мобільність, ефективна організація контенту стає дедалі складнішою і потребує більше ресурсів. Саме тому системи управління контентом, які автоматизують процеси створення, редагування та публікації матеріалів, набувають все більшої ваги. Вони не лише економлять час адміністраторів, але й роблять доступ до інформації більш зручним та структурованим для користувачів.

Зародившись у 90-х роках XX століття як прості редактори вебсторінок, системи управління контентом з часом еволюціонували в потужні та багатофункціональні платформи завдяки стрімкому розвитку вебтехнологій. Одним з перших значних прикладів стала система WordPress, яка дозволяла легко створювати та керувати блогами. Не менш відомою стала система Drupal, яка вражає своєю гнучкістю та можливістю управляти складними вебпроектами різного масштабу.

Сучасні CMS ґрунтуються на передових технологіях та архітектурних рішеннях, таких як фреймворки, ORM, шаблонні системи та хмарні технології (Рисунок 1.1).

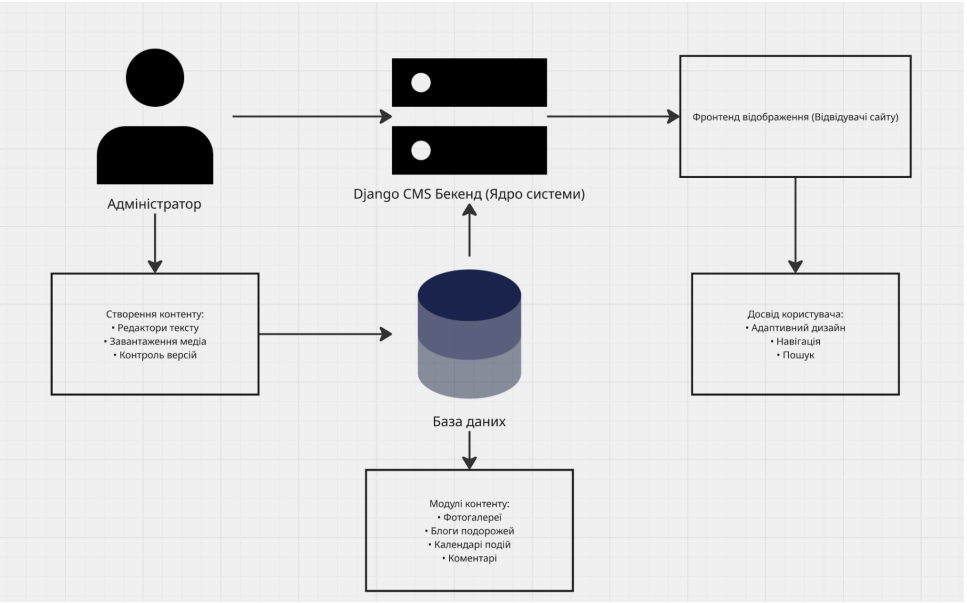


Рисунок 1.1 – Схема роботи системи управління контентом

- Фреймворки: Сучасні CMS побудовані на базі потужних фреймворків, як

Django для Python або Laravel для PHP, що надають надійний фундамент для розробки. Django CMS використовує Django фреймворк, що забезпечує високу продуктивність та безпеку;

- Об'єктно-реляційне відображення (ORM): Ця технологія дозволяє працювати з базами даних на рівні об'єктів, спрощуючи розробку та підтримку. Django ORM в Django CMS забезпечує ефективну роботу з даними про академічну мобільність без необхідності писати складні SQL-запити;
- Шаблонні системи: Технології, які відокремлюють логіку застосунку від його представлення. Django Template Language дозволяє створювати гнучкі та динамічні шаблони для відображення контенту про подорожі та обміни;
- Хмарні технології: Розгортання CMS в хмарному середовищі забезпечує масштабованість та доступність. Сучасні платформи, як AWS або Google Cloud, дозволяють ефективно розміщувати системи управління контентом.

Різні підходи до побудови систем управління контентом мають свої переваги та недоліки:

- Монолітні CMS: Ці системи пропонують готові рішення "з коробки" з мінімальними налаштуваннями. Вони прості у використанні, але можуть бути обмежені у кастомізації та масштабуванні;
- Модульні CMS: Ці системи, як Django CMS, побудовані на основі окремих модулів, які можна додавати або видаляти відповідно до потреб користувача. Вони забезпечують більшу гнучкість, але можуть вимагати більше технічних знань для налаштування;
- Headless CMS: Відокремлюють управління вмісту від його представлення, що дозволяє використовувати різні фронтенд-технології. Такий підхід забезпечує максимальну гнучкість, але вимагає більше ресурсів для розробки та підтримки.

Сьогодні системи управління контентом стали невід'ємною частиною багатьох освітніх та інформаційних порталів:

- WordPress: Найпопулярніша CMS у світі, яка використовується для створення блогів, корпоративних сайтів та онлайн-магазинів. Її простота та

велика кількість плагінів роблять її ідеальним рішенням для багатьох проектів;

- Drupal: Потужна CMS, яка підходить для складних вебпроектів, що вимагають високого рівня безпеки та можливостей кастомізації. Багато урядових та освітніх сайтів використовують Drupal;
- Django CMS: Побудована на базі Django фреймворку, ця система забезпечує високу продуктивність, безпеку та гнучкість. Вона ідеально підходить для проектів з великою кількістю користувачів та складною логікою;
- Joomla: Балансує між простотою WordPress та потужністю Drupal, пропонуючи зручний інтерфейс та хороші можливості для розширення функціональності.

Системи управління контентом для академічної мобільності стикаються з багатьма викликами, які потребують подальших досліджень та інновацій:

- Безпека та конфіденційність даних: Освітні платформи часто містять персональні дані, тому забезпечення їх захисту є критичним завданням. Розробники CMS повинні впроваджувати сучасні методи захисту інформації;
- Інтеграція з іншими системами: Часто виникає необхідність інтеграції CMS з іншими освітніми платформами та сервісами, що може бути складним технічним завданням;
- Адаптивність та доступність: Забезпечення зручного доступу до контенту з різних пристроїв та для користувачів з різними можливостями є важливим аспектом сучасних CMS;
- Масштабованість: Зі збільшенням кількості контенту та користувачів, система повинна залишатися продуктивною та стабільною.

Інтеграція з технологіями штучного інтелекту та машинного навчання відкриває нові можливості для автоматизації та персоналізації контенту. Розвиток технологій блокчейн може забезпечити нові підходи до верифікації та захисту інформації про академічну мобільність.

Завдяки сучасним технологіям та архітектурним рішенням, системи управління контентом стають все більш досконалими, пропонуючи зручні інструменти для створення, редагування та публікації матеріалів про академічну мобільність. CMS –

це не просто програмне забезпечення, це платформа для ефективної комунікації та обміну інформацією, яка може зробити освітні процеси більш прозорими та доступними. Важливо, щоб ці системи були надійними, зручними та адаптованими до потреб конкретних освітніх установ.

1.2 Аналіз бекенд складової

Сучасні системи управління контентом для академічної мобільності потребують надійної та масштабованої бекенд архітектури, яка зможе ефективно обробляти великі обсяги даних та забезпечувати безпеку інформації. Аналіз існуючих рішень показує, що більшість сучасних CMS використовують багатошарову архітектуру, де кожен шар відповідає за певний аспект функціональності системи.

На рівні бази даних виникає необхідність ефективного зберігання та обробки структурованих та неструктурованих даних. Системи академічної мобільності повинні зберігати інформацію про студентів, викладачів, навчальні програми, документи та медіафайли. Традиційні реляційні бази даних, такі як PostgreSQL або MySQL, забезпечують надійне зберігання структурованих даних, але можуть виявитися недостатньо ефективними для роботи з великими об'ємами неструктурованих даних, таких як документи та медіафайли.

Для обробки запитів та бізнес-логіки сучасні CMS використовують потужні ве-фреймворки. Django, як один з найпопулярніших Python-фреймворків, надає комплексний набір інструментів для розробки вебзастосунків. Його архітектура MVT (Model-View-Template) дозволяє чітко розділити логіку обробки даних, представлення та шаблони. Однак, при роботі з великими обсягами даних виникає необхідність оптимізації запитів та використання кешування.

Безпека даних є критичним аспектом бекенд архітектури. Системи академічної мобільності повинні забезпечувати захист персональних даних студентів та викладачів, а також захищати конфіденційну інформацію про навчальні програми та документи. Це потребує реалізації складних механізмів аутентифікації та авторизації,

шифрування даних та захисту від різних типів атак.

Масштабованість системи є ще одним важливим аспектом. Зі збільшенням кількості користувачів та обсягу даних, система повинна зберігати високу продуктивність. Це потребує використання технік горизонтального масштабування, таких як балансування навантаження та розподілене кешування. Також важливим є ефективне управління ресурсами сервера та покращення роботи продуктивності бази даних.

Інтеграція з іншими системами є невід'ємною частиною сучасних CMS. Системи академічної мобільності повинні взаємодіяти з іншими освітніми платформами, системами електронного навчання та зовнішніми сервісами. Це потребує розробки надійних API та механізмів синхронізації даних. REST API та GraphQL стали стандартними інструментами для забезпечення такої інтеграції.

Обробка медіафайлів та документів є особливою складовою бекенд архітектури. Системи повинні ефективно зберігати, обробляти та доставляти різні типи файлів, від текстових документів до відео та аудіо матеріалів. Це потребує використання спеціалізованих систем зберігання файлів та покращення роботи їх доставки користувачам.

Моніторинг та логування є важливими аспектами підтримки бекенд системи. Необхідно відстежувати продуктивність системи, виявляти помилки та аналізувати поведінку користувачів. Це потребує реалізації комплексних систем моніторингу та логування, які дозволяють оперативно реагувати на проблеми та покращити роботу системи.

Розвиток технологій штучного інтелекту та машинного навчання відкриває нові можливості для автоматизації процесів обробки даних. Системи можуть використовувати AI для аналізу контенту, автоматичної категорізації документів та персоналізації контенту для користувачів. Це потребує інтеграції з AI-сервісами та розробки спеціалізованих алгоритмів обробки даних.

Загалом, бекенд архітектура сучасних систем управління контентом для академічної мобільності повинна бути надійною, масштабованою та безпечною. Вона повинна ефективно обробляти великі обсяги даних, забезпечувати безпеку

інформації та надавати можливості для інтеграції з іншими системами. Розвиток нових технологій відкриває нові можливості для покращення функціональності та ефективності таких систем.

1.3 Аналіз фронтент складової

Сучасні системи управління контентом для академічної мобільності потребують інтуїтивно зрозумілого та ефективного користувацького інтерфейсу, який забезпечить зручний доступ до інформації та простий процес управління контентом. Аналіз існуючих рішень показує, що фронтенд архітектура сучасних CMS повинна враховувати різноманітність пристроїв та потреб користувачів.

На рівні користувацького інтерфейсу виникає необхідність створення адаптивного дизайну, який забезпечить оптимальне відображення контенту на різних пристроях - від мобільних телефонів до великих десктопних моніторів. Системи академічної мобільності повинні забезпечувати зручний доступ до інформації незалежно від розміру екрану та типу пристрою. Це потребує використання сучасних підходів до вебдизайну, таких як гнучкі сітки та медіа-запити.

Доступність інтерфейсу є критичним аспектом фронтенд архітектури. Системи повинні бути доступними для користувачів з різними можливостями, включаючи людей з обмеженнями зору, слуху та моторними функціями. Це потребує реалізації стандартів доступності WCAG, семантичної розмітки та підтримки асистивних технологій. Також важливим є забезпечення клавіатурної навігації та правильне використання ARIA-атрибутів.

Візуальний редактор контенту є ключовим компонентом фронтенд частини CMS. Він повинен надавати зручні інструменти для створення та редагування контенту, включаючи форматування тексту, вставку медіафайлів та створення таблиць. Сучасні редактори, такі як CKEditor або TinyMCE, надають багаті можливості для роботи з контентом, але потребують додаткової налаштування та інтеграції з системою.

Управління медіафайлами є важливою складовою фронтенд архітектури. Системи повинні надавати зручний інтерфейс для завантаження, організації та використання різних типів медіафайлів. Це включає підтримку drag-and-drop завантаження, попередній перегляд файлів та ефективну організацію медіабібліотеки. Також важливим є оптимізація зображень та відео для різних пристроїв.

Навігація та структура сайту є ключовими аспектами користувацького досвіду. Системи повинні надавати зручні інструменти для управління структурою сайту, включаючи створення меню, налаштування навігації та організацію контенту. Це потребує реалізації інтуїтивно зрозумілого інтерфейсу для управління структурою та навігацією.

Персоналізація інтерфейсу є важливим аспектом сучасних CMS. Системи повинні дозволяти користувачам налаштовувати інтерфейс відповідно до своїх потреб, включаючи вибір теми, розташування елементів та налаштування відображення контенту. Це потребує реалізації гнучких механізмів налаштування та збереження користувацьких преференцій.

Продуктивність фронтенду є критичним аспектом користувацького досвіду. Системи повинні забезпечувати швидке завантаження сторінок та плавну взаємодію з інтерфейсом. Це потребує покращення роботи завантаження ресурсів, використання кешування на стороні клієнта та ефективної обробки асинхронних запитів.

Інтернаціоналізація та локалізація є важливими аспектами фронтенд архітектури. Системи повинні підтримувати різні мови та регіональні налаштування, включаючи переклад інтерфейсу, форматування дат та чисел, та підтримку різних напрямків тексту. Це потребує реалізації механізмів перекладу та локалізації контенту.

Інтеграція з бекендом є ключовим аспектом фронтенд архітектури. Системи повинні ефективно взаємодіяти з бекенд API, включаючи обробку запитів, валідацію даних та обробку помилок. Це потребує реалізації надійних механізмів комунікації між фронтендом та бекендом.

Розвиток технологій веброзробки відкриває нові можливості для покращення користувацького досвіду. Сучасні фреймворки, такі як React або Vue.js, надають потужні інструменти для створення інтерактивних інтерфейсів. Прогресивні вебзастосунки (PWA) дозволяють створювати досвід, близький до нативних додатків, включаючи офлайн-функціональність та push-сповіщення.

Загалом, фронтенд архітектура сучасних систем управління контентом для академічної мобільності повинна бути зручною, доступною та ефективною. Вона повинна забезпечувати оптимальний користувацький досвід на різних пристроях, надавати зручні інструменти для управління контентом та підтримувати різні мови та регіональні налаштування.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 SQLite

Система використовує SQLite як основну базу даних, що надає ряд значних переваг для розробки та тестування [3]. SQLite відрізняється простотою використання, оскільки не потребує окремого серверного процесу або складного налаштування. Її компактність, де весь код упакований в один файл, значно спрощує розгортання та перенесення бази даних. Система працює "з коробки" без необхідності складної конфігурації або адміністрування, при цьому забезпечуючи надійність через транзакційну цілісність даних та підтримку ACID-властивостей. Крос-платформність SQLite дозволяє системі працювати на різних операційних системах без додаткових модифікацій, а висока продуктивність робить її ідеальним вибором для малих та середніх проектів. Особливо важливою є чудова інтеграція з Django, що робить SQLite оптимальним рішенням для розробки та тестування Django-проектів [1].

Проте, SQLite має певні обмеження, які варто враховувати [3]. Система використовує блокування на рівні файлу, що може обмежувати продуктивність при високому навантаженні та обмежує конкурентний доступ. Розмір бази даних обмежений кількома гігабайтами, що може бути недостатнім для великих проектів. Масштабованість системи також обмежена, оскільки SQLite не підходить для розподілених систем або високонавантажених застосунків. Мережевий доступ обмежений, оскільки система не призначена для одночасного доступу з багатьох мережевих клієнтів. Крім того, відсутність вбудованих інструментів для моніторингу та покращення роботи може ускладнити адміністрування системи.

SQLite є оптимальним вибором для розробки та тестування Django-проектів, малих та середніх вебсайтів, вбудованих систем, мобільних додатків та проектів, де важлива простота розгортання [3]. Однак, для виробничих середовищ з високим навантаженням або великими обсягами даних рекомендується використання більш потужних СУБД, таких як PostgreSQL або MySQL. У контексті розробки системи управління контентом для академічної мобільності, SQLite може бути використана на

етапі розробки та тестування, але для виробничого середовища рекомендується перехід на більш потужну СУБД, таку як PostgreSQL, яка краще підходить для обробки великих обсягів даних та забезпечення конкурентного доступу.

Рисунок 2.1 – Особливості SQLite

2.2 Python

У якості основної мови програмування для розробки системи використовується Python [2]. Ця мова відома своєю простотою, читабельністю та лаконічним синтаксисом, що значно пришвидшує процес розробки та знижує поріг входу для нових учасників команди [10]. Python має велику спільноту розробників, що забезпечує широкий вибір бібліотек, фреймворків та інструментів для вирішення найрізноманітніших завдань - від роботи з базами даних до обробки зображень, машинного навчання та веброзробки [12].

Однією з ключових переваг Python є його чудова інтеграція з Django - потужним фреймворком для створення вебзастосунків [1]. Django дозволяє швидко створювати масштабовані, безпечні та підтримувані вебсайти, використовуючи всі переваги Python, такі як динамічна типізація, багатий стандартний набір бібліотек та підтримка різних парадигм програмування [8]. Python також забезпечує крос-платформність, що дозволяє запускати розроблені додатки на різних операційних системах без необхідності внесення змін у код [2].

Python відзначається високою продуктивністю розробки завдяки великій кількості готових рішень, зручній роботі з пакетним менеджером pip та активній підтримці спільноти [10]. Це робить його ідеальним вибором для швидкого прототипування, розробки MVP, а також для реалізації складних систем із багатим функціоналом. Особливо важливою є підтримка сучасних стандартів безпеки, що дозволяє створювати надійні та захищені вебзастосунки [9].

Водночас, Python має певні обмеження, які слід враховувати [12]. Інтерпретована природа мови може призводити до меншої швидкодії у порівнянні з компільованими мовами, такими як C++ чи Java, особливо у задачах, що вимагають інтенсивних обчислень. Для таких випадків рекомендується використовувати оптимізовані бібліотеки або інтегрувати компоненти, написані іншими мовами. Також, для досягнення максимальної продуктивності у багатопотокових задачах слід враховувати особливості GIL (Global Interpreter Lock).

Python є оптимальним вибором для розробки вебзастосунків, автоматизації, аналізу даних, наукових досліджень, машинного навчання та багатьох інших сфер [10]. У контексті розробки системи управління контентом для академічної мобільності, Python у поєднанні з Django забезпечує швидку розробку, гнучкість та масштабованість рішення, а також легкість підтримки та розширення функціоналу у майбутньому [8].

2.3 Django

Django - це потужний вебфреймворк на мові Python, розроблений для швидкої розробки безпечних та масштабованих вебзастосунків [1]. Він дотримується принципу "batteries included" (все необхідне вже включено), надаючи розробникам повний набір інструментів для створення вебзастосунків [8].

Архітектура Django базується на патерні Модель-Представлення-Шаблон (MVT), який є варіацією MVC [1]. Цей підхід дозволяє чітко розділити логіку застосунку, представлення даних та шаблони, що сприяє кращій організації коду та його підтримці. Однією з ключових переваг Django є його потужна ORM система, яка дозволяє працювати з базою даних через Python-об'єкти, спрощуючи роботу з

даними та забезпечуючи безпеку від SQL-ін'єкцій [7]. Фреймворк автоматично генерує адміністративний інтерфейс для управління даними, що значно прискорює розробку, а його система шаблонів з підтримкою наслідування, фільтрів та тегів надає гнучкість у створенні користувацьких інтерфейсів [6]. Особливу увагу Django приділяє безпеці, включаючи багато вбудованих механізмів захисту від CSRF, XSS, SQL-ін'єкцій та інших поширених вебвразливостей [9].

Django знайшов широке застосування в багатьох великих вебпроектах та сервісах [8]. Серед найвідоміших прикладів - Instagram, Pinterest, Mozilla та Disqus, які демонструють масштабованість та надійність фреймворку. Він також активно використовується в системах управління контентом, таких як Django CMS та Wagtail, соціальних мережах та електронних комерційних платформах, включаючи Saleor та Oscar.

Основні переваги Django включають швидку розробку завдяки "batteries included" підходу, високу безпеку завдяки вбудованим механізмам захисту, масштабованість та гнучкість архітектури [1]. Велика спільнота розробників та наявність багатьох готових рішень, разом з чудовою документацією та навчальними матеріалами, роблять його привабливим вибором для розробників [8]. Проте, фреймворк має і деякі обмеження: він може бути надмірним для простих проектів, має досить круту криву навчання для початківців, пропонує меншу гнучкість порівняно з мікрофреймворками, а використання Python може бути обмеженням для деяких проектів [9].

Загалом, Django є надійним інструментом для розробки вебзастосунків, який надає розробникам потужний набір інструментів для створення безпечних, масштабованих та легко підтримуваних вебзастосунків [1]. Його використання особливо доречне для великих проектів, де важлива безпека, масштабованість та швидкість розробки [8].

Рисунок 2.3 – Логотип Django

2.4 Django Templates

Django Templates - це потужна система шаблонів, вбудована у фреймворк Django для розробки вебінтерфейсів [6]. Вперше представлена разом з Django у 2005 році, ця система дозволяє розробникам створювати динамічні вебсторінки, ефективно розділяючи логіку застосунку від його представлення. Основна ідея полягає у використанні шаблонів для генерації HTML-сторінок з динамічним контентом.

В основі Django Templates лежить концепція шаблонів, які представляють собою HTML-файли зі спеціальними тегами та фільтрами для вставки динамічного контенту [6]. Це дозволяє структурувати вебсторінки на логічні блоки, значно полегшуючи розробку та підтримку . Система використовує власну мову шаблонів (Django Template Language, DTL), яка відрізняється простим та інтуїтивно зрозумілим синтаксисом, що спрощує створення сторінок. Однією з найпотужніших особливостей є підтримка наслідування шаблонів, яка дозволяє створювати базові шаблони та розширювати їх у дочірніх шаблонах, уникаючи дублювання коду та забезпечуючи консистентність дизайну [1].

Дані в шаблони передаються через контекст, що забезпечує безпечне відображення даних та надає контроль над доступною інформацією [6]. Система також надає багатий набір вбудованих фільтрів та тегів для маніпуляції даними та контролю потоку виконання в шаблонах. Ця функціональність знайшла широке

застосування у багатьох відомих вебсайтах, включаючи Instagram, Pinterest та Mozilla, демонструючи свою ефективність у проектах різного масштабу [8].

Django Templates має ряд значних переваг, включаючи простоту використання та низький поріг входу для нових розробників [6]. Система забезпечує потужне наслідування шаблонів, вбудований захист від XSS-атак та тісну інтеграцію з Django ORM та формами. Велика кількість вбудованих фільтрів та тегів значно розширює можливості розробки [1]. Проте, система має і деякі обмеження: обмежену можливість виконання складних операцій безпосередньо в шаблонах, необхідність розуміння Django Template Language та меншу гнучкість порівняно з сучасними JavaScript-фреймворками [9].

Загалом, Django Templates є надійним інструментом для створення ве-інтерфейсів, який пропонує оптимальний баланс між простотою використання, безпекою та інтеграцією з Django [6]. Ці характеристики роблять його популярним вибором серед розробників вебзастосунків, особливо в контексті проектів, де важлива швидкість розробки та надійність[1].

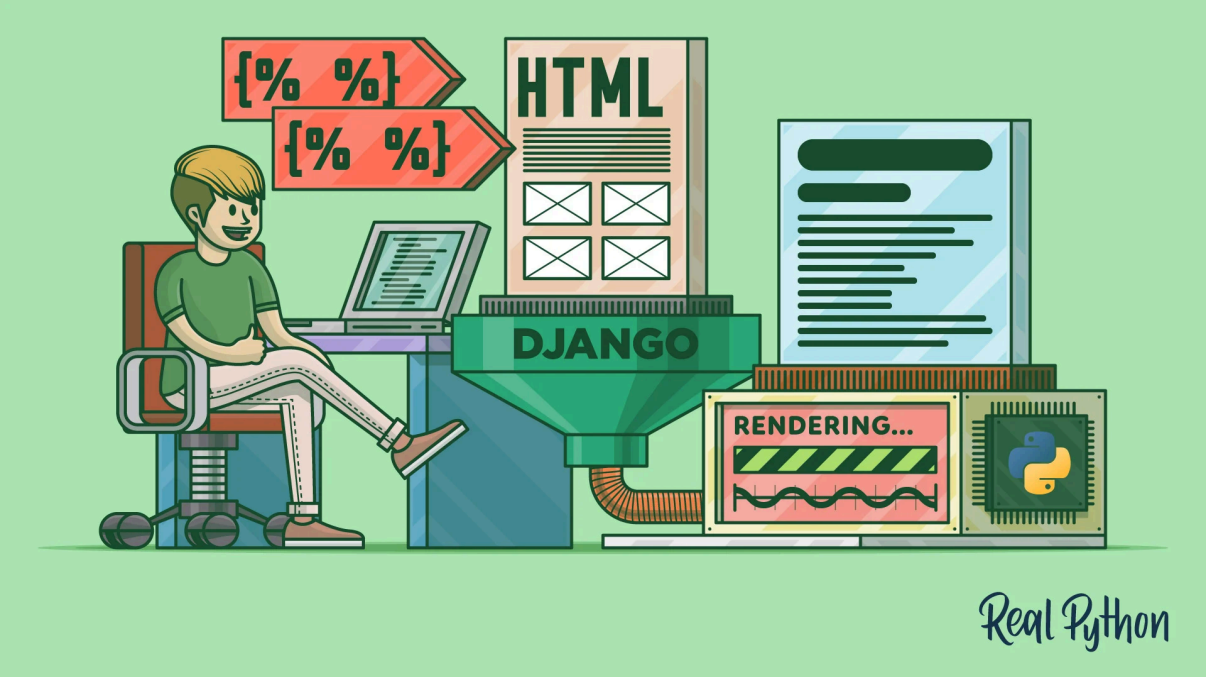


Рисунок 2.4 – Особливості Django Templates

2.5 Django CMS

Django CMS - це система управління контентом, побудована на основі фреймворку Django, яка надає потужні інструменти для створення та управління вебконтентом [5]. Система відрізняється модульною структурою, де кожен функціональний блок, такий як фотогалереї, блоги чи новини, може бути організований в окремий плагін. Такий підхід значно полегшує розширення функціональності та підтримку коду. Завдяки інтеграції з Django Template Language, система надає потужну підтримку шаблонів з можливістю наслідування, використання фільтрів та тегів, що дозволяє створювати гнучкі та динамічні вебсторінки [6].

Однією з ключових особливостей Django CMS є його система плагінів, яка забезпечує простий спосіб розширення функціональності без необхідності модифікації ядра системи [5]. Потужна система версіонування дозволяє ефективно керувати історією змін контенту та підтримувати контроль над публікаціями, що є критично важливим для великих проектів.

Django CMS знайшов широке застосування в різних сферах [5]. Система ідеально підходить для створення корпоративних сайтів, блогів та інформаційних порталів завдяки простому та потужному підходу до управління контентом. В освітній сфері Django CMS забезпечує всі необхідні інструменти для створення різноманітних освітніх ресурсів, від невеликих навчальних сайтів до складних систем управління освітнім контентом. Особливо варто відзначити можливість створення та керування багатомовними сайтами, що сприяє доступності контенту для різних аудиторій.

Система має ряд значних переваг, включаючи модульну структуру, вбудовану підтримку шаблонів та потужну систему плагінів, які роблять Django CMS ефективним інструментом для розробки вебсайтів з управлінням контентом [5]. Проте, варто враховувати, що вивчення системи може зайняти певний час, а також те, що Django CMS, як і будь-який інший інструмент, може не відповідати потребам деяких проектів чи команд розробки [11].

Загалом, Django CMS є надійним рішенням для створення вебсайтів з

управлінням контентом, особливо для проектів, де важлива гнучкість, масштабованість та простота використання [5]. Його тісна інтеграція з Django забезпечує надійну основу для розробки складних вебзастосунків з багатою функціональністю, що робить його привабливим вибором для багатьох розробників та організацій [1].

2.6 Django Filer

Django Filer - це потужний інструмент для управління файлами в Django, який надає розробникам зручний інтерфейс для роботи з медіа-файлами, включаючи зображення, документи та інші типи файлів. Це рішення дозволяє ефективно організовувати, завантажувати та управляти файлами, а також інтегрувати їх у вебзастосунки, відкриваючи широкі можливості для розробників, які працюють з медіа-контентом.

Система відрізняється зручним інтерфейсом для завантаження, організації та управління файлами різних типів. Завдяки тісній інтеграції з адміністративною панеллю Django, Filer забезпечує зручне управління файлами через вебінтерфейс. Система підтримує роботу з різноманітними типами файлів, включаючи зображення, документи, відео та інші медіа-файли. Однією з ключових особливостей є автоматична оптимізація, яка дозволяє Filer автоматично оптимізувати завантажені зображення та створювати їх мініатюри для ефективного використання.

Django Filer знайшов широке застосування в різних сферах. Система дозволяє ефективно керувати медіа-файлами в вебзастосунках, включаючи фотогалереї та бібліотеки документів. Особливо варто відзначити тісну інтеграцію з Django CMS, яка надає зручні інструменти для управління медіа-контентом в системах управління контентом [5]. На основі Django Filer можна розробити потужні системи для створення та управління фотогалереями та медіа-бібліотеками.

Система має ряд значних переваг, включаючи зручний інтерфейс для управління файлами, тісну інтеграцію з Django та автоматичну оптимізацію медіа-контенту. Проте, варто враховувати, що для великих проектів може

знадобитися додаткова налаштування для оптимізації продуктивності, а деякі розширені функції можуть вимагати додаткової конфігурації.

Загалом, Django Filer є надійним рішенням для управління медіа-файлами в Django-проектах, особливо для систем управління контентом, де важлива ефективна робота з медіа-контентом [1]. Його інтеграція з Django CMS та адміністративною панеллю Django робить його ідеальним вибором для проектів, що потребують зручного управління файлами [5]. Система надає розробникам потужний інструментарій для ефективної роботи з медіа-контентом, що робить її незамінною для багатьох сучасних вебпроектів.

2.7 Django ORM

Django ORM (Object-Relational Mapping) - це потужний інструмент для взаємодії з базами даних у Django, який дозволяє розробникам працювати з даними через Python-об'єкти [7]. Однією з ключових переваг Django ORM є його універсальність - система підтримує різні реляційні бази даних, включаючи PostgreSQL, MySQL, SQLite та Oracle. Це дозволяє розробникам використовувати один і той же код для різних систем управління базами даних, значно спрощуючи розробку та підтримку застосунків [1].

Система міграцій Django ORM є особливо потужною, автоматично відстежуючи зміни в моделях та створюючи необхідні SQL-скрипти для оновлення структури бази даних [7]. Розробники можуть використовувати звичний Python-синтаксис для визначення моделей даних та їх взаємодії з базою даних, що робить код більш читабельним та підтримуваним [8]. Django ORM надає розширену підтримку різних типів відносин між моделями, включаючи зв'язки один-до-одного, один-до-багатьох та багато-до-багатьох, а також пропонує потужний API для створення складних запитів [1].

Безпека є одним з пріоритетів Django ORM - система автоматично захищає від SQL-ін'єкцій та інших поширених вразливостей бази даних [7]. Для підвищення продуктивності система підтримує кешування запитів, а також надає можливості для

виконання складних агрегаційних запитів та додавання обчислюваних полів до результатів запитів. Інтеграція з системою тестування Django дозволяє легко створювати та виконувати тести для моделей даних, забезпечуючи надійність та якість коду [9].

Загалом, Django ORM є невід'ємною частиною екосистеми Django, надаючи розробникам потужний та зручний інструмент для роботи з базами даних [7]. Його універсальність, безпека та зручність використання значно спрощують розробку та підтримку вебзастосунків, роблячи його незамінним інструментом для сучасних Django-проектів [1]. Система поєднує в собі потужність реляційних баз даних з простотою та елегантністю Python-програмування, що робить її ідеальним вибором для розробки складних вебзастосунків [8].

2.8 HTML & CSS

HTML (HyperText Markup Language) та CSS (Cascading Style Sheets) є фундаментальними технологіями для створення вебсторінок, які разом формують основу сучасного вебдизайну [4]. HTML відповідає за структуру та семантичне значення контенту вебсторінок, використовуючи систему тегів для визначення різних елементів, таких як заголовки, параграфи, списки та таблиці. Ця семантична структура забезпечує не тільки правильне відображення контенту, але й його доступність для пошукових систем та допоміжних технологій [13].

CSS, як доповнююча технологія, надає потужні інструменти для контролю візуального представлення HTML-елементів [4]. Він дозволяє розробникам визначати кольори, шрифти, відступи, розташування елементів та створювати анімації, перетворюючи базову структуру в привабливий та функціональний інтерфейс. Особливо важливою є можливість створення адаптивних вебсторінок, які коректно відображаються на різних пристроях та роздільних здатностях, що є критичним для сучасного веброзробки [13].

Система CSS побудована на принципах модульності та повторного використання коду [4]. Розробники можуть створювати стилі через систему класів,

ідентифікаторів та селекторів, що дозволяє ефективно організовувати та підтримувати код. Важливу роль відіграє система специфічності та каскаду, яка визначає пріоритет застосування стилів у складних вебзастосунках [13].

Сучасна розроблення вебінтерфейсів значно посилюється завдяки використанню додаткових інструментів [4]. CSS-препроцесори, такі як SASS та LESS, розширюють можливості CSS, додаючи підтримку змінних, вкладених правил та функцій, що значно покращує підтримку коду. Популярні CSS-фреймворки, включаючи Bootstrap та Tailwind CSS, надають готові компоненти та утиліти, що прискорює розробку та забезпечує консистентність дизайну. Ці інструменти стали невід'ємною частиною сучасного веброзробки, дозволяючи створювати складні та адаптивні інтерфейси з мінімальними зусиллями [13].

Загалом, комбінація HTML та CSS, разом з сучасними інструментами розробки, надає потужну основу для створення сучасних вебінтерфейсів [4]. Ці технології постійно еволюціонують, додаючи нові можливості та покращення, що робить їх незамінними для веброзробки [13].

2.9 JavaScript

JavaScript - це мова програмування, яка дозволяє додавати інтерактивність та динамічну поведінку до вебсторінок. Однією з ключових особливостей JavaScript є можливість створення динамічних елементів інтерфейсу, обробки подій користувача та зміни контенту сторінки без її перезавантаження. Ця інтерактивність робить JavaScript незамінним інструментом для створення сучасних вебзастосунків.

Система асинхронного програмування в JavaScript є особливо потужною, надаючи розробникам різні підходи до роботи з асинхронними операціями. Завдяки Promise, async/await та колбекам, розробники можуть ефективно працювати з мережевими запитами та іншими асинхронними задачами. JavaScript також надає потужний API для роботи з Document Object Model (DOM), що дозволяє динамічно змінювати структуру та вміст вебсторінок, створюючи багаті інтерактивні інтерфейси.

Мова підтримує як об'єктно-орієнтоване, так і функціональне програмування, надаючи розробникам гнучкість у виборі підходу до розробки. Об'єктно-орієнтоване програмування реалізовано через підтримку класів, наслідування, інкапсуляції та поліморфізму. Функціональне програмування підтримується через концепції функцій вищого порядку, замикань та чистої функціональності, що дозволяє писати більш декларативний та підтримуваний код.

Сучасний JavaScript підтримує модульну архітектуру через ES6 модулі та системи збірки, що сприяє кращій організації коду та його повторному використанню. Велика екосистема бібліотек та фреймворків, включаючи jQuery, React та Vue.js, значно спрощує розробку, надаючи готові рішення для типових завдань. Ці інструменти стали стандартом у сучасній веброзробці, дозволяючи створювати складні додатки з мінімальними зусиллями.

Безпека є важливим аспектом JavaScript, і мова надає вбудовані механізми захисту від поширених вебвразливостей. Розробники також мають доступ до різних інструментів та технік для оптимізації продуктивності JavaScript-коду, що є критичним для створення швидких та ефективних вебзастосунків.

Загалом, JavaScript є потужною та гнучкою мовою програмування, яка надає розробникам всі необхідні інструменти для створення сучасних вебзастосунків. Його поєднання з HTML та CSS формує основу сучасної веброзробки, дозволяючи створювати інтерактивні, візуально привабливі та функціональні вебзастосунки. Постійна еволюція мови та її екосистеми робить JavaScript одним з найбільш важливих інструментів у арсеналі сучасного веброзробника.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Структура бази даних

База даних для проекту складається з декількох основних таблиць, які забезпечують функціональність вебсайту університету:

1. Review (Відгуки):
- id: унікальний ідентифікатор відгуку (primary key)
- full_name: повне ім'я автора відгуку
- review_text: текст відгуку
- date: дата відгуку
- photo: фотографія автора відгуку

2. PostContentExtension (Розширення контенту сторінок):
- id: унікальний ідентифікатор
- post_content: вміст посту
- extended_object: зв'язок з основною сторінкою CMS

3. CarouselSlide (Слайди каруселі):
- id: унікальний ідентифікатор слайду
- photo: зображення для слайду

4. CooperationPluginModel (Модель для секції співпраці):
- id: унікальний ідентифікатор
- title: заголовок
- description: опис
- image1, image2: зображення для секції

5. EmbeddedVideo (Вбудоване відео):

- id: унікальний ідентифікатор

- video_url: URL відео

Для керування структурою бази даних використовується система міграцій Django [7]. Міграції дозволяють зберігати структуру бази даних в коді та автоматично вносити зміни до бази даних при змінах у схемі. Наприклад, якщо потрібно додати нове поле або змінити тип існуючого поля, створюється нова міграція, яка описує необхідні зміни.

База даних для проекту використовує SQLite як двигун бази даних, що налаштовано в файлі settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / "db.sqlite3",
    }
}
```

Рисунок 3.1 - Оголошення бази данних

Проект використовує Django CMS для управління контентом, що дозволяє легко додавати та редагувати контент через адміністративну панель. Всі моделі інтегровані з Django CMS через систему плагінів, що забезпечує гнучкість у розміщенні контенту на сторінках.

Для зберігання медіафайлів (зображень, відео) використовується наступна конфігурація:

```
MEDIA_URL = '/media/'
MEDIA_ROOT = '/home/emtcnltueduua/public_html/media'
```

Рисунок 3.2 - Конфігурація медіафайлів

3.2 Серверна частина проекту

Під час розробки проекту я використовував Django - потужний Python вебфреймворк, який надав зручні інструменти для створення серверної частини додатку. Django допоміг встановити зв'язок з базою даних та обробляти HTTP-запити ефективно та безпечно.

Модульна структура Django дозволила організувати код додатку у вигляді додатків (apps), моделей та представлень (views), що спростило розробку та підтримку додатку. Для організації маршрутів та обробки HTTP-запитів я використовував вбудовані засоби Django URLconf, що дозволило зосередитися на реалізації бізнес-логіки додатку.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('cms/', include('cms.urls')),
    path('', include('posts.urls')),

]
```

Рисунок 3.3 - Основна структура URL-маршрутів проекту визначена в файлі core/urls.py

Для обробки запитів користувачів використовуються представлення (views), які знаходяться в файлі `posts/views.py`. Основні функції включають:

1. `home` - головна сторінка українською мовою
2. `en_home` - головна сторінка англійською мовою
3. `dynamic_template_view` - динамічне відображення шаблонів для української версії
4. `english_dynamic_template_view` - динамічне відображення шаблонів для англійської версії

Проект підтримує мультимовність через систему шаблонів Django [6]. Для перемикання між мовами використовується спеціальний тег

`change_language_url_lowercase`, який автоматично генерує URL для протилежної мови:

```python
@register.simple_tag(takes_context=True)
def change_language_url_lowercase(context):
    """
    Returns the URL for the current page in the opposite language
    by adding/removing 'en' from the path
    """
    try:
        # Get current path
        path = context['request'].path

        # If we're on an English page (has 'en' in path)
        if '/en/' in path:
            return path.replace('/en/', '/')

        # If we're on a Ukrainian page (no 'en' in path)
        else:
            # Split the path and insert 'en'
            parts = path.split('/')
            # Insert 'en' after the first slash
            parts.insert(1, 'en')
            return '/'.join(parts)

    except Exception:
        return '/'
```

Рисунок 3.4 - оголошення тегу для зміни мови сторінок

Для обробки статичних файлів (CSS, JavaScript, зображення) та медіафайлів налаштовані відповідні URL-патерни:

```python
if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Рисунок 3.5 - Додавання URL-патерни до статичних файлів

Проект інтегрований з Django CMS для управління контентом, що дозволяє легко додавати та редагувати контент через адміністративну панель. Всі моделі інтегровані з Django CMS через систему плагінів, що забезпечує гнучкість у розміщенні контенту на сторінках.

```
CSRF_TRUSTED_ORIGINS = ["*"]
CSRF_COOKIE_SECURE = True
CSRF_USE_SESSIONS = True
CSRF_COOKIE_HTTPONLY = True

CMS_TOOLBAR_SHOW_TOOLBAR_TO_STAFF = True
```

Рисунок 3.6 - Налаштування безпеки

Також було використано Django шаблони та Django CMS для розробки користувацького інтерфейсу [6]. Django CMS є потужним інструментом для створення динамічних та інтерактивних вебзастосунків, який дозволяє легко організовувати компоненти та ефективно керувати контентом [5].

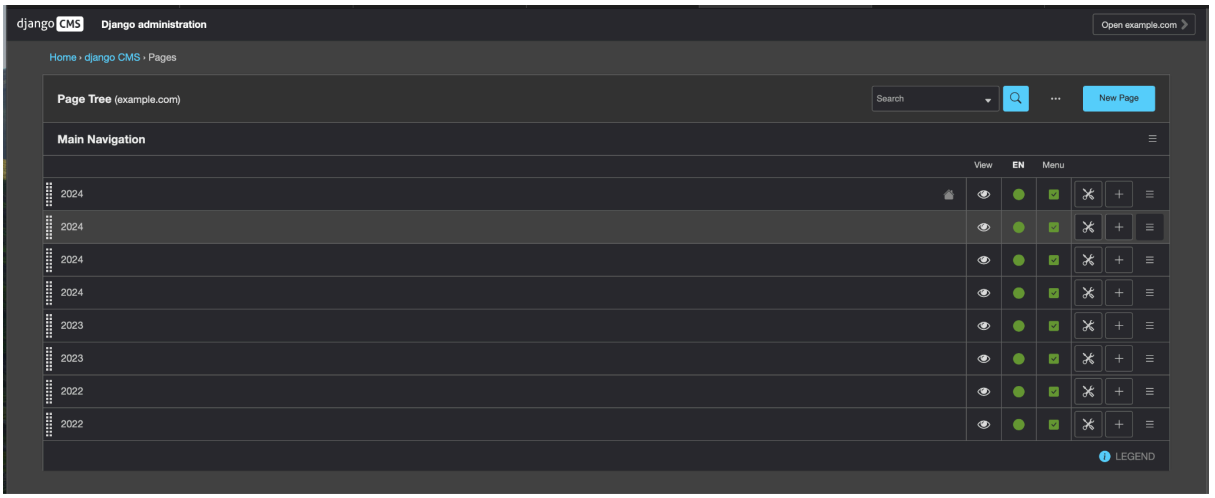


Рисунок 3.7 - Адмін панель сторінок

Django CMS адмін панель дозволяє керувати сторінками сайту, їх основними характеристиками та метаданими [5]. Адміністратор може задавати заголовок сторінки, який відображається для користувачів, а також формувати slug - частину URL, що використовується для ідентифікації сторінки в адресному рядку браузера. Для кожної сторінки доступний вибір шаблону, що визначає її структуру та вигляд, наприклад, можна обрати окремий шаблон для української версії [6].

Передбачена можливість вказати альтернативний заголовок, який буде показано у вкладці браузера або в закладках, що допомагає у SEO та підвищує

зручність для користувачів. Окремо можна додати мета-опис - текст, який використовують пошукові системи для формування короткого опису сторінки у результатах пошуку. Завдяки такому підходу, адмін панель Django CMS забезпечує гнучке та зручне управління контентом, зовнішнім виглядом і оптимізацією сторінок для пошукових систем.

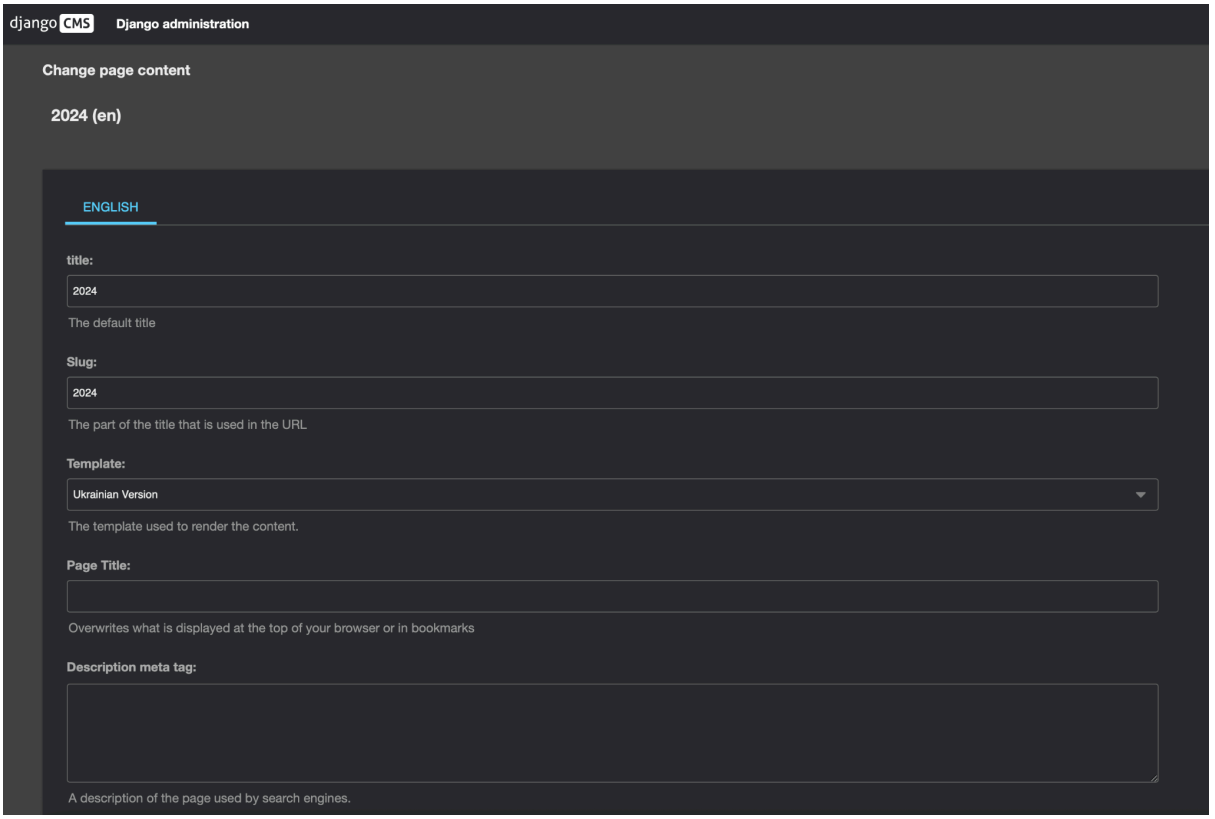


Рисунок 3.8 - Адміністрація інформації сторінки

Django CMS адмін панель забезпечує зручний візуальний інтерфейс для керування структурою та наповненням сторінок сайту. У правій частині екрана розташована панель контенту, де відображається ієрархія всіх секцій та плагінів, які складають сторінку. Кожен елемент, наприклад, текстова секція, секція з текстом і зображенням, відео чи колекція посилань, представлений окремим блоком із можливістю швидкого редагування, переміщення або видалення.

Завдяки такому підходу адміністратор може легко додавати нові блоки, змінювати їх порядок, а також налаштовувати вміст безпосередньо на сторінці. Для кожного плагіна доступні інструменти редагування, що дозволяють швидко змінювати текст, зображення чи інші параметри. Весь контент структурується у

вигляді вкладених секцій, що дає змогу створювати складні сторінки з різноманітними типами інформації[5].

Інтерфейс також дозволяє попередньо переглядати зміни перед їх публікацією, що забезпечує контроль над виглядом сторінки для кінцевого користувача. Така система значно спрощує процес наповнення сайту, робить його інтуїтивно зрозумілим навіть для користувачів без глибоких технічних знань, і дозволяє швидко адаптувати структуру сторінки під потреби проєкту [5].

Використання Django в цьому проекті дало можливість створити потужну та масштабовану серверну частину додатку з чистим та організованим кодом, а також забезпечити високу продуктивність та безпеку [1].

Рисунок 3.9 - Адміністрація елементів сторінки

3.3 Клієнтська частина проекту

На клієнтській стороні сайт, створений на основі Django CMS, виконує роль сучасної платформи для презентації студентських проєктів, досягнень та взаємодії з відвідувачами [5]. Головна сторінка містить зрозумілий заголовок і перелік тематичних звітів, кожен із яких представлений у вигляді інтерактивного посилання.

Це дозволяє користувачам швидко знайти та перейти до цікавого для них матеріалу.

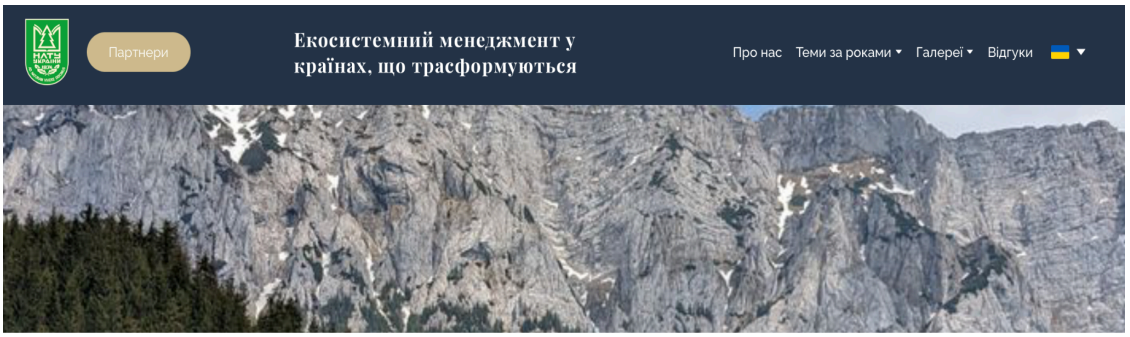
Сайт інтегрує різноманітний мультимедійний контент: фотографії з подій, презентації, а також відео, створені самими студентами. Такий підхід не лише робить сторінку більш живою та привабливою, а й дозволяє повніше передати атмосферу заходів, показати командну роботу та креативність учасників. Відвідувачі можуть переглядати фотозвіти, знайомитися з результатами досліджень, а також отримувати враження від відеопрезентацій, що розкривають тему з різних сторін.

Окремо реалізована система відгуків, функціонал якої базується на моделі Review, що є частиною Django CMS [5]. Це дозволяє гнучко контролювати відображення відгуків студентів і викладачів на сайті, забезпечуючи прозорість та зворотний зв’язок.

Рисунок 3.10 - Відгуки студентів

Рисунок 3.11 - Відгуки викладачів

Після входу на сайт користувач бачить головну сторінку з основною інформацією. Завдяки підтримці двомовності, кожен може перемикатися між українською та англійською версіями інтерфейсу. Це робить ресурс доступним для ширшої аудиторії та сприяє популяризації студентських ініціатив за межами України.

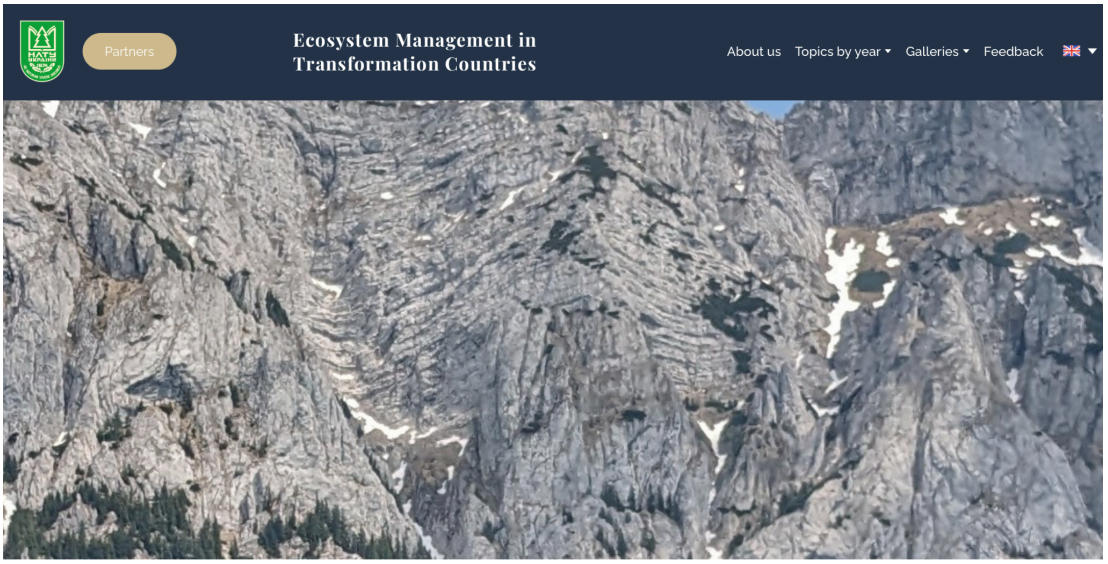


Яка наша мета?

На прикладі регіону в обраній країні, що трансформується, студенти вивчають, наскільки соціоекономічні та політичні процеси впливають на екосистеми та пропонують шляхи покращення якості екосистемного менеджменту, використовуючи отримані знання під час вивчення дисципліни. Студентам надається можливість виявити та застосувати екосистемні та соціоекономічні показники для оцінки потенційних змін у системі.

Крім теоретичної підготовки та загальної інформації про регіон, студенти знаходяться в реальних ситуаціях екосистемного менеджменту на обраних для дослідження територіях Східної Європи та навчаються їх інтерпретувати на основі знань про трансформаційні процеси та на основі обміну думками з місцевими експертами. Перевага надається біосферним заповідникам як об'єктам для вивчення. Досліджується наскільки вони можуть втілити у життя ключовий аспект концепції біосферних заповідників – бути лабораторіями для сталого розвитку за умов соціополітичних та соціоекономічних трансформацій, а також глобальних змін у навколишньому середовищі.

Рисунок 3.12 - Головна сторінка Українською мовою



What is our goal?

Using the example of a region in a selected country undergoing transformation, students learn to what extent socio-economic and political transformation processes affect ecosystems and suggest ways to improve the quality of ecosystem management, using the knowledge gained during the study of the discipline. Students get the opportunity to identify and apply ecosystem and socio-economic indicators to assess potential changes in the system.

Рисунок 3.13 - Головна сторінка Англійською мовою

Додатковою перевагою є адаптивний дизайн: сайт коректно відображається як на комп'ютерах, так і на мобільних пристроях [13]. Це дозволяє зручно користуватися всіма функціями ресурсу незалежно від типу пристрою.

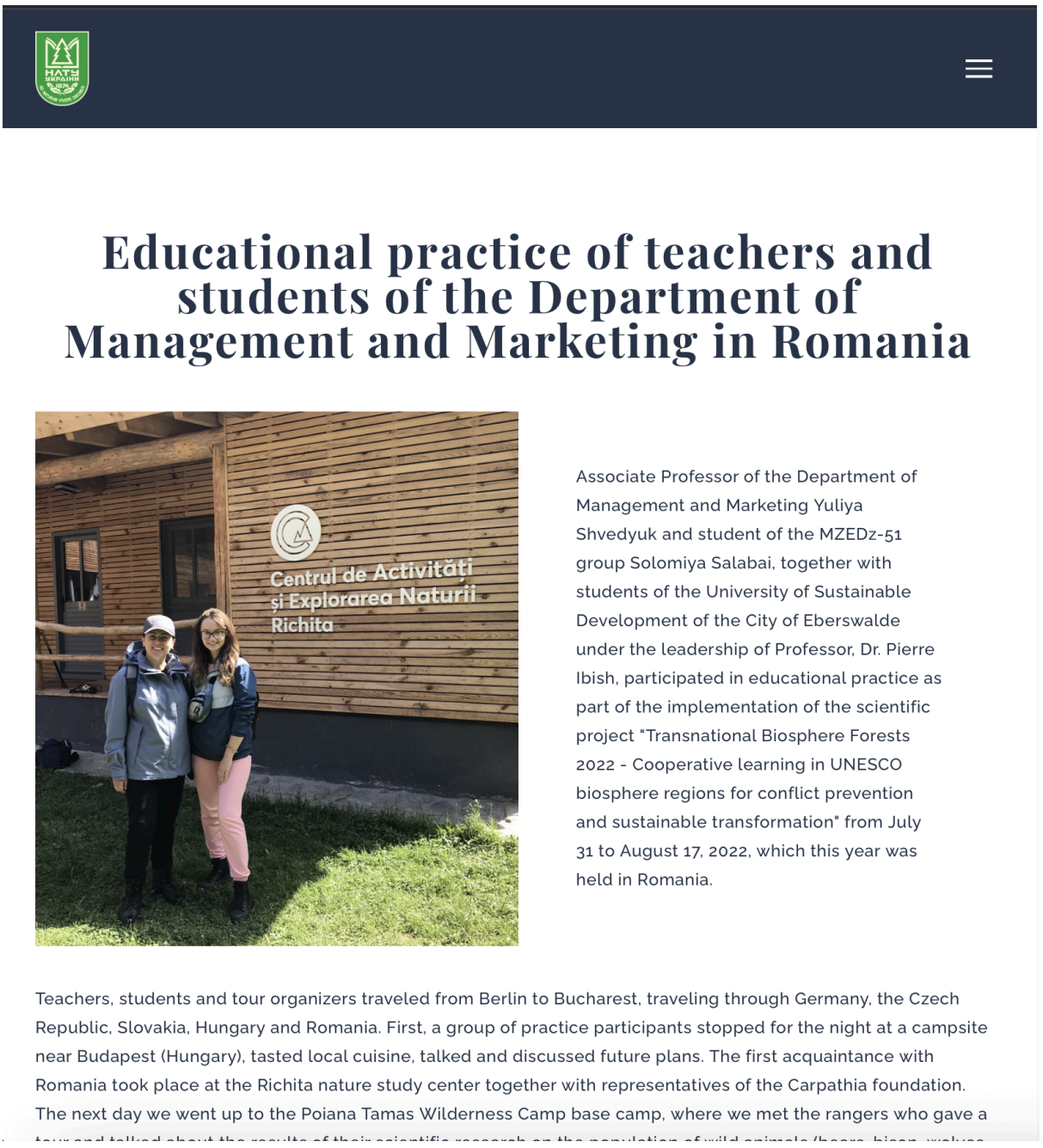


Рисунок 3.14 - Тема 2022 року Англійською мовою в версії для мобільних телефонів

На сторінках сайту реалізовано можливість інтеграції зовнішнього мультимедійного контенту за допомогою ембедингів. Завдяки цьому,

Google-презентації можуть бути відображені безпосередньо на сайті, що дозволяє користувачам переглядати слайди у зручному інтерактивному форматі без необхідності переходу на сторонні ресурси. Також підтримується вбудовування відео з YouTube, завдяки чому відеопрезентації, репортажі чи творчі роботи студентів стають доступними для перегляду прямо на сторінці. Така інтеграція робить сайт більш динамічним, сприяє кращому сприйняттю інформації та дозволяє максимально повно представити результати студентської діяльності у різних форматах.

Рисунок 3.15 - Відображення інтегрованої Google-презентації на сторінці сайту

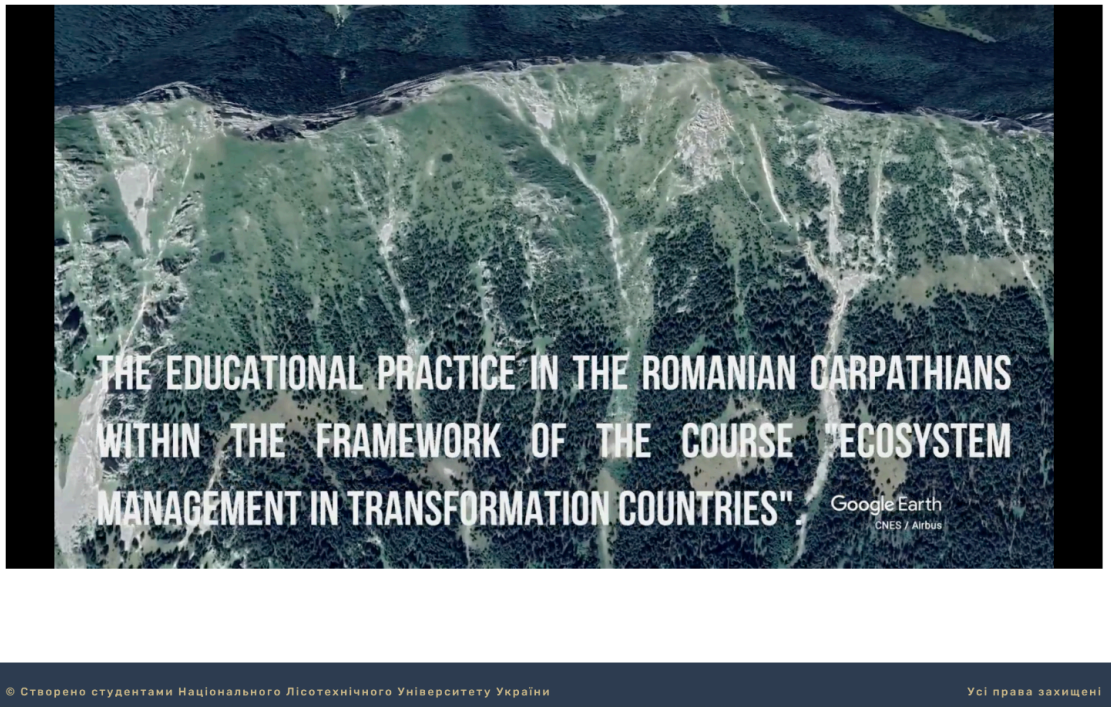


Рисунок 3.16 - Відображення вбудованого відео з YouTube у клієнтському інтерфейсі

Галереї організовані у вигляді сітки зображень, що забезпечує швидкий візуальний огляд усіх доступних фото. Кожне зображення можна збільшити для детальнішого перегляду - при натисканні відкривається окреме вікно з фотографією у великому розмірі, не залишаючи поточної сторінки.

Такий підхід робить знайомство з атмосферою подій максимально наочним і зручним для користувачів. Галереї сприяють популяризації студентських активностей, дозволяють поділитися враженнями з широкою аудиторією та зберігають найяскравіші моменти у відкритому доступі для всіх відвідувачів сайту.

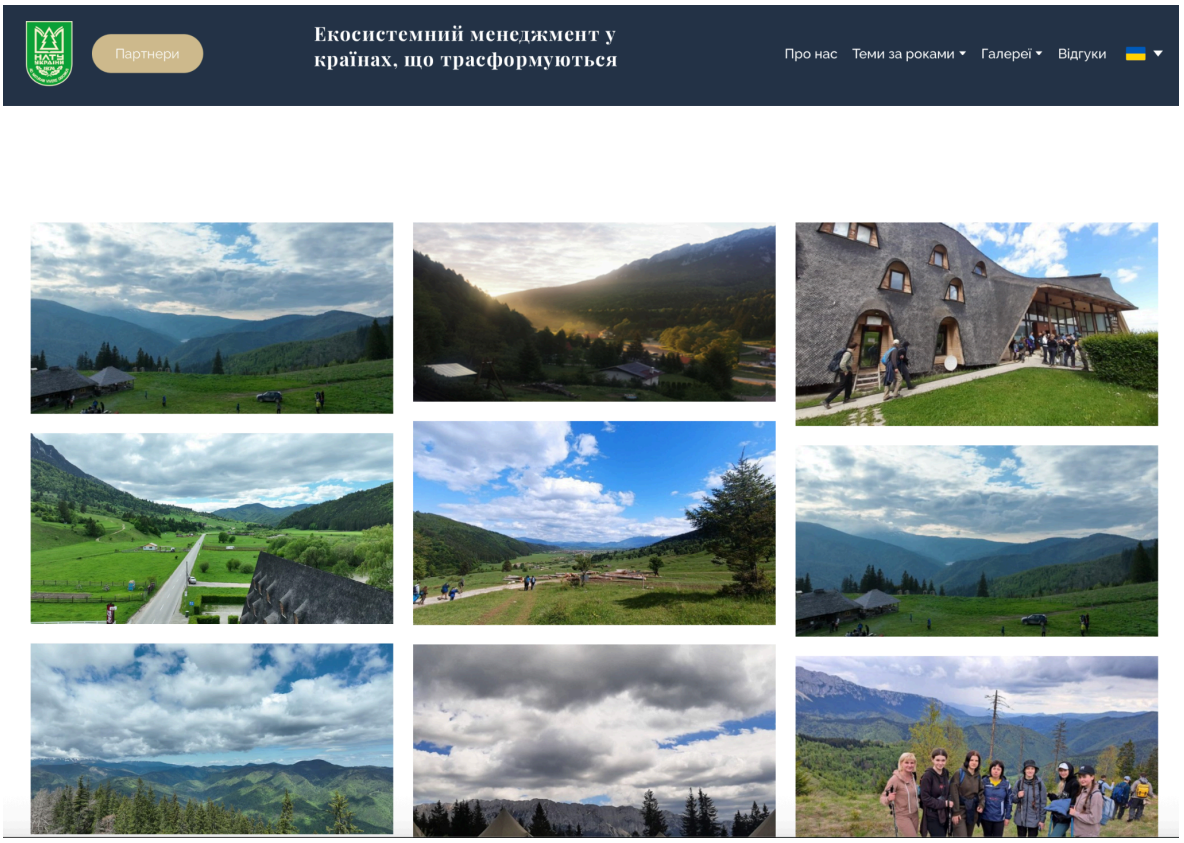


Рисунок 3.17 - Сторінка-галерея з фотографіями студентської навчальної поїздки

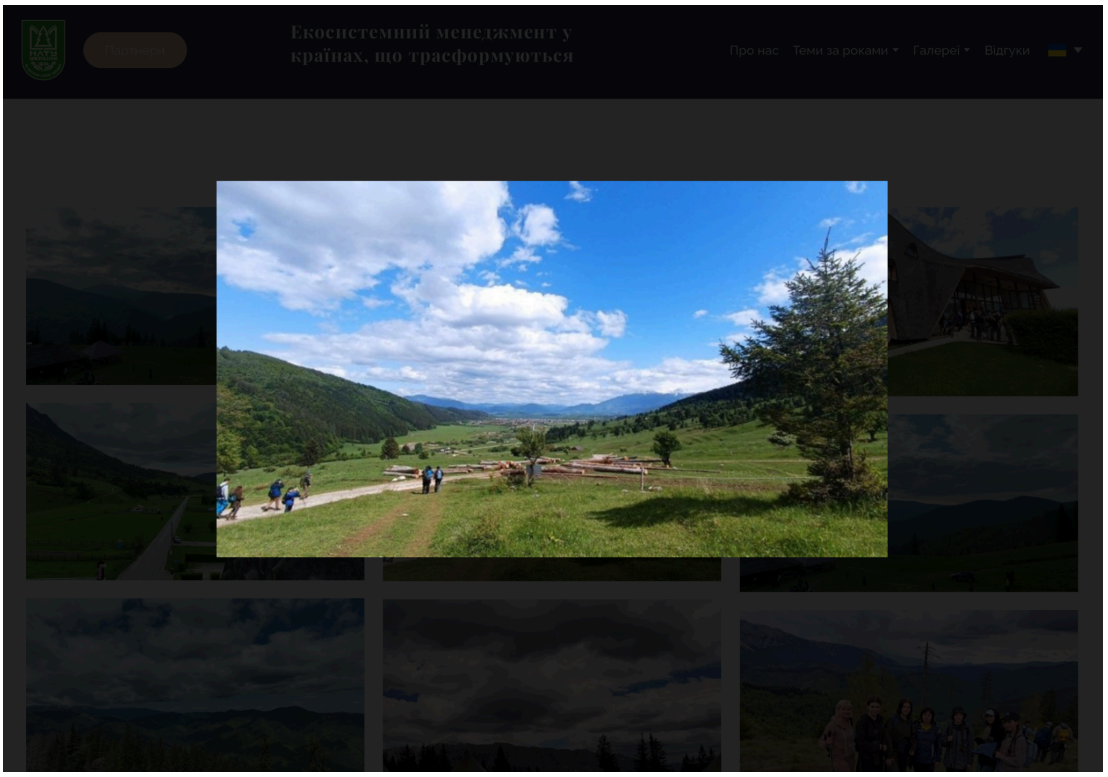


Рисунок 3.18 - Збільшене відображення вибраної фотографії у галереї

Завдяки структурованому розміщенню матеріалів, сучасному дизайну та інтеграції системи відгуків, сайт є зручним для навігації, сприйняття інформації та

взаємодії з аудиторією [5]. Він слугує не лише архівом студентських робіт, а й платформою для обміну досвідом, натхнення та популяризації наукових і творчих здобутків молоді.На сайті реалізовано окремі сторінки-галереї, які дозволяють зручно переглядати фотографії з навчальних поїздок, практик та інших заходів.

ВИСНОВКИ

У процесі розробки вебзастосунку для університету були досягнуті всі поставлені цілі та вирішені всі завдання. Вебзастосунок, створений за допомогою технологій Django, Django CMS та Python, демонструє високу продуктивність та зручність користування. Було розроблено базу даних на основі SQLite, яка забезпечує надійне збереження даних про контент сайту. Використання Django CMS дозволило спростити управління контентом та забезпечити стабільність системи. Інтеграція з системою плагінів забезпечила гнучкість у розміщенні та редагуванні контенту на сторінках.

Серверна частина додатку, розроблена з використанням Django, забезпечує ефективну обробку запитів та безпечне управління контентом. Використання вбудованих інструментів Django полегшило процес розробки, забезпечуючи автоматичне оновлення контенту. Клієнтська частина, розроблена з використанням Django шаблонів, дозволила створити динамічний та зручний для користувачів інтерфейс, який легко масштабується. Впровадження системи відгуків значно покращило користувацький досвід, надаючи можливість залишати відгуки про університет.

Таким чином, створений вебзастосунок не лише відповідає сучасним вимогам до якості та функціональності, але й демонструє високий рівень технічної реалізації, що робить його ефективним інструментом для представлення інформації про університет. Всі поставлені цілі були досягнуті, і проект успішно вирішив поставлені перед ним завдання, що підтверджує його практичну цінність та потенціал для подальшого розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Django Documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/
2. Python Documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/
3. SQLite Documentation [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html
4. HTML5 & CSS3 Documentation [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/
5. Django CMS Documentation [Електронний ресурс]. – Режим доступу: https://docs.django-cms.org/
6. Django Templates Documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/stable/topics/templates/
7. Django Models [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/stable/topics/db/models/
8. Antonio Mele. *Django 4 By Example: Build powerful and reliable Python web applications from scratch*. – Packt Publishing, 2022. – 700 p.
9. William S. Vincent. *Django for Professionals: Production websites with Python & Django*. – WelcomeToCode, 2020. – 300 p.
10. Eric Matthes. *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*. – No Starch Press, 2019. – 544 p.
11. Luke Welling, Laura Thomson. *PHP and MySQL Web Development* (although PHP-focused, contains CMS architecture ideas). – Addison-Wesley, 2019. – 1000+ p.
12. Mark Summerfield. *Programming in Python 3: A Complete Introduction to the Python Language*. – Addison-Wesley, 2018. – 648 p.
13. Jon Duckett. *HTML and CSS: Design and Build Websites*. – Wiley, 2018 (reprint). – 490 p.

ДОДАТОК А
ФРАГМЕНТИ КОДУ ПЛАГІНІВ

```
import os
from cms.models import CMSPlugin
from cms.plugin_base import CMSPluginBase
from django.db import models
from cms.extensions import PageExtension
from django.templatetags.static import static
from django.utils.translation import gettext_lazy as _
from filer.fields.image import FilerImageField
from django.contrib import admin
class PostContentExtension(PageExtension):
post_content = models.TextField()
class MyTextPluginModel(CMSPlugin):
text = models.CharField(max_length=255, verbose_name="Text")
def __str__(self):
return self.text
class CooperationPluginModel(CMSPlugin):
title = models.CharField(_("Title"), max_length=255)
description = models.TextField(_("Description"))
image1 = models.ImageField(
verbose_name=_("Image 1"),
upload_to='uploads/'
)
image2 = models.ImageField(
verbose_name=_("Image 2"),
upload_to='uploads/'
)
def __str__(self):
return self.title
```

```
def get_image1_url(self):
if self.image1:
return os.path.join(static('media'), os.path.basename(self.image1.name))
return None
def get_image2_url(self):
if self.image2:
return static(f'{self.image2.name}')
return None
class TextSectionPluginModel(CMSPlugin):
title = models.CharField(_("Title"), max_length=255, blank=True, null=True)
def __str__(self):
return self.title or "Text Section"
class Paragraph(models.Model):
text_section = models.ForeignKey(
TextSectionPluginModel,
related_name="paragraphs",
on_delete=models.CASCADE
)
content = models.TextField(_("Content"))
def __str__(self):
return self.content[:50]
class ProjectSectionPluginModel(CMSPlugin):
title = models.CharField(_("Title"), max_length=255, blank=True, null=True)
description = models.TextField(_("Description"), blank=True, null=True)
image = FilerImageField(
verbose_name=_("Image"),
on_delete=models.CASCADE,
related_name="project_section_image"
)
def __str__(self):
return self.title or "Project Section"
```

```
class PicturePluginModel(CMSPlugin):
image = FilerImageField(
verbose_name=_("Image"),
on_delete=models.CASCADE,
related_name="picture"
)
class Review(models.Model):
full_name = models.CharField(max_length=255)
review_text = models.TextField()
date = models.DateField(null=True, blank=True)
photo = models.ImageField(upload_to='reviews/')
def __str__(self):
return f"{self.full_name} - {self.date}"
class ReviewPlugin(CMSPlugin):
title = models.TextField(blank=True, null=True)
reviews = models.ManyToManyField(Review, related_name='review_plugins')
class ReviewPluginModel(CMSPluginBase):
model = ReviewPlugin
name = "Review Carousel"
render_template = "plugins/feedbacks.html"
cache = False
class ImageCarousel(CMSPlugin):
title = models.CharField(max_length=255, verbose_name=_("Main Title"))
class CarouselSlide(CMSPlugin):
photo = models.ImageField(upload_to='carousel_slides/')
class TextBesideTheImage(CMSPlugin):
text = models.TextField()
image = models.ImageField(upload_to='pictures/')
class EmbeddedVideo(CMSPlugin):
video_url = models.URLField()
class GalleryImage(CMSPlugin):
```

```
image = models.ImageField(upload_to='gallery/')

class LinkItem(models.Model):

"""Model for individual link items"""

plugin = models.ForeignKey('Link', related_name='link_items', on_delete=models.CASCADE, null=True, blank=True)

text = models.CharField(_("Link Text"), max_length=255)

url = models.URLField(_("URL"))

order = models.PositiveIntegerField(_("Order"), default=0)

class Meta:

ordering = ['order']

def __str__(self):

return self.text

class Link(CMSPlugin):

"""Plugin model for a collection of links"""

def copy_relations(self, oldinstance):

"""Needed to copy relations when copying plugin"""

for link in oldinstance.link_items.all():

link.pk = None

link.plugin = self

link.save()

class LinkItemInline(admin.StackedInline):

model = LinkItem

extra = 1
```

ДОДАТОК Б
ІНСТРУКЦІЯ З КОРИСТУВАННЯ АМДІН-ПАНЕЛЛЮ

Для створення нової сторінки необхідно бути авторизованим у адміністративній панелі сайту. Першим кроком є вхід до системи під обліковим записом адміністратора.

Для цього потрібно перейти за посиланням: https://emtc.nltu.edu.ua/admin/ і ввести відповідні облікові дані адміністратора. Після успішної авторизації відкривається доступ до управління всім вмістом сайту.

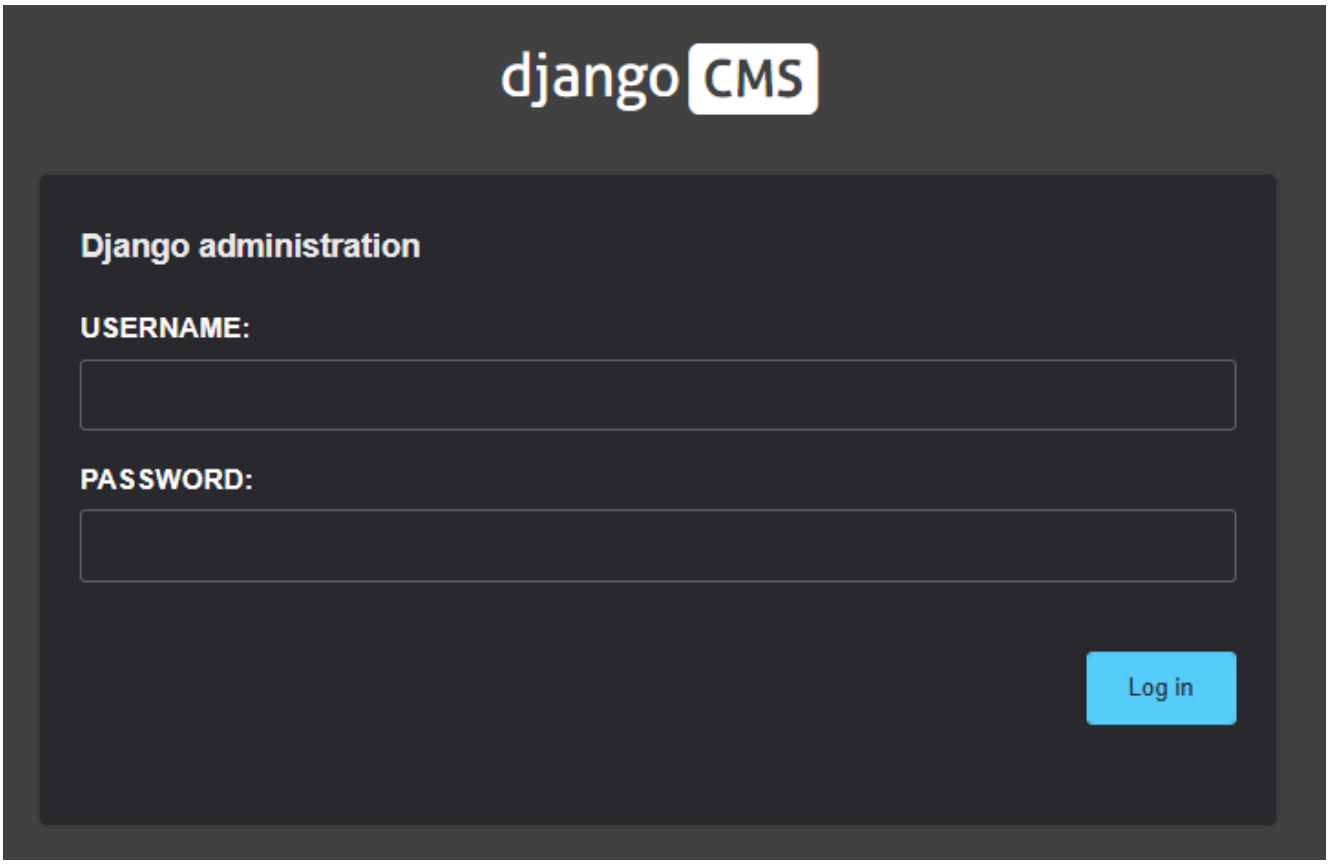


Рисунок Д2.1 - Форма авторизації для адмін панелі

Щоб створити нову сторінку, слід натиснути кнопку **“Create”** на будь-якій сторінці сайту, у спливаючому меню обрати пункт **“New page”**, після чого натиснути кнопку **“Save”** для збереження створеної сторінки.

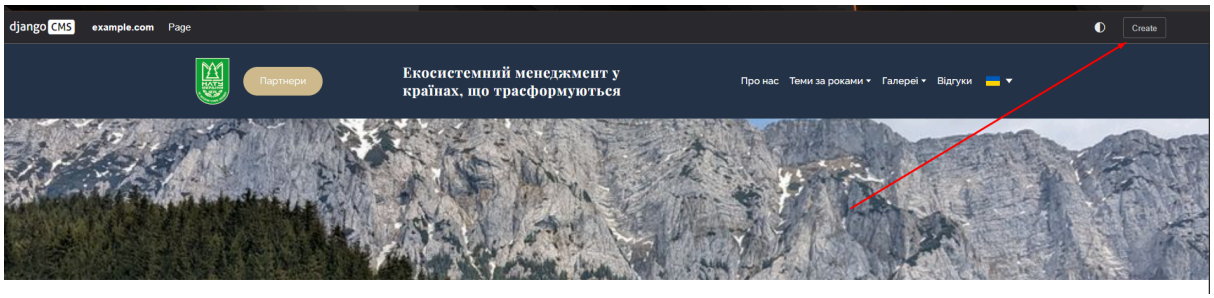


Рисунок Д2.2 - Інтструкція переходу на вікно створення нової сторінки

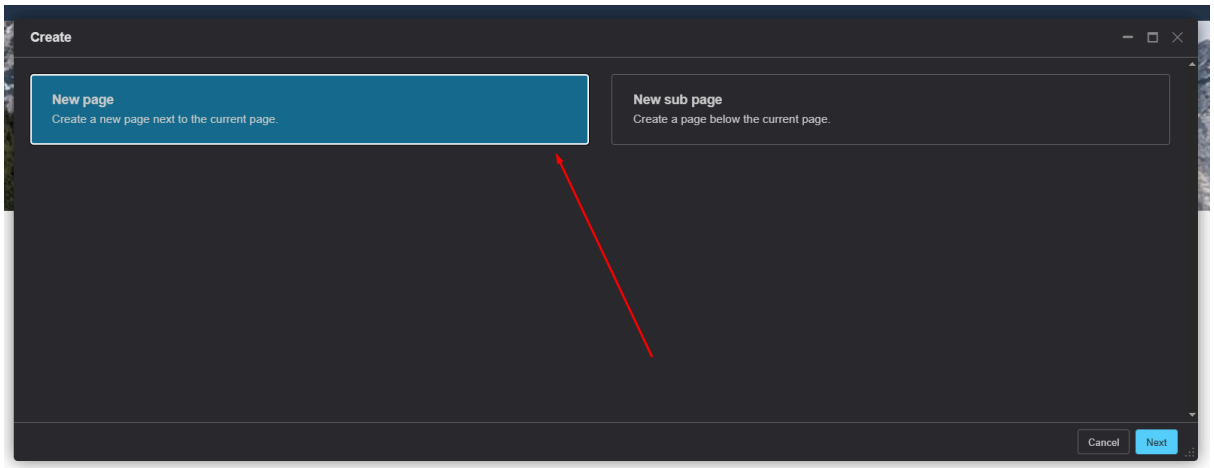


Рисунок Д2.3 - Вікно створення нової сторінки

Після виконання попередніх дій на екрані з’явиться нове вікно, в якому необхідно вказати початкову інформацію для створюваної сторінки. Серед основних параметрів - **заголовок** сторінки (*Title*) та **URL-адреса** сторінки (*Slug*), за якою вона буде доступна.

Після заповнення всіх необхідних полів слід натиснути кнопку **“Create”** для завершення створення сторінки.

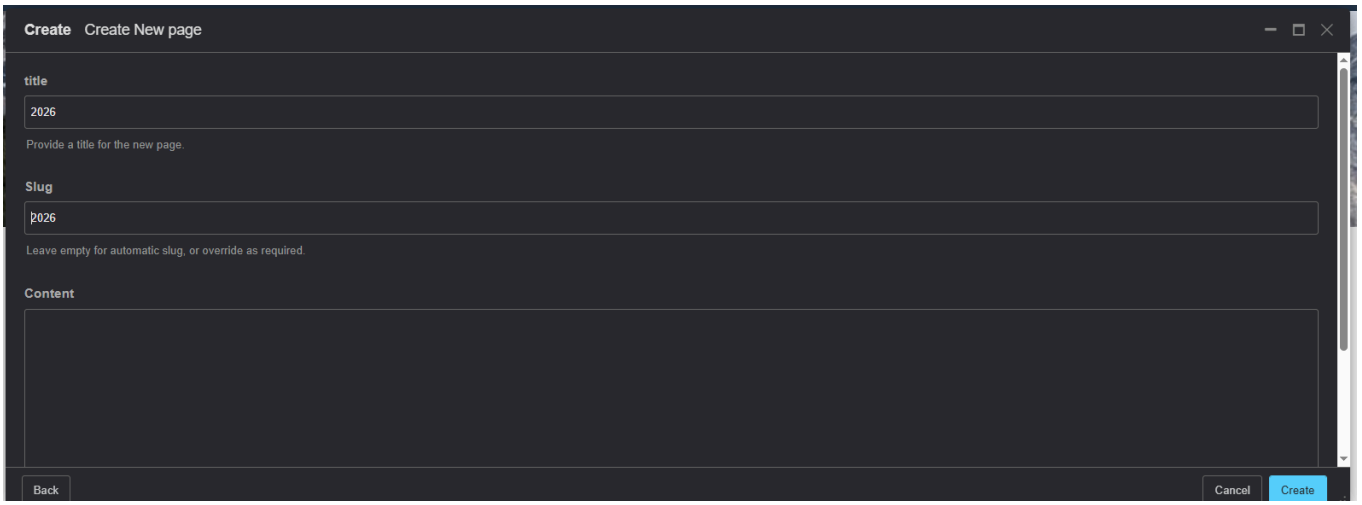


Рисунок Д2.4 - Форма створення нової сторінки

Після створення нової сторінки вона з’явиться у вигляді порожнього шаблону, готового до редагування. Однак перед початком редагування необхідно задати додаткові параметри через адміністративну панель.

Для цього переходимо за посиланням до адмін-панелі та відкриваємо розділ **“Page contents”**. У цьому розділі відкривається панель керування всіма сторінками сайту.

Зі списку обираємо потрібну сторінку, після чого натискаємо кнопку **налаштувань (Settings)** для переходу до її конфігурації.

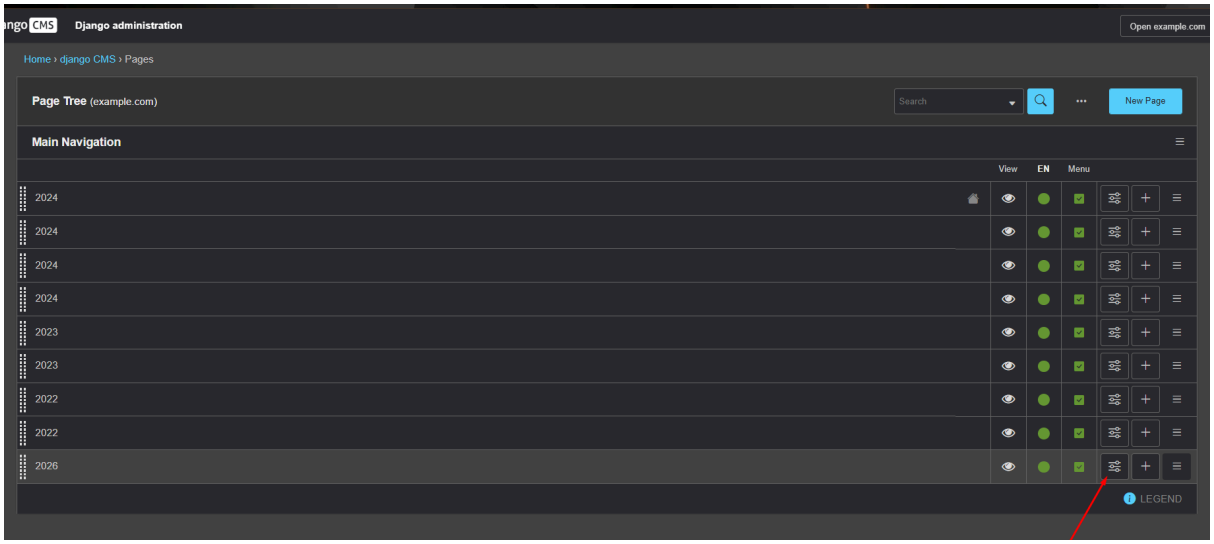


Рисунок Д2.5 - Адмін панель сторінок

У цьому вікні необхідно заповнити ще один важливий параметр, який є ключовим етапом у створенні сторінки. Йдеться про вибір шаблону сторінки.

Потрібно відкрити випадаюче меню **“Templates”**, яке визначає структуру та оформлення майбутньої сторінки. У списку доступні чотири основні шаблони:

- **Ukrainian Version** - українська версія сторінки з темами за роками;
- **English Version** - англійська версія сторінки з темами за роками;
- **Gallery 2023** - українська версія галереї за роками;
- **Gallery 2023 English Version** - англійська версія галереї за роками.

Оскільки ми створюємо сторінку українською мовою з тематикою за роками, обираємо шаблон **“Ukrainian Version”** та натискаємо кнопку **“Save”** для збереження змін.

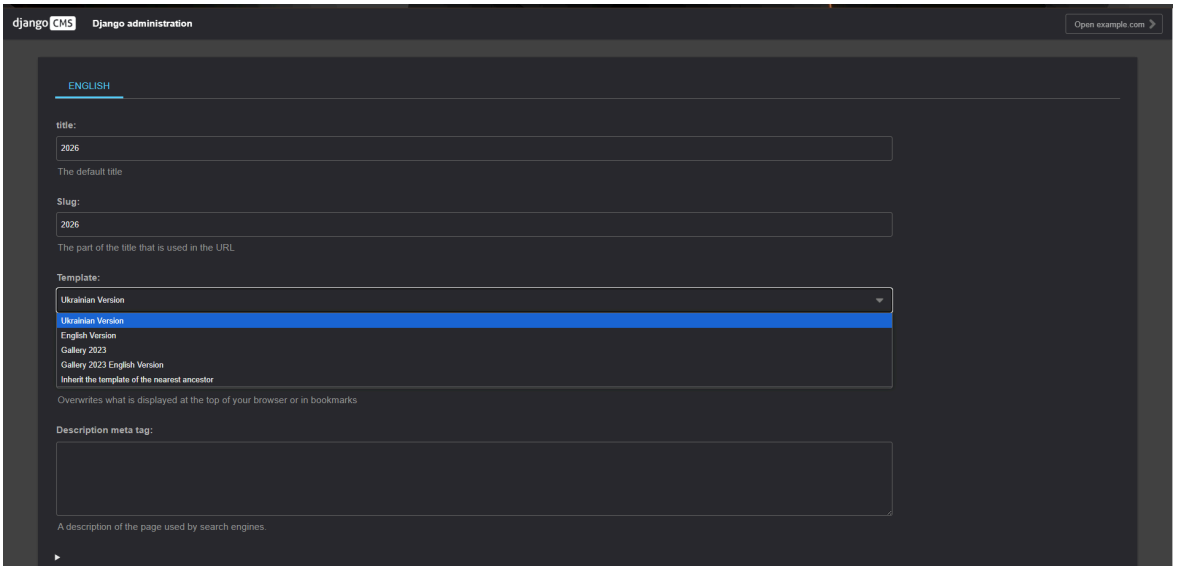


Рисунок Д2.6 - Вибір потрібного шаблона для будування сторінки

Після збереження шаблону наша сторінка буде готова до додавання необхідних елементів.

Тепер переходимо на будь-яку сторінку української версії сайту та відкриваємо випадаюче меню навігації. У ньому з’явиться створена нами сторінка. Залишається лише перейти на неї, щоб розпочати наповнення контентом.

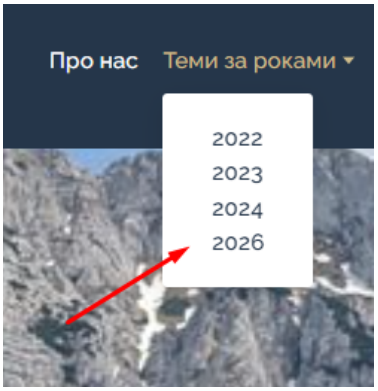


Рисунок Д2.7 - Відображення нової теми за роками в випадаючому меню

Тепер перед нами відображається порожня сторінка, готова до редагування. Щоб розпочати процес, натискаємо кнопку **“Edit”** у верхній панелі.

Після цього відкриється режим редагування, де потрібно натиснути на іконку **“бургер-меню”** (три горизонтальні смужки) - це дозволить відкрити бічне меню з елементами, доступними для додавання на сторінку.

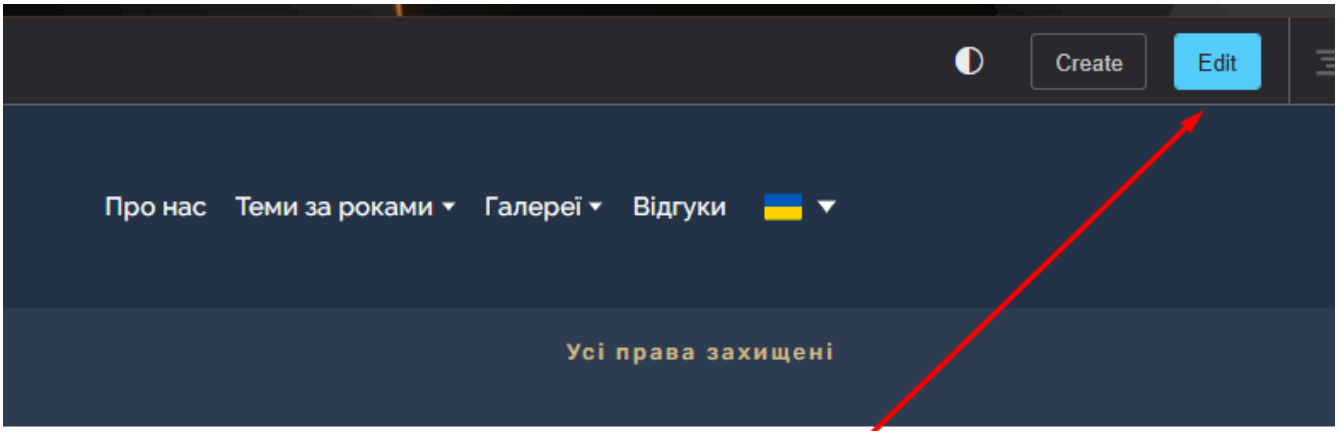


Рисунок Д2.8 - Інструкція з переходу на режим редагування сторінки

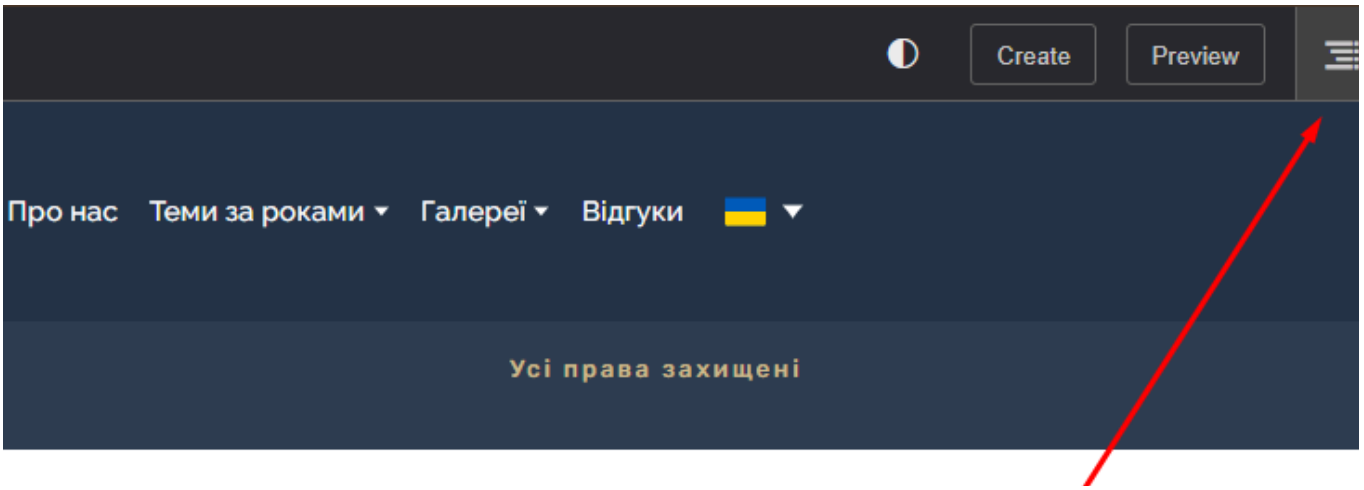


Рисунок Д2.9 - Панель редагування сторінки

Тепер можна додати новий елемент сторінки, натиснувши кнопку **“+”**.

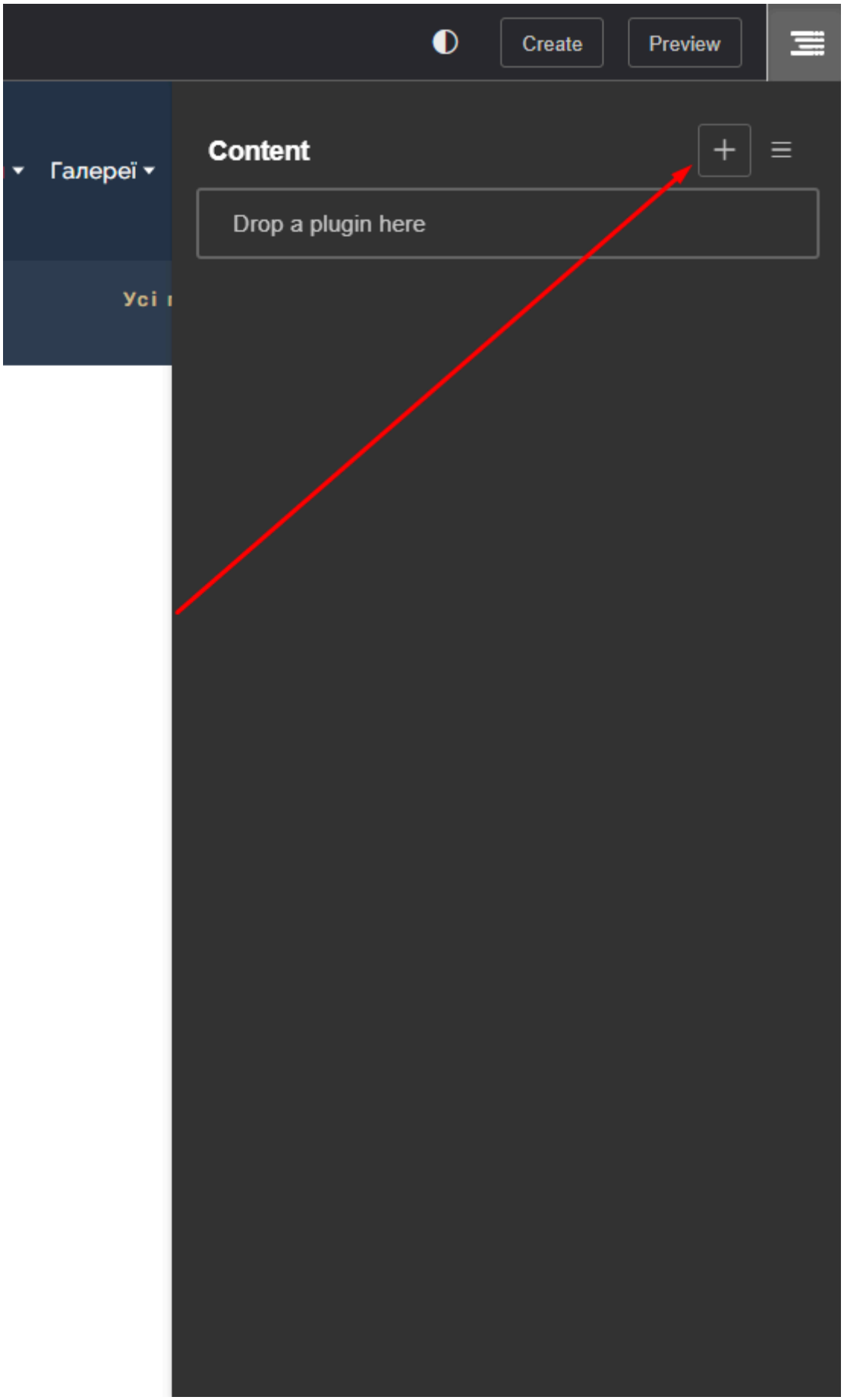


Рисунок Д2.10 - Інструкція з додавання нових елементів на сторінку

У спливаючому меню з’явиться перелік усіх доступних елементів сторінки. Основні елементи, що відповідають стилю сайту, розташовані внизу меню у трьох секціях:

- **Gallery Section** - секція для галерей;
- **Text and Picture Section** - секція для комбінування тексту з зображеннями;
- **University Components** - секція основних компонентів для сторінок університету.

Для тестування було обрано блок **тексту**, який дозволяє додавати абзаци та вставляти зображення.

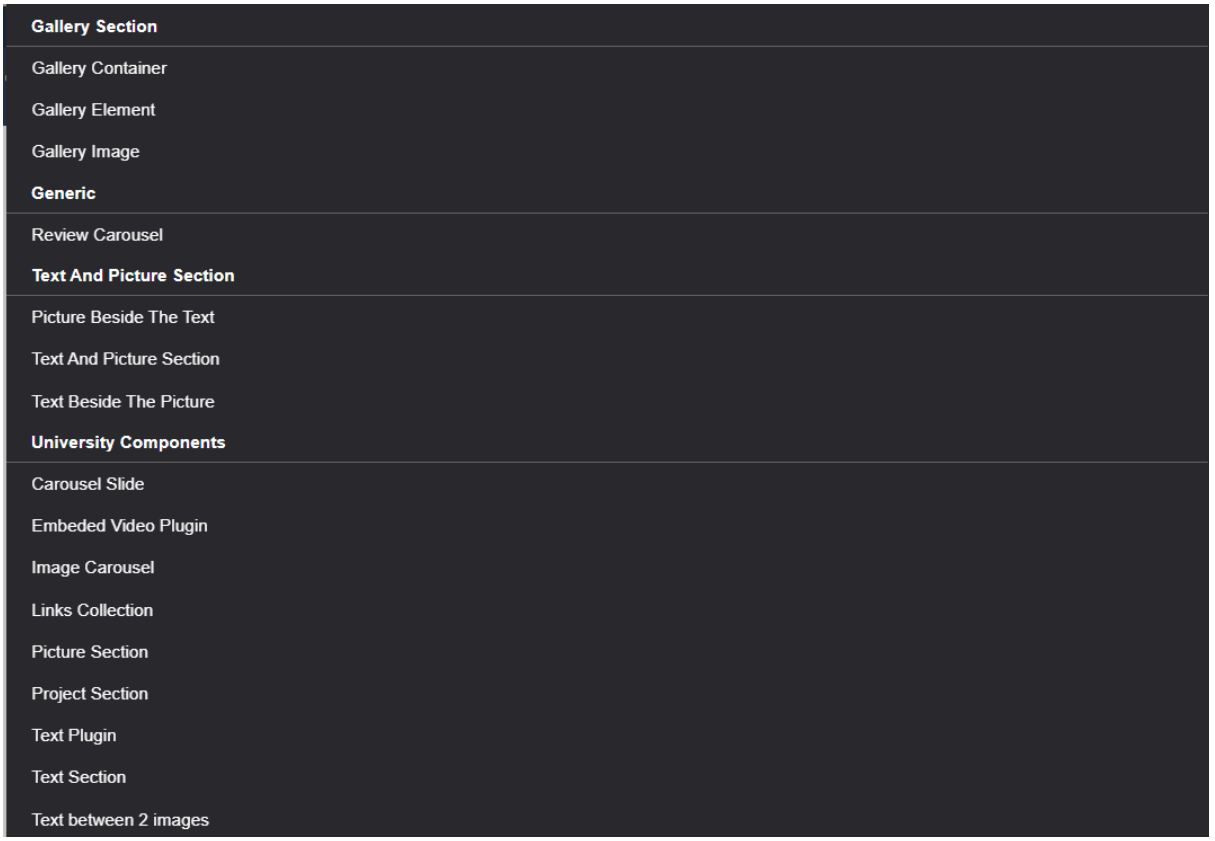


Рисунок Д2.11 - Головні елементи сторінки

У новому меню з’явиться форма для редагування текстового блоку, де можна вказати заголовок блоку (**Title**) та додати абзаци (**Paragraphs**).

Після заповнення необхідних полів натисніть кнопку **“Save”** - і новий елемент одразу з’явиться на сторінці.

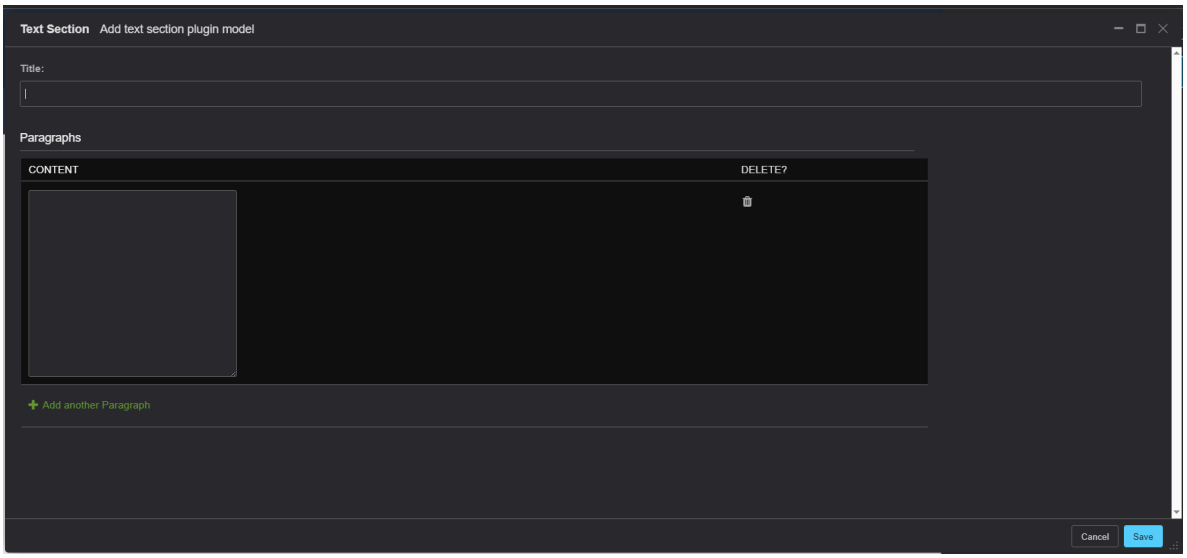


Рисунок Д2.12 - Форма створення секції тексту

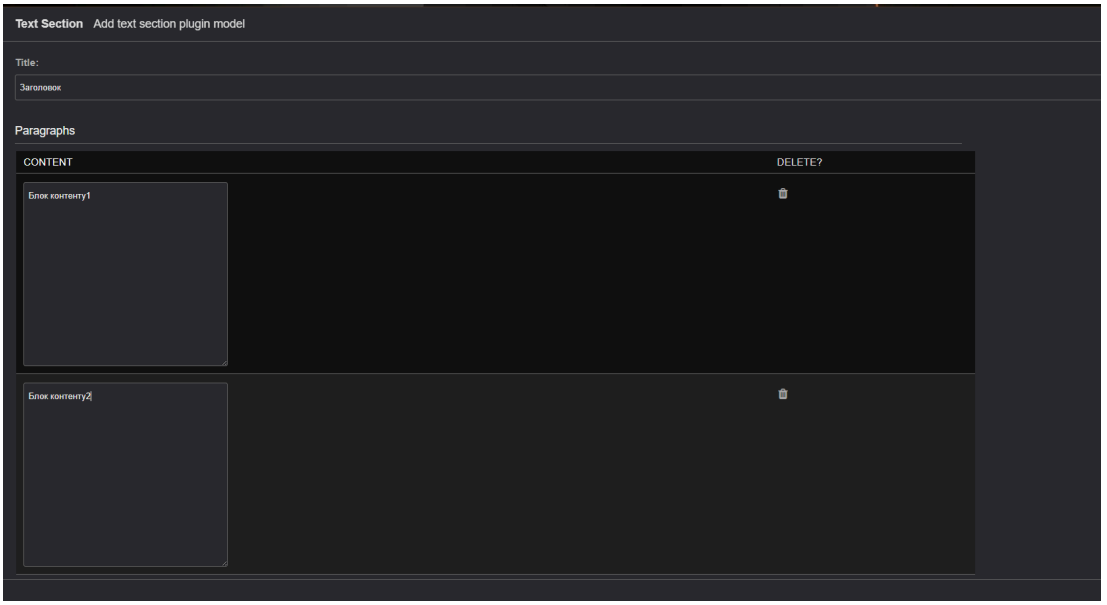


Рисунок Д2.12 - Заповнена форма створення секції тексту

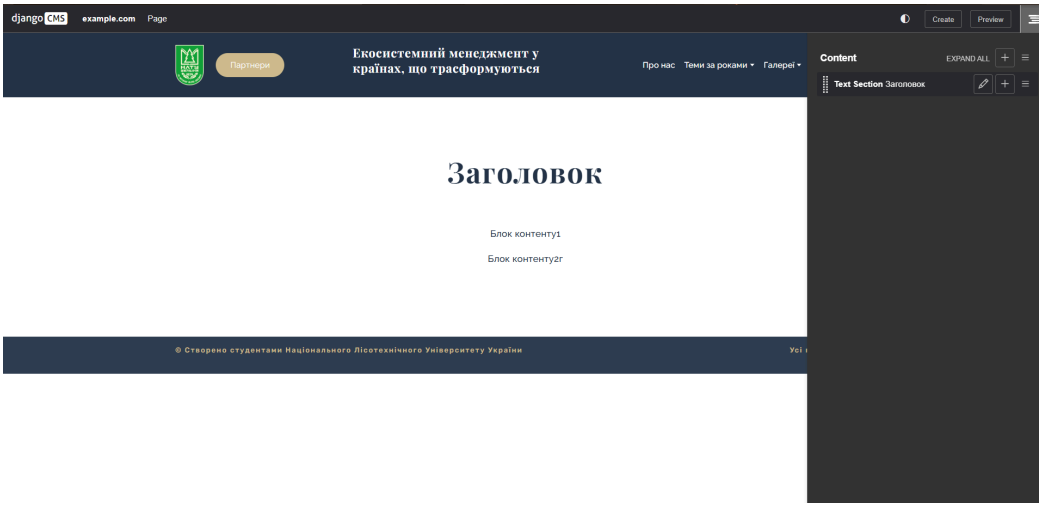


Рисунок Д2.13 - Відображення створенної секції тексту на сторінці

Деякі елементи підтримують додавання вкладених елементів, що дозволяє тримати вміст структурованим і зручним для редагування в межах одного блоку.

Щоб додати новий елемент всередину існуючого, потрібно натиснути кнопку “+” і вибрати потрібний елемент зі списку. У нашому випадку було обрано секцію **“Picture Section”**, яка дає змогу додавати зображення на сторінку.

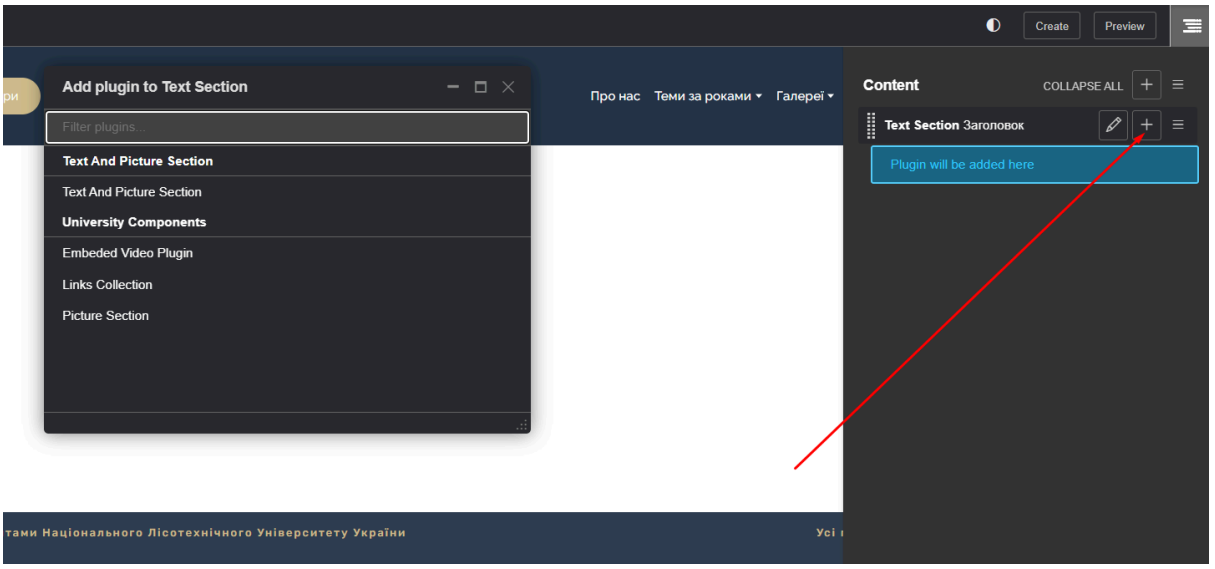


Рисунок Д2.14 - Додавання внутрішніх елементів до секції тексту

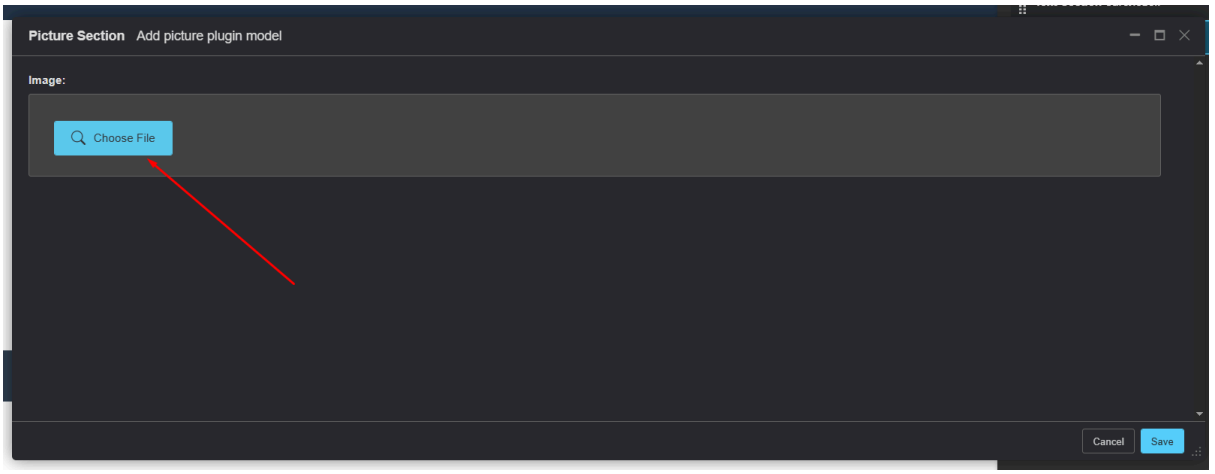


Рисунок Д2.15 - Форма створення секції зображення

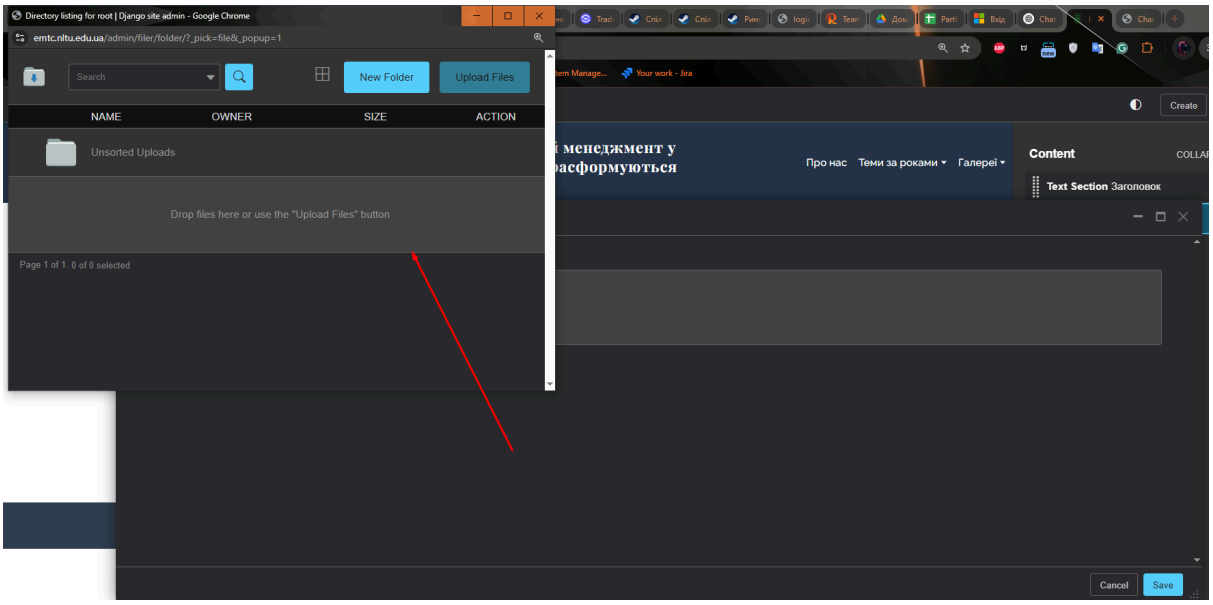


Рисунок Д2.16 - Вивантаження зображення в секцію тексту

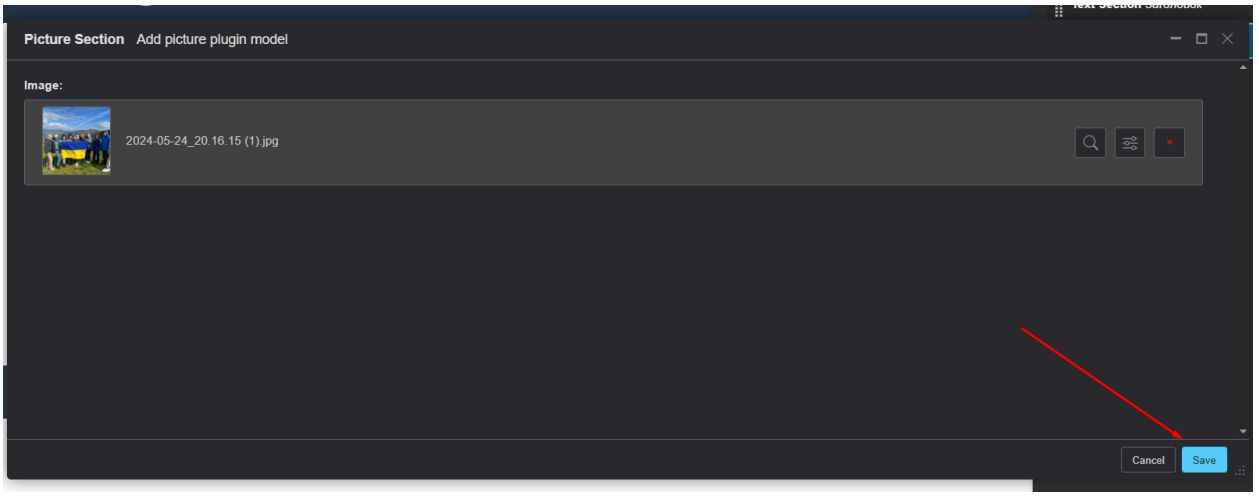


Рисунок Д2.17 - Збереження зображення

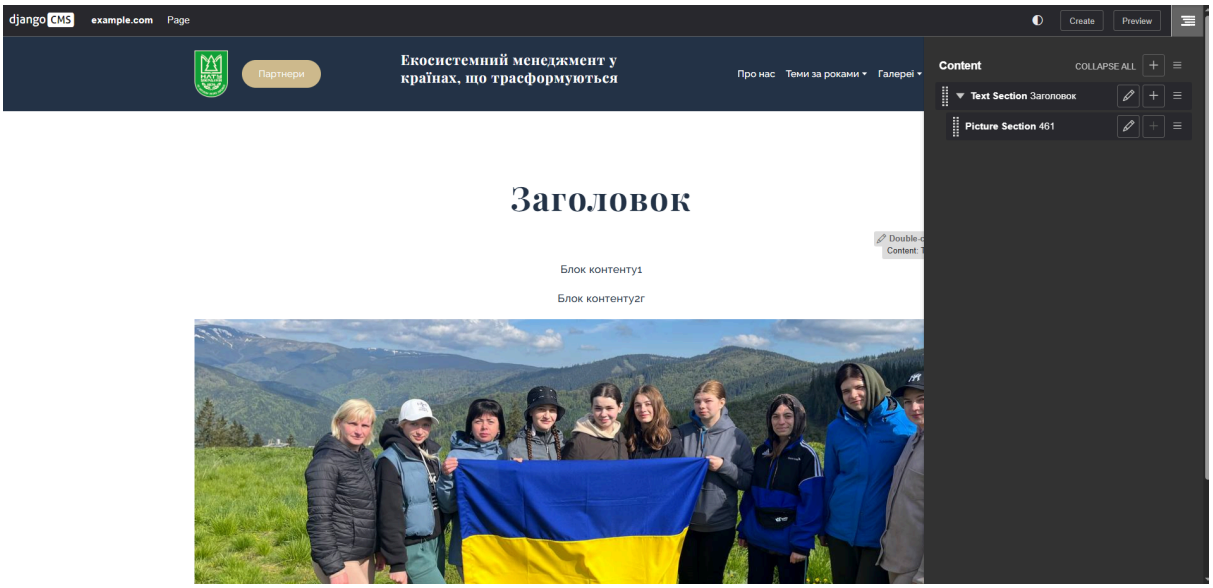


Рисунок Д2.18 - Відображення результату на сторінці

Щоб створити нову сторінку галереї в англомовній версії сайту, адміністратор повинен спочатку створити звичайну сторінку та вказати URL у наступному форматі:

english_version_gallery_2026

(де **“english_version”** вказує сайту, що сторінка має відображатися в англомовній версії, **“gallery”** означає, що сторінка буде розміщена у випадаючому меню галереї, а решта частини посилання є довільною).

Після цього в адміністративній панелі слід обрати відповідний шаблон **“Gallery 2023 English Version”**. На цьому ініціалізація сторінки вважається завершеною.

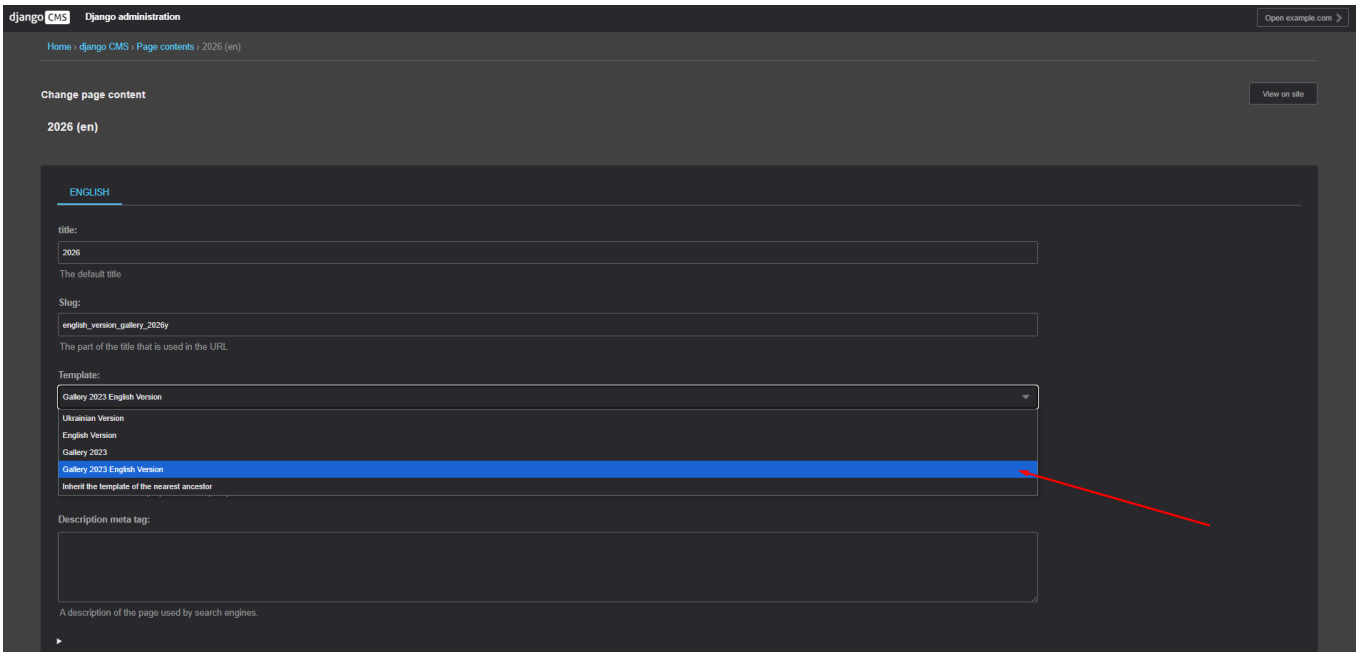


Рисунок Д2.19 - Вибір шаблону англомовної версії галереї для нової сторінки

Далі необхідно перейти на англомовну версію сайту та в розділі галереї обрати створену сторінку.

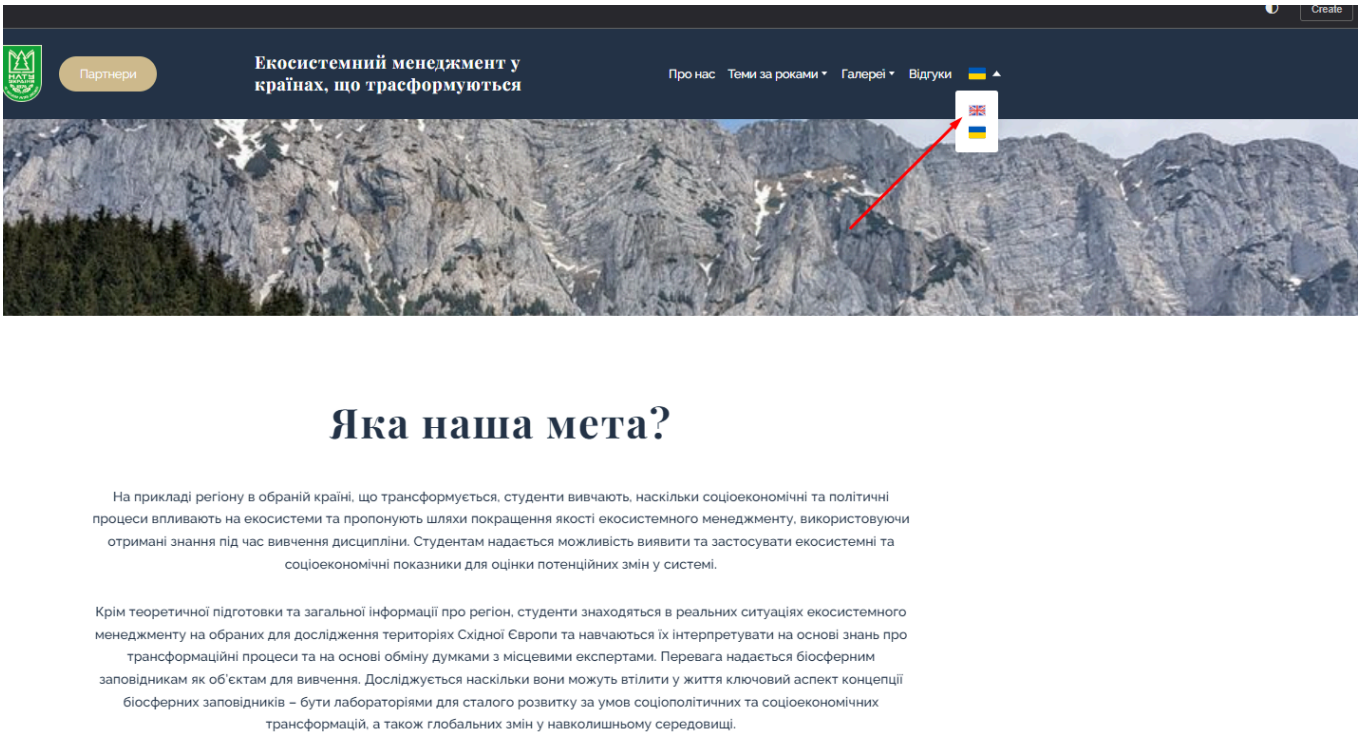


Рисунок Д2.20 - Заміна мови на сайті

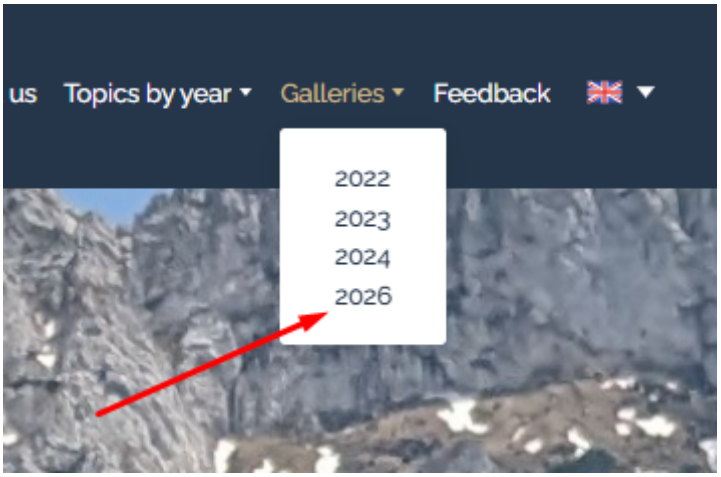


Рисунок Д2.21 - Відображення створеної сторінки галереї в випадаючому меню

Тепер, за звичним сценарієм, додаємо новий елемент на сторінку. Галерея має власну структуру побудови.

Спочатку необхідно додати блок **“Gallery Section”**, у який слід розмістити три елементи **“Gallery Element”**. Кожен із цих елементів може містити довільну кількість зображень - **“Gallery Image”**.

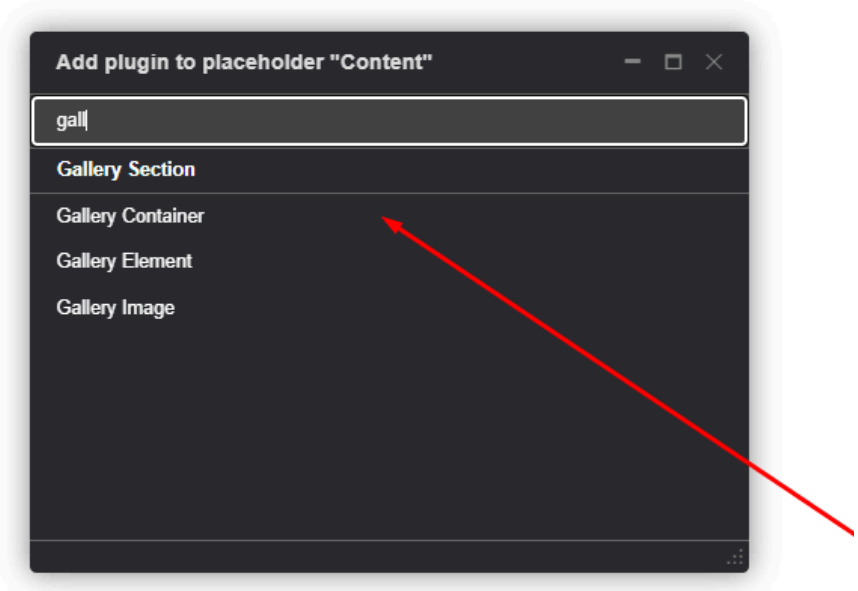


Рисунок Д2.22 - Додавання контейнеру галереї на сторінку

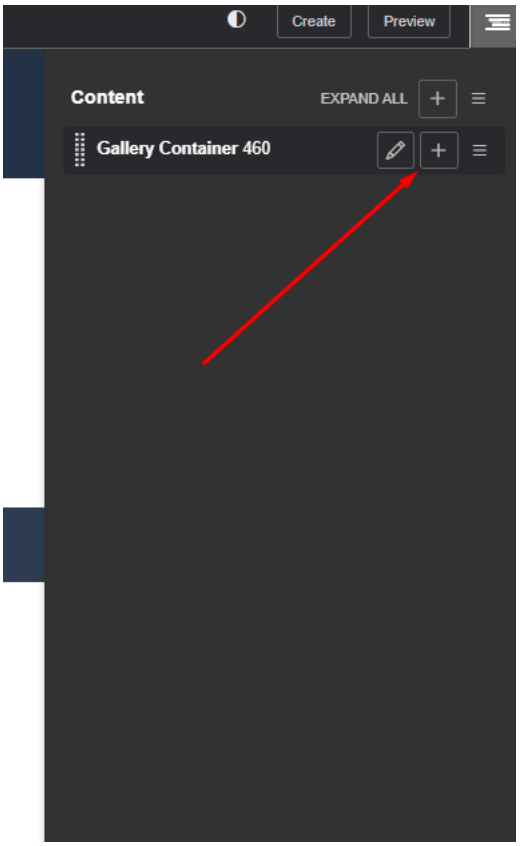


Рисунок Д2.23 - Додавання внутрішнього елементу в контейнер галереї

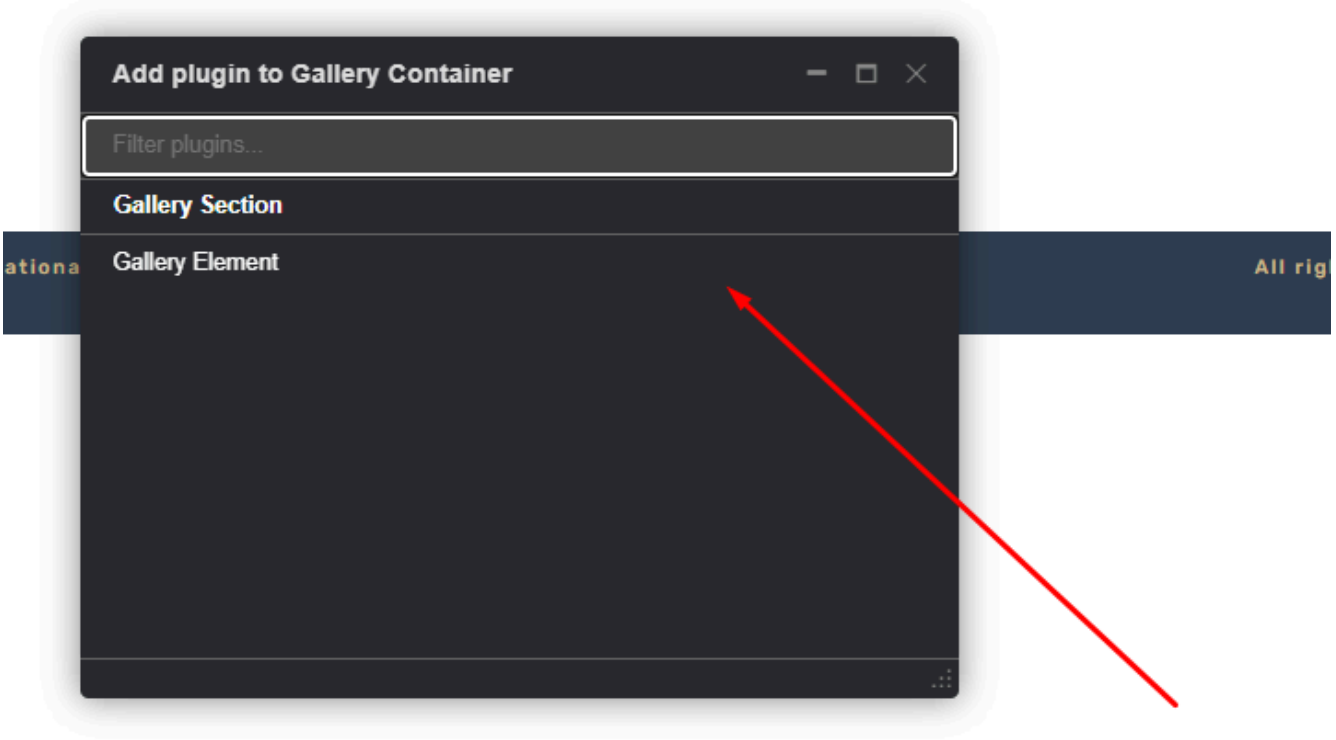


Рисунок Д2.24 - Додавання секції галереї в контейнер

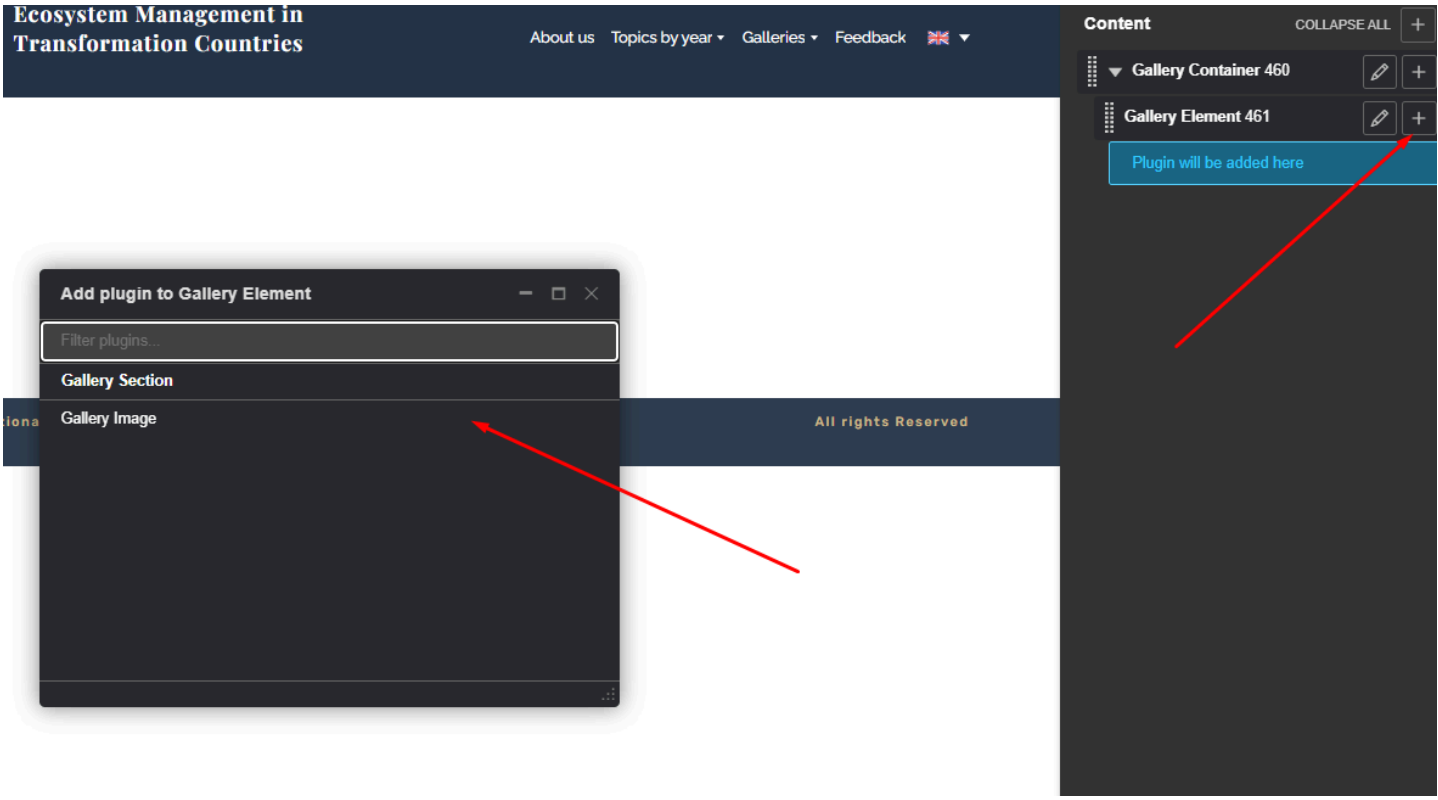


Рисунок Д2.25 - Додавання зображення в елемент галереї

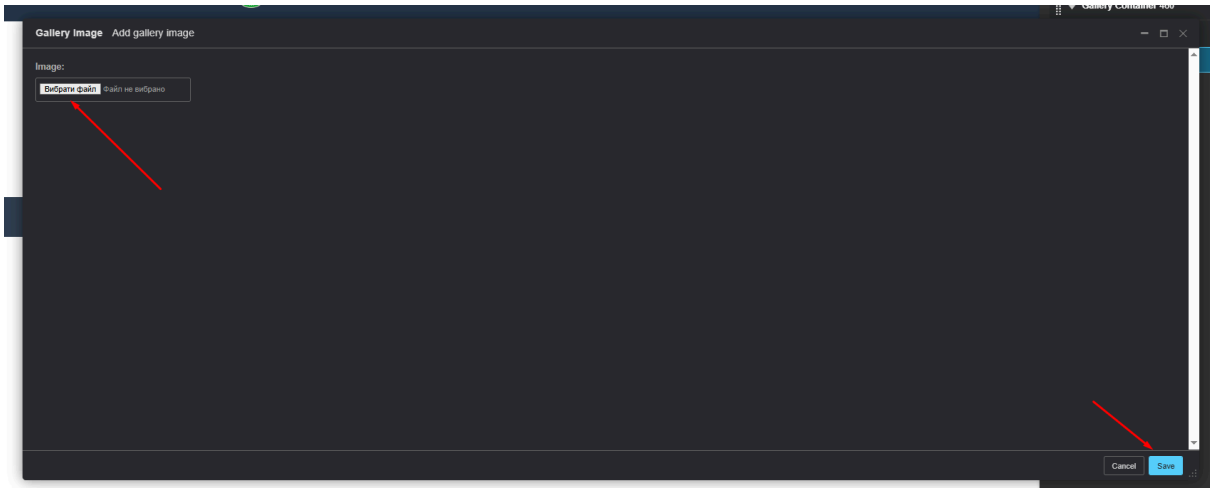


Рисунок Д2.26 - Вибір зображення для відображення на сторінці

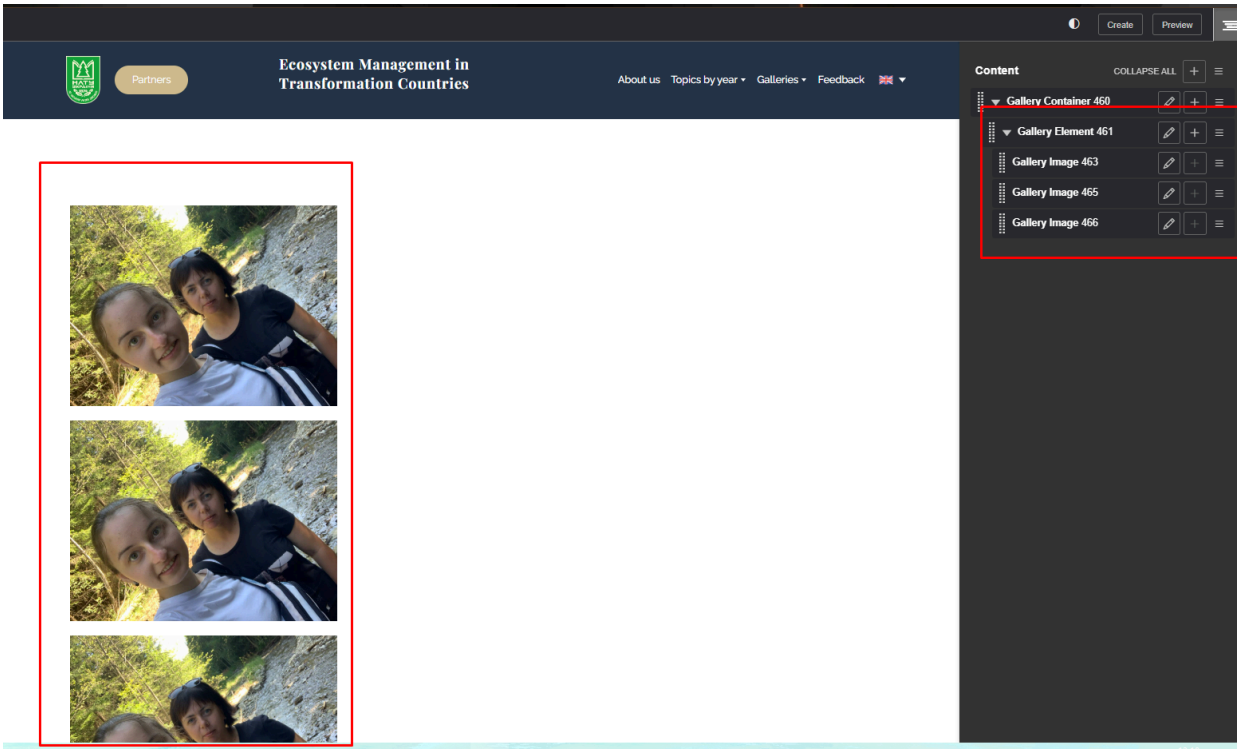


Рисунок Д2.27 - Відображення результату на сторінці

Крім того, функціонал адміністративної панелі передбачає можливість копіювання елементів, що значно спрощує роботу зі сторінками.

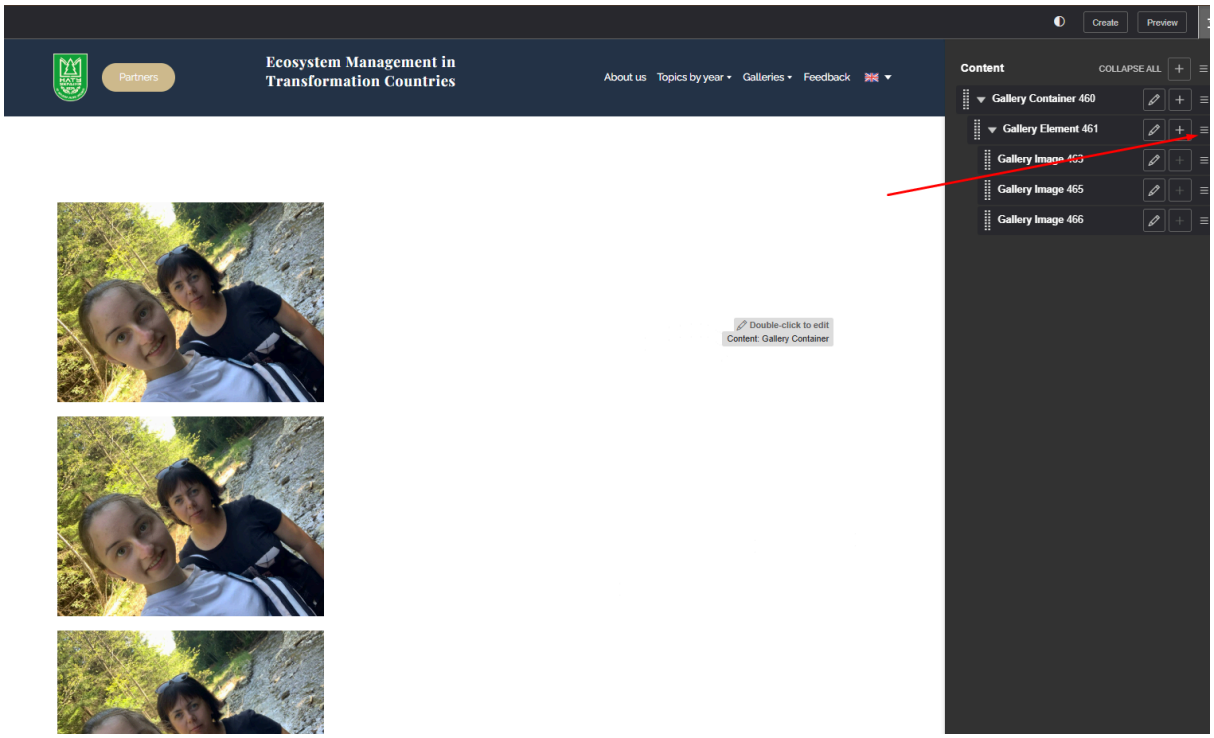


Рисунок Д2.28 - Інструкція з копіювання елементів

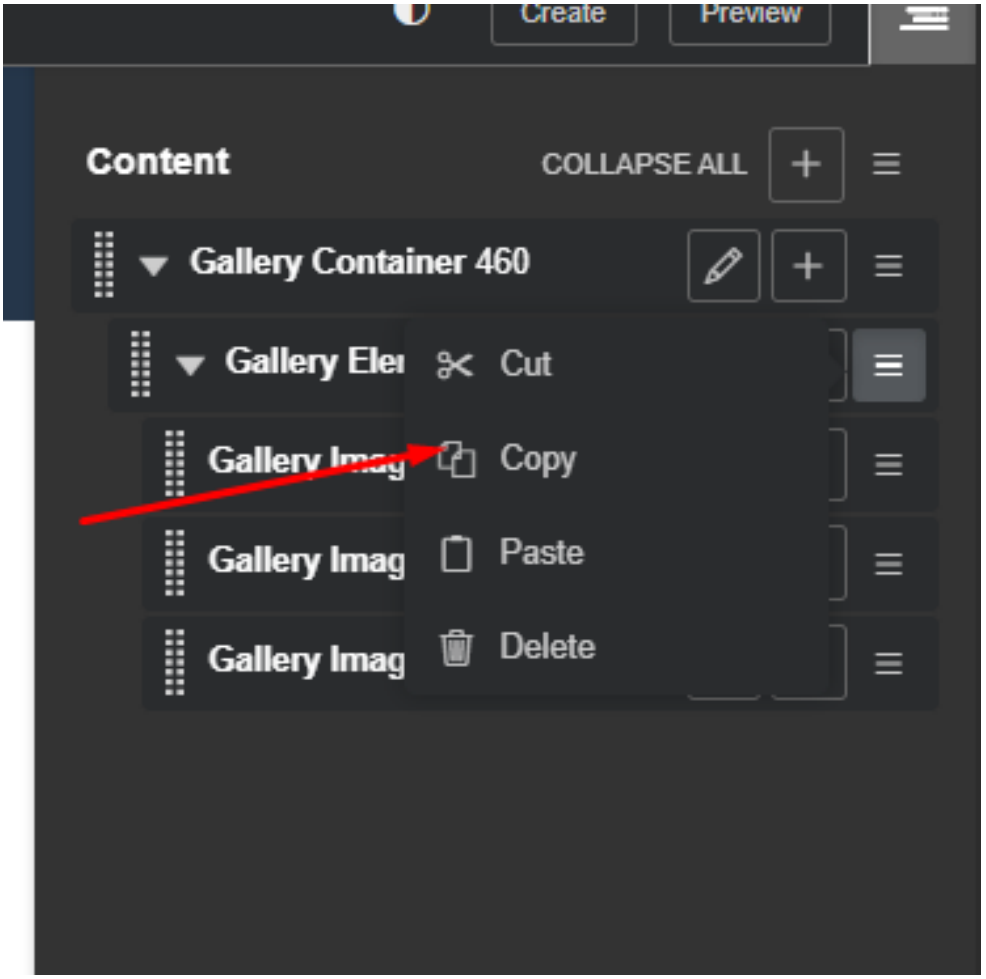


Рисунок Д2.29 - Копіювання елементу

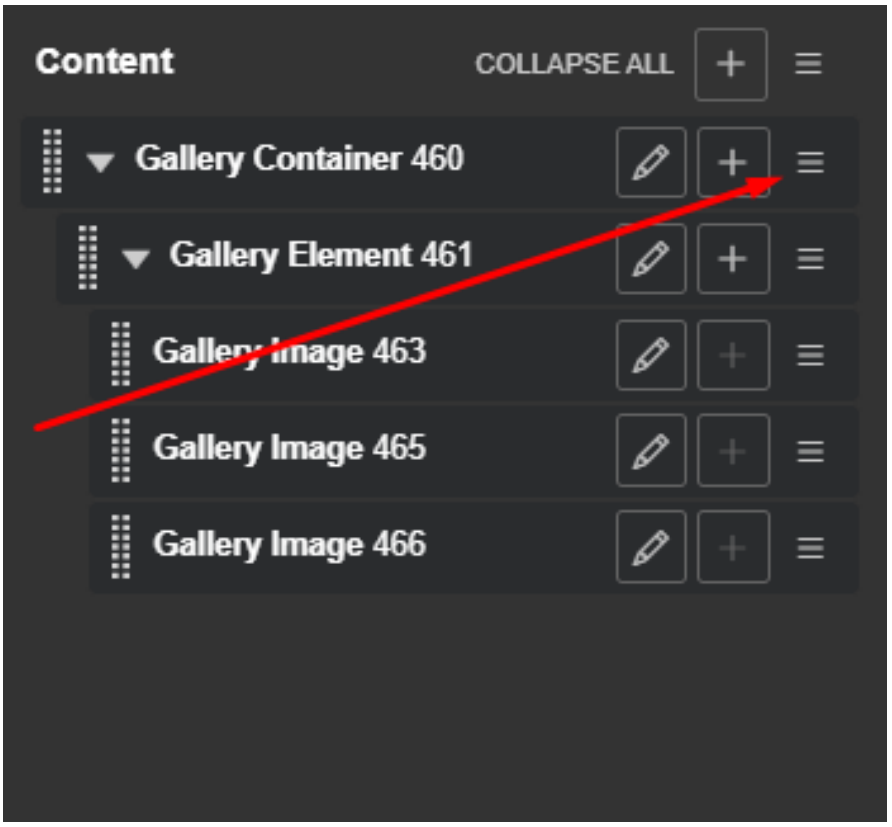


Рисунок Д2.30 - Інструкція з вставлення елементів

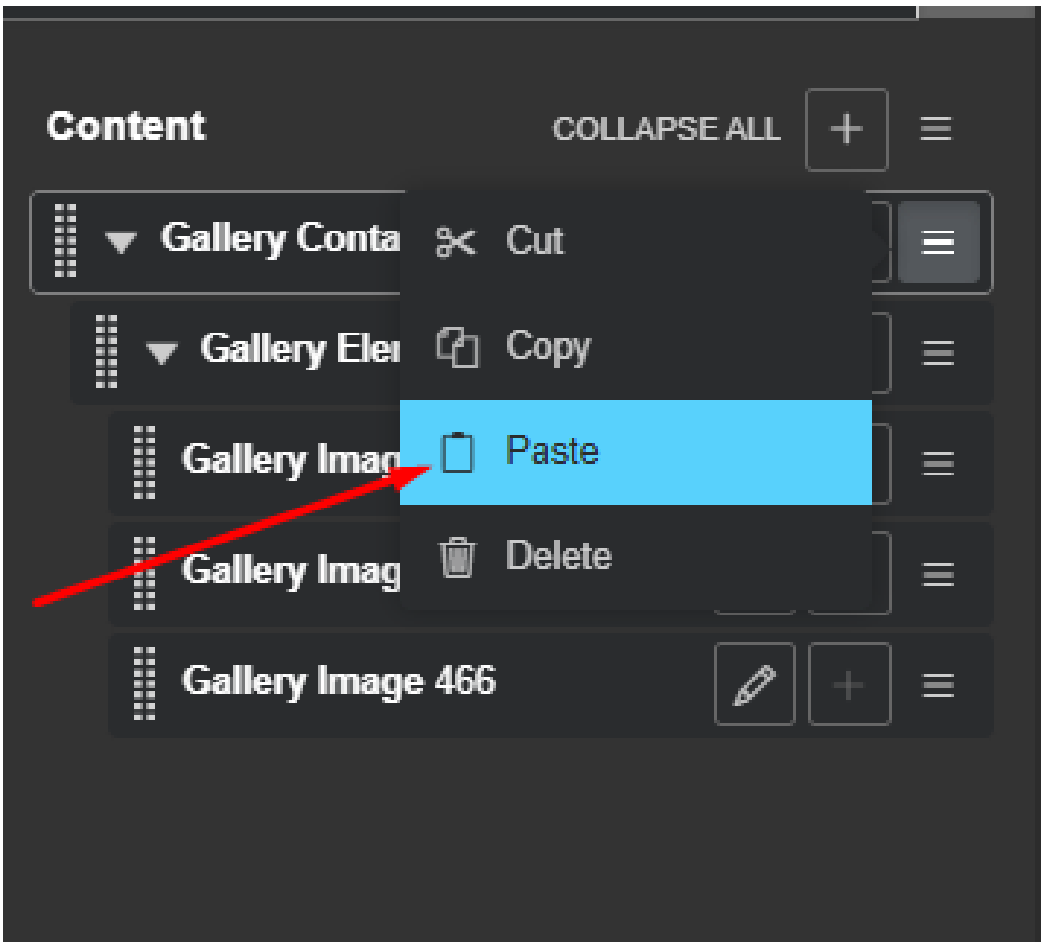


Рисунок Д2.31 - Вставлення елементів

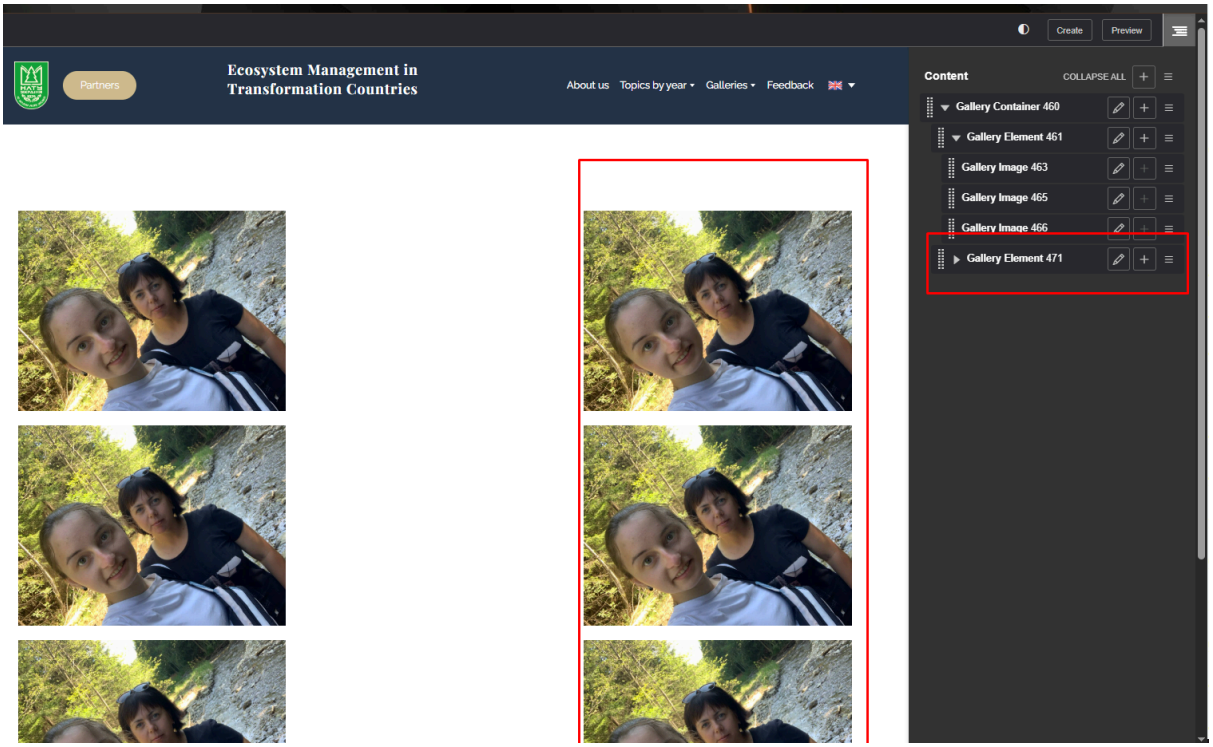


Рисунок Д2.32 - Відображення результату вставлення

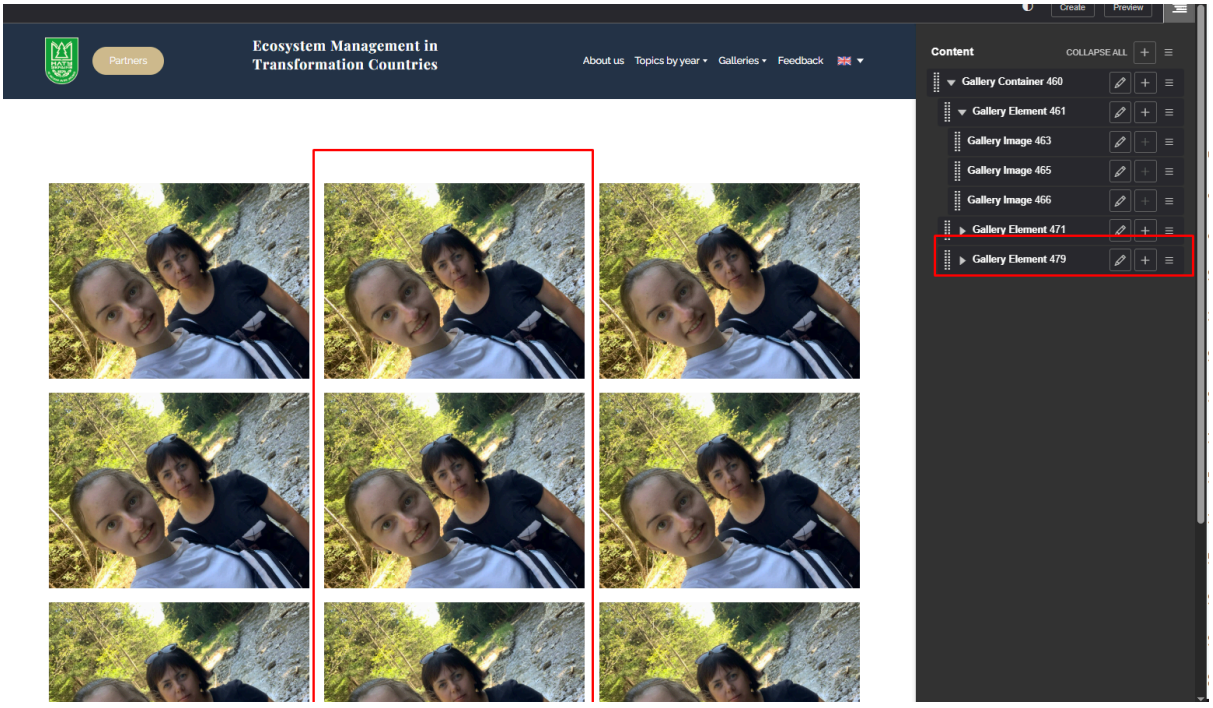


Рисунок Д2.33 - Відображення повної структури контейнера галереї

Додаткові відомості про елементи:

Gallery Section:

- **Gallery Container** - головний блочний елемент на сторінці галерей, який містить по 3 блоки **Gallery Element**.
- **Gallery Element** - блок, що приймає необмежену кількість елементів **Gallery Image**.
- **Gallery Image** - елемент зображення всередині **Gallery Element**.

Generic:

- **Review Carousel** - карусель відгуків, що містить необмежену кількість блоків із зображенням та текстом відгуку.

Text and Picture Section:

- **Text And Picture Section** - блок, який містить елементи **Picture Beside The Text** та **Text Beside The Picture**.
- **Picture Beside The Text** - додає елемент зображення зліва та тексту справа.
- **Text Beside The Picture** - додає елемент зображення справа та тексту зліва.

University Components:

- **Image Carousel** - блок каруселі зображень, який приймає у себе елементи **Carousel Slide**.
- **Carousel Slide** - використовується в **Image Carousel** для створення слайдів із зображеннями.
- **Embeded Video Plugin** - дозволяє додати відео з YouTube на сторінку.
- **Links Collection** - дозволяє додати необмежену кількість текстових елементів з гіперпосиланнями.
- **Picture Section** - дозволяє додати одне зображення.
- **Project Section** - приймає заголовок, один абзац та одне зображення.

- **Text Section** - один із найголовніших елементів, може містити в собі **Text And Picture Section**, **Embeded Video Plugin**, **Links Collection**, **Picture Section**. Дозволяє додавати заголовок та необмежену кількість абзаців.