

Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ВЕБ-САЙТ ДЛЯ КАТАЛОГІЗАЦІЇ ТА
КОМЕНТУВАННЯ ВІДЕОІГОР

WEBSITE FOR CATALOGING AND COMMENTING
ON VIDEO GAMES

спеціальність 122 Комп'ютерні науки

освітня програма «Комп'ютерні науки»

Виконав: здобувач вищої освіти
групи КН-41
Павліха Дмитро Олександрович

(підпис)

Керівник: асистент
Хиць Руслан Андрійович

(підпис)

Кваліфікаційну роботу
допущено до захисту
«__» _____ 2026 р.
Гарант освітньої програми:
д.пед.н., професор
Тулашвілі Юрій Йосипович

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра комп'ютерних наук

Ступінь вищої освіти: бакалавр

Галузь знань: 12 Інформаційні технології

Спеціальність: 122 Комп'ютерні науки

Освітня програма: «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Валерій ЛІЩИНА

«16» січня 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ПЕРШОГО (БАКАЛАВРСЬКОГО) РІВНЯ ВИЩОЇ ОСВІТИ

Павліха Дмитро Олександрович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Веб-сайт для каталогізації та коментування відеоігор»

Керівник асистент Хиць Руслан Андрійович

затверджені наказом закладу вищої освіти від «16» січня 2026 р. № 32/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи «02» червня 2026 р.

3. Вихідні дані до роботи: вебзастосунок для каталогізації та коментування відеоігор, що забезпечує перегляд інформації про ігри, розробників і компанії, фільтрацію за категоріями, систему коментарів користувачів та адміністративну панель для керування контентом. Система реалізована з використанням PHP-фреймворку Laravel, реляційної бази даних MySQL та клієнт-серверної архітектури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану вебзастосунків для каталогізації відеоігор, дослідження існуючих аналогів та методів їх реалізації, вибір засобів і технологій розробки. Проектування архітектури системи та структури бази даних, опис функціонального наповнення вебзастосунку. Розробка серверної та клієнтської частини системи, реалізація адміністративної панелі, механізму коментування та фільтрації даних. Забезпечення безпеки вебзастосунку та описати основні механізми захисту даних користувача.

5. Перелік графічного матеріалу: 24 рисунків, 14 лістингів, 1 таблиця

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
<i>Аналіз проблеми за темою роботи та постановка завдань дослідження</i>	<i>Хиць Р. А.</i>		
<i>Теоретичне дослідження</i>	<i>Хиць Р. А.</i>		
<i>Практична реалізація</i>	<i>Хиць Р. А.</i>		
<i>Спеціальні розрахунки</i>	<i>Хиць Р. А.</i>		
<i>Показник запозичень тексту</i>	%		
<i>Інструментальна перевірка</i>	<i>Кошелюк В. А.</i>		
<i>Нормоконтроль</i>	<i>Сачук В. О.</i>		
<i>Гарант ОПП</i>	<i>Тулашвілі Ю. Й.</i>		

7. Дата видачі завдання «16» січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	<i>Провести огляд літературних джерел по темі кваліфікаційної роботи</i>	<i>до 17.02.2026 р</i>	
2	<i>Провести аналіз загальної проблеми і вибір напрямків дослідження</i>	<i>до 28.02.2026 р.</i>	
3	<i>Розробити функціональну схему роботи програмного продукту</i>	<i>до 12.03.2026 р</i>	
4	<i>Описати засоби розробки об'єкта проектування</i>	<i>до 26.03.2026 р.</i>	
5	<i>Практична реалізація об'єкта проектування</i>	<i>до 30.04.2026 р.</i>	
6	<i>Провести спеціальні розрахунки</i>	<i>до 18.05.2026 р.</i>	
7	<i>Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру</i>	<i>до 02.06.2026 р.</i>	

Здобувач вищої освіти _____ Павліха Дмитро

Керівник роботи _____ Хиць Руслан

АНОТАЦІЯ

Павліха Д. О. Веб-сайт для каталогізації та коментування відеоігор. Кваліфікаційна робота бакалавра ОП «Комп'ютерні науки». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У роботі проведено аналіз предметної області вебресурсів для каталогізації відеоігор, розглянуто існуючі платформи та визначено основні функціональні потреби користувачів, пов'язані з переглядом, пошуком і коментуванням ігор. Описано функціональні та нефункціональні вимоги до системи, структури бази даних, архітектуру вебзастосунку та опис основних модулів системи. Окрему увагу приділено розробці веб-сайту для каталогізації та коментування відеоігор із використанням фреймворку Laravel та мови програмування PHP. Реалізовано систему керування контентом через адміністративну панель, багатомовність українською та англійською мовами, функціонал перегляду інформації про ігри, компанії та авторів, а також систему коментування. Також, приділено увагу питанням захисту інформації у вебзастосунку. У ньому розглянуто основні механізми авторизації, автентифікації користувачів, валідації вхідних даних та захисту системи від несанкціонованого доступу. Також описано використання засобів безпечної взаємодії між клієнтською та серверною частинами застосунку. У висновках узагальнено результати виконаної кваліфікаційної роботи та визначено перспективи подальшого вдосконалення вебзастосунку. Ключові слова: вебзастосунок, відеоігри, каталогізація, коментування, Laravel, PHP 8.3, адміністративна панель, багатомовність, база даних.

ANNOTATION

Pavlikha Dmytro. Website for Cataloging and Commenting Video Games. Bachelor's Qualification Thesis in the Educational Program «Computer Science». Lutsk National Technical University, Lutsk, 2026.

The bachelor's qualification thesis consists of an introduction, four chapters, conclusions, a list of references, and appendices. The thesis analyzes the subject area of web resources for video game cataloging, examines existing platforms, and identifies the main functional needs of users related to browsing, searching, and commenting on games. Functional and non-functional system requirements, database structure, web application architecture, and the main system modules are described. Particular attention is paid to the development of a website for cataloging and commenting on video games using the Laravel framework and the PHP programming language. A content management system through an administrative panel, multilingual support in Ukrainian and English, functionality for viewing information about games, companies, and creators, as well as a commenting system were implemented. Attention was also paid to information security issues in the web application. The thesis considers the main mechanisms of authorization, user authentication, input data validation, and protection against unauthorized access. In addition, secure interaction methods between the client and server sides of the application are described. The conclusions summarize the results of the completed qualification thesis and define prospects for further improvement of the web application. Keywords: web application, video games, cataloging, commenting, Laravel, PHP 8.3, administrative panel, multilingualism, database.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1 Аналіз сучасного стану проблеми.....	9
1.2 Аналіз аналогічних програм, баз даних, систем, обґрунтування вибраного напрямку та вибір методів рішення	11
1.3 Аналіз і вибір засобів проектування вебсайту.....	13
1.4 Постановка завдання на кваліфікаційну роботу бакалавр.....	18
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ	19
2.1 Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання.....	19
2.2 Функціонально-структурна схема роботи об'єкта проектування (блок-схеми алгоритмів, UML діаграми класів, діаграми компонентів).....	21
2.3 Створення моделі бази даних.....	23
РОЗДІЛ 3 РОЗРОБКА ОНЛАЙН ВІДЕОБІБЛІОТЕКИ.....	27
3.1 Практична реалізація об'єкта проектування. Побудова блок-схем модулів програми.....	27
3.2 Тестування та налагодження системи онлайн відеобібліотеки.....	40
3.3 Посібник з інсталяції та використання програмного продукту.....	45
РОЗДІЛ 4 СПЕЦІАЛЬНІ РОЗРАХУНКИ.....	48
ВИСНОВКИ	52
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	55
ДОДАТКИ	57

ВСТУП

У сучасному світі відеоігри стали однією з найпопулярніших форм цифрових розваг та важливою частиною інформаційного простору. Індустрія комп'ютерних ігор постійно розвивається, а кількість нових проєктів, ігрових студій та незалежних розробників щороку стрімко зростає. Користувачі активно шукають інформацію про нові ігри, цікавляться їхніми жанрами, розробниками, компаніями та коментарями інших гравців. У зв'язку з цим виникає потреба у створенні сучасних вебресурсів, які забезпечують зручне відображення, каталогізацію та перегляд детальної інформації про ту чи іншу відеоігру.

Актуальність теми полягає у необхідності створення вебзастосунку, який поєднує функції каталогу відеоігор та інтерактивної платформи для взаємодії користувачів між собою. Більшість існуючих ресурсів мають складний інтерфейс, або не забезпечують достатньо можливостей для адміністрування та локалізації контенту. Саме тому розробка вебзастосунку, яка дозволяє ефективно керувати інформацією про ігри, авторів та їх компанії, а також підтримує багатомовність і можливість коментування, є актуальним завданням.

Метою кваліфікаційної роботи є розробка веб-сайту для каталогізації та коментування відеоігор із використанням сучасних вебтехнологій. Створена система повинна забезпечувати зручний інтерфейс для користувачів, можливість перегляду детальної інформації про ігри, підтримку дво мовного інтерфейсу, а також зручну адміністративну панель для керування контентом на сайті.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз предметної області та існуючих аналогів;
- визначити функціональні та нефункціональні вимоги до системи;
- спроектувати структуру бази даних і архітектуру вебзастосунку;
- реалізувати сторінки для перегляду детальної інформації про відеоігри, компанії та авторів;
- створити адміністративну панель для додавання, редагування та видалення даних;

- реалізувати систему коментування ігор користувачами;
- забезпечити підтримку української та англійської мов;
- протестувати працездатність розробленої системи.

Об'єктом дослідження є процес організації та представлення інформації про відеоігри у вебсередовищі.

Предметом дослідження є методи та засоби розробки вебзастосунків для каталогізації й коментування відеоігор із використанням серверних технологій та систем керування базами даних.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз сучасного стану проблеми

У сучасному світі цифрові технології стрімко розвиваються та охоплюють все більше сфер людської діяльності. Однією з таких сфер є індустрія відеоігор, яка за останні роки перетворилася не лише на вид розваги, а й на масштабний сегмент глобального цифрового ринку. За даними аналітичних компаній, обсяг світового ринку відеоігор перевищує 180 мільярдів доларів США і продовжує активно зростати. Велика кількість користувачів щоденно використовує різноманітні вебресурси для пошуку інформації про ігри, перегляду рейтингів, новин, описів та відгуків інших гравців.

Популярність відеоігор зростає завдяки розвитку персональних комп'ютерів, ігрових консолей та мобільних пристроїв. Сучасні ігри доступні для широкого кола користувачів та охоплюють різноманітні жанри й платформи – від масштабних рольових ігор до мобільних аркад. Через це постійно збільшується обсяг інформації, яку необхідно систематизувати, фільтрувати та надавати користувачам у зручному та зрозумілому вигляді.

На сьогодні існує велика кількість сайтів і платформ, присвячених відеоіграм. Найвідомішими серед них є Metacritic, IGN, Steam, RAWG та GameSpot. Частина з них виконує функцію новинних порталів, інші дозволяють переглядати каталоги ігор або залишати коментарі та оцінки. Проте значна кількість подібних ресурсів має декілька суттєвих недоліків: перевантажений інтерфейс, надмірна кількість реклами, орієнтованість виключно на англomовну аудиторію, а також відсутність зручної системи фільтрації та пошуку. Це створює незручності для користувачів та ускладнює пошук необхідної інформації, особливо для тих, хто не володіє англійською мовою на достатньому рівні.

Окремої уваги заслуговує проблема достовірності та структурованості інформації. На багатьох платформах описи ігор не є уніфікованими: одні містять

лише короткий текст, інші – розгорнуті характеристики, скріншоти, трейлери та системні вимоги. Відсутність єдиного стандарту подання даних ускладнює порівняння ігор між собою та знижує загальну якість користувацького досвіду.

Однією з головних особливостей сучасних вебресурсів є можливість взаємодії між користувачами. Саме тому системи коментування стали важливою частиною більшості інформаційних платформ. Користувачі прагнуть не лише переглядати інформацію про ігри, а й висловлювати власну думку та знайомитись із думками інших гравців. Такий функціонал підвищує залученість аудиторії, формує навколо платформи активну спільноту та робить вебсайт більш інтерактивним і живим ресурсом.

Важливим елементом сучасних вебзастосунків є також підтримка багатомовності. Реалізація декількох мов інтерфейсу дозволяє зробити сайт більш доступним та розширити аудиторію. Для вебресурсів ігрової тематики це має особливе значення, оскільки ігрова індустрія є міжнародною та охоплює користувачів із різних країн і культур. Наявність україномовного інтерфейсу, сприяє популяризації якісних вебсервісів серед вітчизняних користувачів.

Для забезпечення зручного керування контентом невід'ємною частиною будь-якого сучасного вебзастосунку є адміністративна панель. За її допомогою адміністратор може додавати, редагувати та видаляти інформацію про відеоігри, розробників та видавців. Наявність такого функціоналу значно спрощує підтримку актуальності бази даних і зменшує часові витрати на адміністрування ресурсу.

Таким чином, аналіз сучасного стану проблеми свідчить про наявність реальної потреби у розробці зручного, структурованого та багатомовного вебзастосунку для каталогізації та коментування відеоігор. Такий ресурс повинен поєднувати зрозумілий інтерфейс, розширені можливості пошуку й фільтрації, систему коментування та гнучку адміністративну панель – що і є метою даної дипломної роботи.

1.2 Аналіз аналогічних вебресурсів, обґрунтування вибраного напрямку та вибір методів реалізації

У сучасному інформаційному просторі існує велика кількість вебресурсів, які надають користувачам можливість переглядати інформацію про відеоігри, залишати оцінки, коментарі та взаємодіяти з іншими учасниками спільноти. Аналіз існуючих платформ дозволяє визначити основні переваги та недоліки сучасних рішень, а також сформулювати чіткі вимоги до розробки власного вебзастосунку для каталогізації та коментування відеоігор.

Steam є однією з найбільших у світі платформ цифрової дистрибуції ігор, розробленою компанією Valve Corporation. Сервіс забезпечує доступ до великого каталогу ігор, системи рейтингів, відгуків та персоналізованих рекомендацій. Основними перевагами платформи є широкий функціонал, підтримка користувацьких профілів, досягнень, списків бажань та інтеграція повноцінної ігрової спільноти [1]. Проте через надмірну кількість функцій інтерфейс є перевантаженим, особливо для нових користувачів. Крім того, переважна частина функціоналу орієнтована на продаж ігор, а не на зручну та структуровану каталогізацію інформації про них.

IGN – один із найвідоміших медіаресурсів ігрової тематики, який спеціалізується на публікації новин, оглядів, відеоматеріалів та рейтингів відеоігор. Платформа має сучасний дизайн та структуровану систему подачі редакційного контенту [2]. Однак функціонал взаємодії між користувачами є суттєво обмеженим: відсутня повноцінна система каталогізації ігор із фільтрацією та порівнянням, а основний акцент зроблено саме на матеріали редакції, а не на контент, створений самими користувачами.

Metacritic – агрегатор рецензій, який збирає та узагальнює оцінки як професійних критиків, так і звичайних користувачів для ігор, фільмів і музики. Сервіс дозволяє швидко отримати загальну агреговану оцінку продукту та переглянути окремі рецензії [3]. Разом із тим інтерфейс платформи є морально застарілим, система коментування має мінімальний рівень інтерактивності, а

ресурс не надає адміністраторам інструментів для гнучкого керування структурою контенту або адаптації системи під конкретні потреби.

Epic Games Store – відносно молода платформа цифрової дистрибуції від компанії Epic Games, яка активно конкурує зі Steam. Сервіс вирізняється сучасним та лаконічним дизайном, підтримує персоналізацію інтерфейсу та забезпечує швидку роботу вебзастосунку [4]. Проте основна увага приділяється комерційній складовій – продажу цифрових копій ігор та ексклюзивних угодах із розробниками. Через це функціонал каталогізації та коментування реалізований не в повному обсязі, а спільнотна взаємодія між користувачами залишається мінімальною порівняно з конкурентами.

RAWG – база даних відеоігор, орієнтована саме на каталогізацію та відстеження ігрового прогресу користувачів. Платформа має розвинену систему тегів, жанрів та фільтрів, що спрощує пошук. Однак система коментування на ресурсі розвинена слабко, а інтерфейс є функціональним, але позбавленим виразності та дизайнерської індивідуальності.

Узагальнюючи результати аналізу, можна стверджувати, що більшість сучасних вебресурсів мають або надмірно складний функціонал, орієнтований переважно на продаж ігор, або недостатній рівень інтерактивності у частині взаємодії користувачів із контентом. Значна частина сервісів також не дозволяє гнучко налаштовувати структуру даних, систему категорій та механізми адміністрування, а підтримка мов, відмінних від англійської, реалізована лише частково або відсутня взагалі. Порівняльний аналіз розглянутих платформ наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз існуючих вебресурсів

Критерій	Steam	IGN	Metacritic	Epic Games	RAWG
Каталогізація ігор	✓	частково	частково	✓	✓
Система коментування	✓	обмежено	обмежено	✗	✗
Багатомовність	✓	✗	✗	✓	✗
Адмін-панель	✗	✗	✗	✗	✗
Адаптивний інтерфейс	✓	✓	частково	✓	✓
Орієнтація на контент	✗	✓	✓	✗	✓

У результаті проведеного аналізу було визначено основні вимоги до розроблюваного вебзастосунку:

- реалізація зручної та структурованої системи каталогізації відеоігор із підтримкою жанрів, платформ і розробників;
- підтримка багатомовності інтерфейсу, зокрема української та англійської мов;
- можливість залишення коментарів авторизованими користувачами;
- реалізація адміністративної панелі для повноцінного керування контентом;
- підтримка категорій, компаній та розробників як окремих сутностей бази даних;
- адаптивний інтерфейс користувача для коректного відображення на різних пристроях;
- швидке оновлення та публікація нового контенту адміністратором;
- інтеграція з Telegram-ботом для автоматичного сповіщення користувачів про додавання нових ігор.

Визначені вимоги стали основою для вибору технологічного стеку та архітектурних рішень при розробці вебзастосунку, що описуються в наступних розділах роботи.

1.3 Аналіз і вибір засобів проектування вебсайту для каталогізації відеоігор

Під час розробки сучасних вебзастосунків важливу роль відіграє правильний вибір технологій та інструментів, від яких залежить продуктивність, стабільність, зручність підтримки та можливість подальшого розвитку програмного продукту. Для створення вебсайту, призначеного для каталогізації та коментування відеоігор, необхідно використовувати сучасні засоби

веброзробки, які забезпечують ефективну взаємодію між серверною частиною, базою даних та користувацьким інтерфейсом.

Одним із найбільш популярних серверних фреймворків для створення вебзастосунків є Laravel, на рисунку 1.1 зображено офіційний логотип даного фреймворку.



Рисунок 1.1 – Логотип Laravel [5]

Даний фреймворк побудований на основі архітектурного шаблону MVC, що дозволяє розділити логіку застосунку, інтерфейс користувача та роботу з даними. Такий підхід значно спрощує процес розробки, тестування та подальшої підтримки вебпроєкту. На рисунку 1.2 зображено структуру архітектурного шаблону MVC.

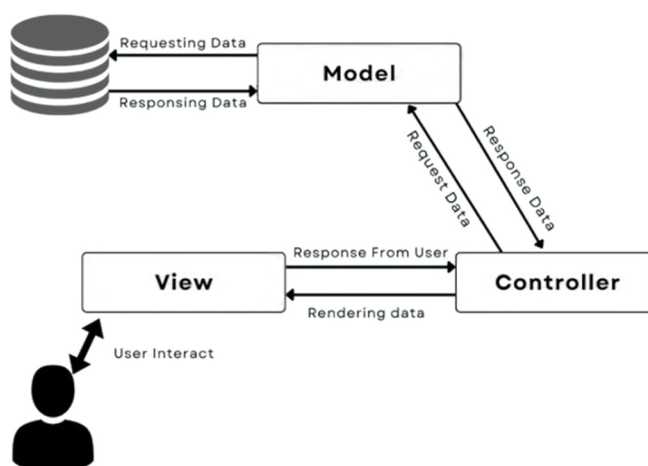


Рисунок 1.2 – Архітектурний шаблон MVC [6]

Фреймворк Laravel містить велику кількість готових інструментів для реалізації сучасних вебсистем. Зокрема, він забезпечує зручну систему

маршрутизації, підтримку шаблонізатора Blade, механізми роботи з базами даних через ORM Eloquent, систему міграцій та готові рішення для автентифікації користувача. Використання Laravel дозволяє скоротити час розробки та зробити структуру проєкту більш організованою та простою.

Для керування залежностями та сторонніми бібліотеками у проєкті використовується менеджер пакетів Composer. Він дозволяє автоматично встановлювати, оновлювати та конфігурувати необхідні бібліотеки для роботи вебзастосунку. Завдяки Composer забезпечується централізоване управління залежностями проєкту, що значно спрощує процес розробки та розгортання системи, а також гарантує сумісність використовуваних компонентів. На рисунку 1.3 зображено логотип composer.



Рисунок 1.3 – Логотип composer [7]

Основною мовою програмування серверної частини було обрано PHP, на рисунку 1.4 зображено логотип мови програмування PHP.



Рисунок 1.4 – Логотип мови програмування PHP [8]

Дана мова є однією з найбільш поширених технологій у сфері веброзробки та активно використовується для створення динамічних вебресурсів. Версія PHP 8.3 забезпечує високу продуктивність, покращену оптимізацію виконання коду та підтримку сучасних можливостей програмування.

Для зберігання інформації про відеоігри, компанії, авторів та коментарі використовується реляційна база даних. База даних дозволяє ефективно організувати структуру інформації, забезпечити швидкий доступ до даних та підтримувати взаємозв'язки між окремими сутностями системи. На рисунку 1.5 зображено логотип бази даних MySQL.



Рисунок 1.5 – Логотип реляційної бази даних Mysql [9]

Під час створення користувацького інтерфейсу використовувалися Blade-шаблони фреймворку Laravel, CSS та JavaScript. Blade є шаблонізатором Laravel, який дозволяє зручно формувати структуру вебсторінок, повторно використовувати та наслідувати компоненти інтерфейсу. Для стилізації сторінок використовувався CSS, а JavaScript застосовувався для реалізації динамічних

елементів інтерфейсу. Використання цих технологій дозволило створити зручний, структурований та сучасний вебінтерфейс для користувачів.

Також для реалізації багатомовності у вебзастосунку було використано бібліотеки Laravel Localization та Laravel Translations. Їх використання дозволяє організувати підтримку декількох мов інтерфейсу, забезпечити зручне перемикання між мовами та зберігати текстові ресурси у структурованому вигляді. Рисунок 1.6 демонструє логотип бібліотеки Laravel Translation.



Рисунок 1.6 – Логотип бібліотеки Laravel Translation [10]

Завдяки цьому користувачі мають можливість переглядати вебсайт українською та англійською мовами, що значно покращує доступність та розширює коло використання системи для аудиторії. Також, для створення автентифікації адміністраторів використано пакет Laravel Breeze. На рисунку 1.7 зображено логотип Laravel Breeze.



Рисунок 1.7 – Логотип Laravel Breeze [11]

Даний пакет надає готову базову реалізацію реєстрації, входу користувача та відновлення пароля. Використання Laravel Breeze дозволяє значно скоротити час розробки системи авторизації та забезпечує відповідність сучасним стандартам безпеки вебзастосунків.

Отож, використання фреймворку Laravel, мови програмування PHP, сучасних вебтехнологій, бібліотек для локалізації та реляційної бази даних є доцільним рішенням для розробки вебсайту каталогізації та коментування відеоігор. Обрані засоби проектування забезпечують ефективність роботи системи, зручність адміністрування, підтримку багатомовності та можливість подальшого розширення функціоналу вебзастосунку.

1.4 Постановка завдання на кваліфікаційну роботу бакалавра

Основною метою даної кваліфікаційної роботи є розроблення вебсайту для каталогізації та коментування відеоігор. Основним призначенням системи є надання користувачам можливості переглядати детальну інформацію про відеоігри, компанії та авторів, а також залишати коментарі та взаємодіяти з контентом.

На вебсайті повинні бути реалізовані:

- головна сторінка з інформацією про відеоігри;
- сторінка компаній;
- сторінка авторів;
- сторінка ігор із можливістю переходу до детальної інформації;
- система коментування;
- адміністративна панель для керування контентом;
- підтримка української та англійської мов інтерфейсу.

Спираючись на основне завдання, було поставлено такі задачі:

- визначити актуальність створення вебсайту для каталогізації відеоігор;
- провести аналіз сучасних технологій та засобів веброзробки;
- сформулювати структуру вебзастосунку та визначити логіку роботи;
- спроектувати структуру бази даних;
- реалізувати сторінки для інформації про ігри, компанії та авторів;
- створити систему коментування користувачів;

- реалізувати адміністративну панель для додавання, редагування та видалення інформації;
- забезпечити підтримку двох мов інтерфейсу;
- провести тестування функціональності вебзастосунку;
- реалізувати механізми захисту та безпеки вебсистеми;
- створити технічну документацію щодо запуску та використання системи.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання

Під час розробки вебсайту для каталогізації та коментування відеоігор одним із найважливіших етапів став вибір технологій та програмних засобів, які забезпечують стабільну роботу системи, зручність адміністрування та можливість подальшого розширення функціоналу. Вибір технологічного стеку здійснювався з урахуванням сучасних тенденцій веброзробки, продуктивності програмного забезпечення та зручності підтримки проєкту в майбутньому.

Для реалізації серверної частини вебзастосунку було обрано фреймворк Laravel.

Використання Laravel дозволяє суттєво спростити створення складних вебсистем завдяки наявності вбудованих механізмів маршрутизації, роботи з базами даних, обробки HTTP-запитів та системи шаблонів Blade. Важливою перевагою фреймворку є підтримка архітектурного шаблону MVC, який забезпечує чітке розділення логіки застосунку, даних та інтерфейсу користувача. На рисунку 2.1 зображено системи шаблонів blade

Laravel Blade Template Engine



Рисунок 2.1 – Система шаблонів Blade [12]

Серверна частина реалізована за допомогою мови програмування PHP. Використання сучасної версії PHP забезпечує швидшу розробку, оптимізацію виконання коду та підтримку нових можливостей мови. Крім того, PHP є однією з найбільш популярних мов веброзробки, що робить її ефективним рішенням для створення динамічних вебзастосунків.

Для зберігання інформації про відеоігри, компанії, авторів та коментарі використовується реляційна MySQL. Такий підхід дозволяє організувати структуроване зберігання інформації, забезпечити швидкий доступ до даних та підтримувати зв'язки між окремими сутностями системи. Робота з базою даних реалізована через ORM Eloquent, яка значно спрощує взаємодію між серверною частиною та таблицями бази даних. На рисунку 2.2 наведено систему об'єктно-реляційне відображення (ORM) в laravel.

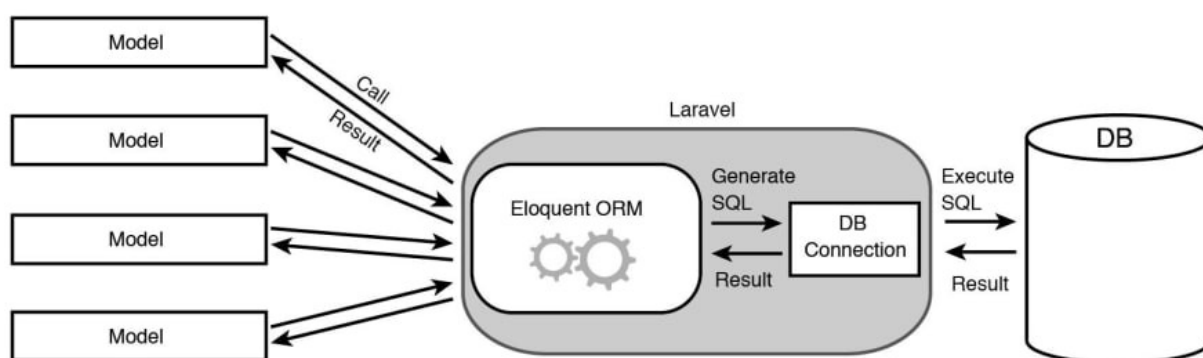


Рисунок 2.2 – Об'єктно-реляційне відображення (ORM) [13]

Під час створення користувацького інтерфейсу використовувалися Blade-шаблони фреймворку Laravel, CSS та JavaScript. Blade дозволяє створювати структуровані шаблони сторінок, повторно використовувати компоненти інтерфейсу та ефективно організовувати відображення даних. Для стилізації вебсторінок використовувався CSS, а JavaScript застосовувався для реалізації інтерактивних елементів і динамічної взаємодії користувача із системою.

Окрему увагу було приділено реалізації багатомовності вебзастосунку. Для цього використовувалися бібліотеки Laravel Localization та Laravel Translations. Їх використання дозволило реалізувати підтримку української та англійської мов інтерфейсу, забезпечити перемикання мов та організувати зберігання текстових ресурсів у структурованому вигляді.

Для забезпечення взаємодії користувачів із контентом у системі реалізовано механізм коментування. Завдяки цьому користувачі можуть залишати власні відгуки про відеоігри та взаємодіяти між собою через систему коментарів. Такий функціонал робить вебресурс більш інтерактивним та підвищує активність користувачів.

Важливою складовою вебзастосунку є адміністративна панель, яка забезпечує керування контентом системи. Адміністратор має можливість додавати, редагувати та видаляти інформацію про ігри, компанії та авторів. Наявність адміністративної панелі значно спрощує підтримку актуальності даних та процес адміністрування системи.

2.2 Функціонально-структурна схема роботи об'єкта проектування (блок-схеми алгоритмів, UML діаграми класів, діаграми компонентів)

Основною метою етапу проектування є створення цілісної вебсистеми, яка забезпечує зручну роботу з інформацією про відеоігри, авторів та компанії. Розроблюваний вебзастосунок повинен забезпечувати ефективну взаємодію між користувацьким інтерфейсом, серверною частиною та базою даних, що дозволяє

реалізувати повноцінний процес перегляду, додавання та редагування інформації.

Архітектура системи побудована за клієнт-серверним принципом. Користувач взаємодіє із вебінтерфейсом через браузер, переглядає інформацію про відеоігри, переходить між сторінками та залишає коментарі. Після виконання певної дії користувачем клієнтська частина формує HTTP-запит до серверної частини застосунку. Схематичне представлення клієнт-серверної архітектури наведено на рисунку 2.3.

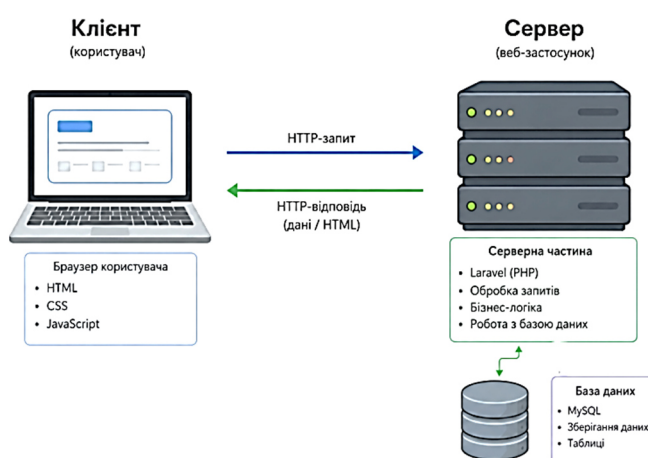


Рисунок 2.3 – Клієнт-серверна архітектура

Серверна частина відповідає за обробку запитів, маршрутизацію, перевірку даних, взаємодію з базою даних та формування відповіді для користувача. Для реалізації серверної логіки використовується фреймворк Laravel, який забезпечує структуровану організацію коду та підтримку архітектури MVC. У системі реалізовано декілька основних функціональних модулів:

- модуль роботи з відеоіграми;
- модуль компаній;
- модуль авторів;
- модуль коментарів;
- модуль багатомовності;

— адміністративний модуль керування контентом.

Модуль відеоігор забезпечує перегляд списку ігор, отримання детальної інформації про конкретну гру, відображення рейтингу, жанру, платформи та опису. Користувач має можливість переходити зі списку ігор до окремої сторінки з детальною інформацією. Модуль компаній призначений для відображення інформації про компанії-розробники. Для кожної компанії передбачено окрему секцію в карточці з описом, назвою компанії та країною в якій вона зареєстрована. Модуль авторів забезпечує відображення інформації про авторів або розробників, пов'язаних із певними іграми. Це дозволяє структурувати інформацію та забезпечити логічні зв'язки між окремими сутностями системи. Система коментування реалізована для забезпечення взаємодії користувачів із контентом. Користувачі можуть залишати текстові коментарі до ігор, переглядати повідомлення інших користувачів та брати участь в обговореннях. Для забезпечення підтримки декількох мов у системі реалізовано модуль локалізації. Завдяки використанню бібліотек Laravel Localization та Laravel Translations користувач може перемикаати мову інтерфейсу між українською та англійською.

Важливою частиною системи є адміністративна панель. Вона дозволяє адміністратору керувати вмістом вебсайту, додавати нові записи, редагувати існуючі та видаляти непотрібний контент. Для забезпечення безпеки системи реалізовано механізми перевірки даних та захисту від несанкціонованого доступу. Доступ до адміністративної панелі мають лише авторизовані користувачі з відповідними правами доступу.

Таким чином, функціонально-структурна схема вебзастосунку забезпечує злагоджену взаємодію між усіма компонентами системи, дозволяє ефективно працювати з інформацією про відеоігри та забезпечує можливість подальшого масштабування проєкту.

2.3 Створення моделі бази даних

Проектування бази даних є одним із найважливіших етапів розробки вебзастосунку, оскільки саме база даних відповідає за зберігання, обробку та структурування всієї інформації системи. Від правильності побудови структури бази даних залежить швидкість роботи вебресурсу, ефективність взаємодії між окремими компонентами системи та можливість подальшого розширення функціоналу. У межах розробки вебсайту для каталогізації та коментування відеоігор було створено реляційну модель бази даних, яка забезпечує збереження інформації про відеоігри, категорії, компанії, авторів та коментарі користувачів. Для роботи з базою даних використовувалися засоби фреймворку Laravel та ORM Eloquent, що дозволяє реалізувати зручну взаємодію між серверною частиною застосунку та таблицями бази даних.

Структура бази даних складається з декількох основних таблиць: categories, creators, games, companies, comments.

На рисунку 2.4 представлено фізичну структуру бази даних вебзастосунку для каталогізації та коментування відеоігор.

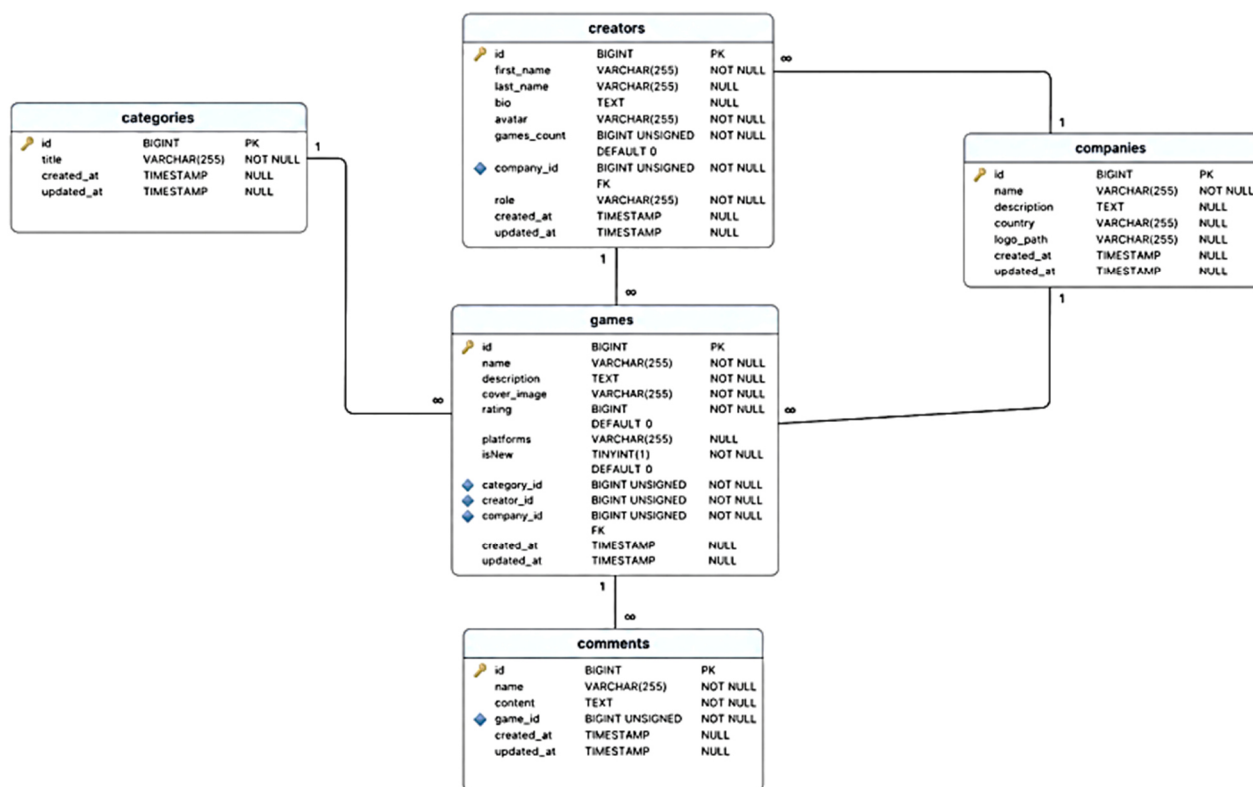


Рисунок 2.4 – Фізична структура бази даних

Малюнок демонструє взаємозв'язки між основними таблицями системи: `games`, `categories`, `creators`, `companies` та `comments`. Центральною таблицею є `games`, оскільки саме вона містить основну інформацію про відеоігри та пов'язана з іншими сутностями через зовнішні ключі. Така структура забезпечує цілісність даних, спрощує отримання інформації та дозволяє ефективно реалізувати функціонал вебзастосунку.

Таблиця `categories` призначена для зберігання жанрів відеоігор. Вона містить унікальний ідентифікатор категорії, назву категорії та службові поля. Використання окремої таблиці категорій дозволяє структурувати інформацію та спрощує сортування ігор за жанрами. Таблиця `creators` використовується для зберігання інформації про авторів відеоігор. У таблиці зберігаються ім'я, прізвище, коротка біографія, зображення автора, кількість створених ігор ним, роль у розробці та ідентифікатор компанії, до якої належить розробник. Наявність зв'язку між авторами та компаніями дозволяє організувати логічну структуру взаємодії між сутностями системи.

Основною таблицею системи є `games`, яка містить інформацію про відеоігри. У таблиці зберігаються назва гри, опис, зображення обкладинки, рейтинг, підтримувані платформи та спеціальне поле `isNew`, яке використовується для позначення нових ігор у системі. Також таблиця містить зовнішні ключі для зв'язку з категорією, автором і компанією. Така структура дозволяє організувати швидкий доступ до інформації та реалізувати детальні сторінки для кожної гри.

Для забезпечення цілісності даних у таблиці `games` використовуються зовнішні ключі:

- `category_id` – зв'язок із таблицею `categories`;
- `creator_id` – зв'язок із таблицею `creators`;
- `company_id` – зв'язок з таблицею `companies`.

Використання зовнішніх ключів дозволяє підтримувати правильність зв'язків між таблицями та забезпечує автоматичне видалення пов'язаних записів при видаленні основного елемента системи.

Таблиця `companies` містить інформацію про компанії-розробники відеоігор. У ній зберігаються назва компанії, опис діяльності, країна розташування та логотип компанії. Використання окремої таблиці компаній дозволяє уникнути дублювання інформації та забезпечує зручне відображення пов'язаних ігор та авторів.

Для реалізації системи коментування використовується таблиця `comments`. У ній зберігаються ім'я користувача, текст коментаря та ідентифікатор гри, до якої належить коментар. Завдяки цьому користувачі можуть залишати власні відгуки про конкретну гру та обговорювати відеоігри безпосередньо на сторінках вебсайту.

Для створення структури таблиць використовувалися міграції `Laravel`. Такий підхід дозволяє автоматизувати процес створення бази даних, контролювати зміни структури таблиць та забезпечує зручність перенесення проєкту між різними середовищами розробки.

Побудована модель бази даних забезпечує ефективне зберігання інформації, підтримку взаємозв'язків між сутностями та стабільну роботу вебзастосунку. Її структура є достатньо гнучкою для подальшого розширення функціоналу системи та додавання нових можливостей у майбутньому.

РОЗДІЛ 3

РОЗРОБКА ВЕБСАЙТУ ДЛЯ КАТАЛОГІЗАЦІЇ ТА КОМЕНТУВАННЯ ВІДЕОІГОР

3.1 Практична реалізація об'єкта проектування. Побудова блок-схем модулів програми

Перед початком розробки програмного забезпечення потрібно було створити базовий проєкт Laravel за допомогою команди «composer create-project laravel/laravel game». Фактично, це процес копіювання репозиторію з бази Laravel. Після успішного створення потрібно було налаштувати файл середовища «.env». Файл середовища, зазвичай з назвою «.env» – файл конфігурації, який служить для зберігання конфіденційних даних, які не повинні бути включені в код додатку. У «.env» для Laravel передбачено багато значень, які потрібно змінити перед запуском системи, чи то на сервері, чи то на локальній машині окремого учасника процесу створення продукту. Більшість з цих значень є індивідуальними для кожного серверу.

Основні значення, що були змінені у даному випадку – це (ключ-значення):

- «APP_NAME» – «Game-4-You»;
- «APP_TIMEZONE» – «Europe/Kyiv»;
- «FILESYSTEM_DISK» – «public»;
- «DB_CONNECTION» – «mysql»;
- «DB_HOST» – «127.0.0.1»;
- «DB_PORT» – «3307»;
- «DB_DATABASE» – «game»;
- «DB_USERNAME» – «root»;
- «DB_PASSWORD» – «root».

Всі ці значення використовуються для конфігурування саме Laravel back-end, і зчитуються файлами конфігурації відповідних підсистем, які знаходяться у папці «config». Після цього було введено серію команд в терміналі, які наведено в лістингу 3.1.

Лістинг 3.1 – Серія команд налаштування проєкту

```
composer require laravel/breeze -dev  
php artisan install:api  
php artisan breeze:install
```

кінець лістингу 3.1

Перша команда робить запит на пакет з Laravel Breeze, робить його необхідним лише для середовища з назвою «dev». Laravel Breeze встановлює вже готову систему автентифікації, на базі Sanctum. Він зручний тим, що при його інсталяції вибирається стек, на базі якого потрібно зробити автентифікацію, і він відразу встановлює всі потрібні для цього пакети.

Друга команда відповідає за встановлення та основного налаштування системи автентифікації Laravel Sanctum, створює правила для гостей та авторизованих користувачів, які можна пізніше використовувати у налаштуванні маршрутизації, та створює файл маршрутизації API.

Третя команда відповідає саме за встановлення та публікацію пакету Laravel Breeze. При виконанні цієї команди, Laravel консоль artisan запитує додаткову інформацію, а саме: стек автентифікації, додаткові рішення, та вибір тестового фреймворку, на якому він відразу публікує готові автотести автентифікації, які, за потреби, пізніше можна використовувати.

Після створення базової структури вебзастосунку було реалізовано структуру бази даних. У Laravel для цього використовуються міграції. Міграції дозволяють створювати та змінювати структуру таблиць за допомогою PHP-класів без необхідності ручного написання SQL-коду. Для створення міграції використовується artisan-команда: «php artisan make:migration create_categories_table». Ця команда генерує файл міграції, який надалі використовується для опису структури таблиці categories та її полів.

Клас міграції складається з двох методів: «up» і «down», які створюють та видаляють таблиці, відповідно. Всю структуру бази даних, яка використовується, було побудовано за допомогою міграцій. У лістингу 3.2 наведено приклад тіла методу «up».

 Лістинг 3.2 – Метод міграції «up», яка створює таблицю «categories»

```
public function up(): void
{
    Schema::create('categories', function (Blueprint $table): void {
        $table->id();
        $table->string('title');
        $table->timestamps();
    });
}
```

кінець лістингу 3.2

Окрім таблиці категорій, у межах системи було реалізовано інші основні сутності, які забезпечують повноцінне функціонування вебзастосунку для каталогізації відеоігор, зокрема: розробники, компанії, ігри та коментарі користувачів. Основною таблицею системи є таблиця «games», яка містить інформацію про відеоігри та є центральним елементом усієї бази даних. Вона забезпечує зв'язок із іншими сутностями, такими як категорії, розробники та компанії. Таблиця зберігає основні дані про кожну гру, включаючи її назву, опис, обкладинку, рейтинг та підтримувані платформи. Також у структурі таблиці передбачено зовнішні ключі для встановлення зв'язків із іншими таблицями системи. Структура таблиці наведена у лістингу 3.3.

 Лістинг 3.3 – Приклад тіла методу міграції «up», яка створює таблицю «games»

```
Schema::create('games', function (Blueprint $table): void {
    $table->id();
    $table->string('name');
    $table->text('description');
    $table->string('cover_image');
    $table->bigInteger('rating')->default(0);
    $table->string('platforms')->nullable();
    $table->unsignedBigInteger('category_id')
        ->constrained()
        ->onDelete('cascade');
    $table->unsignedBigInteger('creator_id')
        ->constrained()
        ->onDelete('cascade');
    $table->unsignedBigInteger('company_id')
        ->constrained()
        ->onDelete('cascade');
    $table->timestamps();
});
```

кінець лістингу 3.3

Таблиця `games` має зв'язки по типу «багато до одного» з таблицями `categories`, `creators` та `companies`. Це означає, що кожна гра належить до однієї категорії, одного розробника та однієї компанії, тоді як кожна з цих сутностей може бути пов'язана з великою кількістю ігор.

Така структура забезпечує нормалізацію бази даних, зменшує дублювання інформації та підвищує ефективність виконання запитів. На рисунку 3.1 наведено структуру бази даних вебзастосунку.

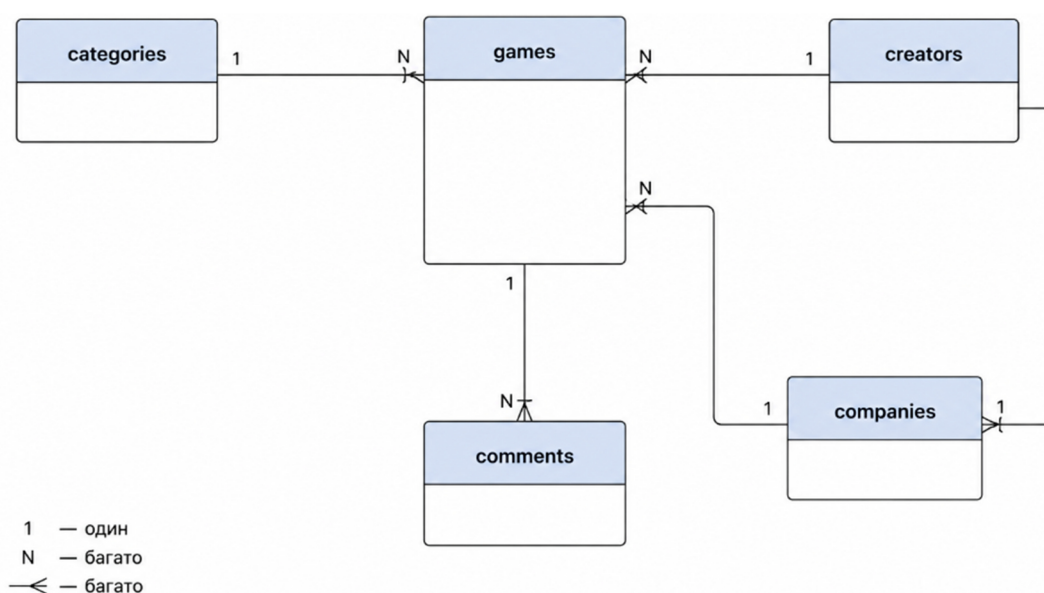


Рисунок 3.1 – Структура бази даних вебзастосунку

Для безпосередньої роботи з даними, у Laravel є об'єктно-реляційний обробник (ORM) Eloquent, що дозволяє використовувати об'єктно-орієнтований підхід. З Eloquent розробники можуть працювати з базою даних, використовуючи моделі, які представляють записи в базі даних як екземпляри класів PHP. Це спрощує виконання запитів до бази даних, тому що дозволяє писати їх у стилі PHP замість SQL. Для створення моделей використовується `artisan`-команда: `php artisan make:model Category`, яка створює модель `category` та надалі дає можливість взаємодіяти з базою даних. У процесі роботи вебзастосунку Laravel автоматично формує SQL-запити на основі викликів моделей Eloquent. Приклад SQL-запиту для отримання списку ігор певної категорії наведено у лістингу 3.4.

Лістинг 3.4 – SQL-запит фільтрації ігор за категорією

```
public function index(Request $request): View
{
    $query = Game::query();
    if ($request->filled('category')) {
        $query->where('category_id', $request->category);
    }
    $games = $query->paginate(12)->withQueryString();
    $categories = Category::all();
    return view('pages.game.games', compact('games', 'categories'));
}
```

кінець лістингу 3.4

Після реалізації структури бази даних, було налаштовано систему маршрутизації вебзастосунку. Для цього використовується файл «web.php», у якому описуються всі доступні маршрути системи. У вебзастосунку реалізовано маршрути для:

- головної сторінки;
- сторінки ігор;
- сторінки компаній;
- сторінки розробників;
- системи коментарів;
- адміністративної панелі.

Маршрути сайту були згруповані з використанням відповідних Middleware. Фрагмент реалізації маршрутизації наведено у лістингу 3.5.

Лістинг 3.5 – Реалізація маршрутизації вебзастосунку

```
Route::group(
[   'prefix' => LaravelLocalization::setLocale(),
    'middleware' => [ 'localeSessionRedirect', 'localizationRedirect',
'localeViewPath' ]
], function(){
    Route::prefix('games-4-you')->group(function (): void {
        Route::get('/', [HomeController::class, 'index']) ->name('home.page');
        Route::get('/creators', [PagesCreatorController::class,
'index'])->name('creator.page');
        Route::get('/company', [PagesCompanyController::class, 'index'])
->name('company.page');
        Route::get('/games', [GamesController::class, 'index'])
->name('games.page');
```

```

        Route::get('/games/{id}', [GamesController::class,
'show'])->name('games.show.page');
        Route::post('/comments', [CommentController::class,
'store'])->name('comments.store');
        Route::get('/category', [PageController::class, 'getGames'])
->name('games');    }); });
Route::middleware(['auth', 'verified'])
prefix('control')->group(function() {
    Route::get('/dashboard', [DashboardController::class,
'index'])->name('dashboard');
    Route::resource('companies', CompanyController::class);
    Route::resource('creator', CreatorController::class);
    Route::resource('category', CategoryController::class);
    Route::resource('game', GameController::class); });

```

кінець лістингу 3.5

Для підтримки багатомовності у проєкті було використано бібліотеку Laravel Localization. Вона дозволяє автоматично змінювати мову інтерфейсу залежно від вибраної локалізації користувача. Для перемикавання мови інтерфейсу користувача реалізовано мовний перемикач (language switcher), який дозволяє змінювати локалізацію вебзастосунку в режимі реального часу. Візуальне представлення перемикача мов наведено на рисунку 3.2.

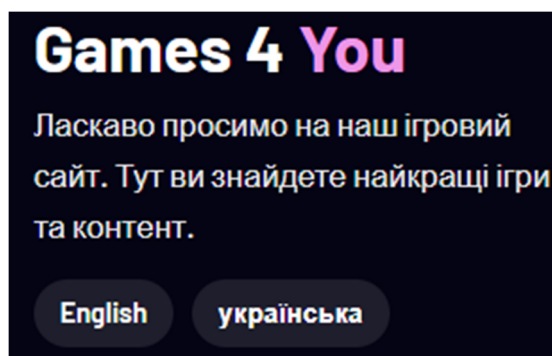


Рисунок 3.2 – Перемикач мов на сайті

Після реалізації маршрутизації було створено контролери для обробки HTTP-запитів користувачів. Основним контролером головної сторінки став «HomeController». У ньому реалізовано отримання списку категорій, вибір ігор за категоріями, пагінацію та виведення нових ігор. Фрагмент реалізації методу index контролера HomeController наведено в лістингу 3.6.

Лістинг 3.6 – Метод index контролера HomeController

```
public function index(Request $request): View
{
    $category_id = $request->get('category_id');
    $categories = Category::all();
    $gamesQuery = Game::latest();
    if ($category_id) {
        $gamesQuery->where('category_id', $category_id);
    }
    $games = $gamesQuery->paginate(6);
    $storeGame = Game::where('isNew', 1)->get();
    return view('pages.home.index', compact('categories',
'category_id', 'games', 'storeGame'));
}
```

кінець лістингу 3.6

У методі використовується модель «Game» для отримання списку ігор з бази даних. Додатково реалізовано фільтрацію за категоріями через параметр «category_id». Для відображення нових ігор використовується поле «isNew», яке міститься у таблиці «games».

Для реалізації сторінки розробників було створено контролер «CreatorController». Він отримує інформацію про розробників разом із пов’язаними іграми та компаніями. У лістингу 3.7 подано реалізацію методу index контролера CreatorController

Лістинг 3.7 – Метод index контролера CreatorController

```
public function index(): View
{
    $creators = Creator::with(['games', 'company'])->paginate(15);
    return view('pages.creator.creators', compact('creators'));
}
```

кінець лістингу 3.7

Аналогічно було створено контролер «CompanyController», який відповідає за відображення списку компаній. Для реалізації сторінки з усіма іграми був створений контролер «GamesController». У ньому реалізовано:

- отримання списку ігор;
- фільтрацію за категоріями;

- пагінацію;
- відкриття сторінки конкретної гри;
- завантаження коментарів до гри.

Фрагмент реалізації методу «show» контролера GameController представлено в лістингу 3.8.

Лістинг 3.8 – Метод show контролера GameController

```
public function show(int $id): View
{
    $game = Game::with(['creator', 'company'])->findOrFail($id);
    $comment = Comment::where('game_id', $id)->get();
    return view('pages.details.detail', compact('game', 'comment'));
}
```

кінець лістингу 3.8

При відкритті сторінки гри система завантажує інформацію про компанію, розробника та коментарі користувачів. Це дозволяє реалізувати повноцінну сторінку деталей гри. Для додавання коментарів було створено «CommentController». У ньому використовується валідація через окремий клас «CommentRequest», після чого коментар записується у базу даних. Реалізацію методу store контролера CommentController наведено в лістингу 3.9.

Лістинг 3.9 – Метод show контролера GameController

```
public function store(CommentRequest $commentRequest)
{
    Comment::create($commentRequest->all());
    return redirect()->back();
}
```

кінець лістингу 3.9

Отож, повертаючись до головної сторінки вебзастосунку, то вона була створена за допомогою шаблонізатора Blade. У ній реалізовано:

- банер головної сторінки;
- список популярних ігор;
- слайдер із зображеннями;
- блок Discord-спільноти;

- фільтрацію ігор за категоріями;
- блок нових ігор.

Відображення головної сторінки вебзастосунку показано на рисунку 3.3

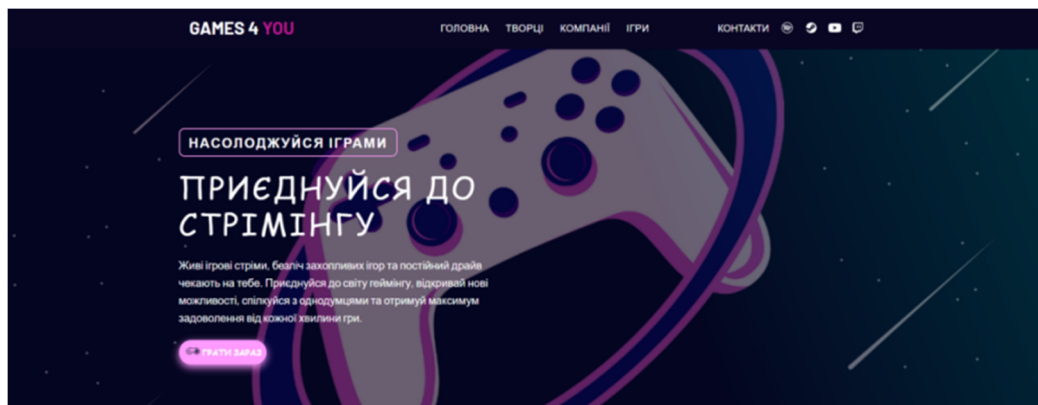


Рисунок 3.3 – Головна сторінка вебзастосунку

На ній присутній список категорій, який формується динамічно за допомогою Blade-шаблонізатора та даних, отриманих із бази даних. Після вибору певної категорії користувач переходить за відповідним маршрутом, де передається параметр «category_id». Даний параметр використовується для фільтрації списку ігор та відображення лише тих записів, які належать до вибраної категорії. Для реалізації інтерактивних елементів сторінки було використано JavaScript. За допомогою JavaScript реалізовано відкриття меню категорій, а також динамічне оновлення списку ігор без повного перезавантаження сторінки. Візуальне представлення меню категорій та перемикавання фільтрації ігор наведено на рисунку 3.4.

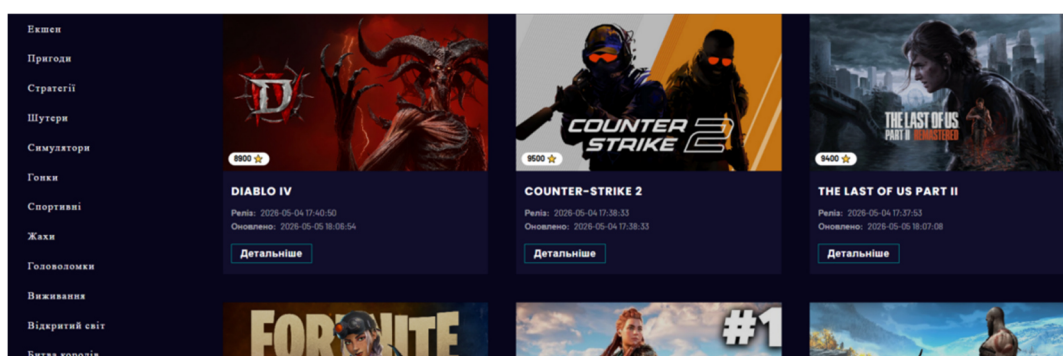


Рисунок 3.4 – Динамічний список з іграми

Додатково у проєкті було реалізовано Telegram-сповіщення. При створенні нової гри в адміністративній панелі система автоматично надсилає повідомлення у Telegram-чат. Це дозволяє оперативно інформувати користувачів або адміністраторів про появу нових ігор у каталозі. Реалізацію Telegram-сповіщення при створенні гри наведено в лістингу 3.10.

Лістинг 3.10 – Реалізація Telegram-сповіщення при створенні гри

```

public $token;
public $chat_id;
public function sendMessage($text)
{
    $token = config('services.telegram.token');
    $chatId = config('services.telegram.chat_id');
    return Http::withOptions([
        'verify' => false,
    ])->post("https://api.telegram.org/bot{$token}/sendMessage", [
        'chat_id' => $chatId,
        'text' => $text,
        'parse_mode' => 'HTML',
    ]);
}
public function sendNewGame(Game $game)
{
    $text = "🎮 Нова гра!\n\n"
        . "Назва гри: {$game->name}\n"
        . "Опис: \n{$game->description}\n"
        . "Рейтинг: {$game->rating} ★\n"
        . "Підтримка платформ: {$game->platforms}\n"
        . "Категорія: {$game->category->title}\n"
        . "Розробник: {$game->creator->first_name}
{$game->creator->last_name}\n"
        . "Компанія: {$game->company->name}\n";
    return $this->sendMessage($text);
}

```

кінець лістингу 3.10

Після реалізації користувацької частини вебзастосунку, було розроблено адміністративну панель для керування контентом сайту. Доступ до панелі адміністратора мають лише авторизовані користувачі. Для цього було використано middleware «auth». Для авторизації користувачів було використано стандартні контролери пакету Laravel Breeze. Маршрути авторизації знаходяться

у файлі «auth.php», які містять функціонал входу та виходу. Фрагмент маршрутів авторизації користувачів наведено в лістингу 3.11.

Лістинг 3.11 – Маршрути авторизації користувачів

```
Route::prefix('control')->group(function () {
    Route::middleware('guest')->group(function () {
        Route::get('login', [AuthenticatedSessionController::class,
        'create'])
            ->name('login');
        Route::post('login', [AuthenticatedSessionController::class,
        'store']);
    });
    Route::post('logout', [AuthenticatedSessionController::class,
    'destroy'])
        ->name('logout');
});
});
```

кінець лістингу 3.11

Усі маршрути самої адміністративної частини згруповано з префіксом «control». Фрагмент реалізації маршрутів адміністративної панелі наведено в лістингу 3.12.

Лістинг 3.12 – Маршрути адміністративної панелі

```
Route::middleware(['auth',
'verified'])->prefix('control')->group(function() {
    Route::get('/dashboard', [DashboardController::class, 'index'])
        ->name('dashboard');
    Route::resource('companies', CompanyController::class);
    Route::resource('creator', CreatorController::class);
    Route::resource('category', CategoryController::class);
    Route::resource('game', GameController::class);
});
```

кінець лістингу 3.12

Адміністративна панель реалізована за допомогою ресурсних контролерів Laravel. Це дозволило автоматично створити стандартний набір методів для роботи з даними: перегляд, створення, редагування та видалення записів. Головна сторінка адміністративної панелі реалізована через контролер

DashboardController, відображення головної сторінки адміністративної панелі, зображено на рисунку 3.5.

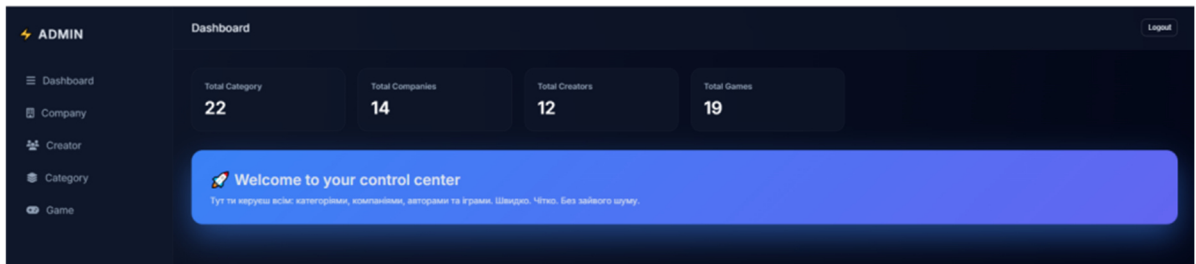


Рисунок 3.5 – Головна сторінка адміністративної панелі

На ній відображено статистичну інформацію про кількість ігор, категорій, компаній та розробників у системі.

Для роботи з компаніями було створено CompanyController. Контролер реалізує повний CRUD-функціонал для створення, редагування, перегляду та видалення компаній. На рисунку 3.6 показано відображення сторінки компанії.

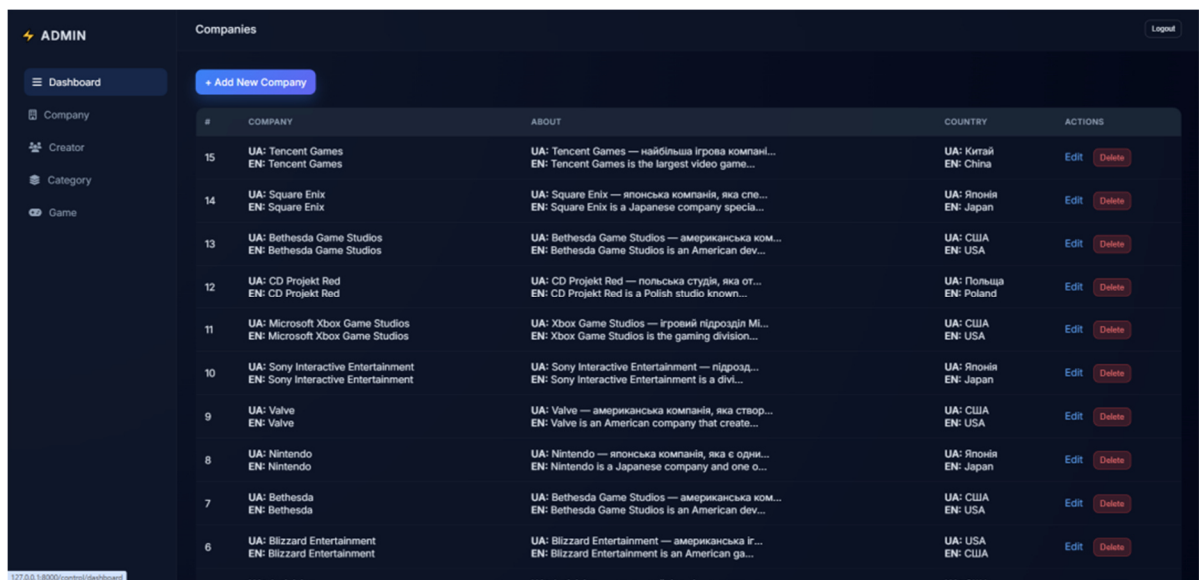


Рисунок 3.6 – Сторінка керування компаніями

Також реалізовано завантаження логотипів компаній на сервер та підтримку мультимовності за допомогою методу `setTranslations()`. Фрагмент реалізації частини контролера компаній наведено в лістингу 3.13.

Лістинг 3.13 – Частина контролера сторінки компаній

```

public function index()
{
    $company = Company::latest()->get();
    return view('dashboard.company.index', compact('company'));
}
public function store(StoreRequest $request)
{
    $data = $request->validated();
    unset($data['name'], $data['description'], $data['country']);
    $path = null;
    if ($request->hasFile('logo_path')) {
        $file = $request->file('logo_path');
        $filename = time().'_'.$file->getClientOriginalName();
        $file->move(public_path('uploads/company'), $filename);
        $path = 'uploads/company/'.$filename;
    }
    $company = new Company($data);
    $company->setTranslations('name', $request->input('name'));
    $company->setTranslations('description',
$request->input('description'));
    $company->setTranslations('country', $request->input('country'));
    $company->logo_path = $path; $company->save();
    return redirect()->route('companies.index');
}

```

кінець лістингу 3.13

Для керування категоріями було створено `CategoryController`. У ньому реалізовано створення, оновлення та видалення категорій ігор. Назви категорій підтримують мультимовність, що дозволяє відображати їх різними мовами інтерфейсу. Фрагмент реалізації контролера категорій наведено в лістингу 3.14.

Лістинг 3.14 – Контролер керування категоріями

```

public function index()
{
    $category = Category::all();
    return view('dashboard.category.index', compact('category'));
}
public function create()
{
    return view('dashboard.category.create');
}
public function store(StoreRequest $request)
{
    $category = new Category();
    $category->setTranslations('title', $request->input('title'));
    $category->save(); return redirect()->route('category.index');
}
public function edit(Category $category)
{
    return view('dashboard.category.edit', compact('category'));
}
public function update(UpdateRequest $request, Category $category)
{
    $category->setTranslations('title', $request->input('title'));
    $category->save(); return redirect()->route('category.index');
}
public function destroy(Category $category) { $category->delete();
    return redirect()->route('category.index');
}

```

кінець лістингу 3.14

Для керування розробниками було реалізовано CreatorController. Контролер дозволяє створювати профілі розробників, завантажувати аватари та прив'язувати розробників до компаній. На рисунку 3.7 показано сторінку керування розробниками.

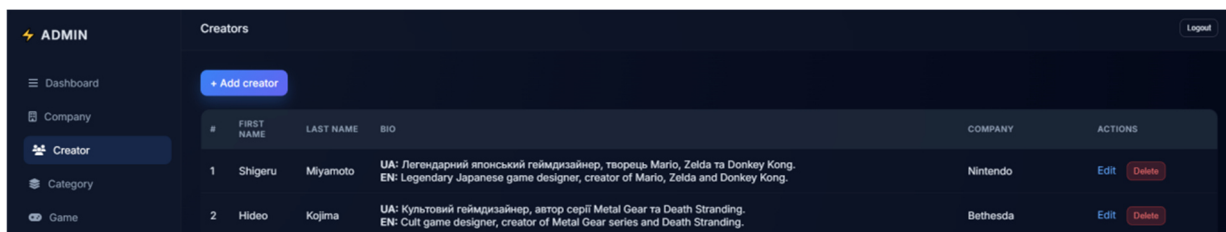


Рисунок 3.7 – Сторінка керування розробниками

Також у системі реалізовано мультимовні поля для опису та ролі розробника. У додатку А, подано фрагмент реалізації контролера для розробників. Основним модулем адміністративної панелі є керування відеоіграми, яке реалізоване через GameController. Контролер дозволяє додавати нові ігри, редагувати інформацію про них, завантажувати зображення та видаляти записи із системи, а також за завантаження обкладинок та інтеграцію із Telegram. Під час створення гри реалізовано зв'язок із категоріями, компаніями та розробниками через Eloquent ORM. Фрагмент реалізації контролера керування іграми наведено в додатку Б. Для створення інтерфейсів адміністративної панелі було використано шаблонізатор Blade та стилі CSS. Усі сторінки адміністративної частини мають єдиний дизайн та структуру, що забезпечує зручність роботи адміністратора із системою.

3.2 Тестування та налагодження вебзастосунку для каталогізації та коментування відеоігор

Процес тестування та налагодження є одним із найважливіших етапів розробки програмного забезпечення, оскільки саме він дозволяє забезпечити стабільність, надійність та коректність роботи вебзастосунку. Під час створення

системи каталогізації відеоігор було проведено комплексне тестування серверної частини, бази даних та користувацького інтерфейсу. Основною метою тестування було виявлення помилок у логіці роботи системи, перевірка взаємодії між модулями та забезпечення безпечної обробки даних користувачів.

Особлива увага приділялася перевірці фільтрації ігор за категоріями, отриманню даних із бази даних та коректному створенню коментарів користувачів. Також проводилося інтеграційне тестування, яке дозволяло перевірити взаємодію між серверною частиною вебзастосунку, базою даних MySQL та інтерфейсом користувача. У процесі тестування перевірялася коректність роботи ORM Eloquent, правильність виконання SQL-запитів та робота зв'язків між таблицями бази даних. Завдяки цьому вдалося забезпечити правильне відображення інформації про ігри, компанії, категорії та розробників.

Для тестування HTTP-маршрутів та API-запитів використовувався інструмент Postman. За його допомогою здійснювалася перевірка CRUD запитів, що дозволило протестувати передачу даних між клієнтською та серверною частинами системи. Також перевірялася робота форм додавання ігор, категорій, компаній та коментарів. Приклад тестування HTTP-запитів у середовищі Postman наведено на рисунку 3.8.

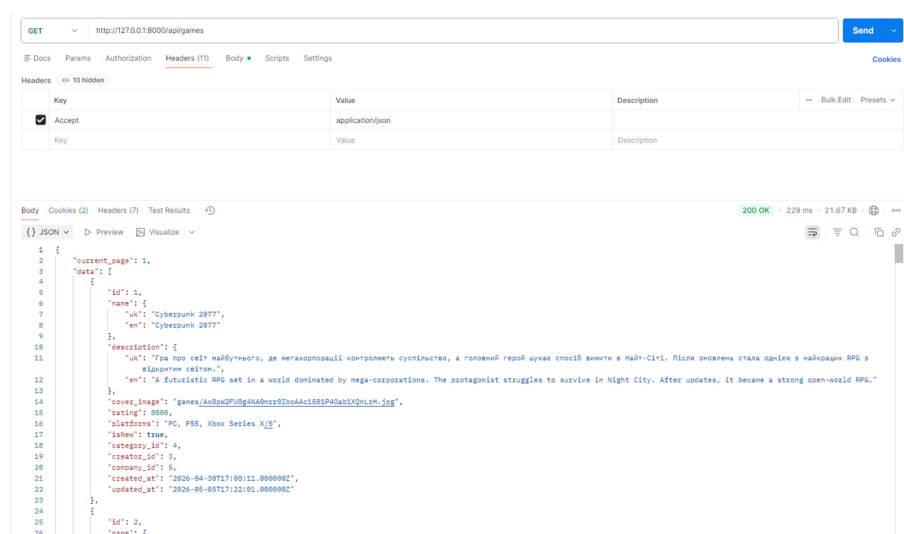
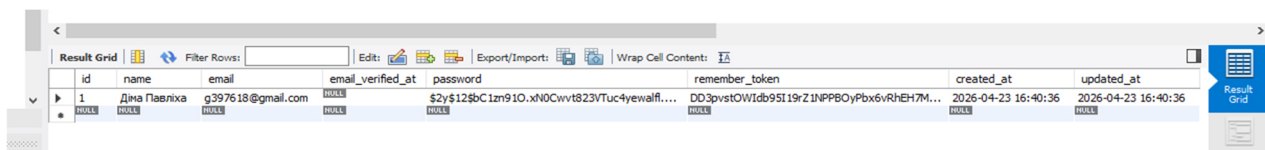


Рисунок 3.8 – Тестування HTTP-запитів у Postman

Особлива увага приділялася тестуванню системи автентифікації Laravel Breeze. Було перевірено:

- реєстрацію користувачів;
- авторизацію;
- вихід із системи.

Реалізацію правильної реєстрації користувача зображено на рисунку 3.9.



id	name	email	email_verified_at	password	remember_token	created_at	updated_at
1	Діна Павліха	g397618@gmail.com		\$2y\$12\$Bc1zn91O..xN0Cwvt823VTuc4yewalf....	DD3pvtOWIdb9SI19zZ1NPPBOyPbx6vRhEH7M...	2026-04-23 16:40:36	2026-04-23 16:40:36

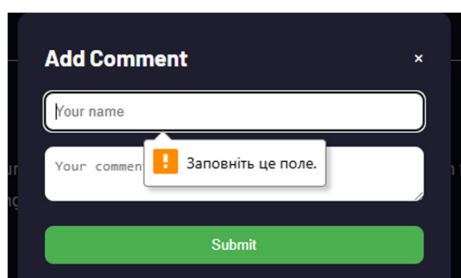
Рисунок 3.9 – Створення користувача в базі даних

Для захисту маршрутів використовувалися middleware «auth». Це дозволило забезпечити доступ до адміністративної частини лише для авторизованих користувачів.

Під час тестування також перевірялася валідація вхідних даних. Для цього використовувалися окремі класи Form Request, які виконують перевірку введеної інформації перед її збереженням у базі даних. Було протестовано:

- перевірку обов’язкових полів;
- обмеження довжини тексту;
- правильність завантаження зображень;
- захист від порожніх значень;
- перевірку форматів даних.

Правильна робота валідації зображена на рисунку 3.10.



Add Comment [X]

Your name [input field]

Your comment [input field] ! Заповніть це поле.

Рисунок 3.10 – Відображення валідації на формі коментарів

Для налагодження серверної частини використовувалася система логування Laravel. Усі помилки та службова інформація автоматично записувалися у файл «laravel.log», що дозволяло швидко знаходити причину несправностей та аналізувати роботу системи. Приклад логування помилок у Laravel наведено на рисунку 3.11.

```
#0 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\View\FileViewFinder.php(79): Illuminate\View\FileViewFinder->findInPaths()
#1 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\View\Factory.php(151): Illuminate\View\FileViewFinder->find()
#2 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Foundation\helpers.php(1062): Illuminate\View\Factory->make()
#3 C:\laravel\Game-4-You\game\routes\web.php(6): view()
#4 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Routing\CallableDispatcher.php(40): Illuminate\Routing\RouteFileRegistrar-
#5 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Routing\Route.php(244): Illuminate\Routing\CallableDispatcher->dispatch()
#6 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Routing\Route.php(215): Illuminate\Routing\Route->runCallable()
#7 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Routing\Router.php(808): Illuminate\Routing\Route->run()
#8 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Pipeline\Pipeline.php(144): Illuminate\Routing\Router->Illuminate\Routing
#9 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Routing\Middleware\SubstituteBindings.php(51): Illuminate\Pipeline\Pipeli
#10 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Pipeline\Pipeline.php(183): Illuminate\Routing\Middleware\SubstituteBindi
#11 C:\laravel\Game-4-You\game\vendor\laravel\framework\src\Illuminate\Foundation\Http\Middleware\VerifyCsrfToken.php(88): Illuminate\Pipeline\
```

Рисунок 3.11 – Логування помилок у файлі laravel.log

У процесі налагодження використовувалося середовище розробки Visual Studio Code. За допомогою вбудованих засобів дебагінгу виконувалася перевірка значень змінних, аналіз роботи контролерів та покрокове виконання програмного коду. Це дозволило значно спростити процес пошуку логічних помилок та оптимізувати окремі модулі системи.

Для тестування клієнтської частини вебзастосунку використовувалися інструменти розробника Google Chrome DevTools. Він дозволяє перевіряти HTML-структуру сторінок, CSS-стилі, JavaScript-скрипти та мережеві запити між клієнтською та серверною частинами системи. Приклад використання Chrome DevTools наведено на рисунку 3.12.

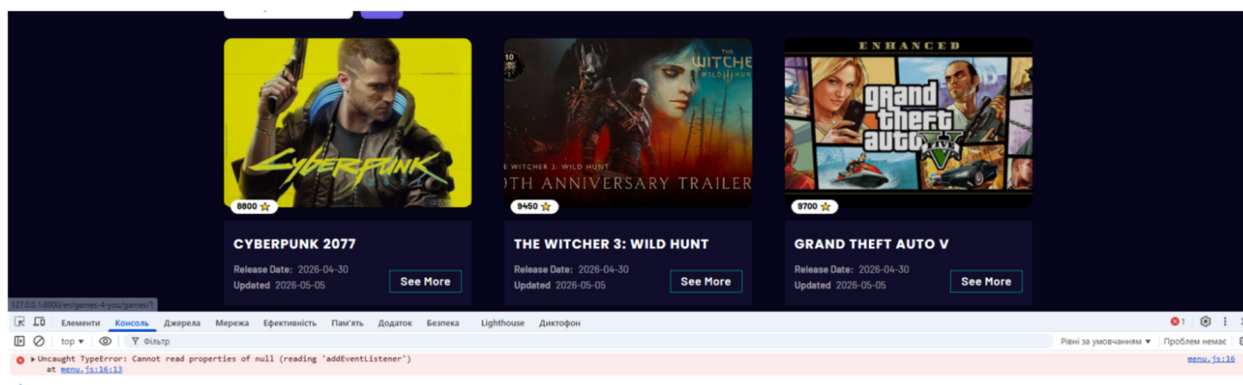


Рисунок 3.12 – Використання Chrome DevTools під час налагодження

Окремо проводилося тестування бази даних MySQL. Перевірялася правильність виконання міграцій, створення зовнішніх ключів та зв'язків між таблицями. Також тестувалося каскадне видалення записів і цілісність даних при взаємодії між таблицями «games», «categories», «companies», «creators» і «comments».

Під час розробки також проводилося тестування Telegram-сповіщень. Система надсилає повідомлення у Telegram-чат після створення нової гри через адміністративну панель. Було перевірено правильність формування повідомлення, передачу даних через Telegram Bot API та стабільність HTTP-запитів. Приклад Telegram-сповіщення наведено на рисунку 3.13.

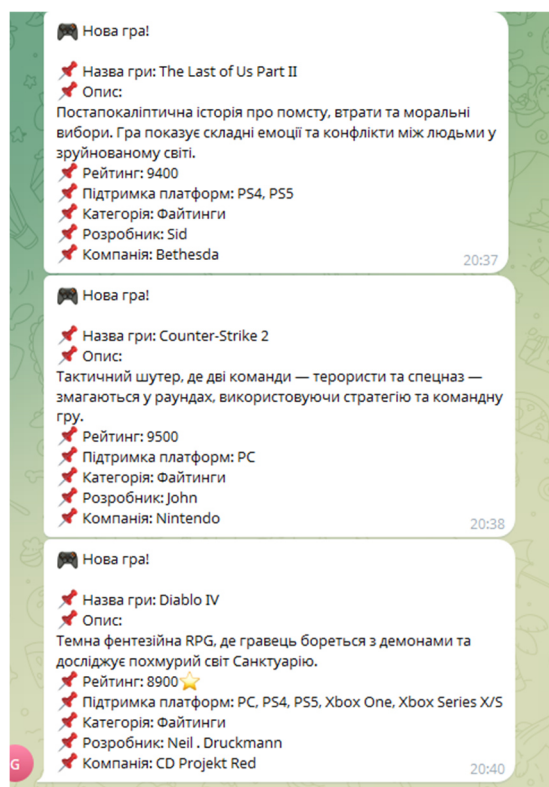


Рисунок 3.13 – Telegram-сповіщення про створення нової гри

У результаті проведеного тестування вдалося виявити та усунути помилки у роботі окремих модулів системи, оптимізувати взаємодію між компонентами та забезпечити стабільну роботу вебзастосунку для каталогізації відеоігор.

3.3 Інструкція користувача з інсталяції та використання програмного продукту

Для встановлення вебзастосунку каталогізації та коментування відеоігор необхідно попередньо встановити програмне забезпечення, яке використовується для роботи Laravel-проєкту. До основних компонентів належать:

- PHP 8.3;
- Composer 2;
- MySQL 8.0;
- Node.js 16;
- npm;
- Git.

Також рекомендується використовувати середовище розробки Visual Studio Code або інший редактор коду з підтримкою PHP та Laravel. Для локального запуску бази даних можна використовувати XAMPP, OpenServer або інші локальні серверні середовища.

Першим етапом є клонування репозиторію вебзастосунку з GitHub. Для цього потрібно відкрити термінал або командний рядок та виконати наступну команду: «git clone https://github.com/Grizerl/Game-4-You.git». Після завершення завантаження необхідно перейти до директорії проєкту за допомогою команди «cd game».

Після переходу до директорії вебзастосунку необхідно встановити всі серверні залежності Laravel. Для цього використовується команда «composer install». У результаті виконання даної команди автоматично встановлюються всі необхідні PHP-бібліотеки та пакети, описані у файлі composer.json.

Наступним етапом є встановлення JavaScript-залежностей, які використовуються для роботи стилів та компіляції фронтенд-ресурсів. Для цього використовується команда «npm install». Після виконання команди система

автоматично встановлює всі необхідні Node.js-пакети, які описані у файлі `package.json`.

Наступним кроком після встановлення залежностей необхідно створити файл конфігурації середовища `«.env»`. Для цього використовується копіювання шаблонного файлу `«.env.example»`. У файлі потрібно вказати параметри підключення до бази даних та змінити поля про, які описувалось раніше. Після налаштування файлу конфігурації необхідно згенерувати унікальний ключ Laravel-додатку. Для цього використовується `artisan`-команда: `«php artisan key:generate»`. Даний ключ використовується для шифрування службових даних, `cookie`-файлів та сесій користувачів. Далі необхідно створити структуру бази даних. Для цього потрібно попередньо створити порожню базу даних у MySQL та виконати міграції Laravel за допомогою команди `«php artisan migrate»`. Після виконання даної команди система автоматично створює всі необхідні таблиці.

Після створення структури бази даних необхідно запустити локальний сервер Laravel. Для цього використовується команда `«php artisan serve»`. Після запуску сервер буде доступний за локальною адресою `http://127.0.0.1:8000`.

Окремо необхідно запустити процес компіляції фронтенд-ресурсів. Для цього використовується команда `«npm run dev»`. Вона відповідає за компіляцію CSS-стилів, JavaScript-файлів та автоматичне оновлення ресурсів під час розробки.

Після відкриття вебзастосунку користувач отримує доступ до основного функціоналу системи. Незареєстровані користувачі можуть:

- переглядати список відеоігор;
- відкривати сторінки окремих ігор;
- переглядати інформацію про компанії та розробників;
- використовувати фільтрацію за категоріями;
- переглядати нові та популярні ігри.

Для доступу до адміністративної панелі користувачу необхідно пройти авторизацію. Після входу в систему адміністратор отримує доступ до функціоналу керування контентом вебзастосунку. Для переходу до

адміністративної панелі використовується адреса: <http://127.0.0.1:8000/control/dashboard>. Під час роботи системи адміністратор може використовувати багатомовний інтерфейс українською та англійською мовами. Перемикання мови здійснюється за допомогою language switcher у нижній частині сайту.

У разі виникнення помилок під час запуску вебзастосунку необхідно перевірити:

- правильність налаштування файлу «.env»;
- роботу сервера MySQL;
- правильність підключення до бази даних;
- встановлення Composer та Node.js;
- наявність усіх залежностей проєкту;
- коректність виконання Laravel-міграцій;
- запуск локального сервера Laravel;
- запуск команди «npm run dev».

Для пошуку помилок у Laravel використовується файл логування «laravel.log», який знаходиться у директорії storage/logs. У даному файлі записуються всі критичні помилки, винятки та системні повідомлення.

РОЗДІЛ 4

СПЕЦІАЛЬНІ РОЗРАХУНКИ

Під час розробки вебзастосунку для каталогізації та коментування відеоігор особливу увагу було приділено питанням інформаційної безпеки, захисту персональних даних користувачів та забезпеченню стабільної роботи системи. Оскільки вебзастосунок працює з базою даних, підтримує систему авторизації, адміністративну панель, завантаження зображень та взаємодію користувачів через систему коментарів, необхідно було реалізувати комплекс механізмів захисту від основних типів вебзагроз.

Для реалізації серверної частини системи використано фреймворк Laravel, який містить вбудовані засоби захисту та дозволяє значно підвищити безпеку вебзастосунку без необхідності реалізації складних алгоритмів вручну. Завдяки використанню Laravel вдалося реалізувати захищену систему автентифікації, контроль доступу до адміністративної панелі, перевірку вхідних даних, захист від SQL-ін'єкцій, CSRF-атак та інших потенційно небезпечних дій.

Одним із найважливіших елементів безпеки є система автентифікації користувачів. Для реалізації входу до системи використано пакет Laravel Breeze, який автоматично створює готову структуру авторизації та реєстрації користувачів. Даний пакет забезпечує:

- реєстрацію користувачів;
- вхід до системи;
- вихід із системи;
- захист сесій користувачів.

Під час реєстрації користувача система перевіряє коректність введених даних, після чого створює запис у таблиці users. Паролі користувачів не зберігаються у відкритому вигляді. Перед записом до бази даних вони автоматично хешуються за допомогою алгоритму bcrypt. Алгоритм bcrypt є одним із найбільш поширених способів захисту паролів, оскільки створює

складний криптографічний хеш, який практично неможливо відновити у вихідний вигляд.

Під час авторизації система виконує порівняння введеного пароля із його хешованим значенням у базі даних. Якщо значення збігаються, користувач отримує доступ до системи та створюється захищена сесія. Для захисту адміністративної панелі використано Middleware «auth». Middleware є спеціальним механізмом Laravel, який перевіряє запит користувача ще до виконання маршруту або контролера. Middleware «auth» перевіряє, чи виконав користувач вхід у систему.

У випадку відсутності авторизації користувач автоматично перенаправляється на сторінку входу. Завдяки цьому вдалося забезпечити додатковий рівень захисту від несанкціонованого доступу.

Адміністративні маршрути були об'єднані у спеціальну групу маршрутів із middleware-захистом. Це дозволило централізовано контролювати доступ до функціоналу керування системою. Доступ до адміністративної панелі мають лише авторизовані користувачі, що значно знижує ризик несанкціонованого втручання у роботу вебзастосунку.

Важливим елементом безпеки є перевірка вхідних даних користувачів. Усі форми у вебзастосунку проходять валідацію за допомогою спеціальних класів Request. У процесі перевірки система контролює:

- правильність введення текстових полів;
- довжину назви гри або коментаря;
- допустимий формат електронної пошти;
- типи та розмір файлів;
- обов'язковість заповнення полів.

Завдяки цьому вдається уникнути передачі некоректних або небезпечних даних до системи. Наприклад, при створенні гри система перевіряє наявність назви, опису, рейтингу та обкладинки. Якщо будь-яке поле заповнене неправильно, користувач отримує повідомлення про помилку.

Для захисту від SQL-ін'єкцій використовується ORM Eloquent. Замість ручного написання SQL-запитів застосовується об'єктно-орієнтований підхід роботи з базою даних. Це дозволяє автоматично екранувати параметри та безпечно передавати дані до MySQL. Крім цього, Laravel використовує prepared statements, підготовлені SQL-запити, які додатково зменшують ризик виконання шкідливого SQL-коду.

У вебзастосунку також реалізовано захист від CSRF-атак. Для захисту він автоматично генерує спеціальний CSRF-токен для кожної HTML-форми. Під час надсилання форми система перевіряє відповідність токена поточній сесії користувача. Якщо токен неправильний або відсутній, запит автоматично блокується.

Усі Blade-шаблони вебзастосунку використовують директиву @csrf, яка автоматично вставляє захисний токен у форму. Це забезпечує безпечну взаємодію між клієнтською та серверною частинами системи.

Окрему увагу було приділено захисту від XSS-атак. XSS-атаки дозволяють вставляти шкідливий JavaScript-код у сторінки вебзастосунку. Для запобігання таким атакам Laravel автоматично екранує HTML-вивід у Blade-шаблонах. Завдяки цьому текст, який вводить користувач, відображається як звичайний текст, а не виконується у браузері.

Додатково система перевіряє файли, які завантажуються користувачами. Для зображень дозволено лише безпечні формати, зокрема JPG, PNG та WEBP. Після перевірки файли зберігаються у спеціальних директоріях файлової системи Laravel. Це дозволяє уникнути завантаження потенційно небезпечних файлів або виконання стороннього коду на сервері.

У вебзастосунку також використовується система логування помилок. Laravel автоматично записує помилки, винятки та критичні події у файл «laravel.log», який знаходиться у директорії logs. Це дозволяє:

- відстежувати помилки під час роботи системи;
- аналізувати причини несправностей;
- перевіряти роботу API та маршрутизації;

— контролювати критичні події.

Під час тестування вебзастосунку логування активно використовувалось для пошуку помилок у маршрутах, контролерах та системі взаємодії з базою даних.

Для забезпечення безпечної передачі даних Laravel використовує механізм шифрування службових cookie-файлів та захист сесій користувачів. Усі сесії зберігаються у захищеному середовищі та автоматично перевіряються системою. Це дозволяє уникнути підміни сесій та викрадення даних авторизації.

Додатково у вебзастосунку реалізовано захист від масового присвоєння даних (Mass Assignment Protection). У моделях Laravel визначаються поля, які дозволено масово змінювати через методи `create()` або `update()`. Це дозволяє уникнути зміни критичних полів через сторонні HTTP-запити.

У процесі роботи вебзастосунку також використовуються механізми локалізації, які не впливають на безпеку системи, однак забезпечують коректне відображення контенту різними мовами. Для цього застосовано бібліотеку `Laravel Localization`.

Таким чином, у результаті використання сучасних засобів захисту фреймворку Laravel вдалося забезпечити високий рівень безпеки вебзастосунку. Реалізовані механізми авторизації, валідації, захисту маршрутів, захисту від SQL-ін'єкцій, CSRF та XSS-атак забезпечують стабільну роботу системи та захист персональних даних користувачів. Використання сучасних підходів до безпеки дозволило створити надійний вебзастосунок для каталогізації та коментування відеоігор із захищеною адміністративною панеллю та безпечним обміном даними між клієнтом і сервером.

ВИСНОВКИ

Результатом виконання даної кваліфікаційної роботи є розроблений вебзастосунок для каталогізації та коментування відеоігор «Game-4-You». У процесі створення системи було використано сучасні вебтехнології та інструменти, які забезпечили зручність користування, швидкодію та можливість подальшого розширення функціоналу.

Під час виконання роботи було проведено аналіз особливостей сучасних вебзастосунків для роботи з ігровим контентом та визначено актуальність створення системи, яка дозволяє користувачам переглядати інформацію про відеоігри, отримувати детальний опис, взаємодіяти з контентом та залишати коментарі.

У ході проектування було сформовано структуру вебзастосунку, яка включає:

- головну сторінку;
- каталог ігор;
- сторінку перегляду гри;
- систему коментарів;
- сторінки компаній та розробників;
- систему категорій;
- адміністративну панель.

Було розроблено структуру бази даних та організовано зв'язки між таблицями для зберігання інформації про ігри, категорії, компанії, розробників і коментарі. Для роботи з базою даних використано ORM Eloquent, що дозволило спростити взаємодію із записами та забезпечити зручну організацію логіки застосунку. У процесі розробки було реалізовано:

- систему автентифікації та авторизації користувачів;
- адміністративну панель для керування контентом;
- CRUD-функціонал для роботи з іграми, категоріями, компаніями та розробниками;

- систему мультимовності;
- завантаження та зберігання зображень;
- систему коментування;
- інтеграцію з Telegram для автоматичного надсилання повідомлень про нові ігри.

Під час створення вебзастосунку було застосовано можливості Laravel Middleware для захисту маршрутів та розмежування доступу між звичайними користувачами та адміністраторами. Також реалізовано механізми валідації даних для перевірки коректності введеної інформації.

У результаті виконання даної кваліфікаційної роботи було створено функціональний вебзастосунок із сучасним інтерфейсом та зручною структурою. Розроблена система може бути використана як основа для подальшого розвитку проєкту, зокрема додавання рейтингової системи, системи рекомендацій, особистих кабінетів користувачів та інших додаткових можливостей.

Також слід відзначити, що під час виконання кваліфікаційної роботи було отримано практичні навички роботи з сучасними вебтехнологіями, зокрема фреймворком Laravel, що включає роботу з маршрутизацією, контролерами, моделями та представленнями. Це дозволило краще зрозуміти принципи побудови сучасних вебзастосунків та організацію їх архітектури.

Окрім цього, було закріплено навички роботи з базами даних та ORM-системою Eloquent, що дало змогу ефективно реалізувати взаємодію між різними сутностями системи. Отриманий досвід проєктування структури бази даних є важливим для подальшої професійної діяльності у сфері веброзробки.

Важливим результатом також є практичне розуміння процесів реалізації систем автентифікації та авторизації користувачів, а також побудови адміністративних панелей для керування контентом. Це є ключовими елементами більшості сучасних вебзастосунків.

Крім того, під час розробки було вдосконалено навички роботи з клієнт-серверною архітектурою, обробкою HTTP-запитів та реалізацією бізнес-логіки

на стороні сервера. Це дозволило сформувати цілісне розуміння принципів роботи вебзастосунків у реальних умовах.

Отримані результати можуть бути використані як основа для подальшого розвитку проєкту та впровадження нових функціональних можливостей, що дозволить підвищити зручність та ефективність використання системи.

Таким чином, усі поставлені завдання були виконані, а основна мета роботи – розробка вебзастосунку для каталогізації та коментування відеоігор успішно досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Steam – сервіс компанії Valve. URL: <https://uk.wikipedia.org/wiki/Steam> (дата звернення: 10.02.2026).
2. IGN – розважальний вебсайт. URL: <https://uk.wikipedia.org/wiki/IGN> (дата звернення: 13.02.2026).
3. Metacritic – американський вебсайт для огляду фільмів. URL: <https://en.wikipedia.org/wiki/Metacritic> (дата звернення: 15.02.2026).
4. Epic Games – американська компанія по розробці відеоігор. URL: https://en.wikipedia.org/wiki/Epic_Games (дата звернення: 21.02.2026).
5. Laravel Documentation – офіційна документація фреймворку Laravel. URL: <https://laravel.com/docs/13.x> (дата звернення: 10.03.2026).
6. Laravel – архітектурний шаблон MVC (Model-View-Controller). URL: <https://laravel.com/learn/getting-started-with-laravel/what-is-mvc> (дата звернення: 18.03.2026).
7. Composer – офіційна документація мови програмування PHP. URL: <https://getcomposer.org/> (дата звернення: 22.03.2026).
8. PHP Documentation – офіційна документація мови програмування PHP. URL: https://www.php.net/docs.php?utm_source=chatgpt.com (дата звернення: 25.03.2026).
9. MySQL Documentation – офіційна документація системи керування базами даних MySQL. URL: https://dev.mysql.com/doc/?utm_source=chatgpt.com (дата звернення: 08.04.2026).
10. Laravel Localization – документація мультимовності у Laravel. URL: <https://laravel.com/docs/13.x/localization> (дата звернення: 18.04.2026).
11. Laravel Breeze Documentation – документація щодо системи автентифікації Laravel Breeze. URL: <https://laravel.com/docs/13.x/starter-kits#laravel-breeze/> (дата звернення: 24.04.2026).
12. Laravel Blade Templates – документація Blade-шаблонізатора. URL: <https://laravel.com/docs/13.x/blade> (дата звернення: 28.04.2026).

13. Laravel Eloquent ORM – документація ORM Eloquent для роботи з базою даних. URL: <https://laravel.com/docs/13.x/eloquent> (дата звернення: 05.05.2026).

ДОДАТКИ

Додаток А

Контролер керування розробниками

```

public function index()
{
    $creator = Creator::all();
    return view('dashboard.creators.index', compact('creator'));
}

public function create()
{
    $companies = Company::select('id', 'name')->get();
    return view('dashboard.creators.create', compact('companies'));
}

public function store(StoreRequest $request)
{
    $data = $request->validated();

    unset($data['bio'], $data['role']);

    if ($request->hasFile('avatar')) {
        $data['avatar'] = $request->file('avatar')
            ->store('creators', 'public');
    }

    $creator = new Creator($data);

    $creator->setTranslations('bio', $request->input('bio'));
    $creator->setTranslations('role', $request->input('role'));

    $creator->save();

    return redirect()->route('creator.index');
}

public function update(UpdateRequest $request, Creator $creator)
{
    $data = $request->validated();

    unset($data['bio'], $data['role']);

    if ($request->hasFile('avatar'))
    {
        if ($creator->avatar) {
            Storage::disk('public')->delete($creator->avatar);
        }
    }
}

```

```
        $data['avatar'] = $request->file('avatar')->store('creators',  
'public');  
    }  
  
    $creator->update($data);  
  
    $creator->setTranslations('bio', $request->input('bio'));  
    $creator->setTranslations('role', $request->input('role'));  
  
    $creator->save();  
  
    return redirect()->route('creator.index');  
}
```

Додаток Б

Контролер керування іграми

```

/**
 * Display a listing of the resource.
 */
public function index()
{
    $game = Game::all();
    return view('dashboard.game.index', compact('game'));
}

/**
 * Show the form for creating a new resource.
 */
public function create()
{
    $categories = Category::select('id', 'title')->get();
    $creators = Creator::select('id', 'first_name')->get();
    $companies = Company::select('id', 'name')->get();
    return view('dashboard.game.create', compact('categories',
'creators', 'companies'));
}

/**
 * Store a newly created resource in storage.
 */
public function store(StoreRequest $request, TelegramService
$telegram)
{
    $data = $request->validated();

    unset($data['name'], $data['description']);

    if ($request->hasFile('cover_image')) {
        $data['cover_image'] =
$request->file('cover_image')->store('games', 'public');
    }

    $game = new Game($data);

    $game->setTranslations('name', $request->input('name'));
    $game->setTranslations('description',
$request->input('description'));

    $telegram->sendNewGame($game);
    $game->save();

    return redirect()->route('game.index');
}

```

```

}

/**
 * Show the form for editing the specified resource.
 */
public function edit(Game $game)
{
    $categories = Category::select('id', 'title')->get();
    $creators = Creator::select('id', 'first_name')->get();
    $companies = Company::select('id', 'name')->get();
    return view('dashboard.game.edit', compact('game', 'categories',
'creators', 'companies'));
}

/**
 * Update the specified resource in storage.
 */
public function update(UpdateRequest $request, Game $game)
{
    $data = $request->validated();

    unset($data['name'], $data['description']);

    if ($request->hasFile('cover_image')) {
        if ($game->cover_image) {
            Storage::disk('public')->delete($game->cover_image);
        }

        $data['cover_image'] =
$request->file('cover_image')->store('games', 'public');
    }

    $game->update($data);

    $game->setTranslations('name', $request->input('name'));
    $game->setTranslations('description',
$request->input('description'));

    $game->save();

    return redirect()->route('game.index');
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(Game $game)
{
    if ($game->cover_image &&
file_exists(public_path($game->cover_image))) {
        unlink(public_path($game->cover_image));
    }
}

```

```
}  
$game->delete();  
return redirect()->route('game.index');  
}
```