НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ

І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри

_____комп'ютерних наук_____
(назва кафедри)

_____ / Голуб Б.Л. /
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Програмне забезпечення веб-орієнтованої інформаційної системи управління контентом новинного порталу»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н, доцент _____ Вайганг Г.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

к.ф.-м.н, доцент _____ Кириченко В.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Гладков М.С.
(підпис) (ПІБ студента)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____комп'ютерних наук_____
(назва кафедри)

___к.т.н., доцент_____ _____Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)

"__" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту

Гладков Марк Самеддінович

Спеціальність 121 – «Інженерія програмного забезпечення»

1. Тема бакалаврської кваліфікаційної роботи «Програмне забезпечення веб-орієнтованої інформаційної системи управління контентом новинного порталу» затверджена наказом ректора НУБіП України від 16.12.2024 № 2249 С

2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)

3. Вихідні дані до роботи: поставлене завдання, вимоги до роботи та методичні вказівки.

4. Перелік питань, що розглядаються:

1. Аналіз предметної області;

2. Проєктування програмно-інформаційної системи;

3. Розробка програмного забезпечення;

4. Впровадження та експлуатація системи.

Дата видачі завдання _____

Керівник бакалаврської кваліфікаційної роботи _____ / Кириченко В.В./
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____ / Гладков М.С./
(підпис)

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві оперативне поширення достовірної інформації відіграє надзвичайно важливу роль у житті суспільства. Новинні портали стали основними джерелами інформування громадськості, тому виникає потреба у розробці функціональних систем управління новинним контентом. Актуальність теми полягає у необхідності створення програмного забезпечення, яке б забезпечувало гнучке, безпечне та зручне з точки зору кінцевого користувача керування інформаційним наповненням веб-орієнтованих систем.

Метою цієї роботи є розробка програмного забезпечення для веб-орієнтованої інформаційної системи управління контентом новинного порталу, яке дозволить адміністративним користувачам редагувати, оновлювати та структурувати інформацію без потреби у глибоких знаннях програмування. Такий підхід сприятиме покращенню якості роботи редакцій, підвищенню оперативності публікацій та загальному вдосконаленню функціонування інформаційних ресурсів.

Для реалізації поставленої мети використовувалися сучасні методи та технології розробки веб-додатків. Зокрема, для створення серверної частини застосовано мову програмування PHP та фреймворк Laravel, що забезпечує зручну архітектуру і високу масштабованість. У якості засобу управління базою даних обрано реляційну MySQL, а для графічного інтерфейсу користувача – фронтенду використано HTML, CSS та JavaScript з адміністративну бібліотекою Bootstrap5, що дозволяє створити адаптивний та інтуїтивно зрозумілий інтерфейс. Для моделювання системи були використані діаграми UML, які дозволяють чітко окреслити структуру та логіку роботи майбутнього програмного продукту.

Програмний додаток було протестовано в межах навчального процесу. Основні результати доповідались під час засідань наукового гуртка та

представлено у вигляді тез доповіді на студентській конференції. Отримані результати засвідчили практичну доцільність і функціональну повноту запропонованого рішення.

Основна частина записка містить 73 сторінок основного тексту, 35 рисунків, 5 таблиці, 20 використаних джерел, а також 4 додатки. У першому розділі здійснено аналіз предметної області, сформульовано вимоги до системи, змодельовано предметну область та виконано аналіз існуючих рішень. Наступний розділ присвячений побудові логічної структури системи засобами об’єктно-орієнтованого моделювання. У третьому розділі описано вибір інструментів розробки, реалізацію бази даних та створення прикладного програмного забезпечення. Четвертий розділ містить опис процесу впровадження, тестування та рекомендації щодо розгортання системи. У висновках наведено підсумки виконаної роботи та перспективи подальшого вдосконалення розробленого програмного продукту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області та огляд бізнес-вимог

Предметна область цього дослідження охоплює діяльність новинної редакції, яка функціонує як частина сучасного медіа-видавництва. Основною метою такої редакції є створення, редагування, перевірка, публікація та просування новинного контенту в цифровому середовищі, зокрема на веб-порталах. У межах редакції взаємодіють різні учасники, серед яких: автори матеріалів, редактори, менеджери з реклами, аналітики, головний редактор, а також директор компанії. Кожен із цих учасників виконує конкретні функції, які забезпечують функціонування всієї системи.

Процес створення новинного матеріалу зазвичай починається з призначення завдання редактором. Автор матеріалів отримує завдання, проводить збір інформації, перевіряє факти, створює текстовий контент. Далі матеріал передається редактору для перевірки на відповідність стандартам і вимогам, після чого відбувається погодження із головним редактором, який відповідає за загальну якість і політику контенту видання. Після затвердження матеріалу він може бути опублікований на новинному порталі.

Публікація матеріалу також пов'язана з процесами просування, що передбачає взаємодію з менеджером з реклами. Менеджер з реклами отримує затверджений контент, розміщує рекламу поряд із матеріалом, залучає рекламодавців, оцінює ефективність рекламних кампаній та формує відповідні звіти. Ці звіти аналізуються аналітиком, який також займається формуванням статистичних звітів, прогнозуванням читабельності матеріалів, аналізом активності аудиторії та оцінкою ефективності контенту загалом.

Результати аналітики передаються директору компанії, який ухвалює стратегічні рішення. До обов’язків директора входить визначення загальної стратегії розвитку, ухвалення фінансових та кадрових рішень, а також координація співпраці з партнерами та інвесторами.

Ключовою особливістю предметної області є тісна взаємодія всіх учасників процесу на різних етапах життєвого циклу новини — від ідеї до публікації та її подальшого аналітичного опрацювання. Відповідно до функціональної моделі, яка представлена на діаграмі, всі ролі виконують свої функції у межах єдиної інформаційної системи. Така система має забезпечити не лише створення та збереження контенту, а й контроль за його якістю, оперативність оновлення, адаптацію до потреб аудиторії, а також ефективну взаємодію з рекламодавцями.

Ще однією важливою складовою предметної області є взаємодія з читачами, які не лише споживають контент, але й можуть брати участь в опитуваннях, переглядати, оформлювати підписки на розсилку по пошті. Це створює зворотний зв'язок, який враховується під час формування контент-стратегії та покращення редакційної політики.

Якщо розглянути предметну область що безпосередньо відносить до програмних рішень, то у сучасних умовах новинний портал повинен функціонувати у режимі реального часу, тому важливою є підтримка постійного оновлення інформації та безперервного публікування матеріалів. Система повинна не тільки зменшити часові витрати на обробку новин, а й створити прозору, контрольовану та ефективну редакційну структуру.

Серед пріоритетних бізнес-вимог можна виділити наступні.

- Підвищення продуктивності редакційної команди. Завдяки автоматизації операцій зі створення, редагування та публікації контенту працівники зможуть працювати швидше, що особливо важливо для новинних ресурсів із високою частотою оновлень;
- Покращення якості контенту. Передбачено використання інструментів перевірки правопису, стилістичних підказок та вбудованого редактора, що забезпечує контроль за граматикою і змістом тексту. Це сприятиме формуванню авторитетного іміджу порталу;

- Оптимізація робочого процесу. Завдяки єдиному веб-інтерфейсу, в якому працюють усі ролі (редактори, автори, адміністратори, контент-менеджери), покращується комунікація та спрощується взаємодія між учасниками процесу. Усі дії в системі будуть відслідковувані та прозорі;
- Безперервне публікування. Автоматизація розкладу публікацій дозволяє уникати пауз в оновленні новинного потоку, забезпечуючи сталу присутність ресурсу в інформаційному просторі;
- Підтримка монетизації. Інтеграція рекламного функціоналу дозволить розміщувати банери, вставляти рекламні блоки та контролювати їх;
- SEO-просування. Наявність засобів для оптимізації контенту під пошукові системи (мета-теги, URL-структура, семантика) сприятиме підвищенню органічного трафіку і розширенню аудиторії.

Таким чином, предметна область охоплює складну систему взаємозалежних процесів, які відбуваються в межах новинного медіа-видавництва. Функціонування такої системи потребує автоматизації ключових процесів, прозорого розподілу обов’язків, централізованого зберігання інформації та можливості її швидкої обробки й аналізу. Усі ці потреби і обґрунтовують актуальність розробки програмного забезпечення, яке стане технологічною основою для підтримки основних функцій редакційної діяльності в цифровому просторі.

1.2 Моделювання предметної області

Для кращого розуміння структури та процесів, які відбуваються в межах новинного медіа-видавництва, застосовано моделювання предметної області за допомогою нотації UML (Unified Modeling Language). UML — це уніфікована мова моделювання, яка дозволяє візуалізувати, описувати та документувати програмні системи та бізнес-процеси. Вона є стандартом де-факто в галузі розробки програмного забезпечення.

У цьому розділі використано такі типи UML-діаграм.

- Діаграма варіантів використання (Use Case Diagram) — дозволяє відобразити основних учасників системи та їхню взаємодію з функціональністю системи;
- Діаграма послідовності (Sequence Diagram) — показує порядок виконання дій та обмін повідомленнями між об’єктами системи в часі;
- Діаграма діяльності (Activity Diagram) — ілюструє логіку виконання процесів та основні етапи робочого потоку.

Ці діаграми дають змогу формалізувати вимоги до системи, виявити ключові взаємозв'язки між учасниками процесу, визначити критичні точки взаємодії, а також закласти основу для подальшого проєктування архітектури програмного забезпечення. Візуалізація також полегшує комунікацію між розробниками та замовниками, оскільки надає узгоджену та зрозумілу схему функціонування системи.

Далі наведено приклади кожного з типів діаграм, що відображають реальні процеси новинного порталу: створення матеріалу, його погодження, публікацію та просування.

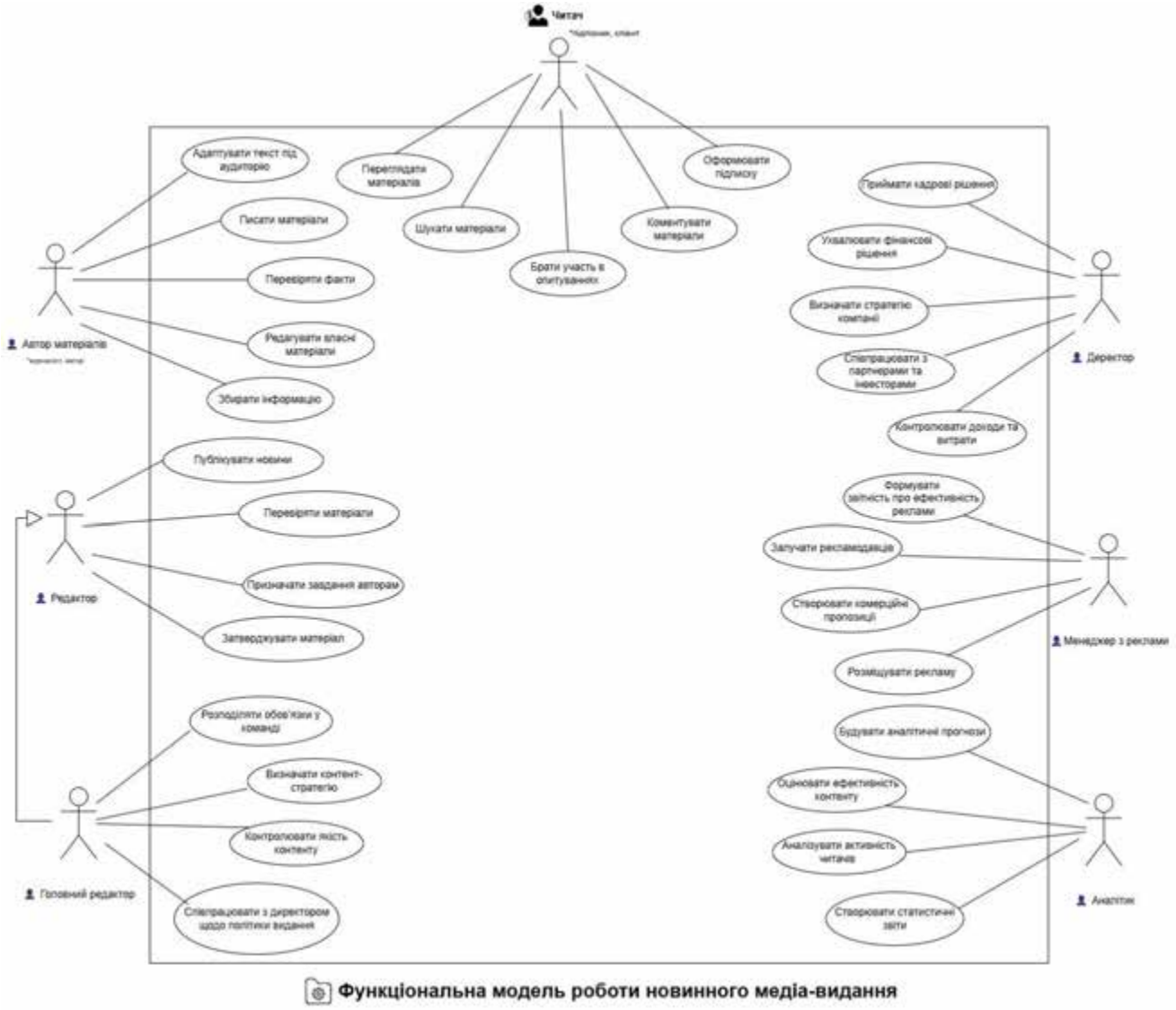

Рис. 1 – Діаграма варіантів використання

Діаграма варіантів використання (див. рис 1).

На діаграмі зображено функціональну модель роботи новинного медіа-видавництва. В центрі схеми розташовані основні функції системи, які згруповані за відповідними ролями користувачів. Серед ключових акторів — Автор матеріалів, Редактор, Головний редактор, Менеджер з реклами, Аналітик, Директор, а також Читач як кінцевий користувач порталу.

Автор займається написанням, редагуванням і адаптацією матеріалів, а також перевіркою фактів. Редактор призначає завдання, перевіряє і затверджує

матеріали. Головний редактор визначає контент-стратегію, координує команду і співпрацює з директором. Менеджер з реклами відповідає за комерційну складову — від пошуку рекламодавців до оцінки ефективності реклами. Аналітик аналізує активність читачів, будує прогнози та формує звіти. Директор ухвалює стратегічні, фінансові та кадрові рішення. Читачі мають змогу переглядати, шукати, коментувати матеріали, брати участь в опитуваннях і оформлювати підписку. Діаграма ілюструє повний функціональний контур редакційної роботи та взаємодії з аудиторією.

Діаграма послідовності (див. рис. 2).

На діаграмі послідовності зображено логіку роботи над новинним матеріалом від його створення до публікації. Послідовність починається з того, що Редактор призначає завдання Автору матеріалів, який далі збирає інформацію, перевіряє факти та створює текст. Готовий матеріал повертається редактору, який його перевіряє і, в разі позитивної оцінки, надсилає Головному редактору для погодження.

Після затвердження редакційною командою, матеріал передається Менеджеру з реклами, який готує рекламу для публікації разом із новиною. Далі менеджер повідомляє про завершення дій з просування. Паралельно в процесі беруть участь Аналітик і Директор, які отримують звіти, формують рішення та визначають стратегію. У фіналі матеріал публікується. Діаграма демонструє послідовність дій і передачу відповідальності між учасниками редакційного процесу.

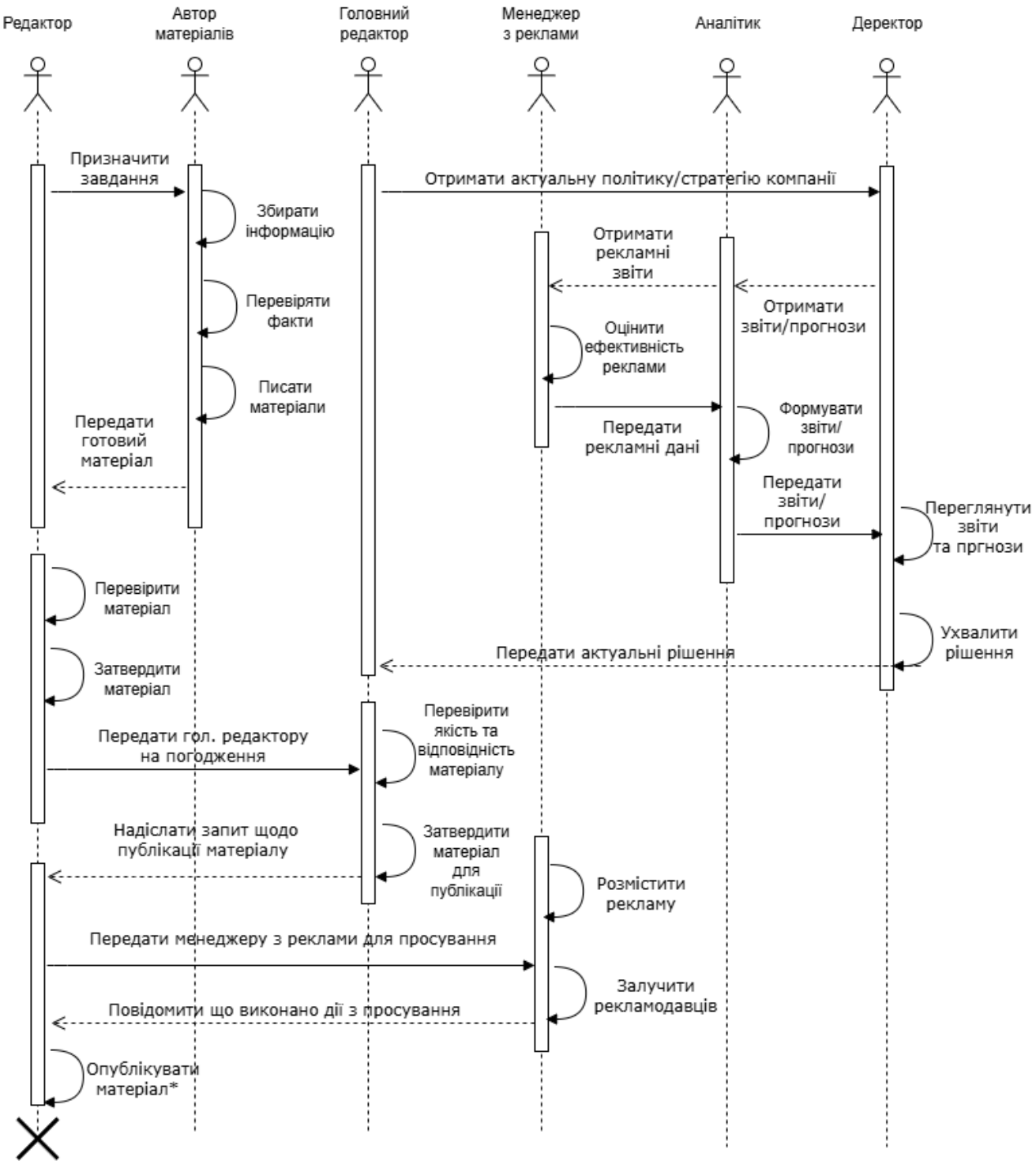

Рис. 2 – Діаграма послідовності

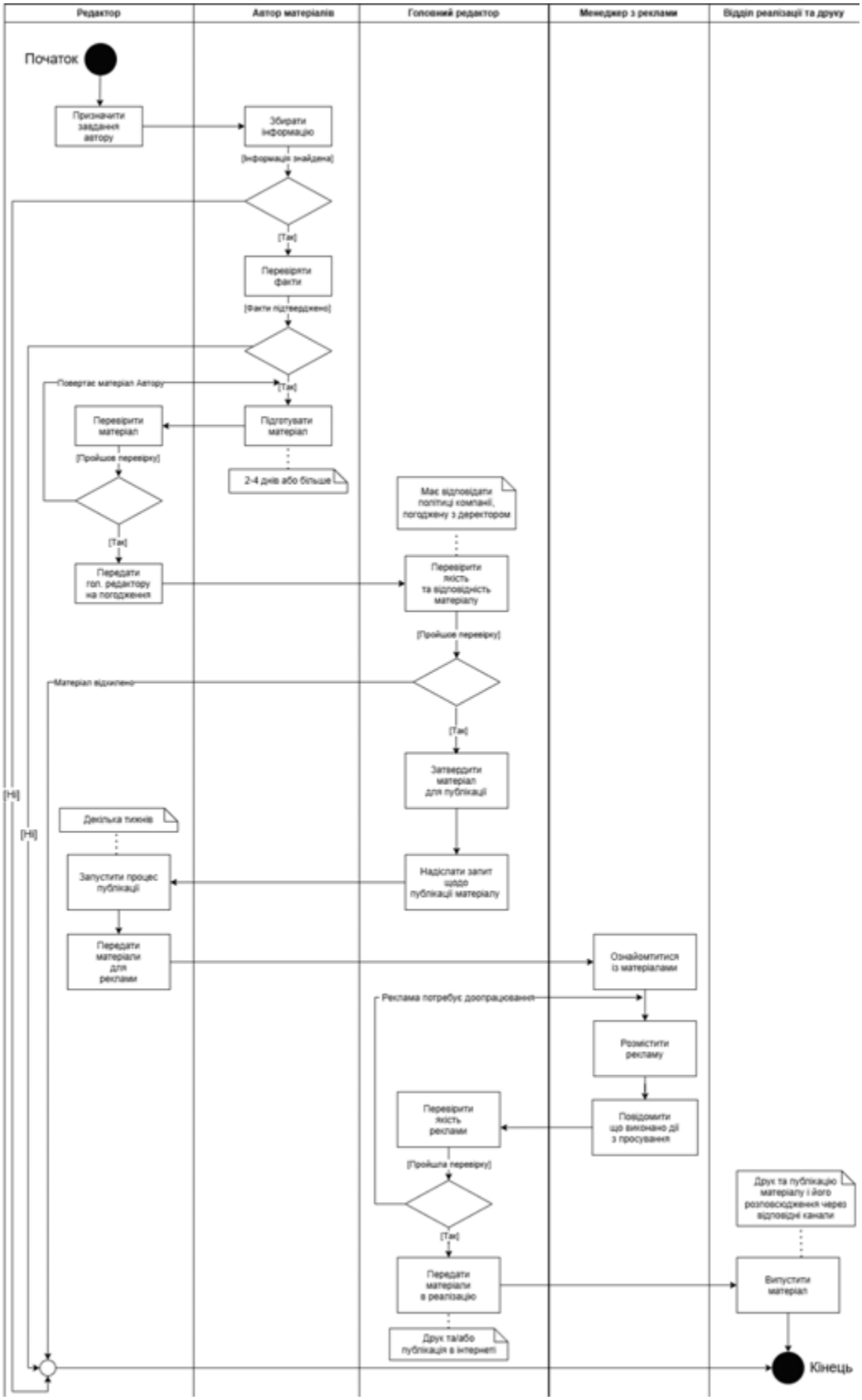

Рис. 3 – Діаграма активності (діальності)

На діаграма діяльності (див. рис. 3) показано ключові бізнес-процеси, що реалізуються в межах роботи новинної редакції. В центрі уваги — створення та

публікація контенту, аналітика, керування рекламною діяльністю та стратегічне управління.

Процеси автора охоплюють написання, редагування і перевірку матеріалів. Редактор керує завданнями, переглядає контент і затверджує його. Головний редактор визначає стратегію, розподіляє обов'язки в команді та координує з директором. Менеджер з реклами відповідає за взаємодію з рекламодавцями, аналітик — за аналіз аудиторії, ефективності контенту та створення звітів. Директор ухвалює управлінські рішення та контролює фінанси. Читач виконує дії з боку користувача — переглядає, коментує, підписується, бере участь в опитуваннях. Таким чином, діаграма дає уявлення про загальну структуру дій усіх учасників системи та логіку їх взаємодії.

Далі буде представлено абстракції предметної області, що відображають ключові ролі користувачів та їхні функції в межах редакційного процесу новинного порталу (див. рис. 4.1-4.4).

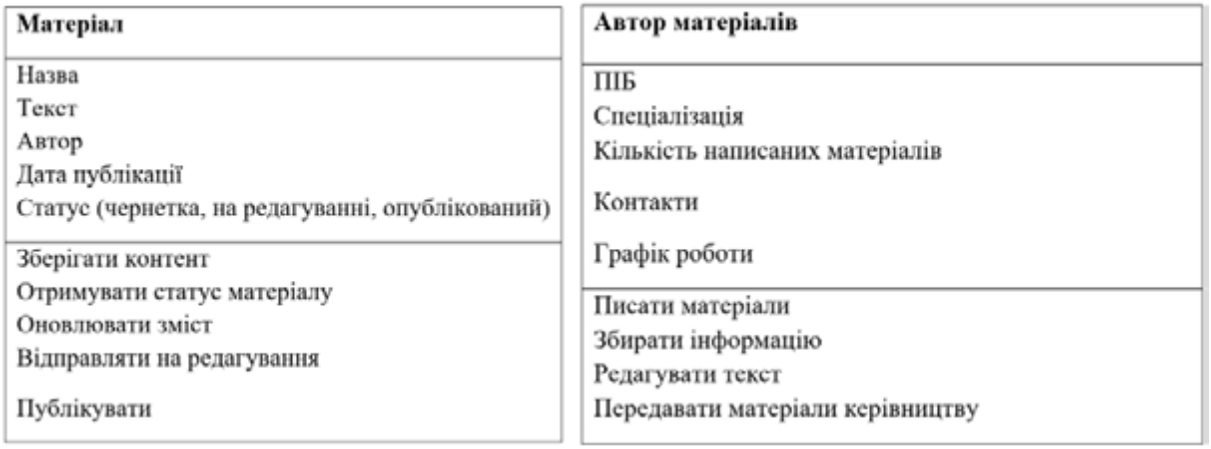

Рис. 4.1 – Абстракції предметної області

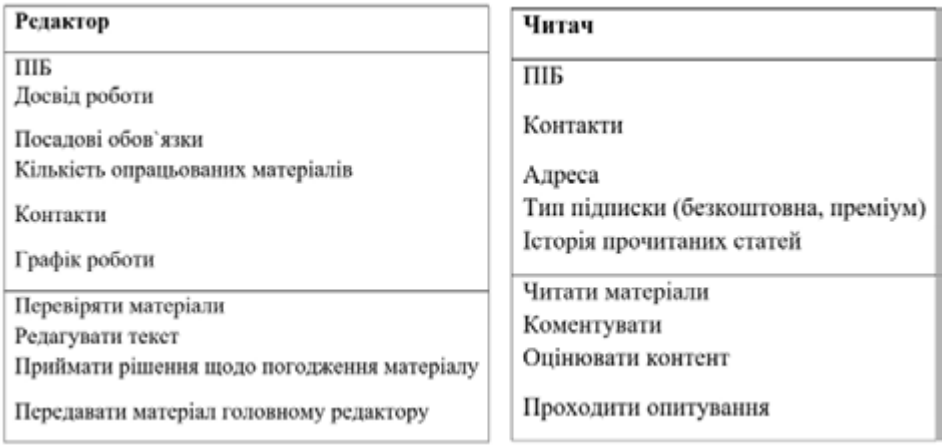

Рис. 4.2 – Абстракції предметної області

Головний редактор
ПІБ
Досвід роботи
Контакти
Графік роботи
Посадові обов`язки
Затверджувати матеріали
Координувати роботу редакторів
Співпрацювати з директором
Контролювати якість
Передавати матеріалів в реалізацію

Аналітик
ПІБ
Спеціалізація
Досвід у дослідженні ринку
Контакти
Графік роботи
Посадові обов`язки
Аналізувати популярність тем
Оцінювати ефективність контенту
Надавати рекомендації щодо заголовків і подачі матеріалів
Готувати статистичні звіти

Рис. 4.3 – Абстракції предметної області

Директор
ПІБ
Досвід управління
Відповідальність за стратегію компанії
Контакти
Графік роботи
Посадові обов`язки
Приймати кадрові рішення
Затверджувати фінансові рішення
Контролювати доходи та витрати
Співпрацювати з партнерами та інвесторами

Менеджер з реклами
ПІБ
Досвід роботи в рекламі
Відповідальність за комерційні пропозиції
Контакти
Графік роботи
Посадові обов`язки
Запускати рекламні кампанії
Створювати комерційні пропозиції
Розміщувати рекламу
Будувати рекламні прогнози

Рис. 4.4 – Абстракції предметної області

1.3 Огляд інформаційних джерел та існуючих рішень

У сфері новинних порталів та інформаційних ресурсів вибір системи управління контентом (CMS) є ключовим для забезпечення ефективної роботи редакційної команди, швидкої публікації матеріалів та зручної взаємодії з аудиторією. Нижче наведено огляд чотирьох популярних CMS потенційних конкурентів, які активно використовуються для створення та підтримки новинних сайтів тощо.

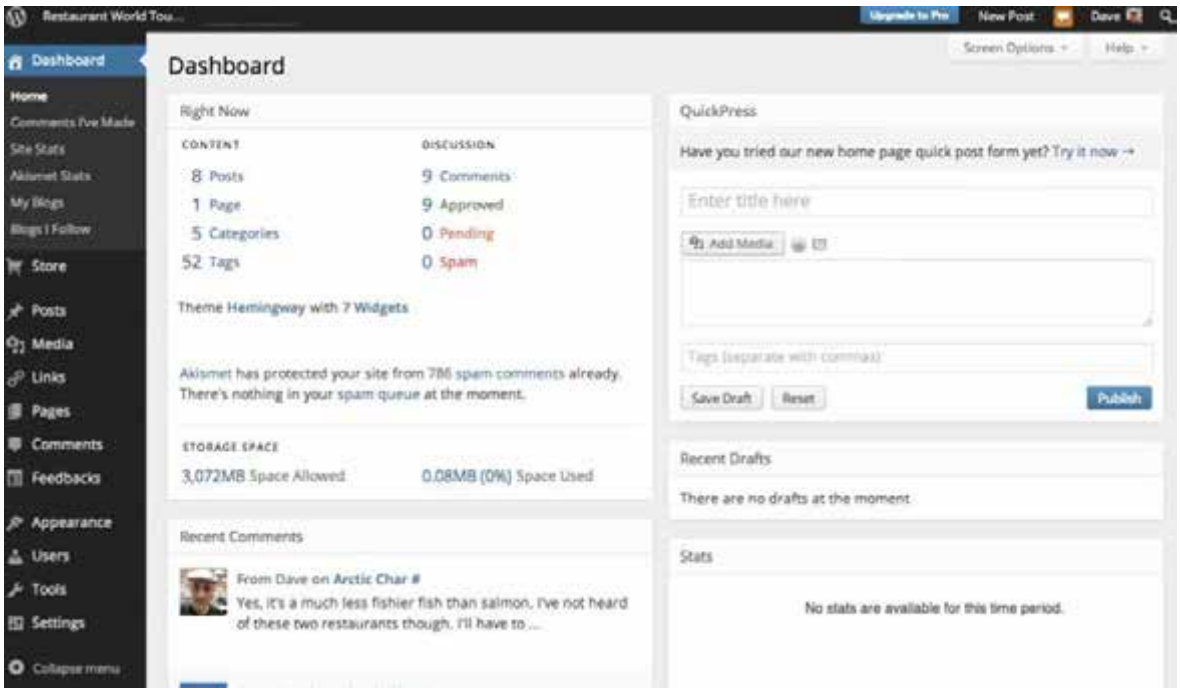

Рис. 5.1 – Приклад адмін панелі WordPress

1. WordPress є однією з найпоширеніших CMS у світі, що забезпечує гнучкість та розширюваність завдяки великій кількості плагінів і тем. Вона підходить як для невеликих блогів, так і для великих новинних порталів. WordPress підтримує багатокористувацький режим, що дозволяє організувати роботу редакційної команди з різними рівнями доступу. [джерело: https://wordpress.org]

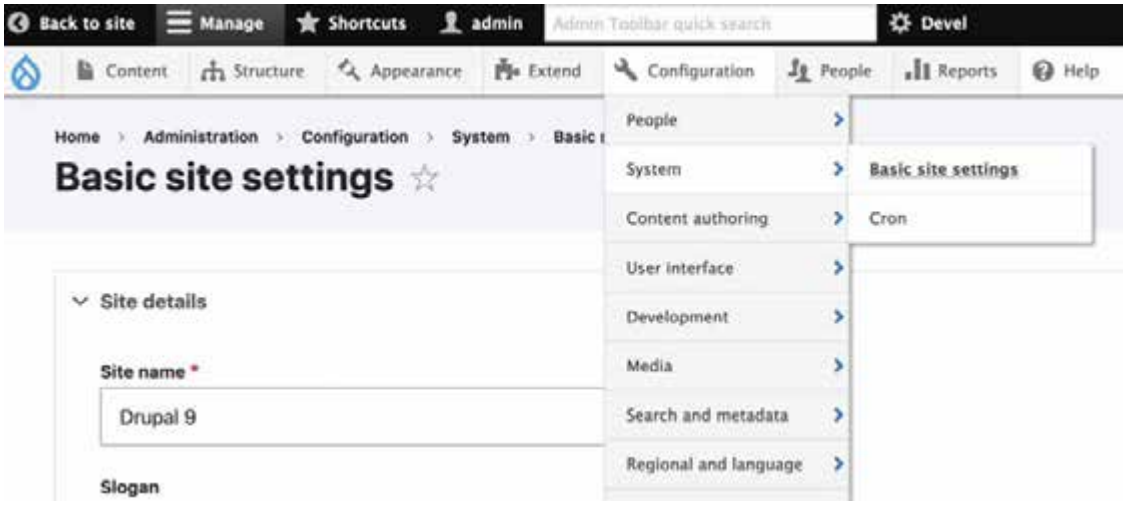

Рис. 5.2 – Панель керування Drupal

2. Drupal — потужна CMS, яка відзначається високою гнучкістю та масштабованістю. Вона особливо підходить для великих новинних організацій з комплексною структурою контенту. Drupal підтримує багатомовність, складні робочі процеси та інтеграцію з різноманітними сервісами, що робить її привабливою для медіа-компаній. [джерело: https://www.drupal.org]

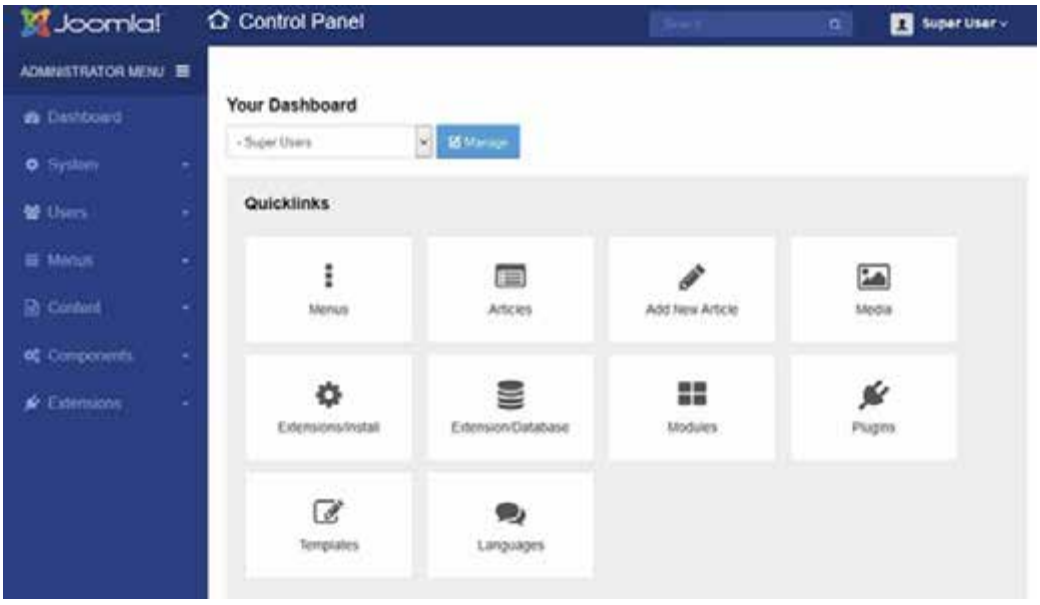

Рис. 5.3 – Панель керування Joomla

3. Joomla є ще однією популярною CMS з відкритим кодом, яка поєднує в собі простоту використання та потужні функціональні можливості. Вона підтримує багатомовність, має вбудовану систему управління користувачами та

дозволяє легко налаштовувати структуру сайту. Joomla підходить для середніх за розміром новинних порталів. [джерело: https://www.joomla.org]

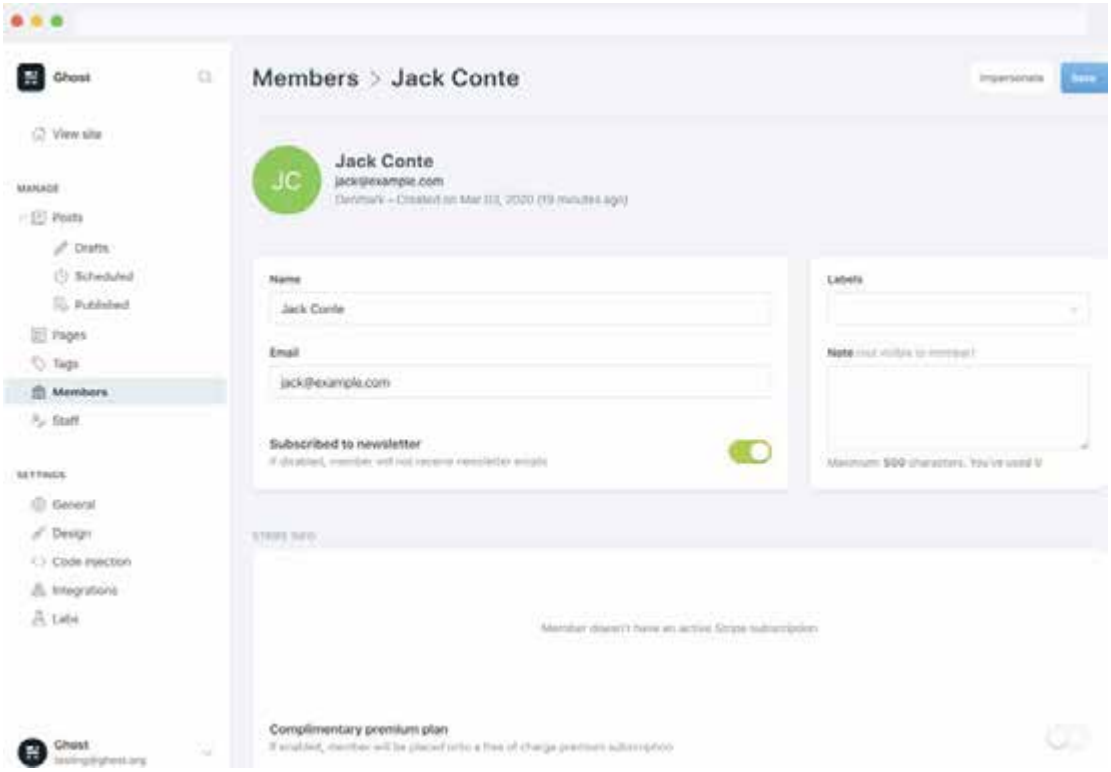

Рис. 5.4 – Лого Ghost

4. Ghost — сучасна CMS, орієнтована на створення блогів та новинних сайтів. Вона відзначається швидкістю роботи, чистим інтерфейсом та зосередженістю на контенті. Ghost підтримує SEO-оптимізацію, інтеграцію з соціальними мережами та має вбудовані інструменти для монетизації контенту. [джерело: https://ghost.org]

Далі буде наведена порівняльна таблиця систем управління контентом (CMS) для новинних порталів. У таблиці використано умовні оцінки рівня (високий, середній, низький) з конкретними поясненнями в дужках для кожного критерію.

Порівняльна таблиця існуючих CMS рішень

Таблиця 1

Критерій	**WordPress**	**Drupal**	**Joomla**	**Ghost**
Простота використання	Висока (інтуїтивний інтерфейс, легке налаштування)	Середня (потребує базових технічних знань)	Середня (потребує ознайомлення з термінологією)	Висока (мінімалістичний дизайн, фокус на контенті)
Масштабованість	Висока (підтримка великих сайтів, багато плагінів)	Висока (підходить для складних структур)	Середня (добре для середніх порталів)	Середня (оптимізована для блогів та новин)
Багатомовність	Висока (підтримка багатьох мов через плагіни)	Висока (вбудована багатомовність)	Висока (вбудована багатомовність)	Низька (обмежена підтримка мов)
SEO-оптимізація	Висока (велика кількість SEO-плагінів)	Висока (гнучкі налаштування SEO)	Середня (менше плагінів для SEO)	Висока (вбудовані SEO-функції)
Розширюваність	Висока (тисячі плагінів і тем)	Висока (велика кількість модулів)	Висока (багато розширень)	Середня (обмежена кількість розширень)
Вартість	Безкоштовна (відкритий код)	Безкоштовна (відкритий код)	Безкоштовна (відкритий код)	Платна (підписка на хостинг)

Сучасні CMS для новинних порталів забезпечують швидке наповнення контентом, зручне управління користувачами, підтримку мультимедіа, SEO-оптимізацію та інтеграцію з соціальними мережами. Найбільш універсальним рішенням є WordPress, тоді як Drupal підходить для складних структур, Joomla — для середніх проєктів, а Ghost — для мінімалістичних новинних сайтів.

1.4 Постановка завдання

Метою створення системи є розробка програмного забезпечення та інформаційної системи, що забезпечить управління новинним контентом. Система повинна дозволяти публікацію, редагування та просування інформаційних матеріалів із залученням учасників редакційного процесу.

Функціональні вимоги до системи.

- можливість створення, редагування, категоризації та публікації новинних матеріалів;
- додавання супровідного контенту (розділи, фото, відео);
- призначення тегів сутностям;
- підтримку SEO-модулю для пошукових систем та соц. мереж;
- підтримка інтерфейсом мультмовності i18n (українська та англійська мови);
- реалізація розділення відповідальності між користувачами, за прикладом матриці-доступу, до того чи іншого програмного модулю;
- конструктор створення форм для збору зворотного зв'язку;
- використання сервісів АПІ-перекладачів для перекладу текстів;
- наявність функціоналу автоматичної генерації карти сайту (XML sitemap).

Нефункціональні вимоги.

- можливість масштабування для зростання кількості користувачів і обсягу контенту;

- адаптивний та інтуїтивно зрозумілий інтерфейс;
- підтримка безпечної авторизації (зокрема використання IP-restricted flow);
- користувач повинен мати можливість навігувати сайтом використовуючи комп`ютерну мишу або сенсорний ввід смартфону (планшету).

Розподіл користувачів у системі.

У системі передбачено кілька типів користувачів, кожен із яких має свій набір функцій і рівень доступу:

- Контент-менеджер — створює контент, прикріплює мультимедійні елементи, наповнює розділи сайту, керує програмними модулями та формами;
- Редактор — працює із новинними матеріалами та тегами;
- Головний редактор — затверджує остаточну версію публікації, має права на рівні адміністратору за винятком конфігураційних вкладок;
- Адміністратор порталу — налаштовує технічну інфраструктуру, створює облікові записи, керує конфігураціями сайту, ключами та доступами тощо.

У результаті постановки завдання визначено ключові вимоги до функціональності, структури та користувацької взаємодії в системі. Чіткий розподіл ролей, сучасні інструменти управління контентом та нефункціональні стандарти дадуть змогу створити ефективну систему, здатну задовольнити потреби редакції новинного порталу та забезпечити взаємодію з аудиторією.

2 ПРОЄКТУВАННЯ ПРОГРАМНО-ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Логічна модель даних у вигляді ER-діаграми

На діаграмі (рис. 6) зображено основні сутності, які використовується для організації та збереження контенту в інформаційній системі новинного порталу.

Сутність «Новина» містить такі атрибути, як заголовок, короткий опис, основний текст, головне зображення, дата і час публікації, а також SEO-дані, теги й статус активності. Вона є центральною частиною системи, навколо якої будуються інші об’єкти.

«Сторінка» (розділ сайту) визначає структуру сайту — містить назву, пріоритет, батьківський розділ, шаблон та параметри відображення. Зі сторінкою пов’язані модулі сторінки, які дозволяють виводити окремі блоки з інформацією (наприклад, новини або відео).

Сутність «Відеоматеріал» включає заголовок, посилання на відео, дату публікації, прев’ю зображення та теги. Аналогічно, «Фотогалерея» об’єднує набір зображень у логічну групу. Самі зображення зберігаються в «Фотобанку», який містить шлях до файлу, параметри зображення (ширина, висота), ALT-опис, джерело тощо.

Окремо представлено сутність «Форма», яка використовується для збору інформації від користувачів (наприклад, коментарі, підписка, зворотний зв’язок). Форма містить поля і збережені дані.

Система тегування реалізована через сутність «Тег», яка дає змогу категоризувати контент. Кожна новина, відео чи галерея може містити кілька тегів для зручного пошуку та навігації.

Сутність «Модуль метаданих SEO» відповідає за оптимізацію кожної сторінки під пошукові системи, включаючи теги <title>, <meta description>, ключові слова, заголовки та додаткові параметри для пошукових роботів.

Загалом діаграма демонструє логічно структуровану та масштабовану модель, яка дозволяє керувати як контентом, так і його поданням та пошуковою

оптимізацією. Така побудова забезпечує гнучкість системи і зручність у подальшій реалізації функціоналу.

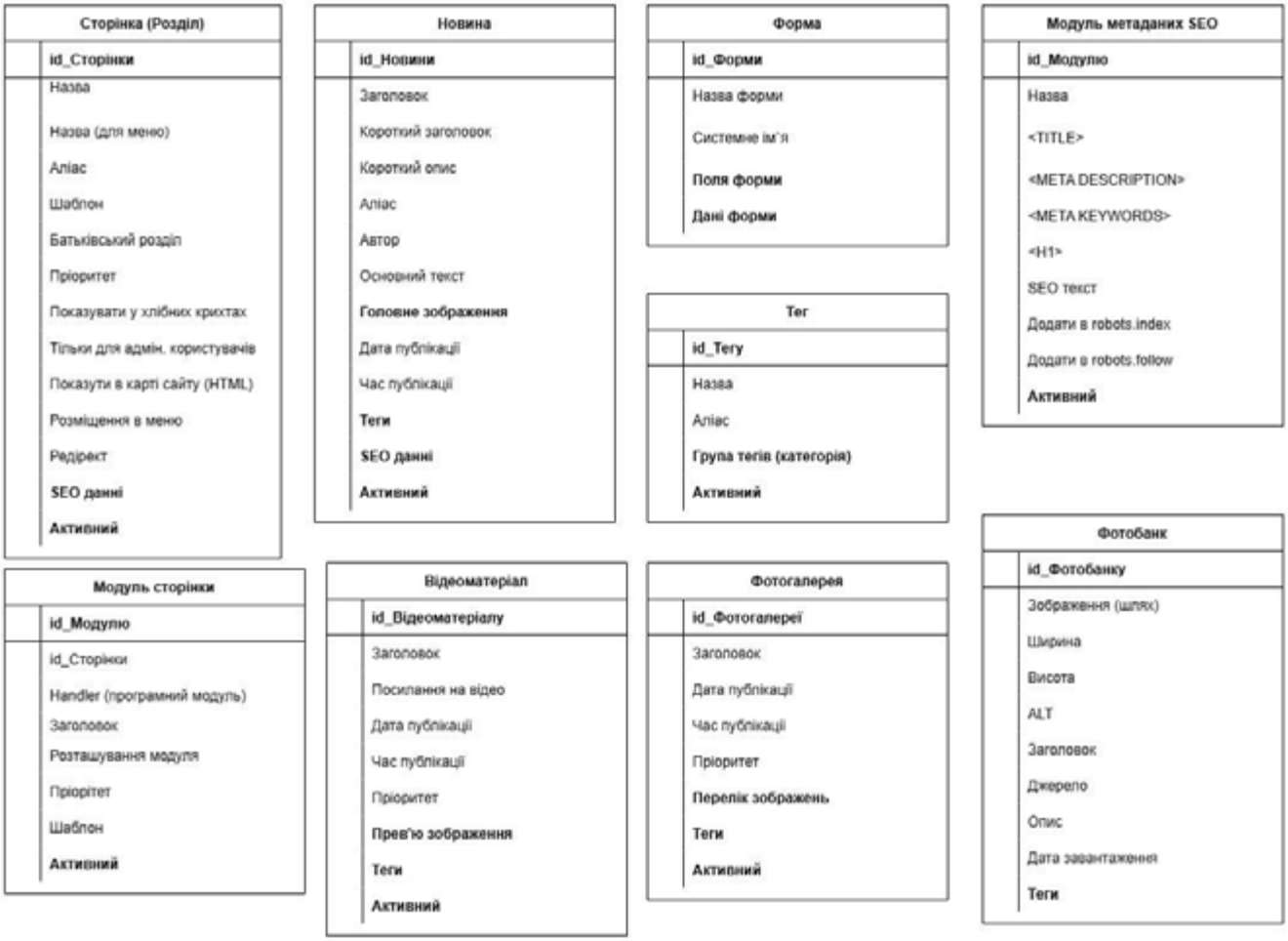

Рис. 6 – Основні сутності

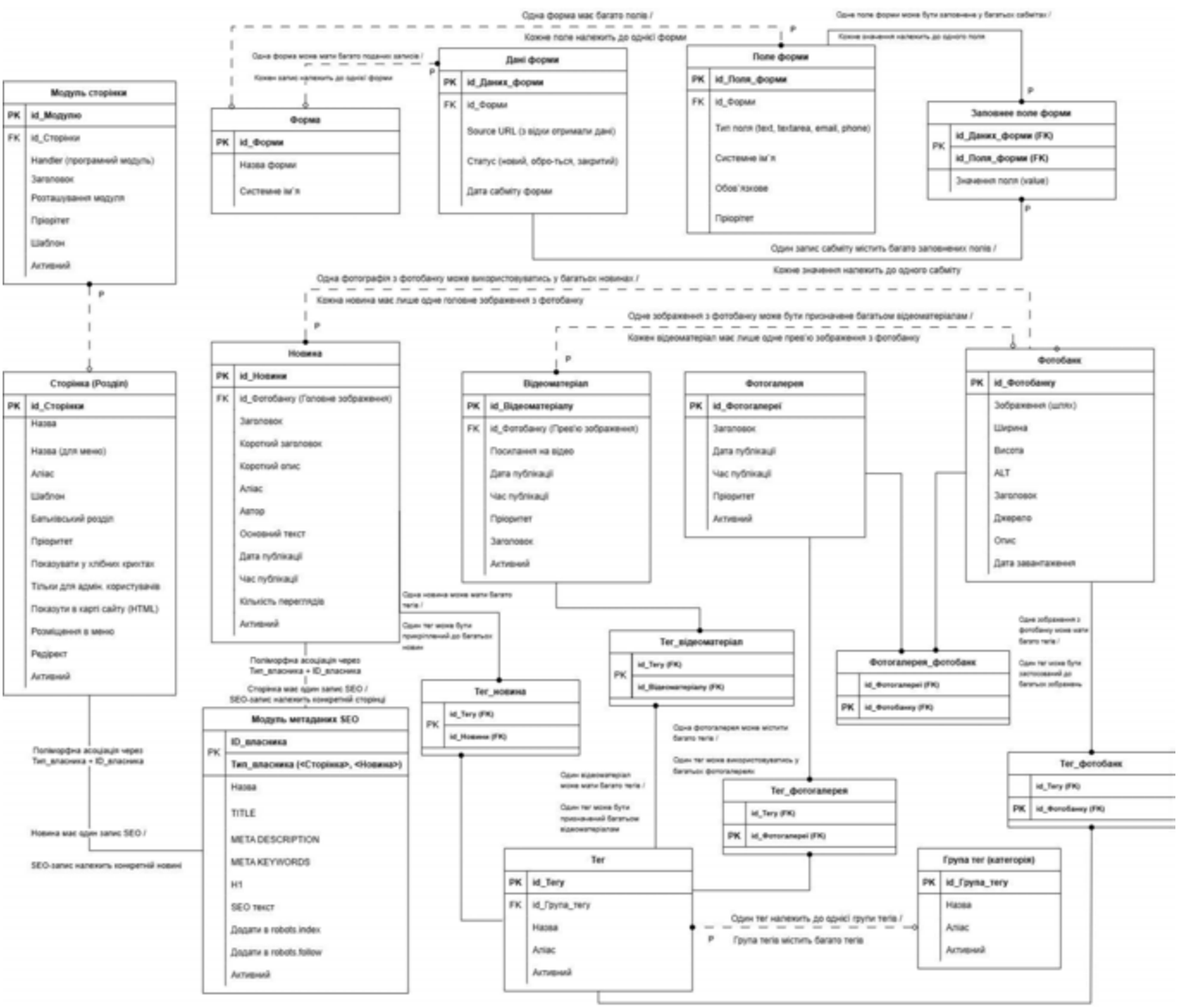

Рис. 7 – ER-діаграма логічного рівня

На даній ER-діаграмі (рис. 7) представлено логічний рівень моделювання бази даних інформаційної системи новинного порталу. Діаграма описує структуру об'єктів предметної області, їх атрибути та взаємозв'язки між сутностями. Фізичну реалізацію цієї моделі буде наведено у Розділі 3 при описі розробки програмного забезпечення. Більш детально ця діаграма представлена в Додатку Г.

Центральними сутностями є «Новина», «Сторінка», «Фотобанк», «Відеоматеріал», «Фотогалерея» та «Форма». Кожна новина має заголовок, короткий опис, основний текст, дату публікації, пов’язане головне зображення

(через зовнішній ключ до фотобанку), теги, SEO-дані та статус активності. Подібним чином побудовано структуру для відеоматеріалів і фотогалерей.

Для керування зображеннями створено сутність «Фотобанк», що зберігає технічні характеристики файлів та метадані, а також пов’язується з іншими об’єктами через посилання (наприклад, прев’ю, галереї, відео тощо).

Теги організовані через таблиці зв’язків «Тег_новини», «Тег_відеоматеріалу», «Тег_фотогалереї» із віднесенням тегів до категорій. Це забезпечує зручну фільтрацію та категоризацію контенту.

Сутність «Форма» дає змогу створювати кастомізовані форми з довільною кількістю полів, які визначаються через допоміжні таблиці. Вона призначена для збору зворотного зв’язку чи інтерактивної взаємодії з користувачами.

Модуль SEO-метаданих дає змогу зберігати оптимізаційні параметри (теги title, meta, keywords тощо) для різних типів записів (сторінки, новини), що дозволяє покращити видимість ресурсу в пошукових системах.

Таким чином, логічна модель чітко відображає структуру контенту, підтримує гнучке управління матеріалами, зображеннями, SEO та формами збереження даних. Усі взаємозв'язки реалізовані через зовнішні ключі, багато-з-багатьох асоціації описані за допомогою проміжних таблиць.

2.2 Діаграма класів та кооперацій

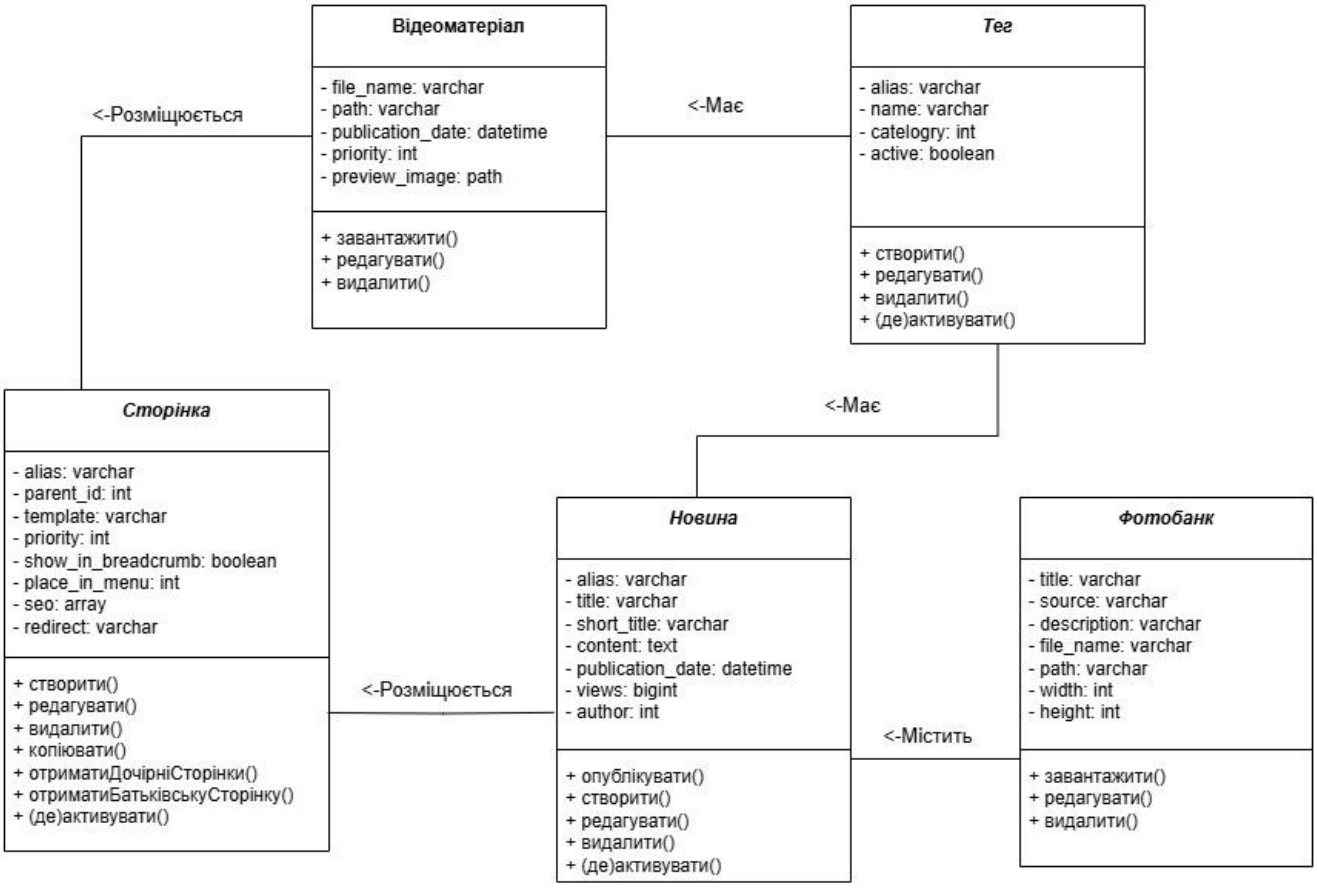

Рис. 8 – Діаграма класів

На діаграмі (рис. 8) представлено логічну структуру класів системи, які реалізують основні об’єкти новинного порталу: новину, сторінку, відеоматеріал, тег та зображення з фотобанку. Дана діаграма описує їх атрибути, методи та взаємозв’язки між класами на рівні об'єктно-орієнтованого програмування.

Клас Новина містить основні поля для текстового контенту (заголовок, короткий заголовок, зміст, дата публікації, кількість переглядів, автор), а також методи для створення, редагування, публікації та деактивації новин. Клас пов’язаний із Фотобанком, що містить зображення, які використовуються як медіа-вміст до публікацій.

Клас Сторінка відповідає за організацію структури сайту. Він містить поля для ієрархії (батьківська сторінка, позиція в меню), шаблон, пріоритет, SEO-

параметри та методи для керування сторінками (створення, редагування, отримання дочірніх сторінок тощо).

Клас Відеоматеріал зберігає інформацію про відеофайли (шлях, файл, дата публікації, зображення-прев'ю) і має методи завантаження, редагування та видалення. Як і новини, відео також можуть мати пов'язані теги.

Клас Тег використовується для категоризації контенту. Він зберігає назву, ідентифікатор категорії, статус активності та методи для управління тегами.

Класи пов'язані між собою асоціаціями: новини та відео можуть «мати» теги, новини та відео можуть «містити» зображення з фотобанку, а самі новини та відео «розміщуються» на певних сторінках сайту.

Дана діаграма відображає об'єктно-орієнтовану модель взаємодії між сутностями в системі та формує основу для реалізації внутрішньої логіки програмного коду.

На кооперaційній діаграмі (рис. 9) показано взаємодію класу «Новина» з іншими класами в системі, що формують повний життєвий цикл новинного матеріалу. Діаграма демонструє, як новина функціонує в контексті системи, які об'єкти впливають на її створення, публікацію та супровід.

Клас «Новина» є підкласом узагальненого класу «Новинний матеріал», до якого також належать статті, відеоматеріали та репортажі. Це дозволяє уніфікувати атрибути (заголовок, текст, дата публікації, перегляди тощо) та методи (створити, редагувати, опублікувати) для різних типів контенту.

Новина взаємодіє з об'єктом «Редактор», який керує її створенням і модифікацією. Усі дії, пов'язані зі змінами в новині, фіксуються в «Журналі змін», що зберігає історію редагування, дату і виконавця.

Окрему роль виконує «Стрічка новин», яка містить список новин та дозволяє додавати, видаляти або сортувати їх за заданими критеріями. Це забезпечує динамічне формування стрічки публікацій на сайті.

Також до новини можуть додаватись «Коментарі», які містять автора, вміст, дату і статус модерації. Це забезпечує зворотний зв'язок від читачів.

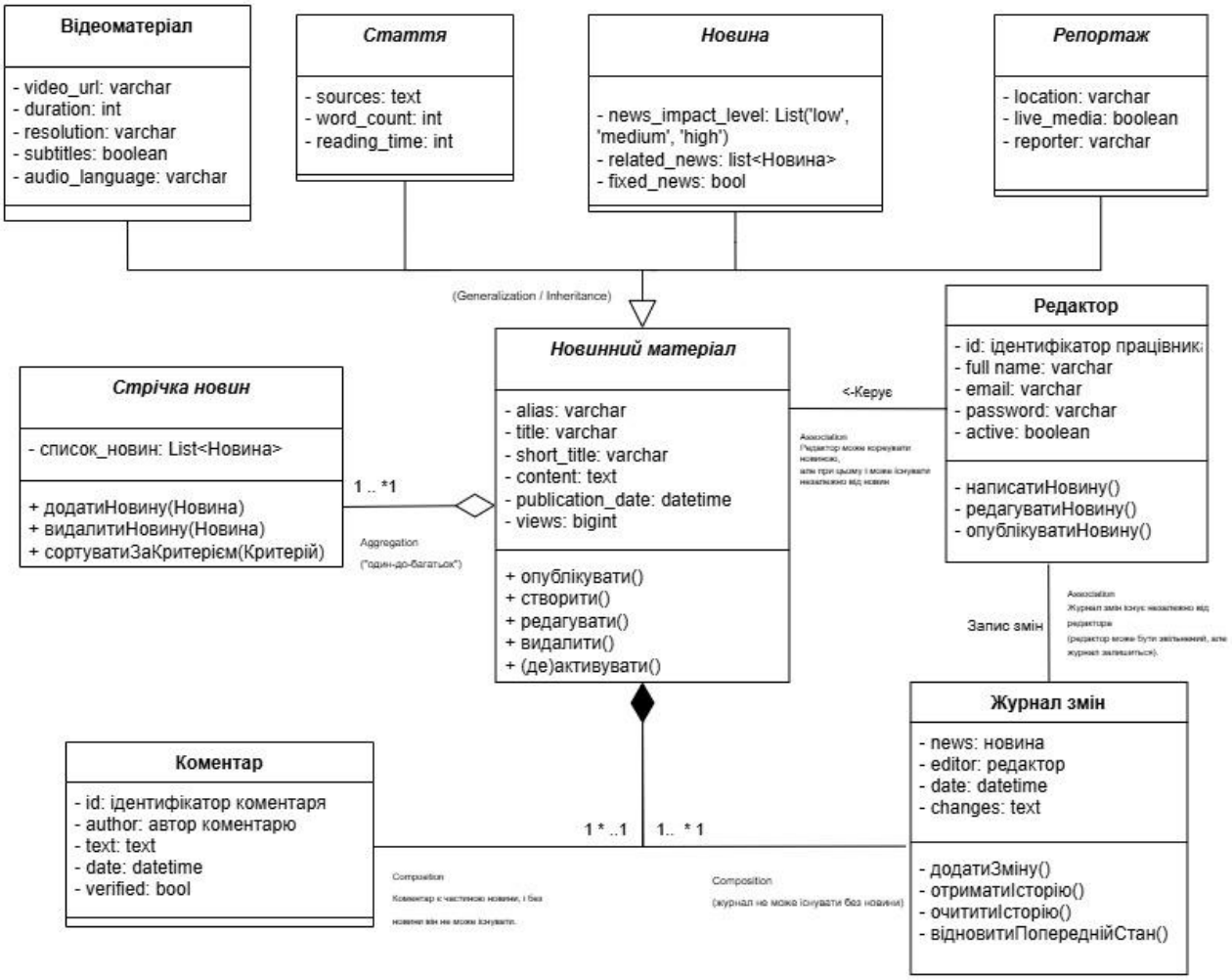

Рис. 9 – Діаграма кооперації клас Новина

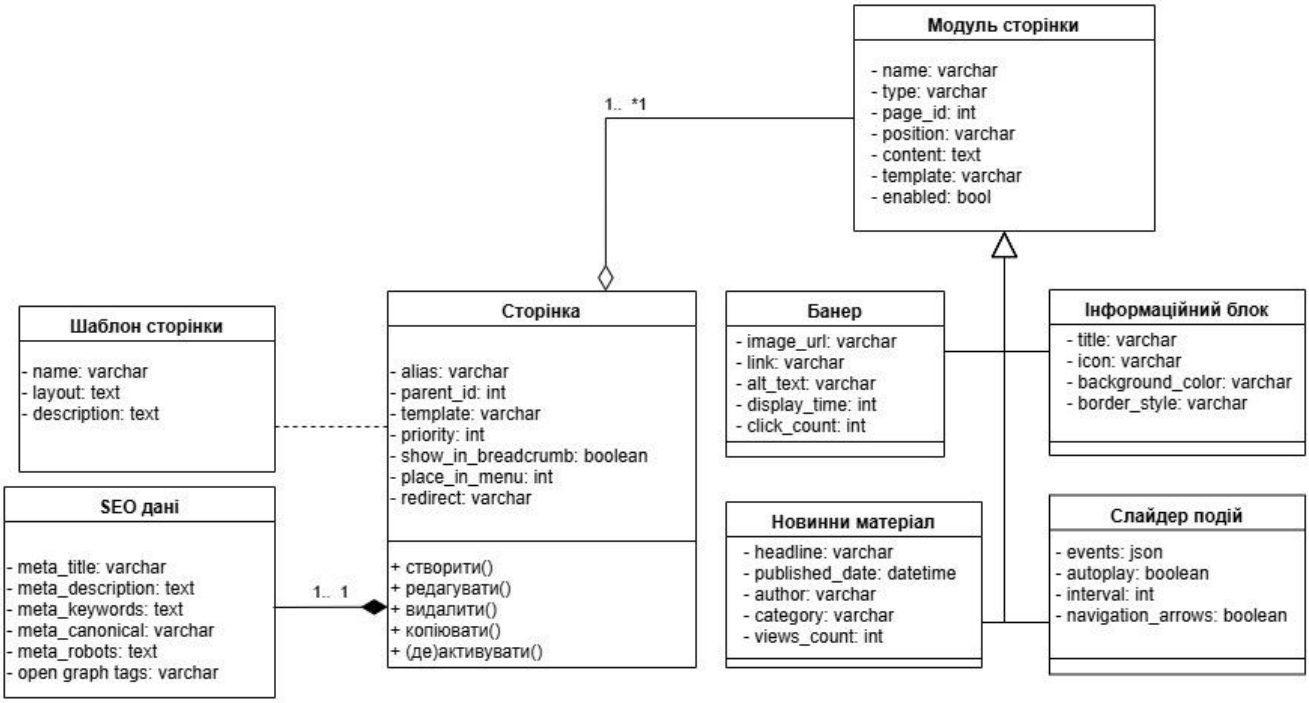

Рис. 10 – Діаграма кооперації клас Сторінка

На діаграмі кооперації класу Сторінка (рис. 10) зображено взаємодію класу «Сторінка» з іншими об’єктами, які забезпечують її вивід, структуризацію та наповнення в інформаційній системі. Клас «Сторінка» виступає центральною одиницею веб-структури, що об’єднує в собі зміст, дизайн і SEO-наповнення.

Сторінка містить основні параметри: псевдонім (alias), шаблон, пріоритет, розташування в меню, ознаку показу у хлібних крихтах тощо. Вона асоціюється з єдиним набором SEO-даних, які включають заголовок сторінки, опис, ключові слова, канонічне посилання, директиви robots та теги для соціальних мереж.

Шаблон сторінки задає її візуальне оформлення — це загальна схема компонування елементів (layout), що використовується для уніфікації вигляду.

2.3 Діаграма пакетів

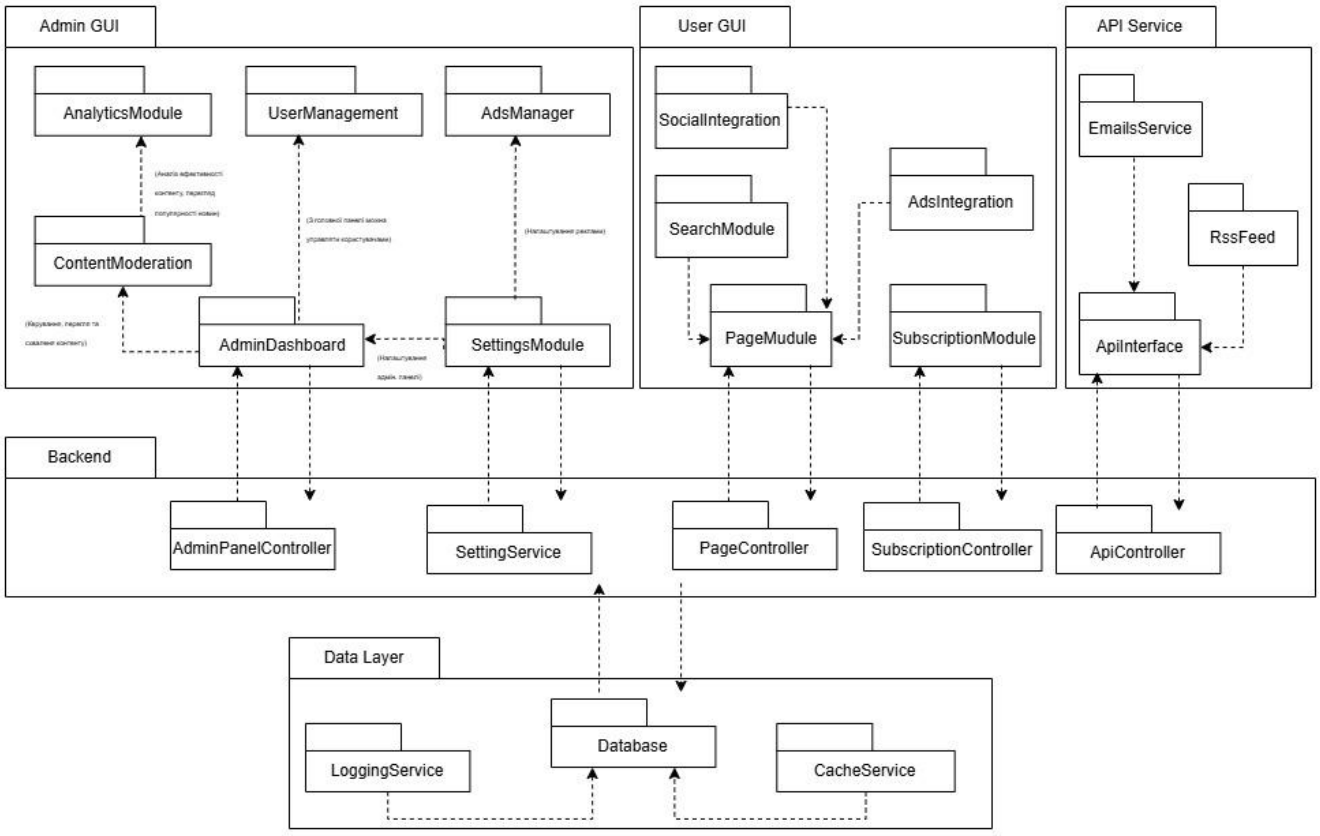

Рис. 11 – Діаграма пакетів

Діаграма пакетів — це тип UML-діаграми, що використовується для візуалізації структури програмної системи на рівні модулів або підсистем. Вона показує, як логічно згруповані класи, компоненти чи функціональні блоки об'єднуються в пакети (модулі), та які залежності існують між ними. Така діаграма допомагає зрозуміти загальну архітектуру системи, полегшує її підтримку та розвиток.

На діаграмі (рис. 11) зображено архітектурну структуру системи у вигляді діаграми пакетів, яка демонструє розподіл компонентів програмного забезпечення за логічними модулями. Така структура дозволяє побачити взаємозв'язки між підсистемами, залежності та рівні відповідальності окремих частин системи.

Система умовно поділена на чотири основні пакети.

- Admin GUI — інтерфейс адміністратора, який містить модулі для аналітики, управління користувачами, рекламою, модерацією контенту та налаштуваннями. Центральним вузлом є AdminDashboard, який координує взаємодію між модулями;
- User GUI — інтерфейс користувача з модулями пошуку, інтеграції із соціальними мережами, рекламою та підпискою. Взаємодія організована через PageModule та SubscriptionModule;
- API Service — набір зовнішніх сервісів, зокрема EmailService та RssFeed, які взаємодіють через ApiInterface;
- Backend — серверна частина, що включає контролери (AdminPanelController, SettingService, PageController, SubscriptionController, ApiController), які обробляють запити від інтерфейсів і забезпечують логіку взаємодії.

У нижньому рівні розміщено Data Layer — рівень даних, який представлений базою даних та допоміжними сервісами: LoggingService та CacheService. Вони відповідають за збереження, кешування та аудит подій у системі.

Стрілки вказують на напрямок залежності та виклику між модулями. Суцільні лінії з стрілками показують прямий виклик, пунктирні — залежності або використання функцій.

Загальна структура є модульною, масштабованою та ізольованою за рівнями, що спрощує підтримку, розширення та тестування системи.

2.4 Діаграма компонентів

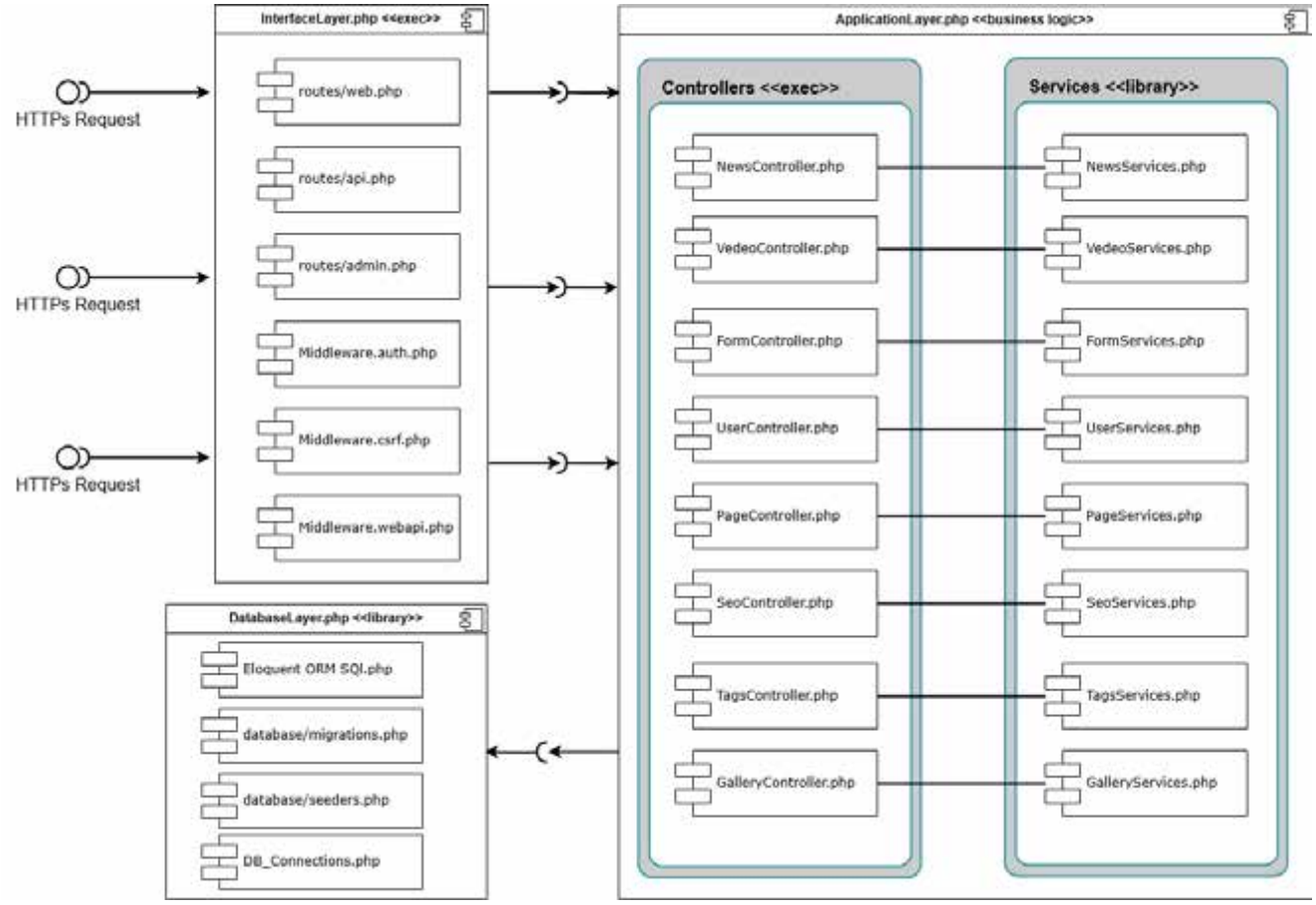

Рис. 12 – Діаграма компонентів

Запропонована система побудована за принципами багаторівневої архітектури, яка розділяє обробку запитів, логіку бізнес-процесів та роботу з базою даних на окремі рівні.

1. Рівень інтерфейсу (Interface Layer)

Цей рівень відповідає за обробку вхідних HTTP(S)-запитів і визначення маршруту, за яким вони мають бути оброблені. Він включає кілька файлів маршрутизації:

- routes/web.php — обробка запитів з веб-інтерфейсу користувача;
- routes/api.php — маршрути для API-запитів;
- routes/admin.php — адміністраторська частина.

Крім того, тут використовуються middleware-компоненти, які реалізують попередню обробку запитів:

- Middleware.auth.php — перевірка автентифікації;
- Middleware.csrf.php — захист від CSRF-атак;
- Middleware.webapi.php — загальні правила для роботи з API.

Ці елементи визначають, як саме запит буде проходити через систему перед тим, як потрапить до основної логіки.

2. Прикладний рівень (Application Layer)

На цьому рівні розміщені основні компоненти, які реалізують бізнес-логіку системи. Він умовно поділяється на дві групи:

Контролери (Controllers) — обробники запитів, які отримують дані з інтерфейсного рівня, передають їх до відповідних сервісів, і повертають результат користувачу або API.

- NewsController.php — керує створенням, редагуванням та публікацією новин;
- VideoController.php — керує відеоматеріалами;
- FormController.php — обробляє користувацькі форми;
- UserController.php — відповідає за управління користувачами;
- PageController.php — відповідає за структуру сторінок;
- SeoController.php, TagsController.php, GalleryController.php — додаткові компоненти, що забезпечують відповідні функції.

Сервіси (Services) — окремі модулі, що реалізують основну бізнес-логіку. Вони викликаються з контролерів і взаємодіють з базою даних.

Наприклад, NewsService.php реалізує логіку створення новини, її збереження, генерацію URL, індексацію в пошукових системах тощо.

Такий поділ дозволяє мінімізувати дублювання коду, спростити супровід системи та підвищити повторне використання логіки.

3. Рівень бази даних (Database Layer)

Цей рівень відповідає за збереження, оновлення та читання даних з бази. Усі дії з базою виконуються через стандартизовані бібліотеки та інструменти:

- Eloquent ORM SQL.php — основна ORM-бібліотека Laravel, яка забезпечує зручну роботу з моделями;
- database/migrations.php — опис структури таблиць;
- database/seeders.php — заповнення бази тестовими або початковими даними;
- DB_Connections.php — налаштування з’єднання з базою даних.

Цей рівень також може включати допоміжні сервіси для кешування (CacheService) та логування (LoggingService), що дозволяє оптимізувати навантаження і забезпечити збереження історії дій.

Всі HTTPS-запити потрапляють спершу в маршрутизатори та middleware (інтерфейсний рівень).

Звідти вони направляються до відповідного контролера (прикладний рівень).

Контролер викликає відповідний сервіс, який, своєю чергою, взаємодіє з базою даних для читання або запису інформації.

2.5 Архітектура та структура програмного забезпечення

При побудові системи планується використати наступний підхід: монолітної архітектури з використанням MVC-підходу (Model-View-Controller). Це класичний варіант побудови веб-додатків, який дозволяє чітко розділити відповідальність у коді між трьома основними складовими:

1. Model (модель) – відповідає за роботу з базою даних, тобто зберігання, оновлення, видалення і отримання інформації;
2. View (подання, вид) – відповідає за те, як інформація буде відображатися користувачу, це HTML-шаблони з динамічними вставками;
3. Controller (контролер) – обробляє запити користувача, отримує дані з моделі і передає їх до View.

Усі три частини працюють разом в рамках одного додатку, тобто бекенд і фронтенд розгортаються разом. Це не окремі сервери як у випадку, коли фронтенд роблять на React чи Vue, а єдина система, яка генерує сторінки прямо на сервері. Для цього використовується шаблонізатор Blade, який входить у стандарт того ж фреймворку Laravel.

Такий підхід спрощує розробку і розгортання, бо немає потреби налаштовувати окремий сервер для API або окремо збирати інтерфейс. Усе працює в одному місці і взаємодіє напряму. Це зручно і стабільно, особливо для новинних сайтів, де важлива швидкість завантаження сторінок і правильна індексація в пошукових системах.

Опис файлової структури проєктованої системи (див. рис. 13)

Core – це центральна частина системи, що зв’язує між собою інші модулі, такі як база даних, логіка застосунку і ресурси.

Database містить усе, що стосується роботи з базою даних.

- dbConn.php – налаштування підключення;
- migration.php – створення таблиць бази;
- seeds.php – наповнення таблиць початковими даними.

Resources містить зберігаються ресурси для інтерфейсу користувача.

- стилі (style.css, file.sass);
- скрипти (script.js);
- тут же можуть бути шаблони для сторінок.

Application містить логіку для роботи з основними розділами системи.

- UserComponent.php – управління користувачами;
- NewsComponent.php – робота з новинами;
- FormsComponent.php, TagsComponent.php, PhotobankComponent.php,

VideosComponent.php та інші – окремі компоненти для кожної сутності.

Інші важливі файли.

- composer.json – опис залежностей PHP;
- package.json – фронтенд-залежності;
- artisan.php – консольна утиліта Laravel;
- route.php – налаштування маршрутів.

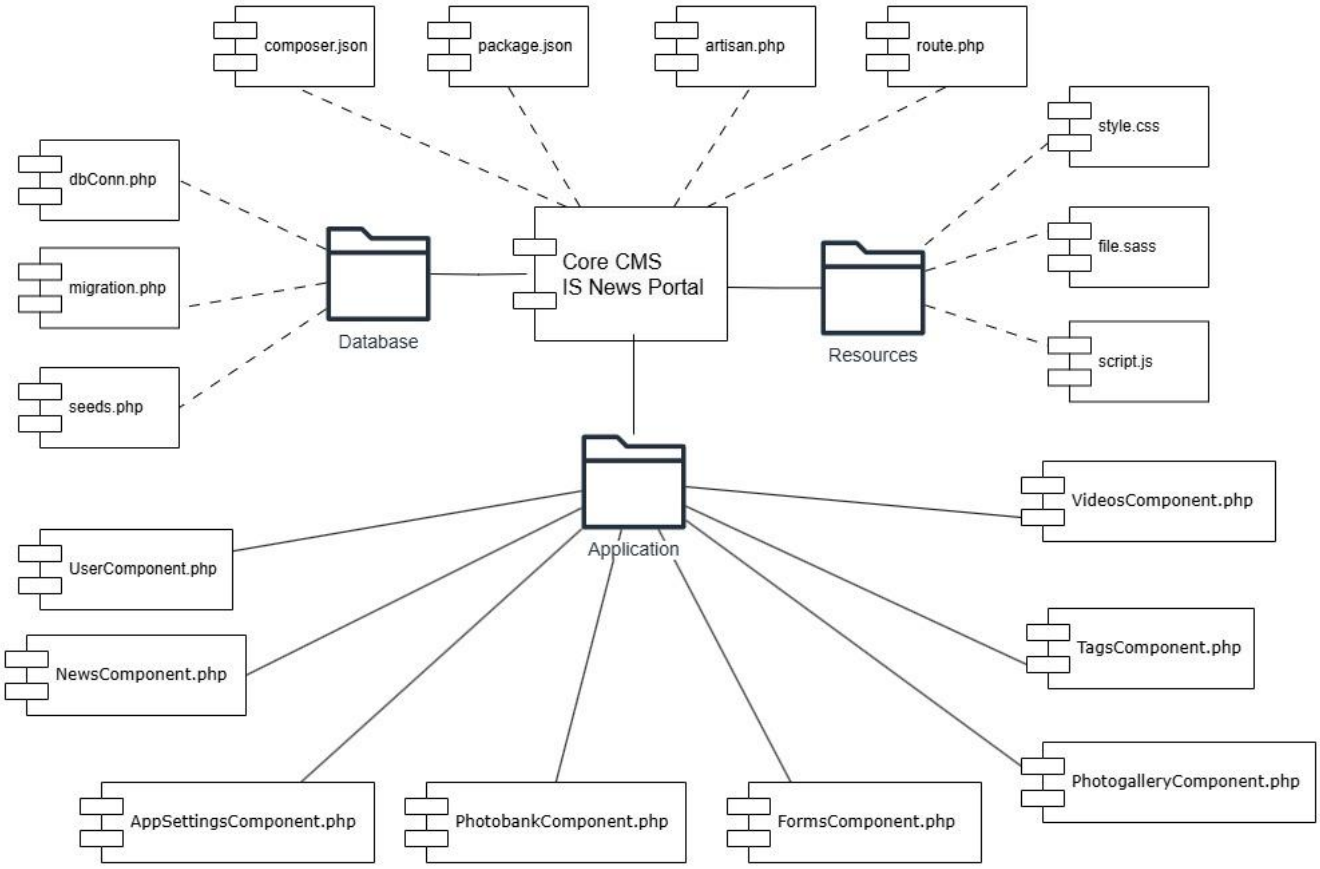

Рис. 13 – Наближений вигляд файлової структури розроблювальної системи

Загалом, архітектура системи є зрозумілою, сталою і простою в підтримці. Монолітний підхід дозволяє швидко розгортати проєкт без складної інфраструктури. MVC-модель чітко відокремлює логіку, дані і інтерфейс, що робить код більш впорядкованим і легким для розробки. Така структура чудово підходить для новинного порталу, де важливо мати стабільну систему з можливістю легкої модифікації і масштабування.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір інструментів для створення програмної системи

У процесі розробки інформаційної системи новинного порталу був застосований набір сучасних і перевірених інструментів та бібліотек, які забезпечують стабільність, продуктивність і масштабованість рішення. Основою програмної системи є фреймворк Laravel, побудований на мові програмування PHP. Усі компоненти були підібрані з урахуванням сумісності, підтримки, гнучкості налаштувань та відповідності вимогам проєкту.

Рис. 14 – Фреймворк: Laravel лого

Laravel — це потужний та зручний PHP-фреймворк, який реалізує архітектуру MVC та надає широкий набір готових рішень для веб-розробки. Він є ядром програмної системи. Проєкт зібраний на версії Laravel 10, яка підтримує сучасний PHP (версія 8.1.9 і вище) та має активну підтримку спільноти.

Laravel забезпечує.

1. Роботу з маршрутизацією, контролерами, шаблонізатором Blade;
2. Інтеграцію з ORM Eloquent для зручної роботи з базою даних;
3. Підтримку авторизації (Sanctum), соціальної аутентифікації (Socialite), API, черг, повідомлень;
4. Консольний інструмент Artisan для швидкої генерації класів, команд, міграцій тощо.

Основні залежності (composer), у файлі composer.json описані бібліотеки, які використовуються як обов'язкові.

- laravel/framework, illuminate/support — основні компоненти фреймворку;
- intervention/image — робота з зображеннями (обрізка, масштабування, водяні знаки);
- guzzlehttp/guzzle — HTTP-клієнт для інтеграції з API;
- spatie/laravel-newsletter, mailchimp-api — підписка на розсилки;
- laravel/ui, adminlte, filemanager — інтерфейс адміністратора;
- laravelcollective/html — формування HTML через Blade;
- nesbot/carbon, jenssegers/date — робота з датами;
- laravel/sanctum — захист API через токени;
- deepl-php, google/cloud-translate — автоматичний переклад контенту;

Інструменти для розробки.

- PHPStorm — основне середовище розробки (IDE), яке має підтримку Laravel, Eloquent, Blade, автодоповнення та інтеграцію з Git.
- Phpunit, mockery, collision — бібліотеки для тестування коду.
- Laravel Debugbar, IDE Helper, Ignition — полегшують налагодження проєкту.

Під час розробки планується використати систему контролю версій Git, зберігання та відновлення ведеться через GitHub.

Це дозволяє відслідковувати зміни, проводити pull-запити та рев'ю коду, відновити попередні версії при потребі.

У репозиторії налаштовано .gitignore, щоб уникнути комітування службових файлів, таких як vendor, node_modules, .env.

Робота зі стилями і скриптами: webpack.mix.js + node_modules

Для збирання frontend-ресурсів (JS, SASS, CSS) використовується webpack.mix.js, який є обгорткою над Webpack і входить до Laravel за замовчуванням. Усі залежності описані у package.json і встановлюються через npm install.

Після компіляції ресурси зберігаються у public/ — це:

- app.js, bootstrap.js — для JavaScript;
- app.css, admin.css — для стилів.

Дотримання конвенцій коду Laravel.

Усі файли, класи, методи і змінні пишуться згідно зі стайлгайдом Laravel.

- назви класів — PascalCase;
- методи — camelCase;
- Blade-файли — snake_case;
- структура контролерів, сервісів, ресурсів і моделей відповідає стандарту Laravel;
- суворе дотримання SRP (Single Responsibility Principle) — кожен клас відповідає за одну задачу.

Веб-сервер: Apache або Nginx. Серверна частина працює під керуванням Apache або Nginx, залежно від середовища. Обидва варіанти повністю сумісні з

Laravel. Apache використовує .htaccess для маршрутизації. Nginx — потребує ручного налаштування location-блоків.

Порівняння серверів Apache та Nginx

Таблиця 2

Критерій	**Apache**	**Nginx**
Продуктивність при великому трафіку	Добре на малих навантаженнях, але може «захлинутися» при пікових	Висока стабільність під навантаженням, добре масштабований
Обробка статичних файлів	Повільніше (більше RAM, CPU)	Дуже швидка (ідеально підходить як CDN, проксі або фронт)
Ресурсозатрати (CPU/RAM)	Вища, особливо при великій кількості одночасних з'єднань	Низька, оптимізована архітектура
Продуктивність з PHP (через PHP-FPM)	Добра, але вимагає fine-tuning	Відмінна — розроблений для роботи з PHP-FPM
Веб-сокети / Real-time	Працює, але менш ефективно	Висока ефективність, оптимізований для real-time
Популярність у хостингах	Часто встановлений за замовчуванням на shared-хостингу	Часто використовується в DevOps, Docker, Kubernetes

Підводячи підсумок, вибрані інструменти дозволили реалізувати гнучкий і розширюваний проєкт з чистою структурою. Laravel у поєднанні з PHP 8.1, Composer, Git, Webpack та сучасними бібліотеками забезпечує високу продуктивність, швидку розробку та зручне обслуговування системи.

3.2 Алгоритмізація та програмування програмного забезпечення

Розробка програмного забезпечення для інформаційної системи новинного порталу відбувалась у кілька основних етапів: налаштування середовища, реалізація базових структур, створення програмних модулів, побудова адміністративної панелі та публічної частини сайту. Кожен етап супроводжувався здебільшого мануальним тестуванням, логічним структуруванням коду та дотриманням принципів архітектури MVC та код стайлінгу PHP і Laravel.

Першим кроком було розгортання нового проєкту Laravel за допомогою Composer. Після ініціалізації проєкт було відкрито в середовищі розробки PHPStorm. Laravel автоматично сформував базову структуру директорій, файлів конфігурацій, маршрутів і шаблонів (див. рис. 15).

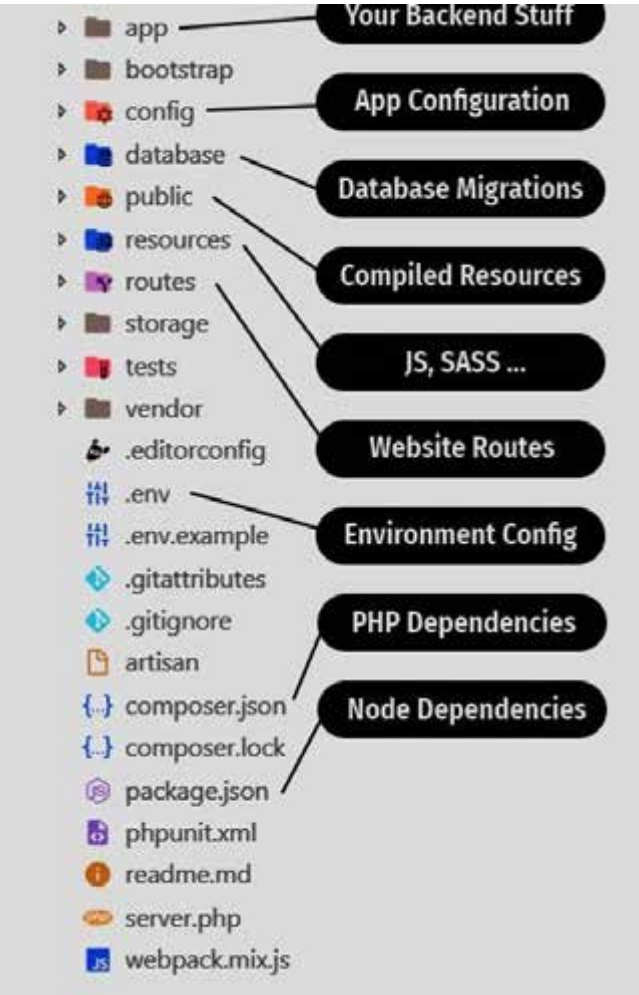

Рис. 15 – Структура Laravel Application

Для розробки локального середовища було налаштовано веб-сервери Apache і nginx. Вони працювали паралельно: nginx — як проксі, Apache — як сервер PHP. Домен ic-newsportal.local було додано у файл hosts, і після запуску серверів на екрані з’явилась привітальна сторінка Laravel (див. рис. 16).

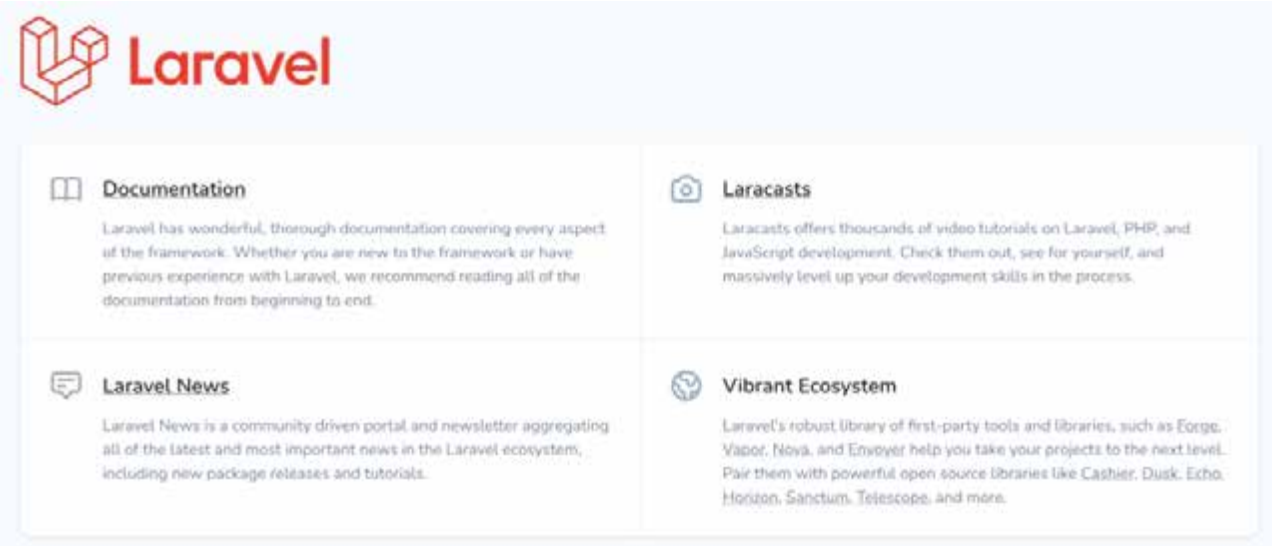

Рис. 16 – Laravel welcome сторінка

На цьому етапі почалась активна розробка ядра системи. Створено глобальні класи, зокрема BaseService для генерації псевдонімів сторінок, BaseController з базовою логікою для контролерів, загальні middleware та базові налаштування. Визначено дві основні групи маршрутів: web (для користувачів) та admin (для адміністраторів), які розділено в окремі файли та підключено в RouteServiceProvider (приклад маршрутів див. рис. 17).

```
/** Модуль Новини */
Route::namespace( value: 'News')->prefix( prefix: 'news')->group(function() {
    Route::get( uri: '', action: 'NewsController@index')->name( name: 'admin.news.list');
    Route::get( uri: 'create', action: 'NewsController@create')->name( name: 'admin.news.create');
    Route::get( uri: '{id}', action: 'NewsController@show')->name( name: 'admin.news.show');

    Route::post( uri: 'create', action: 'NewsController@store')->name( name: 'admin.news.store');
    Route::get( uri: 'edit/{id}', action: 'NewsController@edit')->name( name: 'admin.news.edit');
    Route::patch( uri: 'update/{id}', action: 'NewsController@update')->name( name: 'admin.news.update');
    Route::delete( uri: 'delete/{id}', action: 'NewsController@delete')->name( name: 'admin.news.delete');
});
```

Рис. 17 – Маршрути для модуля Новин

Після створення інфраструктури системи, розпочалась робота над ключовими модулями (див. список модулів рис. 18).

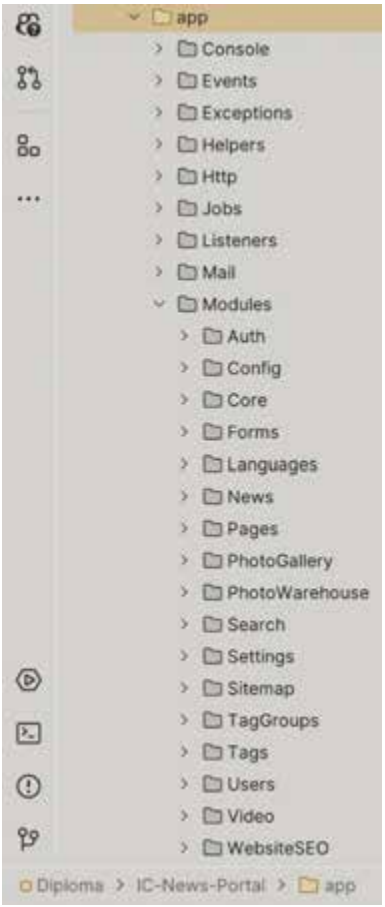

Рис. 18 – Програмні модулі системи

Було реалізовано такі основні функціональні модулі системи: користувачі та авторизація, сторінки (розділи сайту), новини, теги, відеоматеріали, форми (зворотній зв'язок, підписка), фотобанк (загальний банк зображень), налаштування сайту (див. рис. 19).

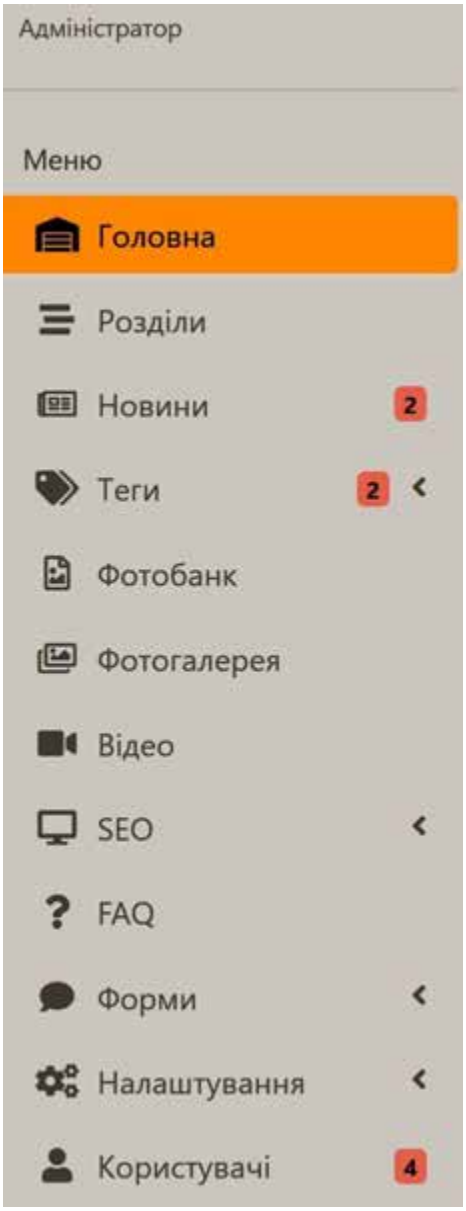

Рис. 19 – Навігація адмін панель, перелік програмних

Екранні форми та вигляд вкладок адмін панелі можна буде переглянути в Додатку В (екранні форми).

Для кожного з модулів створювалась власна модель Eloquent ORM (ООП підхід по роботі із SQL СУБД), приклад коду моделі Tag (див. рис. 20).

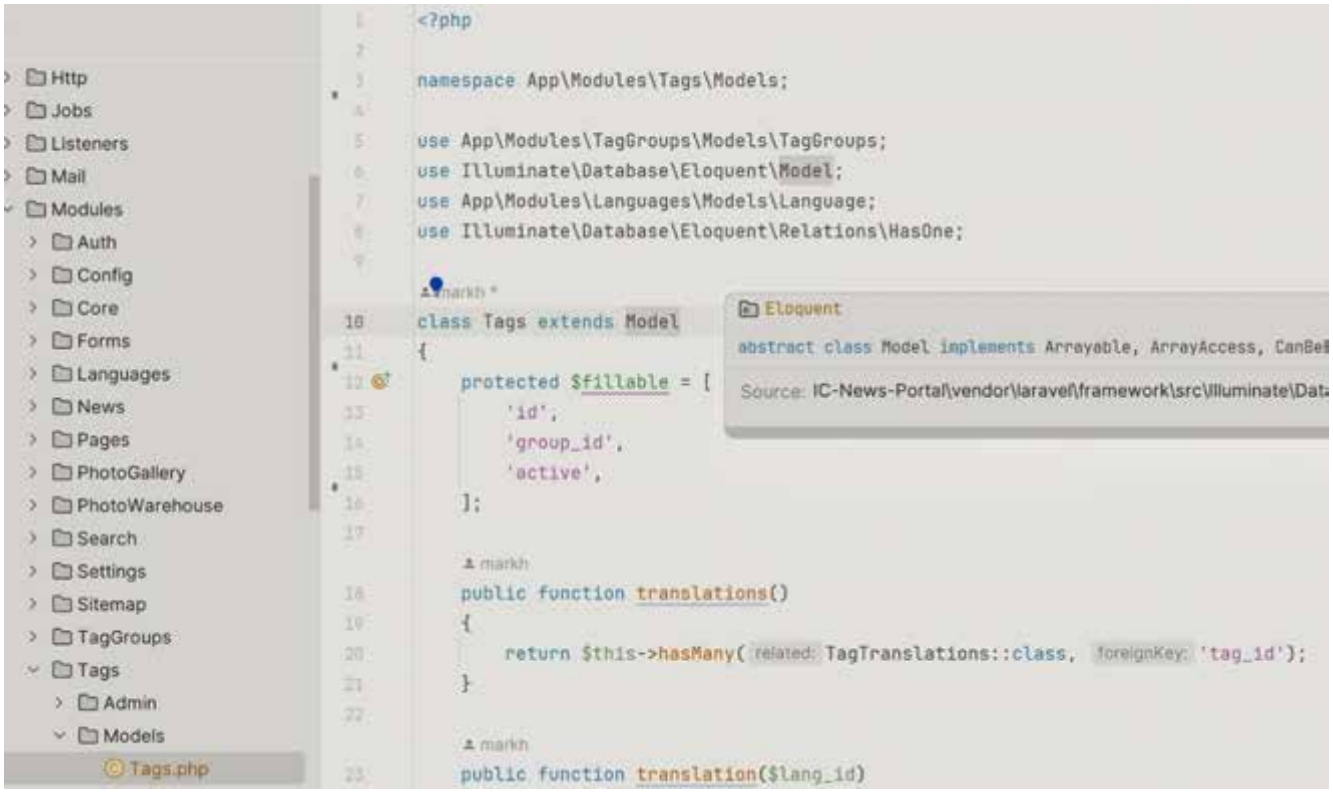

Рис. 20 – Модель таблиці Тег

Міграція для створення таблиць, контролер для обробки запитів, сервіс для логіки взаємодії з моделями, репозиторій для абстрагування запитів до БД (приклад міграцій та запитів наведено в розділі 3.4). Також розроблялись Blade-шаблони для адміністративної частини з використанням AdminLTE: сторінки перегляду списків, форм редагування та створення, меню у навігації. Для кожного програмного модуля створено свій набір базових шаблонів, нижче буде наведено приклад для модуля новин існує декілька шаблонів виводу новин, три або чотири колонки, з пагінацію чи без, слайдер тощо, відповідно при додавання модуля в систему на кожній окремій сторінці можна обрати який шаблон буде задіяно (приклад шаблону див. рис. 21).

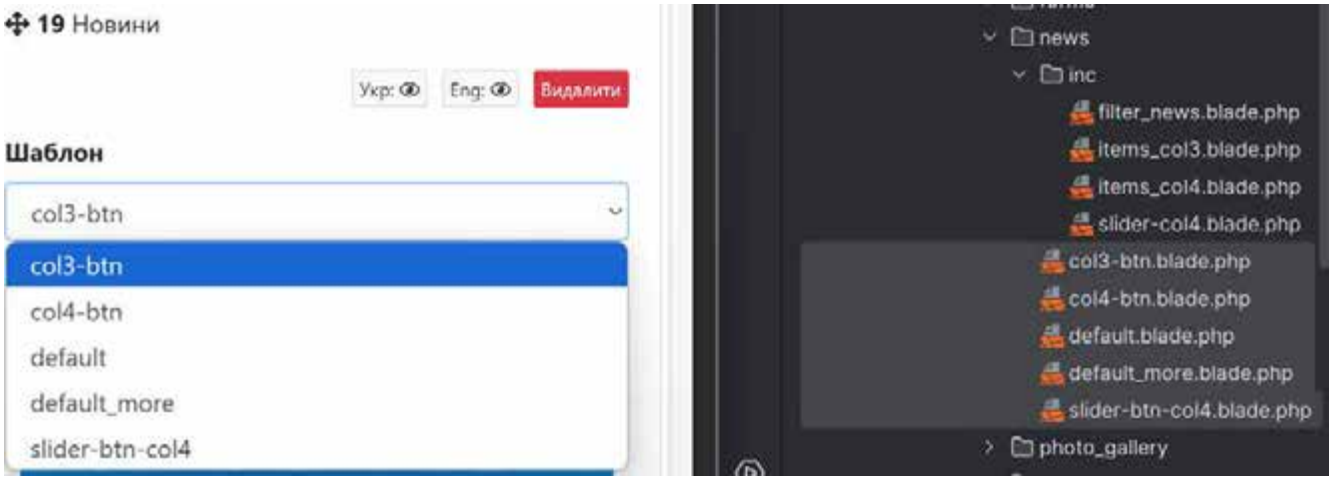

Рис. 21 – Шаблони модуля новин (з ліва обираємо, з право усі що є в проєкті)

Після реалізації всіх CRUD-операцій у кожному модулі адміністративна панель була протестована. Перевірялась навігація між модулями, збереження та видалення даних, робота з формами, генерація SEO-мета, підключення файлів з фотобанку та взаємодія з тегами.

Наступним етапом була реалізація публічної частини сайту. Для цього створено шаблони Blade для головної сторінки, перегляду новин, відео, тегів, сторінок розділів, фотогалерей та форм. Для оформлення використовувались SCSS-файли, які компілювались за допомогою webpack.mix.js у фінальні app.css та app.js. JavaScript відповідав за взаємодію з формами, слайдерами, адаптивним меню та іншими елементами інтерфейсу, нижче буде наведено скріншот із PHPStorm, де продемонстровано як через webpack JavaScript файли компілюються в один оптимізований файл js/admin, який використовується для реалізації інтерактивності в адмін панелі (див. рис. 22).

Рис. 22 – Приклад використання webpack для зборки JS-коду

Майже усі основні шаблони оптимізовано для SEO так як використовується один єдиний SeoService.php який формує дані для шаблонів, використано OpenGraph, теги title і description формуються динамічно. Також реалізовано sitemap.xml і robots.txt через відповідні компоненти.

Для побудови sitemap.xml треба запустити консольну команду: php artisan sitemap:create. Ця консольна команда запустить handle метод класу class GenerateSitemap extends Command, який в свою чергу проаналізує наявні сутності в базі даних (новини, відеоматеріали, сторінки тощо) і сформує XML структуру (див. рис. 23).

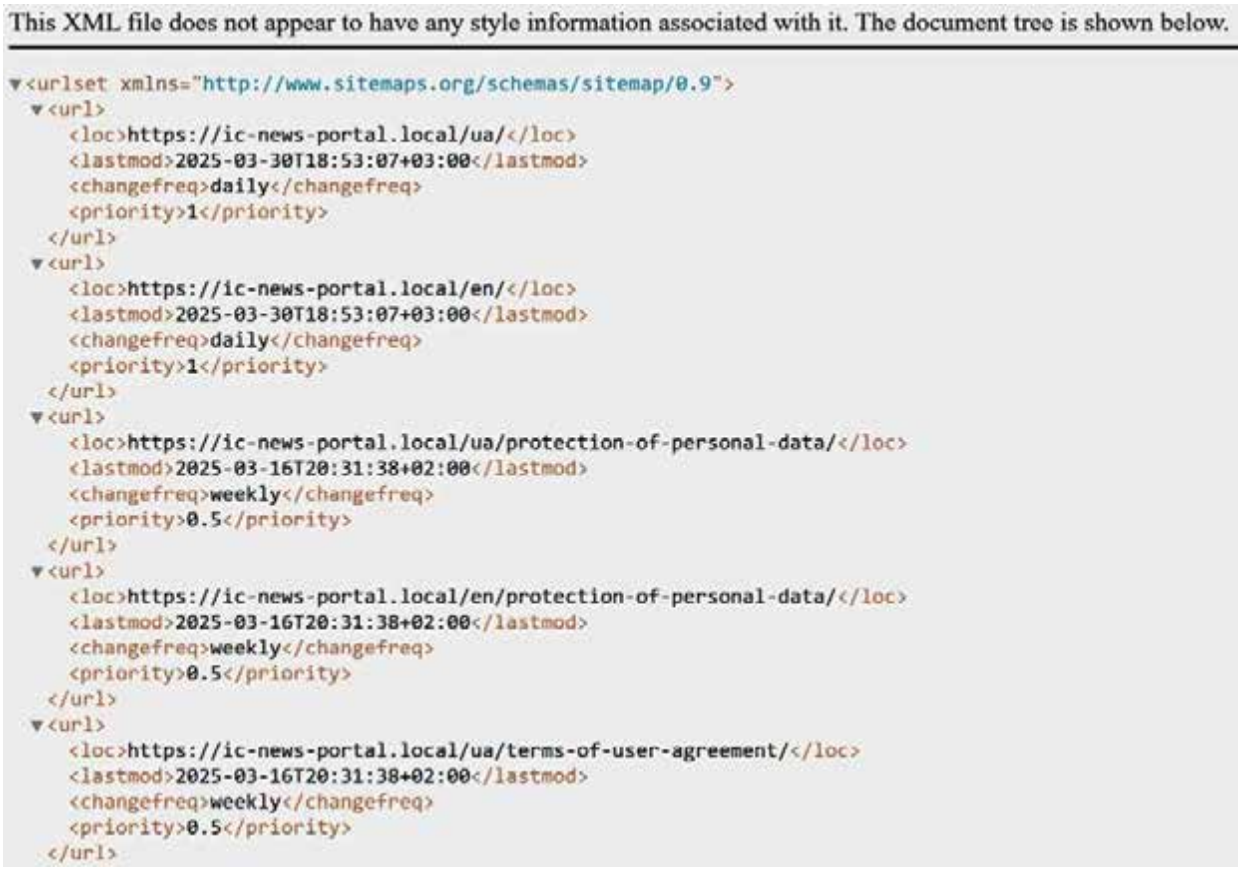

Рис. 23 – Згенерована карта сайту

В свою чергу запити, які шукають сторінки та інші активні сутності в базі даних, які потім будуть додані до сайтмапу виглядають наступним чином:

```
public function getActive(array $notIds = []): array
{
    $pages = Page::whereNotIn('id', $notIds)
        ->where('restrict_access', '=', 0)
        ->select(['parent_id', 'id', 'updated_at'])
        ->with(['translations' => function ($q) {
            $q->select(['page_id', 'lang_id', 'slug', 'redirect_url', 'in_sitemap_xml'])->where('active', 1);
        }])->orderBy('id')->get();

    return $this->prepareData($pages);
}
```

Рис. 24 – Eloquent ORM запит для отримання активних сторінок

Також було розроблено конфігураційний файл robots.txt для пошукових ботів, в цьому файлі можна вказувати різного роду інструкції та вказівки для пошукових систем, в тому числі можна забороняти індексувати певні сторінки

сайту, розширення та вказувати посилання де знаходить карта сайту, яку будуть зчитувати пошукові боти для кращої індексації ресурсів порталу (див. рис. 25).

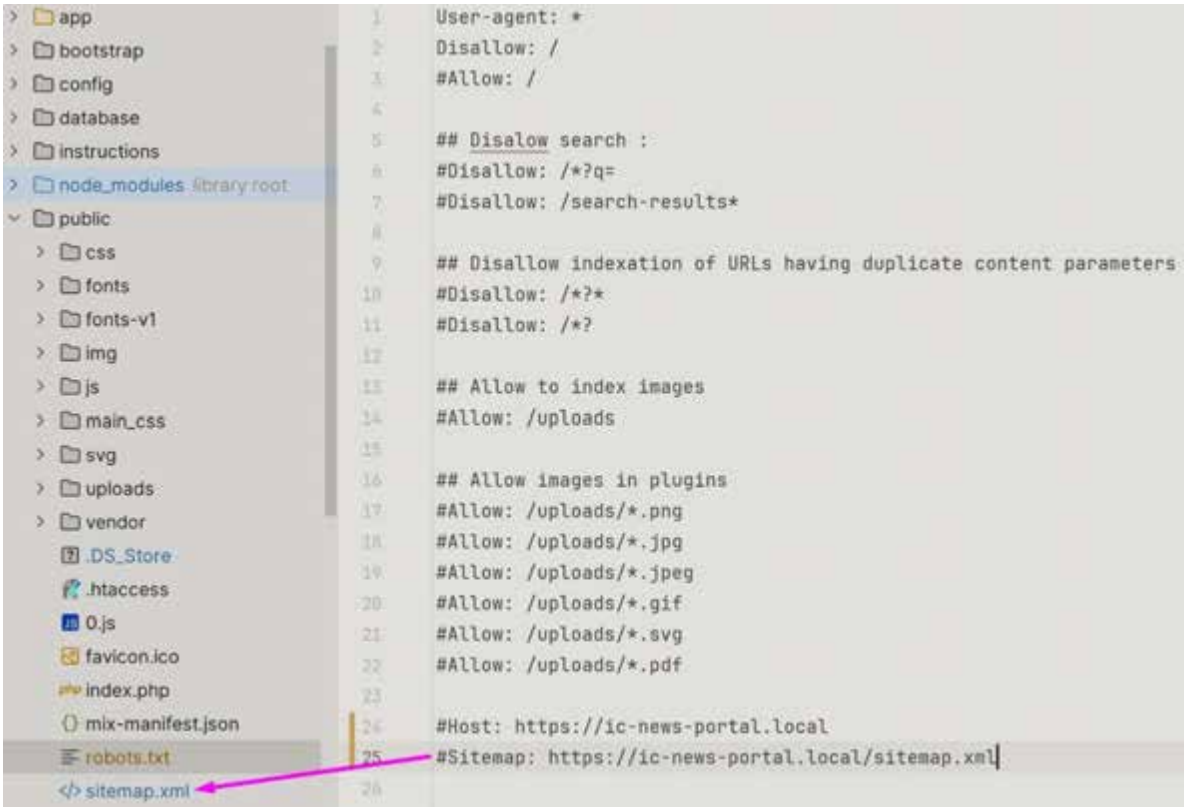

Рис. 25 – Приклад конфігураційного файлу robots.txt

Останнім етапом стало фінальне тестування, налагодження, виправлення виявлених дрібних помилок, перевірка швидкодії. Завдяки чіткій структурі коду та єдиній архітектурі система вийшла монолітною, масштабованою та зручною для подальшого розвитку.

3.3 Вибір системи управління базою даних

У процесі розробки інформаційної системи для новинного порталу ключовим етапом є вибір відповідної системи управління базами даних (СУБД).

Враховуючи характер даних, вимоги до продуктивності та інтеграції з веб-фреймворком, було прийнято рішення на користь реляційної СУБД, зокрема MySQL.

Реляційні СУБД підходять для систем, де дані мають чітку структуру та взаємозв'язки. У випадку новинного порталу, сутності такі як новини, користувачі, коментарі, категорії та теги мають визначені атрибути та зв'язки між собою. Реляційна модель дозволяє ефективно організувати ці дані, забезпечуючи цілісність та узгодженість інформації.

Порівняльний аналіз СУБД

Таблиця 3

Критерій	**MySQL**	**PostgreSQL**	**MS SQL Server**
Ліцензія	GPL / Open Source	Open Source	Комерційну ліцензію (SQL Server Standard / Enterprise)
Продуктивність (читання)	Висока (≈ 32,000 оп/сек)	Висока (≈ 29,000 оп/сек)	Висока (≈ 31,000 оп/сек)
Продуктивність (запис)	Середня (≈ 10,000 оп/сек)	Середня (≈ 9,800 оп/сек)	Висока (≈ 11,000 оп/сек)
Транзакції/секунда	≈ 2,200	≈ 1,900	≈ 2,300
Масштабованість	Висока (реплікація, шардінг)	Висока (Citus, Postgres-XL)	Висока (Always On, реплікація)
Інтеграція з Laravel	Повна підтримка	Повна підтримка	Часткова підтримка (через SQLSRV драйвер)
Простота налаштування	Висока	Середня	Середня (вимагає ліцензування, Windows або Docker)

Рис. 26 – СУБД MySQL лого

Причини обрати MySQL для даного проєкту.

- Інтеграція з Laravel: Laravel має вбудовану підтримку MySQL, що забезпечує легке налаштування та використання. Фреймворк надає зручні інструменти для роботи з базою даних, такі як Eloquent ORM, міграції та сидери, що значно спрощує розробку;
- Продуктивність: MySQL демонструє високу продуктивність при операціях читання, що є критичним для новинного порталу з великим обсягом трафіку. Завдяки оптимізаціям у рушії зберігання InnoDB, система ефективно обробляє запити;
- Масштабованість: MySQL підтримує вертикальне та горизонтальне масштабування через реплікацію та шардінг, що дозволяє системі адаптуватися до зростаючих навантажень;
- Простота налаштування та підтримка: MySQL має простий процес встановлення та налаштування, а також велику спільноту користувачів, що забезпечує доступ до численних ресурсів та швидке вирішення проблем.

3.4 Фізична модель бази даних та її реалізація

Раніше в розділі: 2.1 (див. рис. 7), було побудовано логічну модель даних, в свою чергу в цьому розділі буде зазначено яким чином було реалізовано фізичну структуру таблиць та зв`язків бази даних новинного порталу. Фізична модель бази даних реалізована у вигляді реляційних таблиць із чітким визначенням типів даних що наявні в обраній у пункті 3.3 СУБД MySQL, первинних та зовнішніх ключів для забезпечення. (див. рис. 27.1).

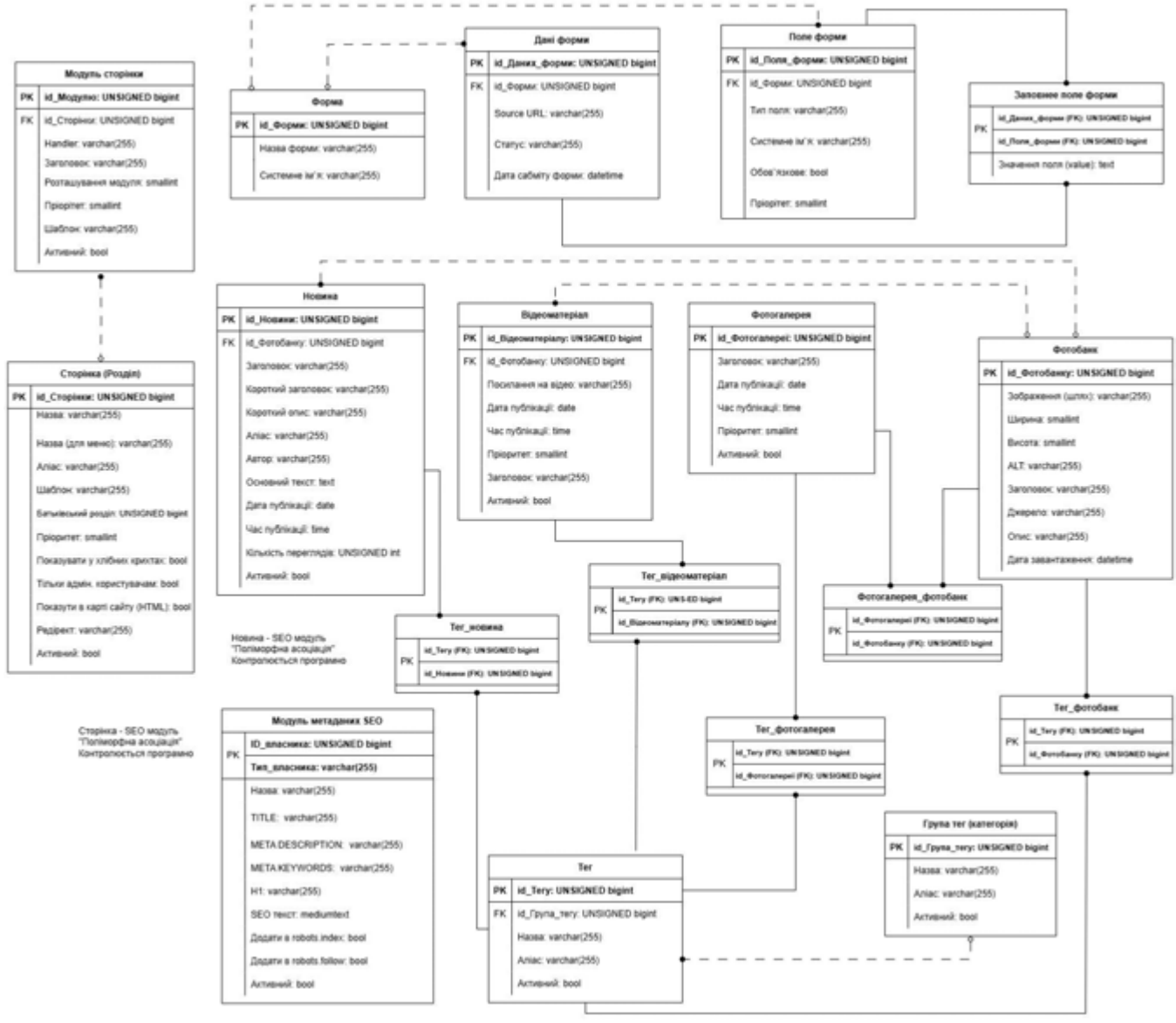

Рис. 27.1 – Фізична структура таблиць

У реалізації використовуються типи даних UNSIGNED bigint для ідентифікаторів, varchar для текстових полів (до 255 символів), text для довших описів, bool для логічних значень, smallint для збереження числових пріоритетів, а також date, time та datetime для збереження дат та часу. Завдяки цьому структура бази є гнучкою, масштабованою та оптимізованою для швидкого пошуку і збереження інформації.

Наступним кроком після побудови фізичної моделі бази даних стало написання SQL-запитів, які відповідають структурі, поданій на ER-діаграмі. Це дозволило створити таблиці з чітко визначеними зв’язками через первинні та зовнішні ключі. Приклад таких SQL-запитів (див. рис. 28).

```sql
CREATE DATABASE IF NOT EXISTS news_portal_ic CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE news_portal_ic;

CREATE TABLE IF NOT EXISTS mos_forms (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  title VARCHAR(255) NOT NULL,
  system_name VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS mos_form_fields (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  form_id BIGINT UNSIGNED NOT NULL,
  field_type VARCHAR(100) NOT NULL,
  input_name VARCHAR(255) NOT NULL,
  is_required tinyint(1) DEFAULT FALSE, -- BOOL
  priority SMALLINT DEFAULT 0,
  PRIMARY KEY (id),
  FOREIGN KEY (form_id) REFERENCES mos_forms(id) ON DELETE CASCADE
);
```

Рис. 27.2 – SQL-запити для створення бази даних

Однак, оскільки в проєкті використовується не чистий SQL, а фреймворк Laravel з об'єктно-реляційним перетворенням (ORM — Eloquent), то створення та керування структурою бази даних реалізовано через міграції у каталозі database/migrations (див. рис. 28).

Рис. 28 – Приклад Laravel міграції для створення таблиць БД

Для кожної таблиці було створено окрему міграцію, що описує структуру таблиці за допомогою PHP-коду. Це дало можливість легко змінювати структуру БД, відслідковувати її зміни в системі контролю версій та автоматично розгортати базу на будь-якому середовищі за допомогою команди: php artisan migrate. В результаті виконання цієї команди Laravel створює всю структуру бази даних (див. рис. 29) згідно з описаними міграціями.

Крім того, для кожної таблиці було створено окрему модель (Model) Laravel (див. розділ 3.2, рис. 20), яка відображає відповідну таблицю у вигляді класу. Це дозволяє працювати з даними як з об'єктами в стилі ООП: створювати, редагувати, видаляти або отримувати записи за допомогою методів класу. Наприклад, модель Form відповідає таблиці mos_forms, і через неї можна виконувати запити типу:

```
$forms = Form::all();
$newForm = Form::create([...]);
```

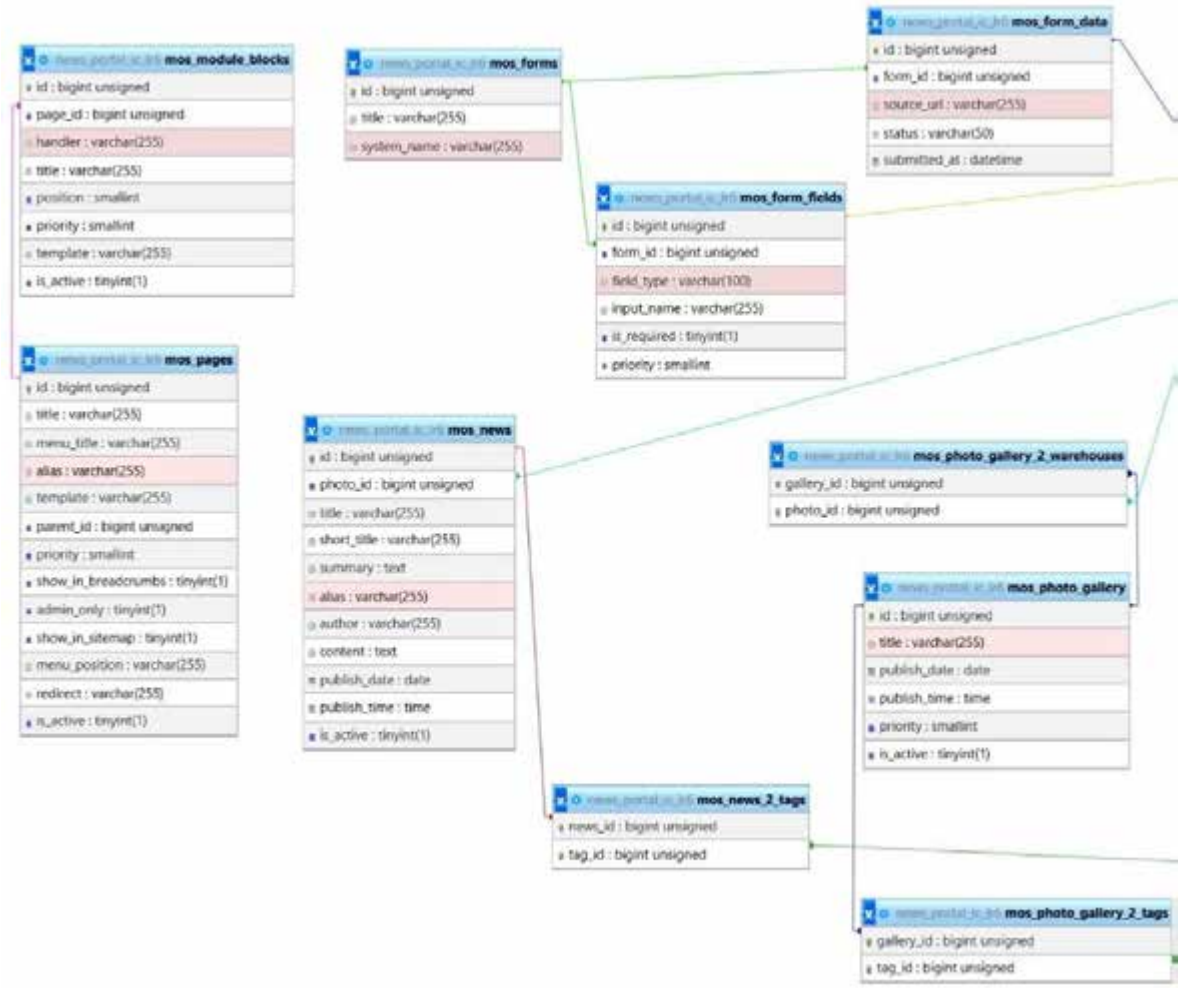

Рис. 29 – Фізична структура побудованої БД (phpMyAdmin diagram)

Після успішного запуску команди php artisan migrate, яка створила всі необхідні таблиці в базі даних, наступним кроком було виконання команди: php artisan db:seed.

Ця команда використовується для запуску сідів (seeds) — спеціальних PHP-класів, які автоматично наповнюють базу тестовими або стартовими даними. Сіди дозволяють швидко заповнити таблиці типовим вмістом, необхідним для початкової роботи системи.

Рис. 30 – LanguageSeeder стартове заведення англ. та укр. мови в системі

Файл database/seeds/LanguageSeeder.php (див. рис. 30) відповідає за наповнення таблиці мов (languages) початковими значеннями — українською та англійською мовами. Кожна мова має ключові поля: назва (title), повна назва (title_full), ISO-код (code_iso_639), активність (active), пріоритет тощо. У методі run() ці значення передаються у таблицю за допомогою методу Language::insert(...), який використовує модель Language.

Цей підхід значно спрощує налаштування середовища для розробки або тестування, дозволяючи швидко отримати базові дані після розгортання проєкту.

4 ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЯ СИСТЕМИ

4.1 Вимоги до апаратного та програмного забезпечення

Вимоги до веб-сервера (хостингу). Для забезпечення стабільної та безперебійної роботи інформаційної системи типу CMS новинного порталу на основі Laravel, необхідно використовувати сервер із такими мінімальними характеристиками:

Апаратні вимоги.

- Процесор (CPU): мінімум 2 ядра (рекомендовано від 4 ядер для середнього навантаження);
- Оперативна пам’ять (RAM): мінімум 6 ГБ (рекомендовано від 8–16 ГБ для продуктивного середовища);
- Місце на диску (SSD): мінімум 20 ГБ вільного простору (залежить від обсягу контенту, медіафайлів та резервних копій);
- Тип накопичувача: SSD (для швидшого читання/запису та кращої продуктивності).

Програмні вимоги.

- Операційна система: Linux (Ubuntu 20.04 або новіше рекомендовано);
- Веб-сервер: Apache 2.4 або Nginx 1.18+;
- PHP: версія 8.1.9;
- MySQL: версія 8.0 або MariaDB 10.6+;
- Composer: остання стабільна версія для керування залежностями PHP;

- Підтримка HTTPS: через встановлений SSL-сертифікат;
- Додаткове ПЗ: доступ до SSH для адміністрування.

У разі навантаження на портал (велика кількість одночасних відвідувачів, завантаження фото/відео, використання API) бажано використовувати VPS (віртуальний приватний сервер, який надає користувачеві ізольоване серверне середовище з власними ресурсами (CPU, RAM, диск) на фізичному сервері, спільно з іншими користувачами) або виділений сервер. Для високонавантажених систем — хмарні рішення на базі AWS, DigitalOcean, Hetzner тощо.

Далі буде наведено вимоги до пристрою користувача.

Інформаційна система є веб-додатком, тому не потребує встановлення на комп’ютер користувача. Для роботи достатньо сучасного браузера та стабільного доступу до Інтернету.

Програмні вимоги: Браузер: будь-який сучасний веб-браузер з підтримкою HTML5, CSS3, JavaScript (ES6). Підтримуються: Google Chrome (версія > 100), Mozilla Firefox (версія > 100), Microsoft Edge на базі Chromium (версія > 100), Safari на пристроях Apple (версія > 13), Opera (версія > 85).

Мінімальна роздільна здатність екрана: 1024×768 пікселів (адаптивна верстка також дозволяє комфортне використання на менших екранах)

Інтернет-з’єднання.

- Мінімальна швидкість — 2 Мбіт/с;
- Рекомендована швидкість — від 10 Мбіт/с і вище для комфортного завантаження фото, відео та інтерактивних елементів.

Підтримувані пристрої.

- Настільні комп’ютери та ноутбуки: з сучасним браузером і ОС (Windows, macOS, Linux);
- Мобільні пристрої: смартфони на базі Android 9+ або iOS 13+;
- Планшети: Android або iPadOS з актуальними версіями браузерів.

Приклади пристроїв.

- ПК/ноутбук: Lenovo IdeaPad, MacBook Air, Dell Inspiron, HP Pavilion;
- Смартфон: iPhone 12/13/14, Samsung Galaxy S21/S22, Google Pixel 6;
- Планшет: iPad (8-го покоління і вище), Samsung Galaxy Tab A/S.

Це забезпечує кросплатформенність системи та її доступність для широкого кола користувачів. Усі функціональні можливості доступні як із десктопів, так і з мобільних пристроїв, завдяки адаптивному дизайну та підтримці сучасних вебтехнологій.

4.2 Тестування та опис роботи системи

У процесі розробки програмного забезпечення тестування здійснювалося на кількох етапах з використанням різних підходів і інструментів. Початкове тестування проводилося безпосередньо під час написання коду за допомогою вбудованих засобів відлагодження. Основним інструментом виступала функція dd() (dump and die), яка дозволяла наочно перевіряти проміжні значення змінних, структуру масивів та об’єктів, стан сесій, запити до бази даних тощо. Це дозволяло оперативно виявляти та усувати логічні помилки на ранніх етапах реалізації функціональності.

Також під час розробки активно використовувались інструменти середовища розробки PhpStorm, яке забезпечує широкі можливості для аналізу коду, статичної перевірки, трасування та контролю за виконанням програмних інструкцій. Інтеграція PhpStorm з системами контролю версій, середовищем

виконання PHP та базами даних значно полегшила процес локального тестування та верифікації логіки застосунку.

Після завершення реалізації основного функціоналу було проведено мануальне (ручне) тестування усіх розділів адміністративної панелі.

Перевірялися такі аспекти, як.

- коректне функціонування форм створення, редагування та видалення даних;
- відповідність елементів інтерфейсу очікуваній поведінці (наприклад, наявність валідації, повідомлень про помилки, підтвердження дій);
- правильність навігації між сторінками;
- робота фільтрів, сортування та пагінації;
- доступність функцій лише для авторизованих користувачів з відповідними правами.

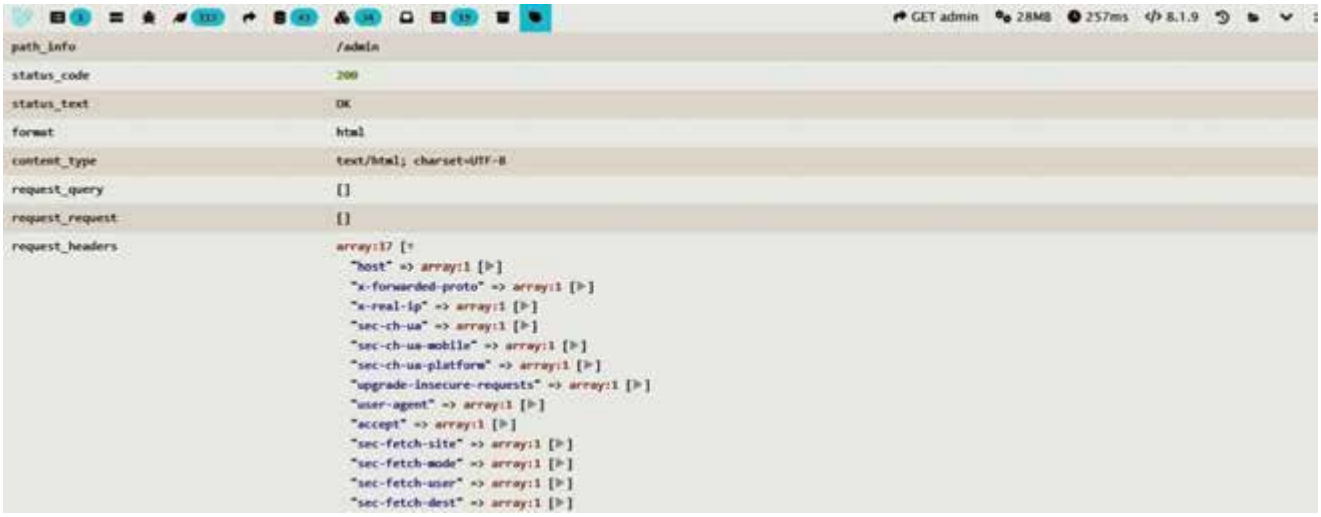

Рис. 31.1 – Laravel debugbar панель

Також при тестуванні сайту на Laravel значну допомогу надала панель Laravel Debugbar. Це інструмент для налагодження, який автоматично

виводиться на фронтенді під час розробки. Він інтегрується з Laravel і надає розробнику візуальну інформацію про ключові аспекти роботи застосунку. Це суттєво пришвидшує виявлення проблем.

Таким чином, проведене поетапне тестування дозволило виявити та усунути основні логічні, технічні та інтерфейсні помилки ще до розгортання проєкту на сервері. Результати тестування підтвердили функціональну придатність реалізованої системи в межах поставлених завдань.

Інструкція користувача до адміністративної панелі. Авторизація та ролі.

Вхід до системи здійснюється через форму авторизації, де потрібно ввести логін і пароль. Після успішного входу користувач отримує доступ до функціональності відповідно до своєї ролі. Інформація для доступу до системи для кожної ролі наведена в таблиці 4.

Базові ролі користувачів та облікові дані для доступу до системи

Таблиця 4

Роль	**Логін**	**Email**	**Пароль**
Super Admin	admin	admin@mail.com	admin
Main Editor	main-editor	main@mail.com	main-editor
Editor	editor	editor@mail.com	editor
Content Manager	content-manager	content@mail.com	content-manager

Відповідно кожна роль має різний рівень доступу до ресурсів системи. Перелік доступності модулів для тієї чи іншої ролі можна побачити в таблиці 5.

Перелік ролей та їх доступ до модулів

Таблиця 5

Роль	**Значення**	**Доступ до модулів**
superAdmin	1000	Усі модулі (*)
mainEditor	900	Усі модулі окрім configs
editor	500	dashboard, news, photo-warehouse, tag, video
contentManager	100	banner, constants, dashboard, faq, forms, languages, pages, photo-warehouse, photo_gallery, seo, tag

При спробі користувача зайти ресурс який йому не належить система відмовить у доступі та поверне HTTP-код відповіді 403 (Forbidden) (див. рис. 31.2).

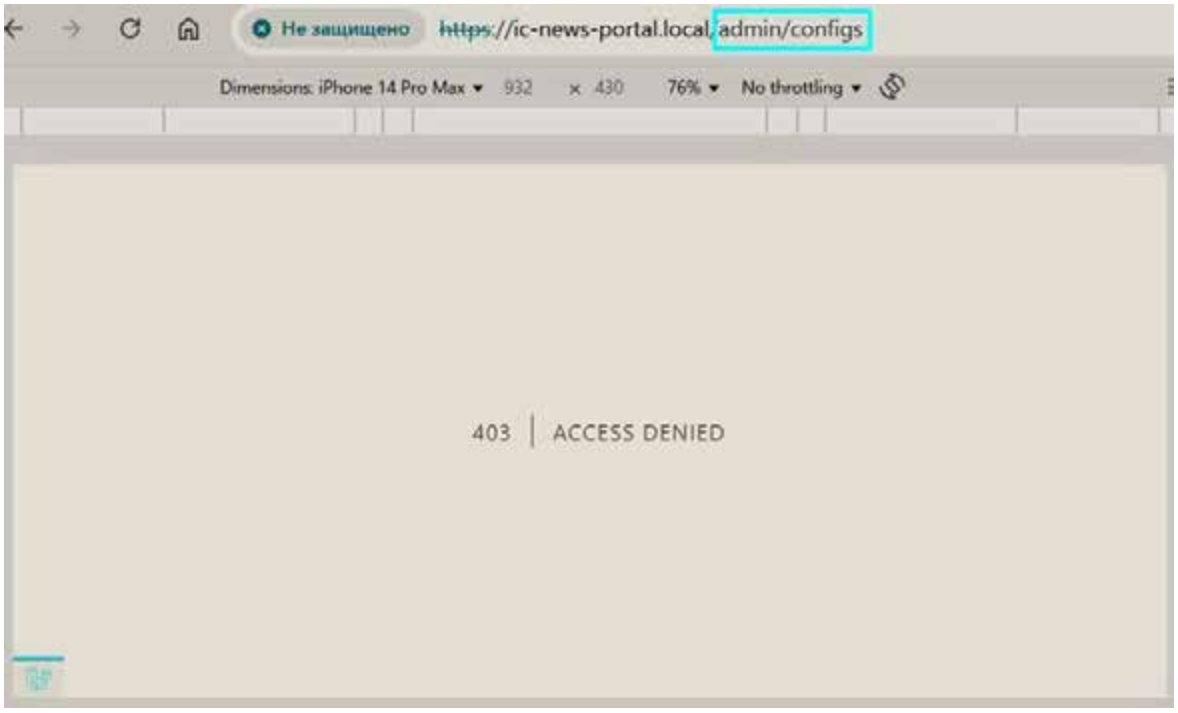

Рис. 31.2 – Access Denied модуль конфігурація для користувача редактор

Для кожного користувача власна область видимості вкладок, тобто користувачі не бачать вкладки та програмні модулі які їм не належать (див. рис 32).

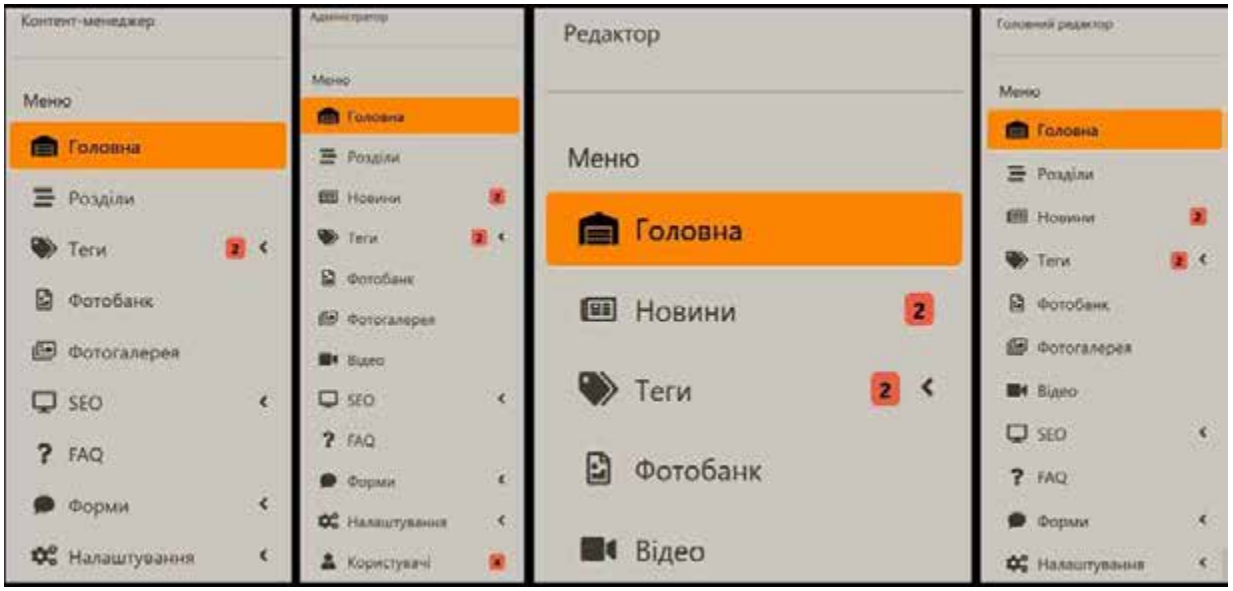

Рис. 32 – Порівняння області видимості вкладок для різних ролей

Основні вкладки системи.

- Головна — відображення статистики, дій користувачів, коротких зведень, швидкий доступ до основних модулів;
- Розділи — створення, редагування та ієрархічне впорядкування сторінок сайту, підключення модулів до сторінок;
- Новини та теги — додавання, редагування, видалення новин, класифікація новин за допомогою тегів, вивантаження матеріалів у візуальному редакторі;
- Фотобанк та фотогалерея — завантаження й зберігання зображень у репозиторії, формування тематичних галерей з описом і сортуванням;
- Форми — конструктор форм із можливістю налаштування полів, збір та перегляд запитів користувачів;
- Налаштування (configs, constants, languages, seo, banner, faq) — системні параметри (тільки для superAdmin), глобальні текстові змінні, мультимовна підтримка, керування мета-тегами сторінок, банери на сайті, питання та відповіді;
- Користувачі (users) — додавання, редагування користувачів та керування ролями, забезпечення контролю доступу до модулів згідно з призначеною роллю.

Далі на рис. 33, 34.1, 34.2, буде наведено скрін-каст інструкцію того як наповнюється основна сутність системи “Новина”. При роботі з іншими розділами можна орієнтуватися на логіку, використану для новини, так як для усіх розділів використані однакові елементи керування та дизайнерські рішення наповнення контенту.

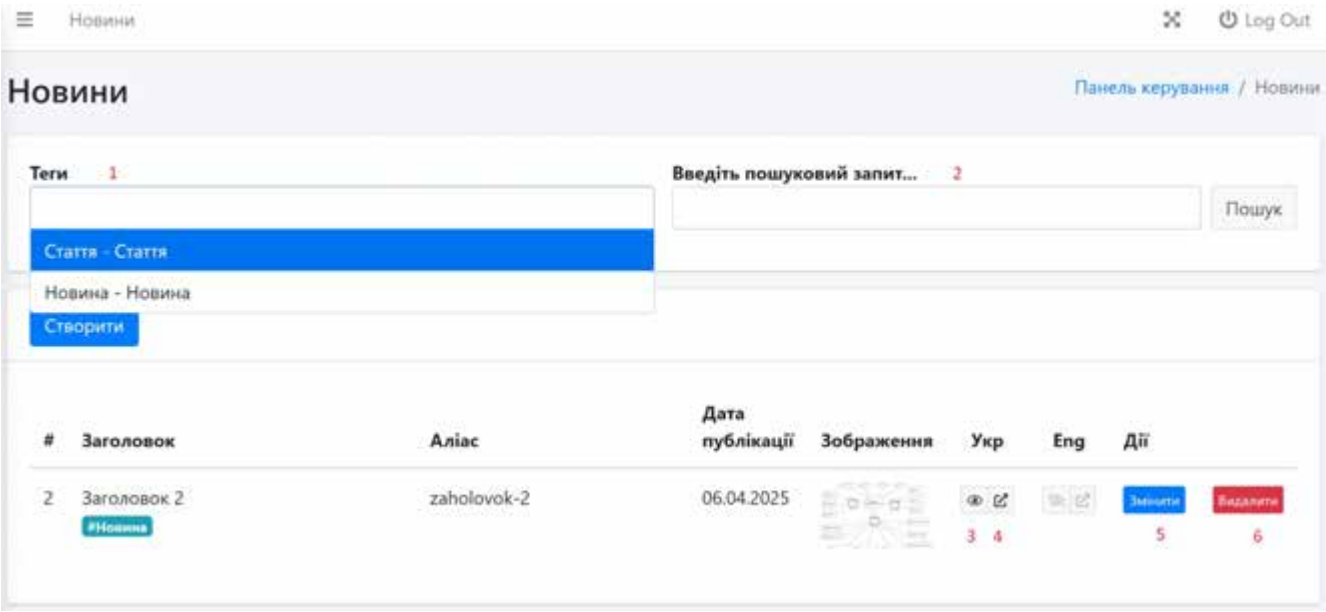

Рис. 33 – Список новин

Сторінка списку – відображає всі новини у вигляді таблиці. Тут можна: шукати, фільтрувати, переглядати, редагувати або видаляти новини, бачити статус мовних версій, швидко переходити до редагування потрібного елементу тощо.

1. Теги – фільтр за тегами: можна вибрати тип новини (наприклад, “Стаття” або “Новина”), щоб відібрати відповідні записи;
2. Пошук – поле для пошуку новин за ключовими словами;
3. Перегляд української версії – іконка “очі” відкриває перегляд новини українською мовою;
4. Перегляд англійської версії – іконка “перекреслене око” означає, що англійська версія ще не створена;
5. Змінити – кнопка для редагування наявної новини;
6. Видалити – кнопка для видалення новини.

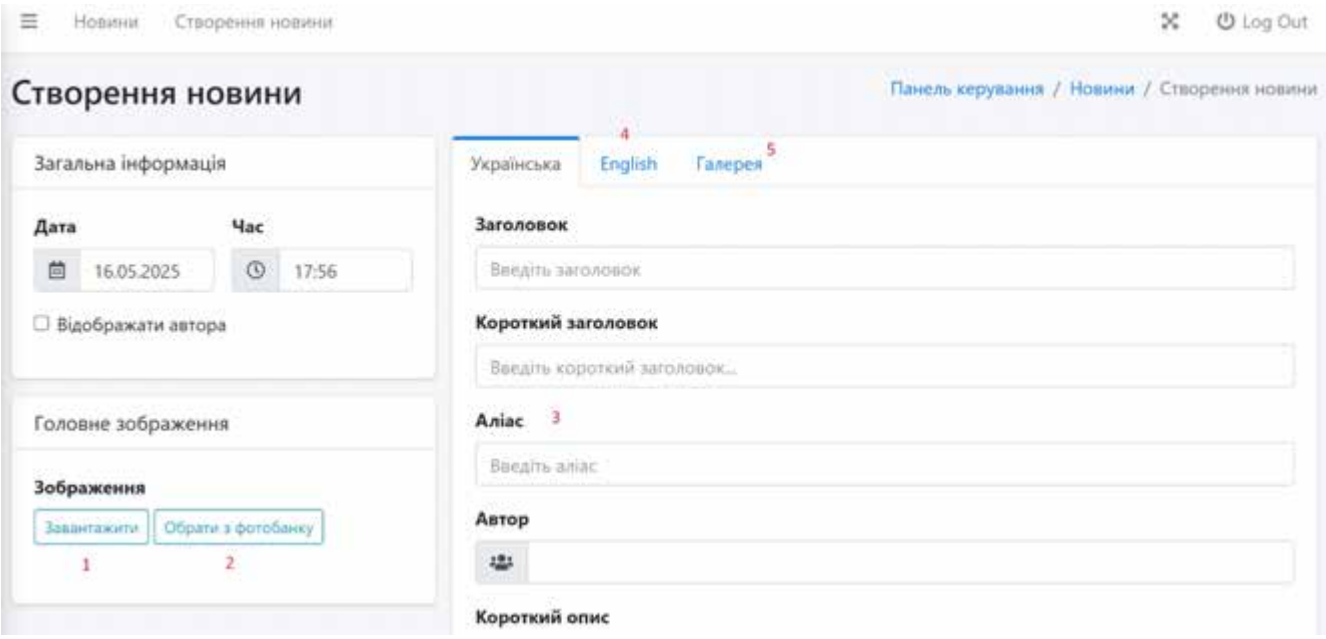

Рис. 34.1 – Створення/редагування новини (частина 1)

Сторінка елементу (редагування/створення новини) – дозволяє створити або змінити новину. Тут можна: завантажити зображення, ввести тексти українською та англійською мовами, додати теги, SEO-дані та галерею, керувати URL-адресою (аліасом) та публікацією.

1. Завантажити зображення – кнопка для завантаження головного зображення з комп’ютера;
2. Обрати з фотобанку – вибір зображення з наявної бази зображень;
3. Аліас – поле для введення унікального посилання (URL-ідентифікатора) новини;
4. Перемикання мов – вкладки для введення вмісту українською або англійською;
5. Галерея – вкладка для завантаження додаткових зображень до новини.

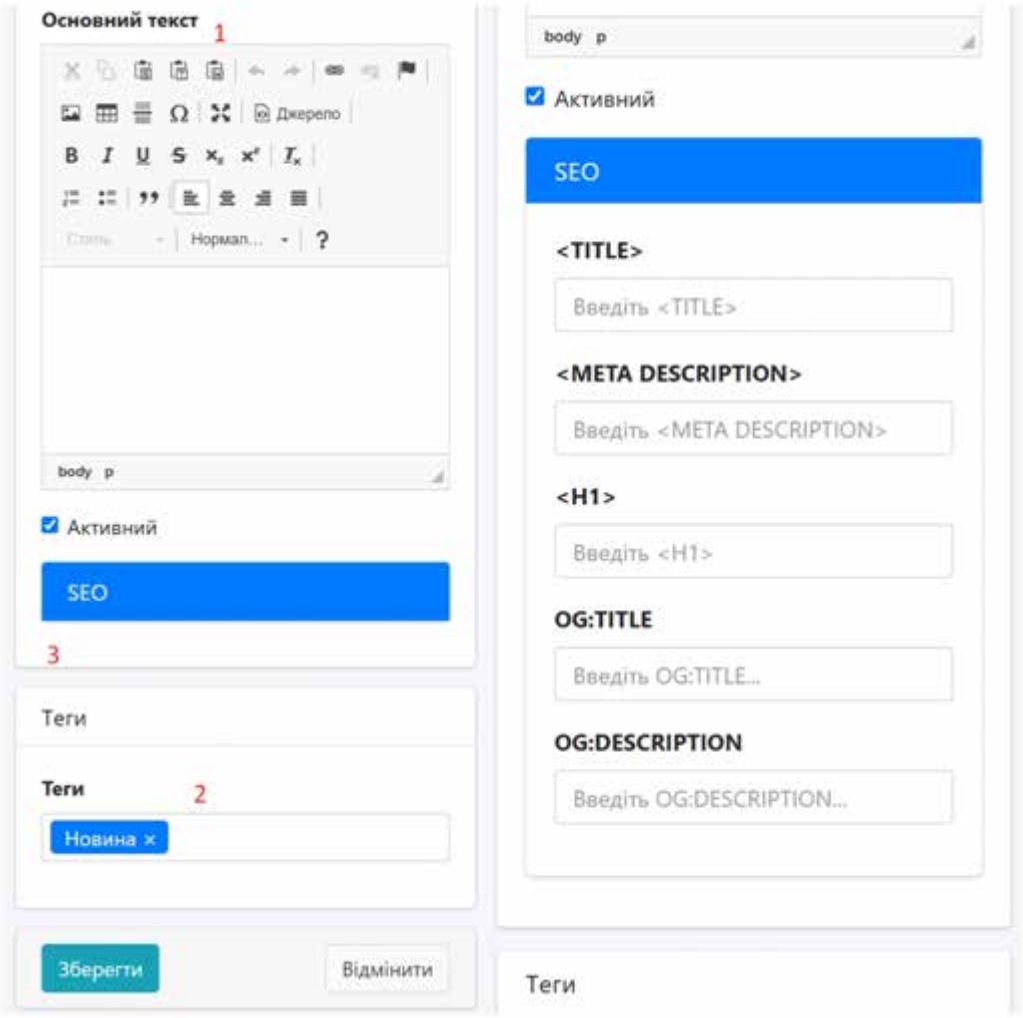

Рис. 34.2 – Створення/редагування новини (частина 2)

1. Основний текст – редактор для введення основного вмісту новини;

2. Теги – введення або вибір тегів, за якими класифікується новина;

3. SEO – відкриває розділ для введення метаданих SEO (теги <title>, meta description, заголовок H1, OG:Title та OG:Description).

Таким чином, було розглянуто основні можливості сторінки списку та сторінки редагування новин. Інтерфейс дозволяє керувати новинами — створювати, редагувати, фільтрувати, публікувати й видаляти. Для інших сутностей керування реалізовано за аналогічним принципом, тому можна орієнтуватися на функціонал новин.

4.3 Склад інсталяційного пакету та рекомендації щодо розгортання

Для кінцевого користувача (тобто відвідувача сайту) не потрібно встановлювати жодного додаткового програмного забезпечення. Усе, що потрібно — це наявність сучасного веб-браузера та стабільного підключення до інтернету.

Для розробників або системних адміністраторів (див. рис. 35).

Для того щоб система запрацювала на реальному сервері (хостингу), необхідно виконати такі кроки:

Підготувати хостинг-середовище, вибрати та зареєструвати доменне ім’я (наприклад: ic-newsportal.com) у будь-якого з реєстраторів.

Придбати хостинг або орендувати VPS-сервер із підтримкою.

- PHP не нижче версії 8.1;
- бази даних MySQL або MariaDB;
- вебсервера Apache або Nginx;
- встановленого Composer (для керування залежностями Laravel);
- можливості встановлення SSL-сертифікатів.

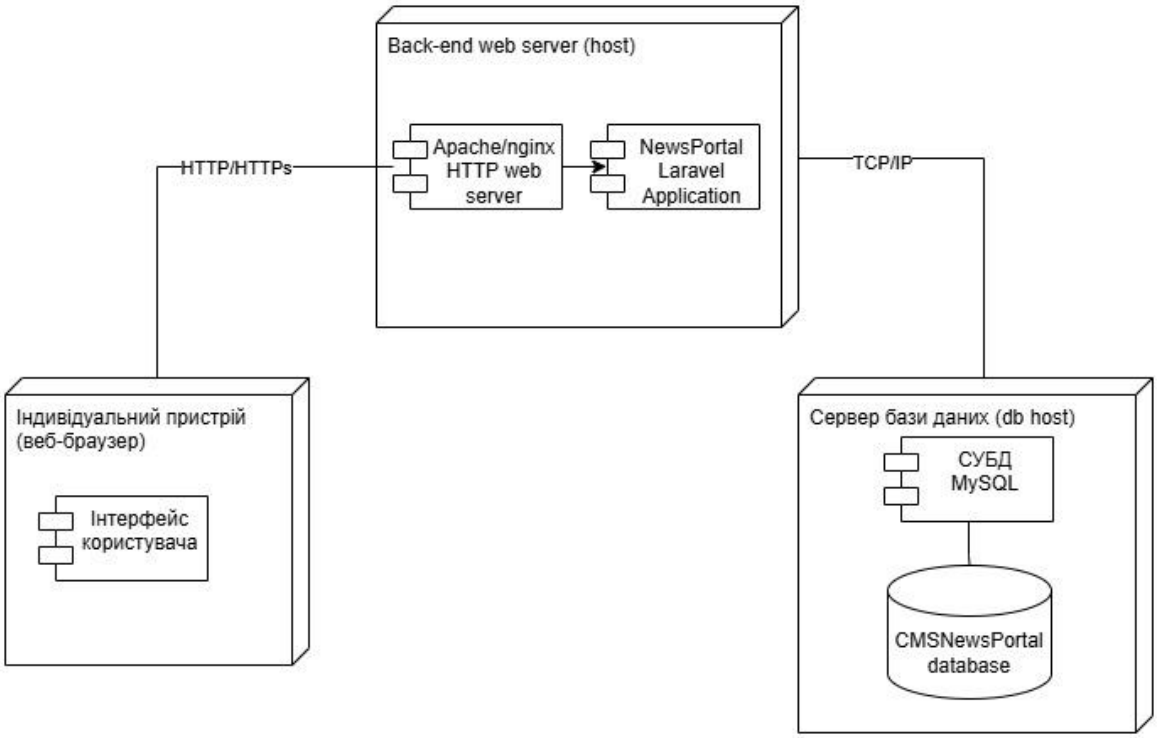

Рис. 35 – Діаграма розгортання системи

Створити базу даних на хостингу через панель управління або консоль.

Завантажити архів із програмним кодом (інсталяційний пакет) через FTP/SFTP або git-клонування у вказану кореневу директорію на хостингу, наприклад: /var/www/ic-newsportal. Налаштувати вебсервер на вказану директорію (наприклад, підключити домен до директорії public/, яка є точкою входу в Laravel-додаток).

Налаштування середовища, відредагувати файл .env, вказавши.

- дані підключення до бази даних (DB_HOST, DB_PORT, DB_DATABASE, DB_USERNAME, DB_PASSWORD);
- APP_URL з доменним іменем сайту;
- ключ програми (php artisan key:generate), якщо потрібно.

Наступний крок, це виконати розгортання бази даних, запустити Laravel-міграції: php artisan migrate. Якщо передбачено заповнення тестовими або стартовими даними, виконати сидери: php artisan db:seed.

Підключення додаткових сервісів, налаштувати SSL-сертифікати (наприклад, через Let's Encrypt) для захищеного з'єднання через HTTPS.

Підключити Google Analytics для збору аналітики по користувачах сайту.

Поради по підключенню Google Search Console.

- підтвердити права на домен;
- додати сайт до індексації;
- подати карту сайту (sitemap.xml), якщо вона згенерована автоматично або вручну.

Таким чином, процес інсталяції проєкту досить стандартний для Laravel-додатків, але потребує певних технічних знань з адміністрування серверів або хостингу. Усі кроки бажано виконувати в точній послідовності, щоб уникнути помилок під час розгортання.

ВИСНОВКИ

У ході виконання дипломного проєктування було здійснено повний цикл розробки програмного забезпечення для веборієнтованої інформаційної системи, призначеної для організації та управління публікаціями новинного порталу.

Реалізація проєкту базувалась на архітектурі MVC із використанням фреймворку Laravel та бази даних MySQL. Це дозволило забезпечити високу гнучкість, масштабованість і підтримку сучасних вимог до системи керування контентом.

Серед основних функціональних можливостей варто виділити підтримку багатомовності, зокрема української та англійської мов, що розширює потенціал залучення міжнародної аудиторії. Також реалізовано можливість категоризації, тегування, редагування новинного контенту, додавання мультимедійних матеріалів (зображення, відео), та управління публікаціями. Вбудований SEO-модуль дозволяє підвищити видимість ресурсу у пошукових системах, а генерація XML-карти сайту автоматизує процес індексації контенту.

Іншою вагомою складовою стало забезпечення високого рівня безпеки через IP-restricted авторизацію та розмежування прав доступу за ролями користувачів. В системі реалізовано чітку ієрархію доступу: контент-менеджери мають змогу створювати й редагувати публікації, редактори — працювати із тегами та корекцією текстів, головний редактор — затверджувати матеріали, а адміністратор — управляти технічною конфігурацією, обліковими записами та API-ключами.

Особливу увагу було приділено зручності взаємодії з інтерфейсом. Сайт адаптовано до використання як на настільних комп’ютерах, так і на мобільних пристроях із сенсорним введенням, що сприяє кращій доступності ресурсу для широкої аудиторії. Інтерфейс системи інтуїтивно зрозумілий і візуально привабливий, що сприяє ефективному виконанню робочих задач.

Серед додаткових функцій варто зазначити наявність конструктора форм для збору зворотного зв’язку, що дозволяє оперативно взаємодіяти з відвідувачами сайту, а також використання API перекладачів, що автоматизує процес локалізації текстових матеріалів.

Завдяки розробці цієї системи стало можливим не лише спростити процес публікації новин, а й впровадити ефективну організацію командної роботи у редакції. Впровадження ролей і розподілу доступу значно знижує ризик людських помилок та сприяє дотриманню редакційної політики. Водночас система залишається достатньо гнучкою для майбутньої модернізації та адаптації під змінні вимоги.

Завершальним етапом стало формування структури для розгортання проєкту на хостингу. Було описано порядок встановлення: завантаження проєкту на сервер, розгортання бази даних, підключення змінних середовища у .env, запуск міграцій та сидів для створення початкової структури БД, налаштування SSL-сертифікату, Google Analytics, robots.txt та засобів для індексації у пошукових системах.

Таким чином, дипломний проєкт відповідає початковому технічному завданню, реалізує визначену програмну архітектуру, містить усі необхідні компоненти для подальшого розгортання та підтримки, а також створює платформу для можливого подальшого масштабування та доповнення новими модулями. Поставлені на початку роботи завдання були виконані у повному обсязі.

Перспективи розвитку програмної системи включають розширення функціональності за рахунок інтеграції з додатковими API (наприклад, соціальні мережі, зовнішні аналітичні сервіси). У разі підвищеного навантаження система може бути масштабована через перехід на мікросервісну архітектуру. Додатково розглядається підтримка адміністративної аналітики на основі BI-інструментів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lucidchart – Introducing Types of UML Diagrams – [Електронний ресурс] – Режим доступу: https://www.lucidchart.com/blog/types-of-UML-diagrams
2. Creately – Learn About All 14 Types of UML Diagrams – [Електронний ресурс] – Режим доступу: https://creately.com/blog/diagrams/uml-diagram-types-examples/
3. W3Schools українською – SQL Підручник – [Електронний ресурс] – Режим доступу: https://w3schoolsua.github.io/sql/sql_intro.html
4. Laravel Documentation – [Електронний ресурс] – Режим доступу: https://laravel.com/docs/11.x/readme
5. AdminLTE Documentation – [Електронний ресурс] – Режим доступу: https://adminlte.io/docs/3.0/
6. Webpack Documentation – [Електронний ресурс] – Режим доступу: https://webpack.js.org/
7. Sass Documentation – [Електронний ресурс] – Режим доступу: https://sass-lang.com/documentation/
8. PHP Manual – [Електронний ресурс] – Режим доступу: https://www.php.net/docs.php
9. MySQL Documentation – [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/
10. Apache HTTP Server Documentation – [Електронний ресурс] – Режим доступу: https://httpd.apache.org/docs/
11. Google Search Console Tools – [Електронний ресурс] – Режим доступу: https://search.google.com/search-console/about
12. Основи збірки Frontend проектів на Webpack – [Електронний ресурс] – Режим доступу: https://victoriaweb.me/webpack-basics/

13. PHP 8.1: Що нового та що змінено – [Електронний ресурс] – Режим доступу: https://promoter.net.ua/index.php/articles/php-81-shho-novogo-ta-shho-zmineno.html

14. Основи Webpack 2 | DevZone – [Електронний ресурс] – Режим доступу: https://devzone.org.ua/post/osnovy-webpack-2

15. Beginner's Guide – nginx – [Електронний ресурс] – Режим доступу: https://nginx.org/en/docs/beginners_guide.html

16. [GA4] Set up Analytics for a website and/or app – Google Help – [Електронний ресурс] – Режим доступу: https://support.google.com/analytics/answer/9304153?hl=en

17. ukrainian.md – alexeymezenin/laravel-best-practices – GitHub – [Електронний ресурс] – Режим доступу: https://github.com/alexeymezenin/laravel-best-practices/blob/master/ukrainian.md

18. PhpStorm Documentation – JetBrains – [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/help/phpstorm/getting-started.html

19. Best WordPress Alternatives in 2025 – WPBeginner – [Електронний ресурс] – Режим доступу: https://www.wpbeginner.com/showcase/wordpress-competitors-23-popular-alternatives-to-wordpress/

20. Web API Design Best Practices – Azure Architecture Center – [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

Додаток А

Представлення програмного коду окремих компонентів системи

Сторінок: 9

Київ 2025

Контролер модуля новин.

```
<?php
/**
 * Class NewsController
*/
class NewsController extends BaseAdminController
{
    protected TagRepository $tagRepository;
    /**
     * NewsController constructor.
     */
    public function __construct()
    {
        $this->baseTemplatePath = 'news.admin';
        $this->repository       = resolve(NewsRepository::class);
        $this->tagRepository    = resolve(TagRepository::class);
    }
    /**
     * Render the list on news in admin.
     *
     * @Route("/news", methods={GET}, name="admin.news.list")
     * @param  Request $request
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        $models = $this->repository->listNewsByFilters($request->all());

        $tags = $this->tagRepository->getTags();

        $data = [
            'models' => $models,
            'tags' => $tags
        ];

        return $this->renderAdminWithBaseData("{$this->baseTemplatePath}.list",$data);
    }
    /**
     * @return Response
     * @throws \Psr\SimpleCache\InvalidArgumentException
     */
    public function create()
    {
```

```
        $dataToView['model']    = new News();
        $dataToView['tagsData'] = Tags::with(['activeTranslation:tag_id,title'])->find([
            Tags::NEWS_TAG_ID
        ], ['id']);

        return $this->renderAdminWithBaseData("{$this->baseTemplatePath}.create",
$dataToView);
    }
    /**
     * Save new news
     *
     * @Route('admin/news/create', methods={POST}, name="admin.news.store")
     * @param  CreateNewsRequest $request
     *
     * @return View
     */
    public function store(CreateNewsRequest $request)
    {
        $form = $request->except(['_token']);

        $form['news']['show_author'] = isset($form['news'] ['show_author']) ? 1 : 0;
        $form['i18n']               = NewsLocalizedFormParser::parse($form['i18n']);
        $form['seo']                = NewsSEOParser::parse($form['seo']);
        $form['newsTagsGroup']      = NewsTagsGroupParser::parse($form['newsTagsGroup']);

        $isCreated = NewsSaver::run($form);

        if($isCreated) {
            return redirect(route('admin.news.list'))->with('success',
__('admin/news.creating_success'));
        } else {
            return redirect()->back()
                        ->with('error', __('admin/news.creating_error'))
                        ->withInput();
        }
    }
```

```
/**
     * @param $id
     * @return Response
     * @throws \Psr\SimpleCache\InvalidArgumentException
     */
    public function edit($id)
    {
        $dataToView['model'] = News::with([
                'translations',
                /*'images' => function($q) {
                    return $q->with('translations');
                },*/
                'contents',
                'seo',
                'news2MainTagGroupId'
            ]
        )->findOrFail($id);

        $dataToView['tagsData'] = Tags::with(['activeTranslation:tag_id,title'])
                                    ->join('news_2_tags AS n2t', 'n2t.tag_id', '=', 'tags.id')
                                    ->where('n2t.news_id', $id)
                                    ->select(['tags.id'])
                                    ->get();
        $dataToView['tagsData'] = $dataToView['tagsData'] ??
Tags::with(['activeTranslation:tag_id,title'])->find([
            Tags::NEWS_TAG_ID
        ], ['id']);
        return $this->renderAdminWithBaseData("{$this->baseTemplatePath}.edit", $dataToView);
    }
    /**
     * @param  EditNewsRequest $request
     * @param   $id
         * @return \Illuminate\Http\RedirectResponse|\Illuminate\Routing\Redirector
     * @throws \Throwable
     */
    public function update(EditNewsRequest $request, $id)
    {
        $model     = News::findOrFail($id);
        $formData  = $request->validated();
        $isUpdated = false;

        DB::transaction(function() use (&$model, &$formData, &$isUpdated) {
            $service = new NewsUpdaterService($model);

            $service->update($formData);
```

```
            $service->updateSeoData($formData);
            $service->updateLocalizedTextData($formData);
            $service->updateChainWithTags($formData);
            $service->updateChainWithTagGroups($formData);
            $service->updateMainPhotoWithData($formData);

            $isUpdated = true;
        });

        if($isUpdated) {
            return redirect(route('admin.news.list'))->with('success',
__('admin/news.updating_success'));
        } else {
            return redirect()->back()->with('error', __('admin/news.updating_error'))->withInput();
        }
    }
    /**
     * Generate news entity.
     *
     * @Route('admin/news/delete/{id}', methods={DELETE}, name="admin.news.delete")
     * @param  string $id
     * @return string
     */
    public function delete($id)
    {
        if(News::destroy($id)) {
            $newsImage = NewsImage::where('news_id', $id)->first();

            if(isset($newsImage)) {
                $newsImage->delete();
            }

            return redirect(route('admin.news.list'))->with('success',
__('admin/news.deleting_success'));
        }

        return redirect(route('admin.news.list'))->with('error', __('admin/news.deleting_error'));
    }
```

```
/**
     *  Get news by id via ajax.
     * @Route('admin/news/{id}', methods={GET}, name="admin.news.show")
     * @return \Illuminate\Contracts\Routing\ResponseFactory|\Illuminate\Http\Response
     */
    public function show(Request $request, $id)
    {
        if($request->ajax()) {
            $model = (new NewsRepository())->getModel($id);
            return response()->json($model);
        } else {
            return response('Method not allowed', 403);
        }
    }
```

Форма редагування/створення новини.

```
@extends('layouts.admin.main')
@section('title',  __('admin/news.news_edit'))
@section('content_header')
    @php($activeTranslation = $model->translations->where('lang_id', $mainLangId)->first())
    @include('layouts.admin.inc.breadcrumbs', $breadcrumbs = [
        [
            'label' =>  __('admin/news.title'),
            'url' => route('admin.news.list')
        ],
        [
            'label' =>  __('admin/news.news_edit') . ' "' . (isset($activeTranslation) &&
isset($activeTranslation->title) ? $activeTranslation->title : '') . '"',
        ]
    ])
@endsection
@section('content')
    <form class="content form" action="{{ route('admin.news.update', ['id' => $model->id]) }}"
method="POST" enctype="multipart/form-data">
        @method('PATCH')
        <input type="hidden" name="entity_id" value="{{ $model->id }}">
        @include('news.admin.inc.form')
    </form>
@endsection
```

Laravel seeder для створення базових ролей.

```
<?php

use Illuminate\Database\Seeder;

class UserSeeder extends Seeder
{
   /**
    * Run the database seeds.
    * @return void
    */
   public function run()
   {
      DB::table('users')->insert([
         [
            'login'    => 'admin',
            'email'    => 'admin@mail.com',
            'password' => bcrypt('admin'),
            'role'     => 1000,
            'active'   => true,
            'created_at' => now(),
            'updated_at' => now(),
         ],
         [
            'login'    => 'main-editor',
            'email'    => 'main@mail.com',
            'password' => bcrypt('main-editor'),
            'role'     => 900,
            'active'   => true,
            'created_at' => now(),
            'updated_at' => now(),
         ],
         [
            'login'    => 'editor',
            'email'    => 'editor@mail.com',
            'password' => bcrypt('editor'),
            'role'     => 500,
            'active'   => true,
            'created_at' => now(),
            'updated_at' => now(),
         ],
         [
            'login'    => 'content-manager',
            'email'    => 'content@mail.com',
```

```
            'password' => bcrypt('content-manager'),
            'role'     => 100,
            'active'   => true,
            'created_at' => now(),
            'updated_at' => now(),
        ],
    ]);
  }
}
```

Handler для побудови карти сайту (sitemap:generate-cron)

```
class GenerateSitemap extends Command
{
  /**
   * The name and signature of the console command.
   * @var string
   */
  protected $signature = 'sitemap:generate-cron';
  /**
   * The console command description.
   * @var string
   */
  protected $description = 'Command description';
  protected $defaultLocale;
  protected $domain;
  protected $redirectRepository;
  protected $notIdsPageToSitemap = [];
  /**
   * Create a new command instance.
   * @return void
   */
  public function __construct()
  {
    $this->defaultLocale      = $mainLang->slug ?? '';
    $this->domain             = config('app.url');
    $this->redirectRepository = new RedirectRepository();

            parent::__construct();
  }
  public function handle()
  {
    $pagesData = (new PageSitemapRepository)->getActive($this->notIdsPageToSitemap);
    $newsData = (new NewsSitemapRepository)->getActive();

    $data = array_merge($pagesData, $newsData ?? []);
```

```
        $this->buildXML($data);
    }
    /** Generate XML
     * @param array $data
     * @return void
     */
    protected function buildXML(array $data): void
    {
        $langs = $this->getLangs();
        $chunks = array_chunk($data, 49000);
        foreach ($chunks as $index => $chunk){
            $base = '<?xml version="1.0" encoding="UTF-8"?><urlset
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"></urlset>';
            $xmlbase = new \SimpleXMLElement($base);
            $itemsRedirect = $this->prepareRedirectLinks($chunk);

            foreach ($chunk as $item) {
                if (isset($langs[$item['langId']])) {
                    $loc = $itemsRedirect[$item['loc']] ?? $item['loc'];

                    $row = $xmlbase->addChild("url");
                    $row->addChild("loc", $this->generateUrl($loc));
                    $row->addChild("lastmod", $item['lastmod']->format('c'));
                    $row->addChild("changefreq", $item['changefreq']);
                    $row->addChild("priority", $item['priority']);
                }
            }

            $filename = $index === 0 ? 'sitemap.xml' : 'sitemap'. ($index + 1) .'.xml';
            $xmlbase->saveXML(public_path() . '/' . $filename);
        }
    }
}
```

i18n переклади модуль Теги (для реалізація мультимовності)

Українська мова (ua/uk)

```
<?php return
['add_tag' => 'Додати тег',
'adding_exist_tag' => 'Пошук існуючих тегів',
'adding_new_tag' => 'Додавання нових тегів',
'already_exists' => 'Тег з такою назвою вже існує для мовної версії:',
'create' => 'Створити тег',
'creating' => 'Створення групи тегів',
'form_search_by_group' => 'За групою тегів',
'form_search_by_tag' => 'За назвою тега',
'info_communicate' => 'Використовуються для звя\'зку новин та відображення по группі-тега',
'invisible' => 'Невидимий',
'legend' => 'Позначення групп тегів. Клікніть на группу-тег для зміни',
'list' => 'Групи тегів',
'main_group_tag_full' => 'Головна группа тегів (для показу в хлібних крихтах)',
'news_count' => 'Новин',
'parent_group' => 'Група тегів',
'search_form_title' => 'Пошук',
'searching_adding_label' => 'Пошук/Додавання тегів',
'slug_tag' => 'Аліас',
'specify_title_for' => 'Вкажіть назву для мовної версії:',
'start_type_tag_name' => 'Почніть вводити назву тега...',
'tag' => 'Тег',
'tag_creating' => 'Створення тегу',
'tag_group_create' => 'Створити групу тегів',
'tag_updating' => 'Редагування тегу',
'tags' => 'Теги',
'type_name' => 'Введіть назву групи тегів',
'type_tag_name' => 'Введіть назву тега',
'unsorted_label' => 'Невідсортовані теги',
'updating' => 'Редагування групи тегів',
'visible' => 'Видимий'];
```

Англійська мова (en)

```
<?php return
['add_tag' => 'Add tag',
'adding_exist_tag' => 'Search for existing tags',
'adding_new_tag' => 'Adding new tags',
'already_exists' => 'A tag with this name already exists for the language version:',
'create' => 'Create a tag',
'creating' => 'Creating a tag group',
'form_search_by_group' => 'By tag group',
'form_search_by_tag' => 'By tag name',
'info_communicate' => 'Used to link news and tag group display',
'invisible' => 'Invisible',
'legend' => 'Tag group designations. Click on the tag group to change',
'list' => 'Tag groups',
'main_group_tag_full' => 'Main tag group (for display in breadcrumbs)',
'news_count' => 'News',
'parent_group' => 'Tag group',
'search_form_title' => 'Search',
'searching_adding_label' => 'Search / Adding tags',
'slug_tag' => 'Alias',
'specify_title_for' => 'Specify a name for the language version:',
'start_type_tag_name' => 'Start entering a tag name...',
'tag' => 'Tag',
'tag_creating' => 'Creating tag',
'tag_group_create' => 'Create a tag group',
'tag_updating' => 'Editing tag',
'tags' => 'Tags',
'type_name' => 'Enter a name for the tag group',
'type_tag_name' => 'Enter a tag name',
'unsorted_label' => 'Unsorted tags',
'updating' => 'Editing a group of tags',
'visible' => 'Visible'];
```

Додаток Б

Представлення коду SQL та Laravel migration для створення таблиць БД системи

Сторінок: 6

Київ 2025

SQL-запити для створення БД.

```
CREATE DATABASE IF NOT EXISTS news_portal_ic CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;

USE news_portal_ic;

CREATE TABLE IF NOT EXISTS mos_forms (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  title VARCHAR(255) NOT NULL,
  system_name VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS mos_form_fields (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  form_id BIGINT UNSIGNED NOT NULL,
  field_type VARCHAR(100) NOT NULL,
  input_name VARCHAR(255) NOT NULL,
  is_required tinyint(1) DEFAULT FALSE, -- BOOL
  priority SMALLINT DEFAULT 0,
  PRIMARY KEY (id),
  FOREIGN KEY (form_id) REFERENCES mos_forms(id) ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS mos_form_data (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  form_id BIGINT UNSIGNED NOT NULL,
  source_url VARCHAR(255),
  status VARCHAR(50),
  submitted_at DATETIME,
  PRIMARY KEY (id),
  FOREIGN KEY (form_id) REFERENCES mos_forms(id) ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS mos_form_data_fields (
```

```
  form_data_id BIGINT UNSIGNED NOT NULL,
  field_id BIGINT UNSIGNED NOT NULL,
  field_value TEXT,
  PRIMARY KEY (form_data_id, field_id),
  FOREIGN KEY (form_data_id) REFERENCES mos_form_data(id) ON DELETE CASCADE,
  FOREIGN KEY (field_id) REFERENCES mos_form_fields(id) ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS mos_pages (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  title VARCHAR(255),
  menu_title VARCHAR(255),
  alias VARCHAR(255),
  template VARCHAR(255),
  parent_id BIGINT UNSIGNED,
  priority SMALLINT,
  show_in_breadcrumbs tinyint(1), -- BOOL
  admin_only tinyint(1), -- BOOL
  show_in_sitemap tinyint(1), -- BOOL
  menu_position VARCHAR(255),
  redirect VARCHAR(255),
  is_active tinyint(1), -- BOOL
  PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS mos_module_blocks (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  page_id BIGINT UNSIGNED,
  handler VARCHAR(255),
  title VARCHAR(255),
  position SMALLINT,
  priority SMALLINT,
```

```
  template VARCHAR(255),
  is_active tinyint(1), -- BOOL
  PRIMARY KEY (id),
  FOREIGN KEY (page_id) REFERENCES mos_pages(id) ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS mos_photo_warehouses (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  path VARCHAR(255),
  width SMALLINT,
  height SMALLINT,
  alt_text VARCHAR(255),
  title VARCHAR(255),
  source VARCHAR(255),
  description TEXT,
  uploaded_at DATETIME,
  PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS mos_news (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  photo_id BIGINT UNSIGNED,
  title VARCHAR(255),
  short_title VARCHAR(255),
  summary TEXT,
  alias VARCHAR(255),
  author VARCHAR(255),
  content TEXT,
  publish_date DATE,
  publish_time TIME,
  is_active tinyint(1), -- BOOL
  PRIMARY KEY (id),
```

```
  FOREIGN KEY (photo_id) REFERENCES mos_photo_warehouses(id) ON DELETE SET NULL
);
CREATE TABLE IF NOT EXISTS mos_videos (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  photo_id BIGINT UNSIGNED,
  video_url VARCHAR(255),
  publish_date DATE,
  publish_time TIME,
  priority SMALLINT,
  title VARCHAR(255),
  is_active tinyint(1), -- BOOL
  PRIMARY KEY (id),
  FOREIGN KEY (photo_id) REFERENCES mos_photo_warehouses(id) ON DELETE SET NULL
);
CREATE TABLE IF NOT EXISTS mos_tags (
  id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  group_id BIGINT UNSIGNED,
  name VARCHAR(255),
  alias VARCHAR(255),
  is_active tinyint(1), -- BOOL
  PRIMARY KEY (id),
  FOREIGN KEY (group_id) REFERENCES mos_tag_groups(id) ON DELETE SET NULL
);
CREATE TABLE IF NOT EXISTS mos_seo_modules (
  owner_id BIGINT UNSIGNED NOT NULL,
  entity VARCHAR(255) NOT NULL, -- ENUM('page', 'news' і тд.)
  name VARCHAR(255),
  title VARCHAR(255),
  meta_description VARCHAR(255),
  meta_keywords VARCHAR(255),
```

```
  h1 VARCHAR(255),
  seo_text MEDIUMTEXT,
  robots_index tinyint(1), -- BOOL
  robots_follow tinyint(1), -- BOOL
  is_active tinyint(1), -- BOOL
  PRIMARY KEY (owner_id, entity)
);
```

Приклад таблиця для реалізації багато до багатьох:

```
CREATE TABLE IF NOT EXISTS mos_photo_gallery_2_warehouses (
  gallery_id BIGINT UNSIGNED,
  photo_id BIGINT UNSIGNED,
  PRIMARY KEY (gallery_id, photo_id),
  FOREIGN KEY (gallery_id) REFERENCES mos_photo_gallery(id) ON DELETE CASCADE,
  FOREIGN KEY (photo_id) REFERENCES mos_photo_warehouses(id) ON DELETE CASCADE
);
```

Laravel migration для створення таблиць.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
// Таблиця сторінок (розділи)
class CreatePagesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('pages', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->unsignedInteger('parent_id')->default(0)->index();
            $table->integer('depth')->default(0);
            $table->string('template')->default('default');
            $table->unsignedSmallInteger('no_link')->index()->default(0);
            $table->integer('priority')->default(0);
            $table->boolean('show_in_breadcrumb')->default(true)->index();
            $table->boolean('sitemap_html_view')->default(true);
            $table->boolean('restrict_access')->nullable(false)->default(false);
            $table->unsignedInteger('core_count')->index()->default(0);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('pages');
    }
}
```

Додаток В

Представлення деяких екранних форм на різних пристроях (Desktop/mobile/table), демонстрація адаптивності GUI

Сторінок: 4

Київ 2025

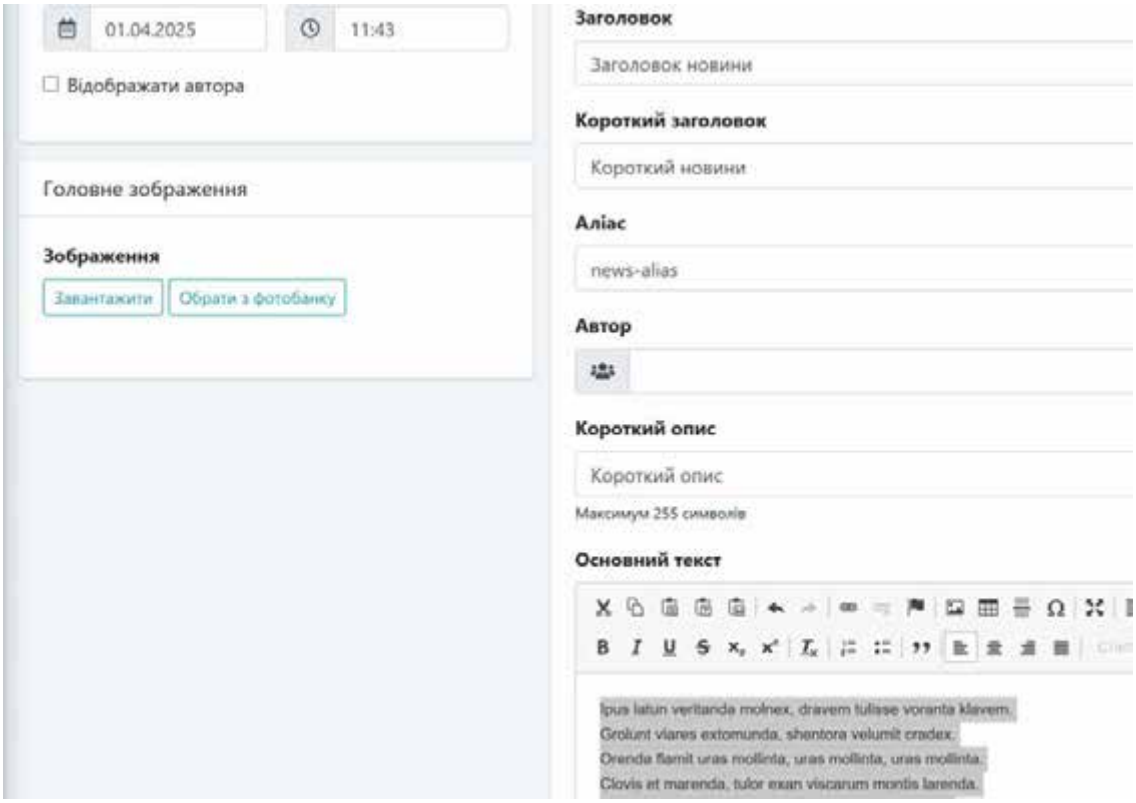

Рис. В.1 – Керування новиною ПК версія (20 дюймів екран)

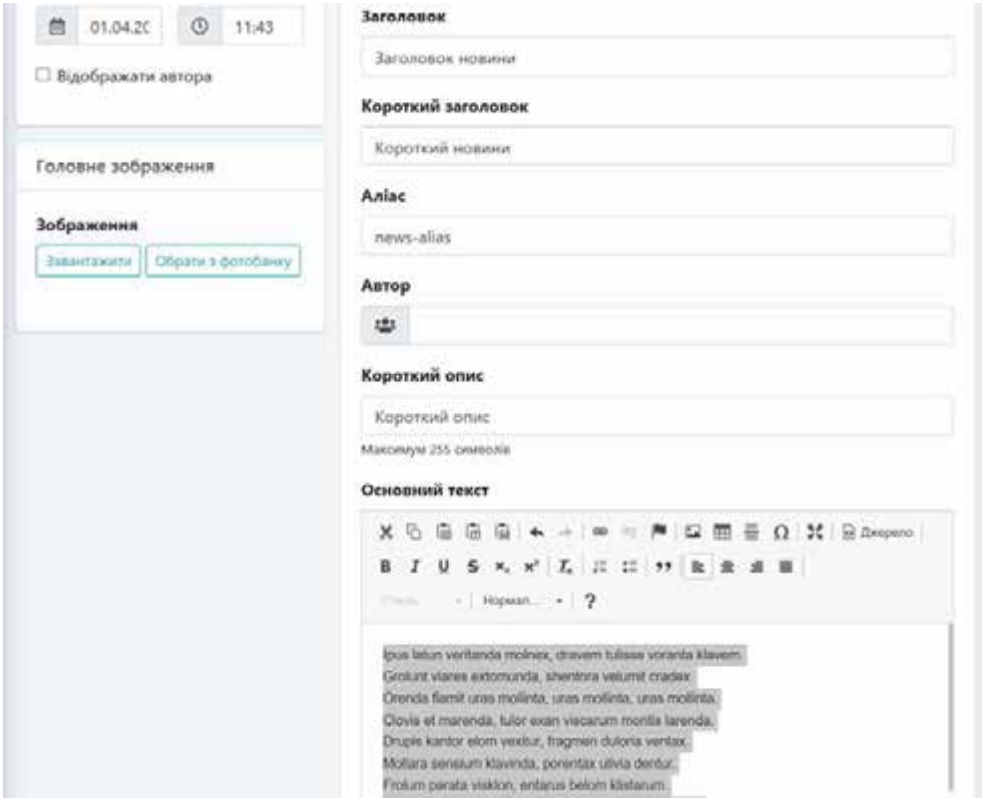

Рис. В.2 – Керування новиною Планшетна версія (iPad mini)

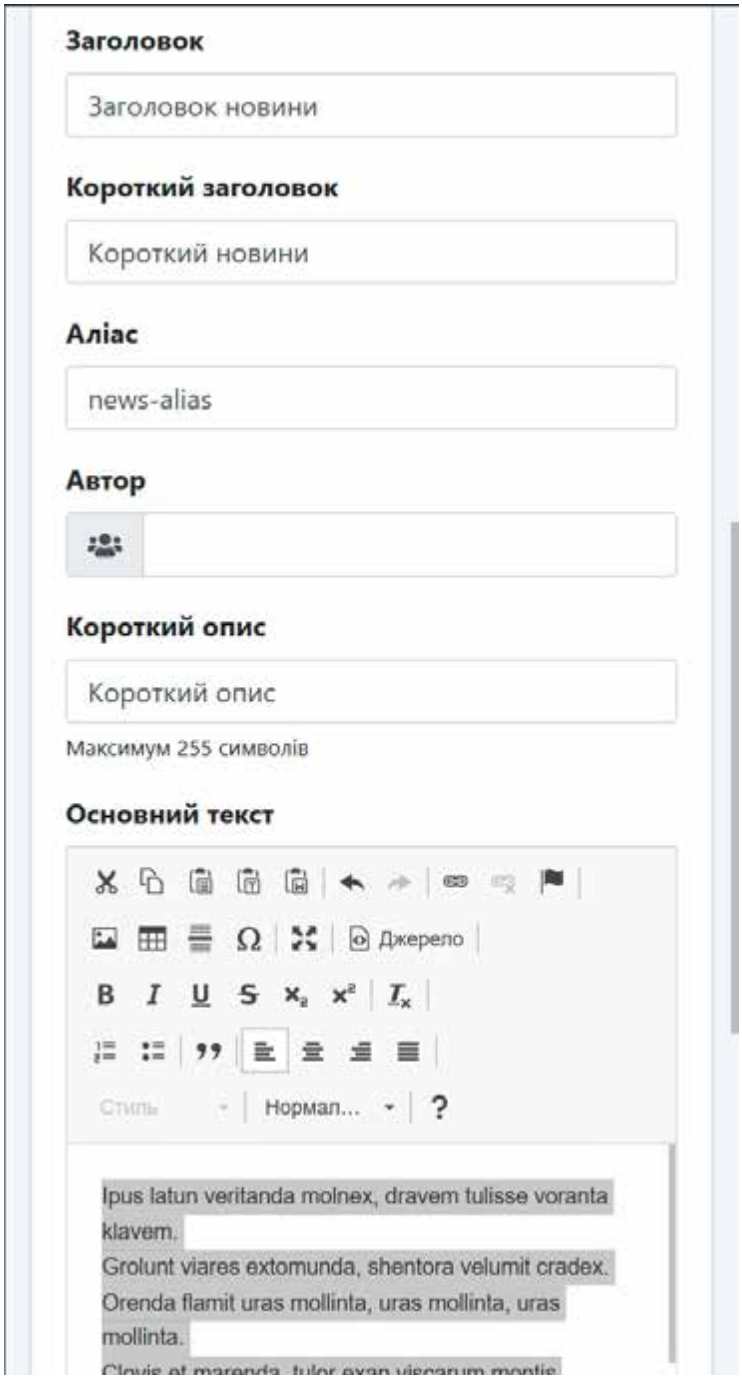

Рис. В.3 – Керування новиною Мобайл версія (iPhone 14 Pro)

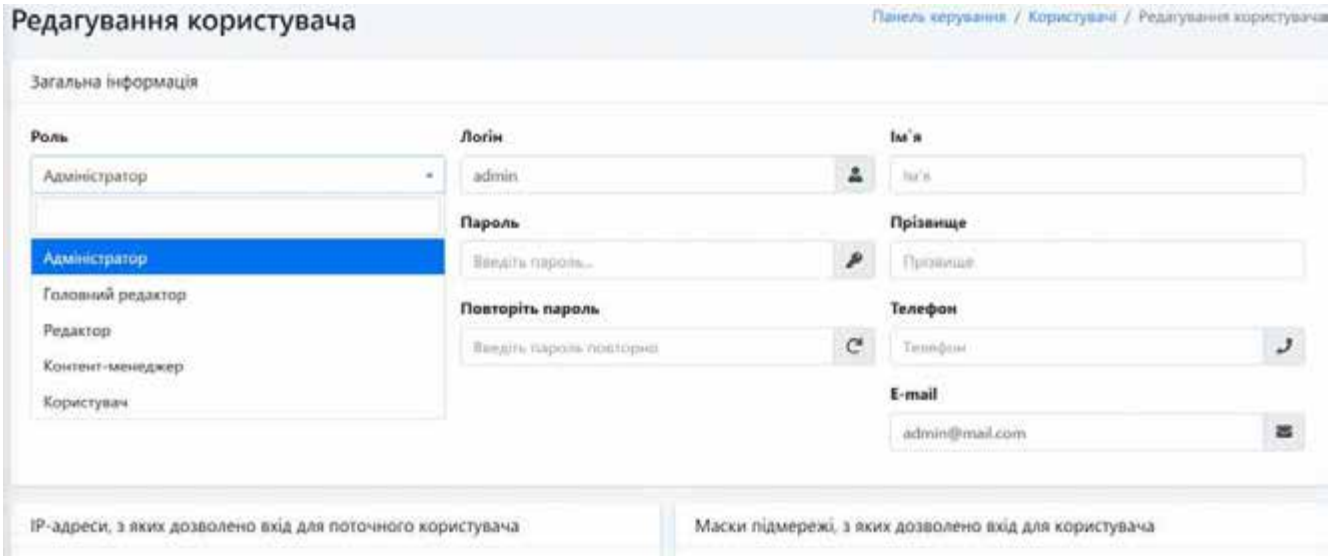

Рис. В.4 – Керування користувачами ПК версія (20 дюймів монітор)

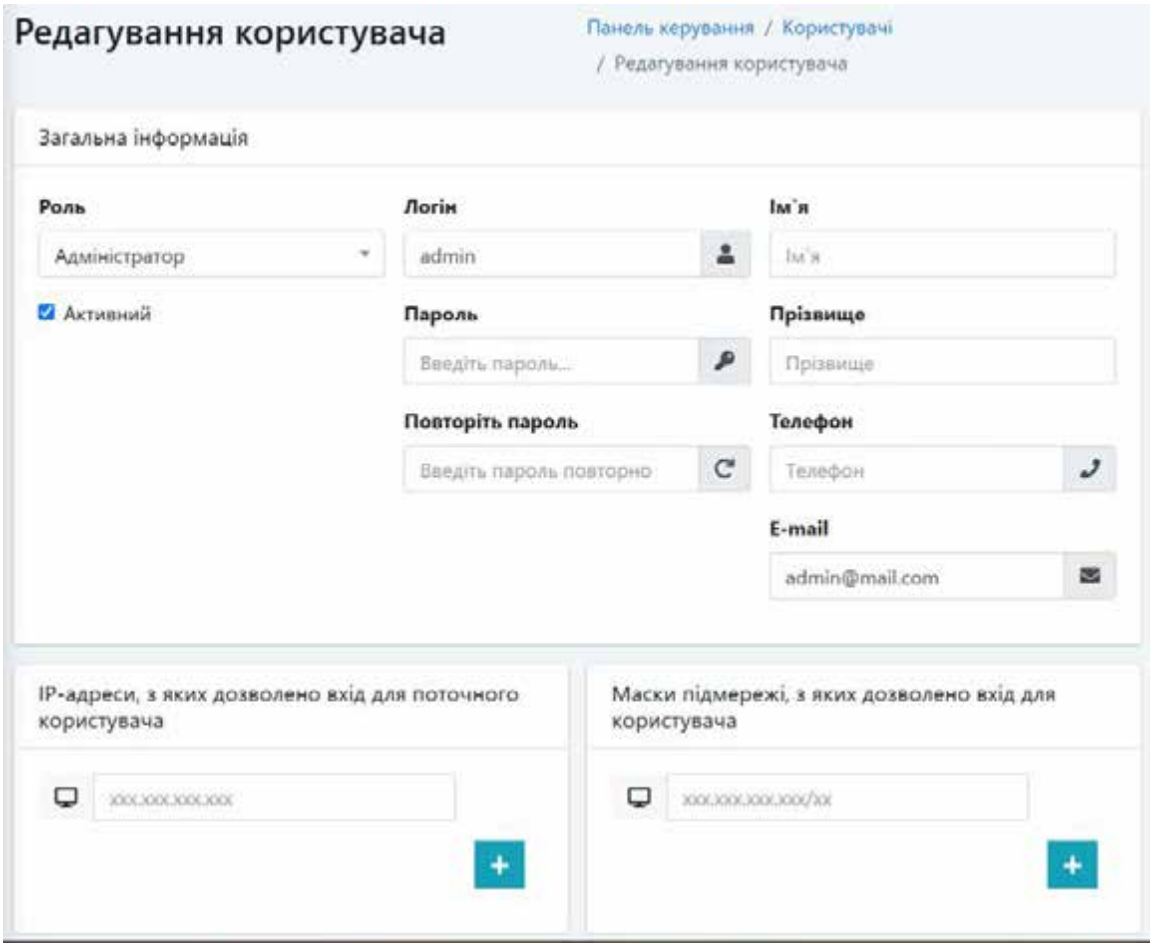

Рис. В.5 – Керування користувачами Планшетна версія (iPad mini)

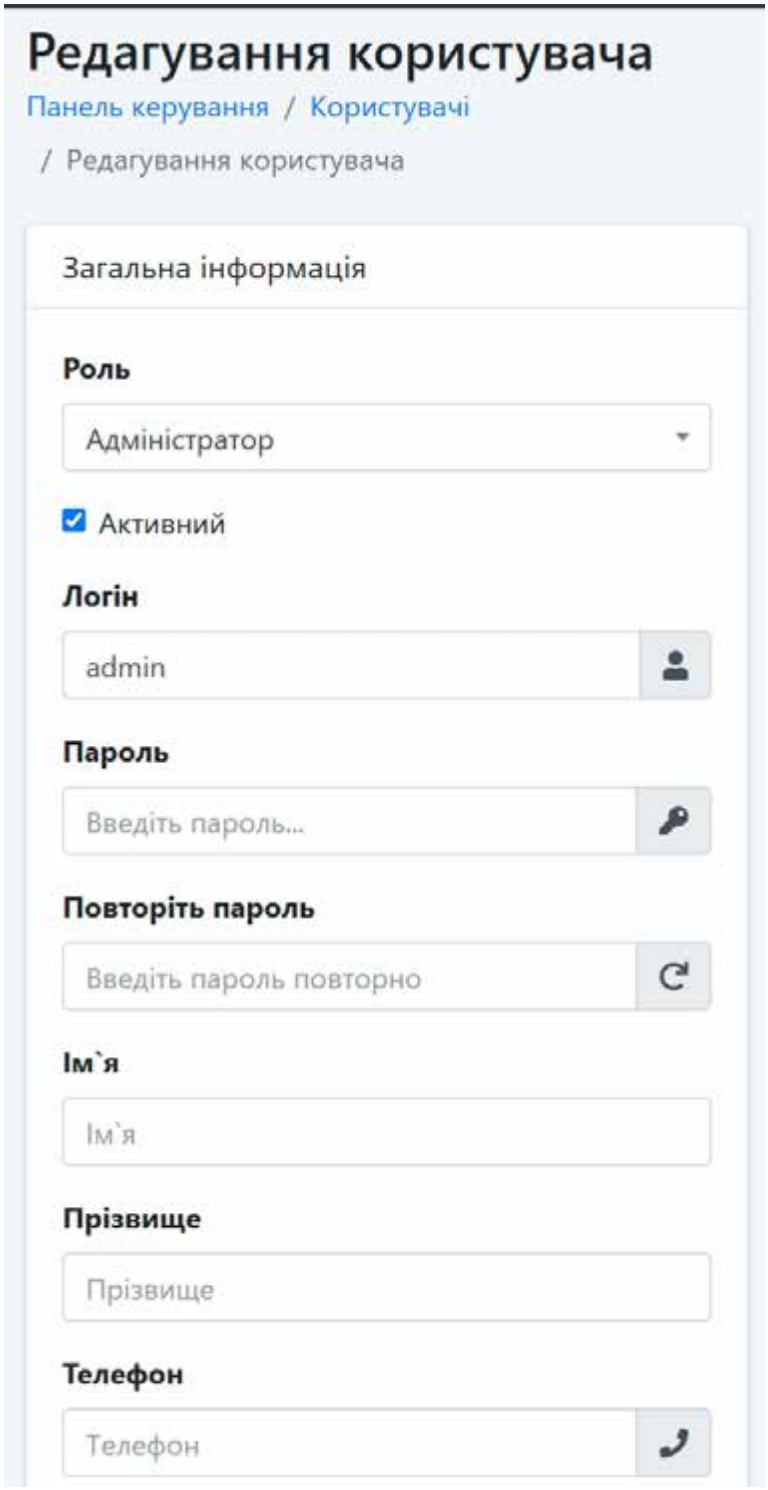

Рис. В.6 – Керування користувачами Мобайл версія (iPhone 14 Pro)

Додаток Г

Представлення окремих UML діаграм

Сторінок: 4

Київ 2025

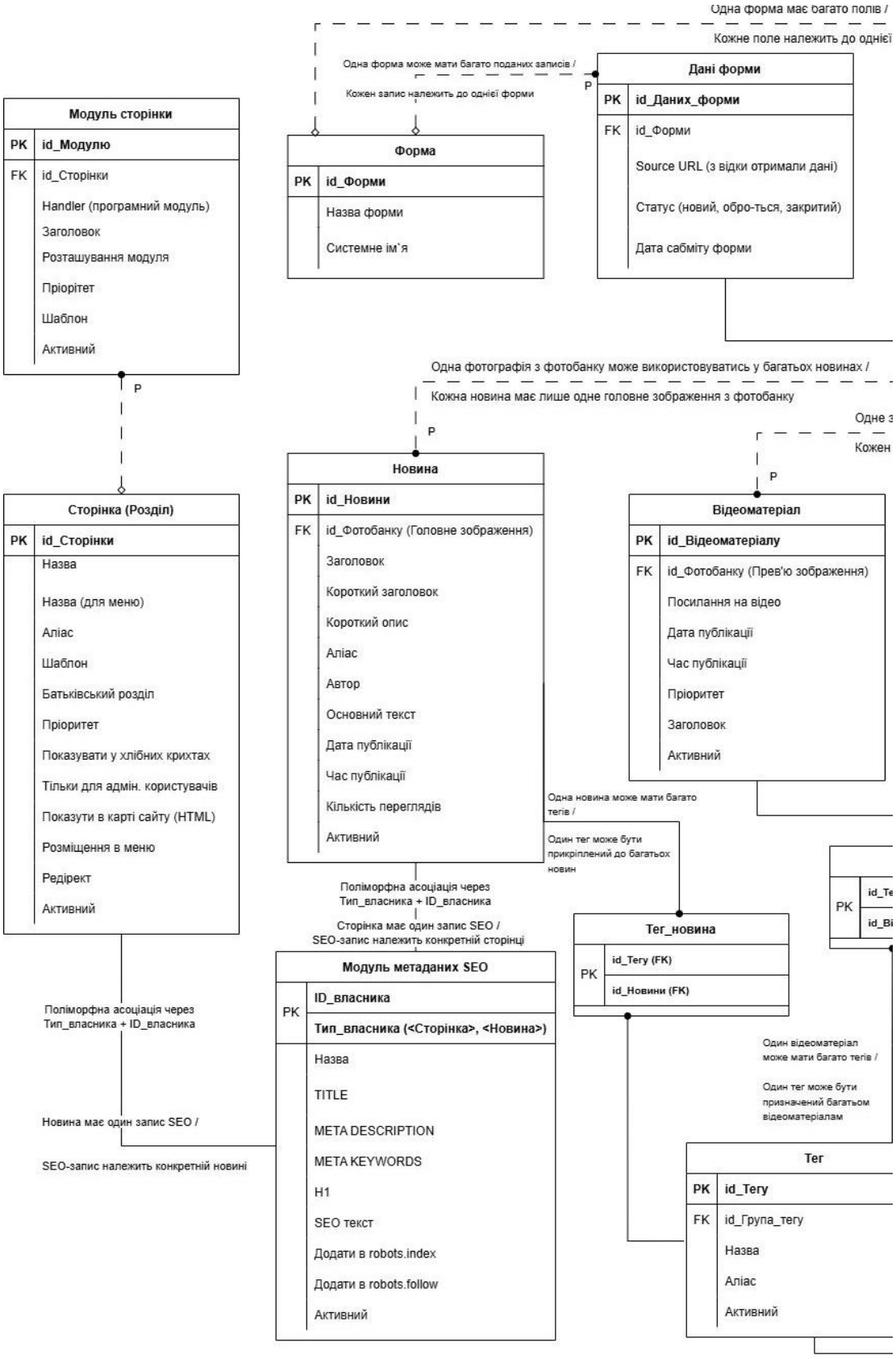

Рис. Г.1 – Збільшена версія ER-діаграма логічного рівня (частина 1)

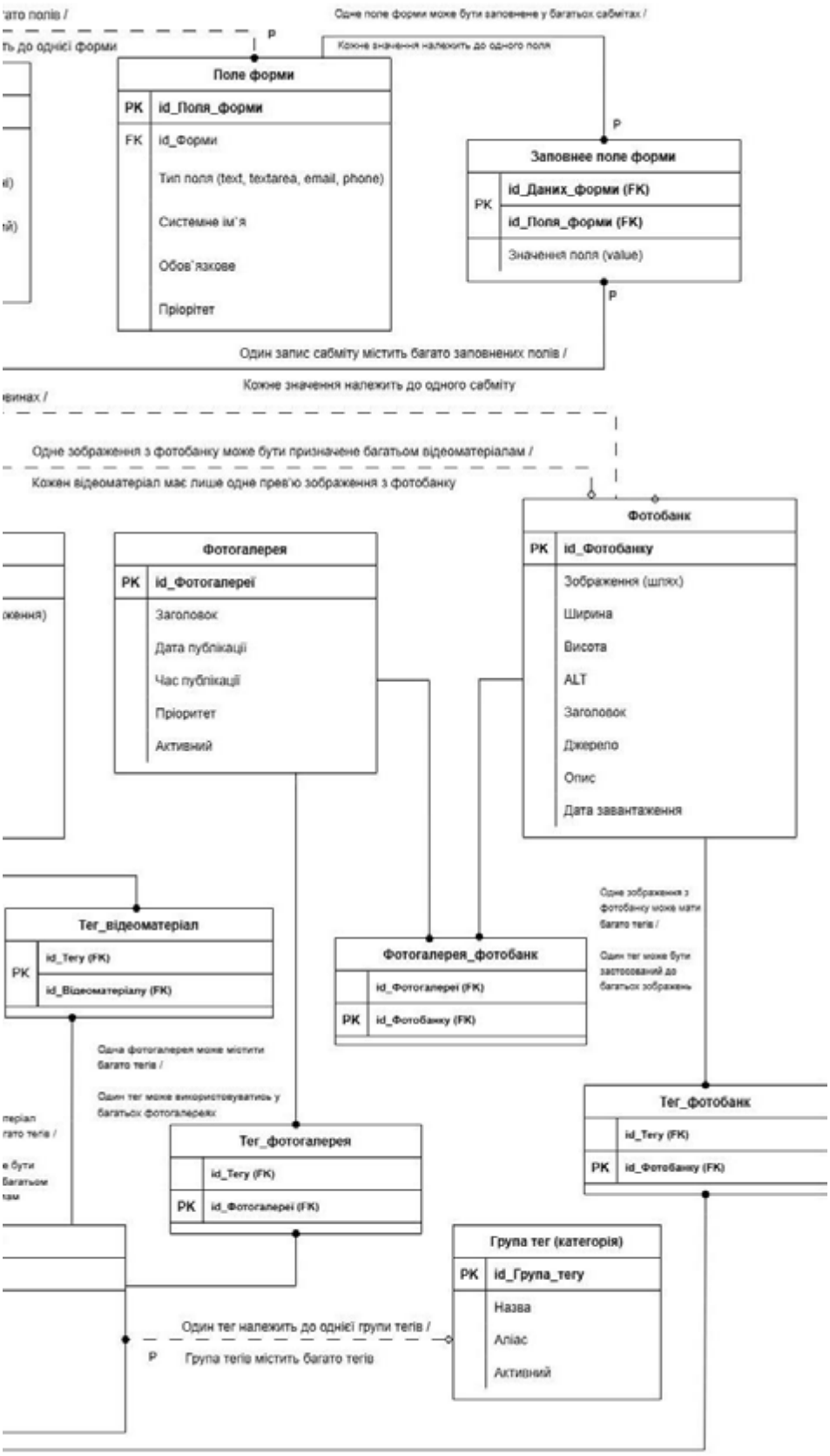

Рис. Г.2 – Збільшена версія ER-діаграма логічного рівня (частина 2)

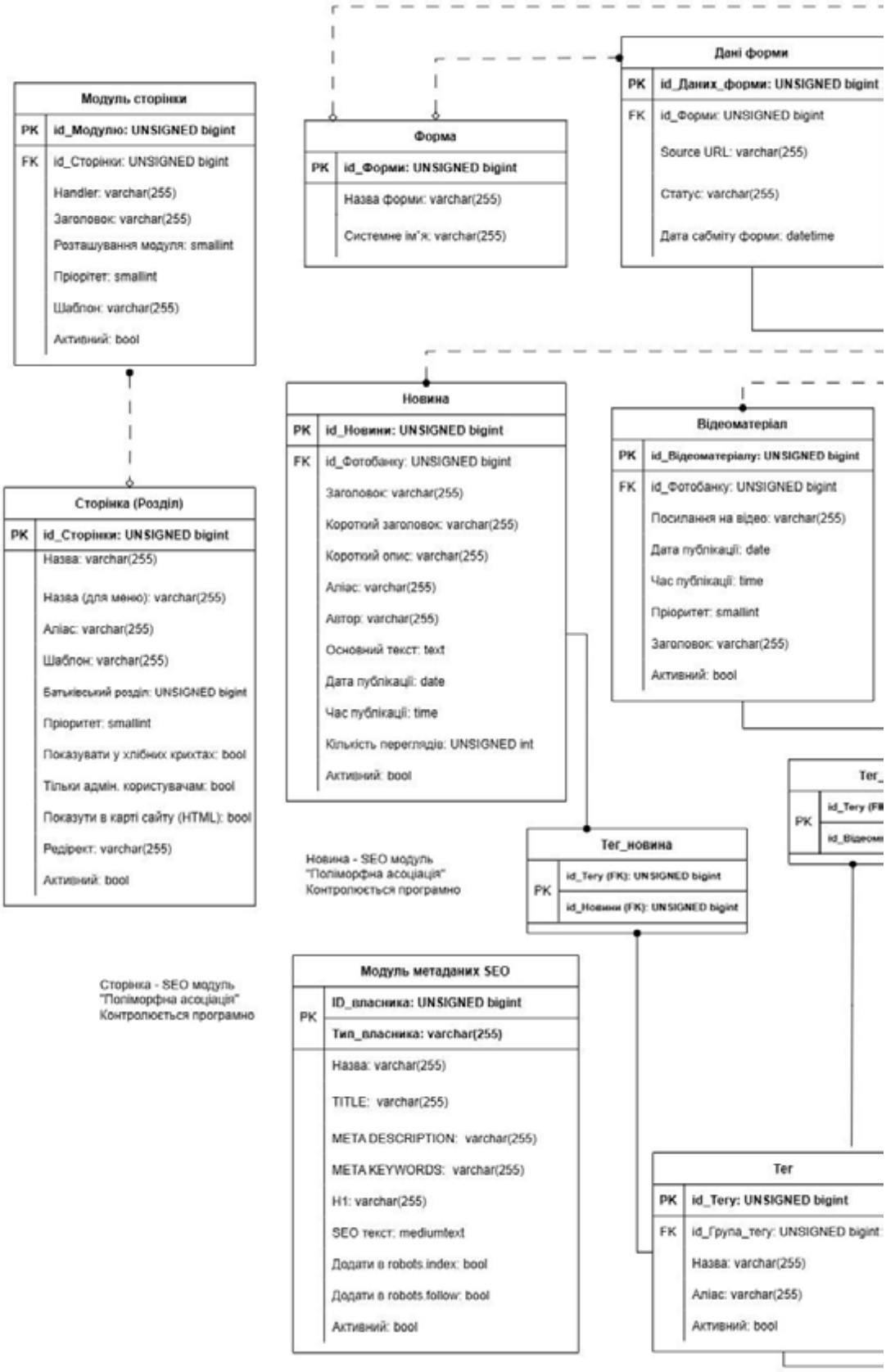

Рис. Г.3 – Збільшена версія фізичного рівня бази даних(частина 1)

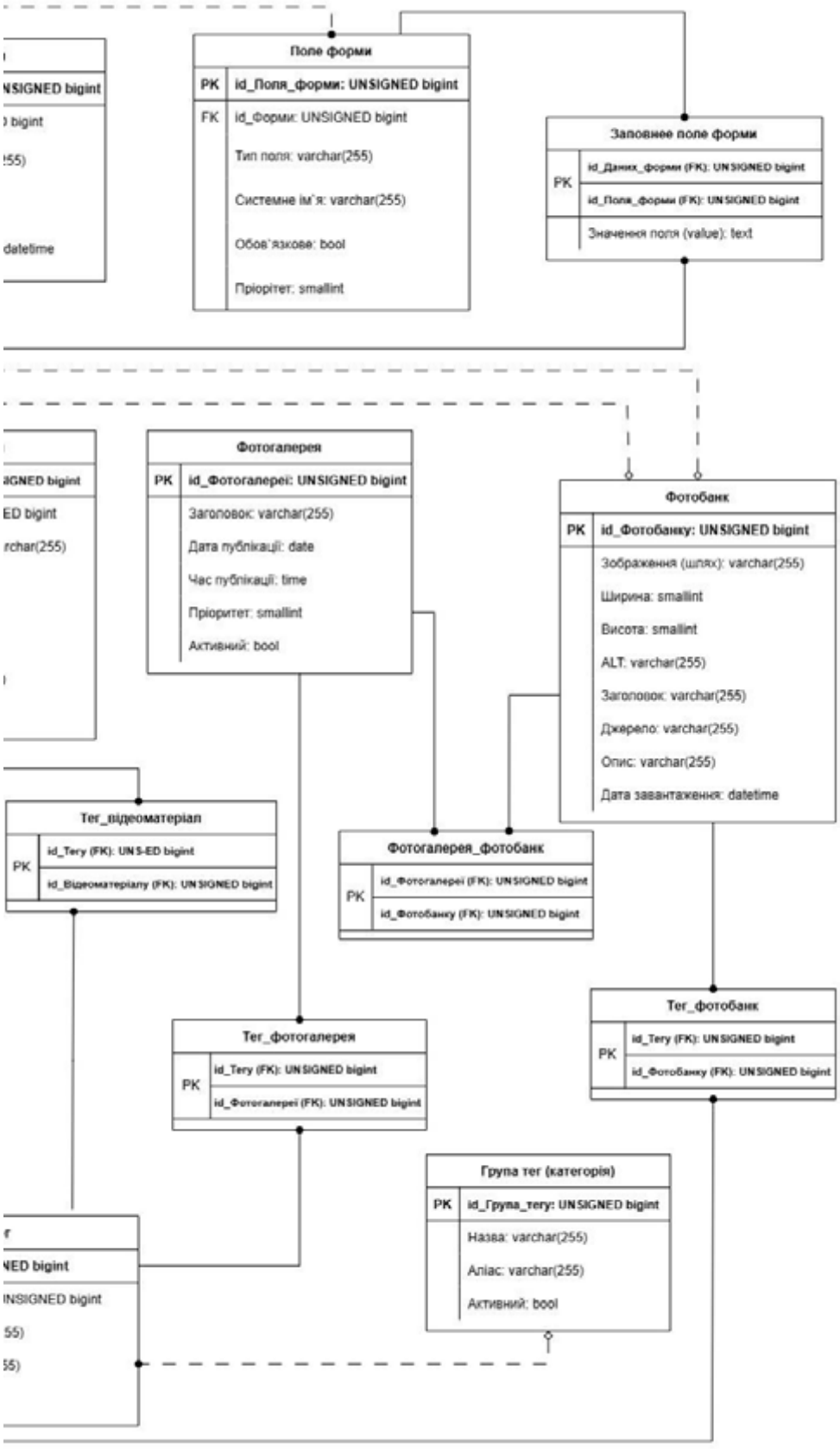

Рис. Г.4 – Збільшена версія фізичного рівня бази даних(частина 2)