**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ**
**Кафедра програмної інженерії**

# КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНОГО ВЕБЗАСТОСУНКУ ЗАСОБАМИ SYMFONY»**

Виконала: студентка 2 курсу, групи 8.1213-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
М.О. Суворова
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

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

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

# З А В Д А Н Н Я

## НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Суворовій Марії Олександрівні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційно-комунікаційного вебзастосунку засобами Symfony

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Огляд технологій та інструментів розробки.
3. Проєктування застосунку.
4. Розробка та тестування.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 07.03.2024 р.

## К А Л Е Н Д А Р Н И Й  П Л А Н

| № | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Розробка плану роботи. | 18.03.2024 | |
| | | | |
| 2. | Збір вихідних даних. | 13.05.2024 | |
| | | | |
| 3. | Обробка методичних та теоретичних | 28.06.2024 | |
| | джерел. | | |
| | | | |
| 4. | Розробка першого та другого розділу. | 09.09.2024 | |
| | | | |
| 5. | Розробка третього та четвертого розділу. | 28.10.2024 | |
| | | | |
| 6. | Оформлення та нормоконтроль | 20.11.2024 | |
| | кваліфікаційної роботи магістра. | | |
| | | | |
| 7. | Захист кваліфікаційної роботи. | 13.12.2024 | |

Студент _____ М.О. Суворова
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

## Нормоконтроль пройдено

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

# РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка інформаційно-комунікаційного вебзастосунку засобами Symfony»: 53 с., 20 рис., 1 табл., 10 джерел, 1 додаток.

ВЕБЗАСТОСУНОК, КОМІКСИ, СИСТЕМА УПРАВЛІННЯ КОНТЕНТОМ, СТВОРЕННЯ ЗАСТОСУНКУ, ТЕХНОЛОГІЇ ВЕБРОЗРОБКИ, ФРЕЙМВОРК SYMFONY, CSS, HTML, JAVASCRIPT ,MVC.

Об’єкт дослідження – створення вебзастосунків для читання графічних романів онлайн.

Предмет дослідження – створення вебзастосунків з використання фреймворку Symfony.

Мета роботи – розробити вебзастосунок для читання коміксів, який забезпечить зручний інтерфейс, ефективну роботу з базою даних, а також авторизацію та аутентифікацію користувачів.

Метод дослідження – аналіз існуючих платформ для читання коміксів, проєктування вебзастосунку, розробка та впровадження основних функціональних елементів через використання технологій Symfony, HTML, CSS, JavaScript, а також практичне застосування бази даних MySQL.

У кваліфікаційній роботі розглянуто розробку вебзастосунку для читання коміксів з використанням фреймворку Symfony. Розглядається проєктування структури даних через моделі та взаємодія з базою даних MySQL, а також реалізація архітектури MVC та маршрутизації. Методи дослідження включають аналіз існуючих рішень на ринку, проєктування системи та практичну реалізацію ключових компонентів вебзастосунку.

# SUMMARY

Master’s qualifying paper “Development of an Information and Communication WebAPP Using Symfony”: 53 pages, 20 figures, 1 table, 10 references, 1 supplement.

APPLICATION, APPLICATION DEVELOPMENT, COMICS, CONTENT MANAGEMENT SYSTEM, CSS, HTML, JAVASCRIPT, MVC, WEB DEVELOPMENT TECHNOLOGIES, WEB SYMFONY FRAMEWORK.

Object of research – a web application for reading comics, created using the Symfony framework.

The aim of the work – development of a web application for reading comics, which will provide a user-friendly interface, efficient database management, and user authentication and authorization.

Research method – analysis of existing comic reading platforms, web application design, development and implementation of key functional elements using Symfony, HTML, CSS, JavaScript, and practical use of the MySQL database.

The qualification work examines the development of a web application for reading comics using the Symfony framework. It addresses the design of the data structure through models and interaction with a MySQL database, as well as the implementation of the MVC architecture and routing. The research methods include the analysis of existing market solutions, system design, and the practical implementation of key components of the web application.

**ЗМІСТ**

Завдання на кваліфікаційну роботу ……..……..2
Реферат ..... 4
Summary ..... 5
Вступ ..... 8
1 Аналіз предметної області та постановка задачі ..... 10
1.1 Характеристика коміксів як медіа-продукту та їх популярність ..... 10
1.2 Огляд існуючих платформ для читання коміксів ..... 11
1.2.1 ReManga ..... 11
1.2.2 MangaLib ..... 12
1.2.3 Порівняння платформ ..... 13
1.3 Вимоги до вебзастосунку ..... 14
1.4 Постановка задачі розробки ..... 16
1.5 Висновки до розділу 1 ..... 17
2 Технології та інструменти розробки ..... 18
2.1 Огляд фреймворку Symfony та його можливостей ..... 18
2.2 Робота з базами даних (MySQL) ..... 19
2.2.1 Створення структури бази даних ..... 19
2.2.2 Інтеграція з Symfony через Doctrine ORM ..... 20
2.2.3 Оптимізація запитів та продуктивність ..... 20
2.3 Бібліотеки та сторонні інструменти ..... 20
2.4 Системи управління проєктами (Jira) ..... 22
2.5 Висновки до розділу 2 ..... 23
3 Проєктування вебзастосунку ..... 25
3.1 Основні діаграми проєктування ..... 25
3.1.1 ER-діаграмма ..... 25
3.1.2 Діаграма класів ..... 27
3.1.3 Діаграма послідовності ..... 30

3.2 Висновки до розділу 3 ..... 32
4 Розробка вебзастосунку ..... 34
4.1 Розробка бекенд-частини ..... 34
4.1.1 Основи архітектури MVC у Symfony ..... 35
4.1.2 Розробка моделей ..... 36
4.1.3 Розробка контролерів ..... 38
4.1.4 Маршрутизація ..... 39
4.1.5 Реалізація авторизації та аутентифікації ..... 41
4.2 Розробка фронтенд-частини ..... 44
4.3 Тестування ..... 45
4.4 Висновки до розділу 4 ..... 46
Висновки ..... 48
Перелік посилань ..... 50
Додаток А Сторінки застосунку ..... 52

# ВСТУП

У сучасному світі інформаційні технології стрімко інтегруються в різні аспекти нашого життя, включаючи розваги. Одним із популярних способів проведення дозвілля стало читання коміксів, які, завдяки своїй яскравій візуальній складовій, приваблюють широкий спектр аудиторії – від дітей до дорослих. У зв'язку з цим зростає попит на сучасні цифрові платформи, які дозволяють зручно й ефективно організовувати процес читання коміксів онлайн.

Мета – розробити вебзастосунок для читання коміксів, який забезпечить зручний інтерфейс, ефективну роботу з базою даних, а також авторизацію та аутентифікацію користувачів.

Досягнення поставленої мети передбачає реалізацію наступних завдань:

– проаналізувати існуючих конкурентів, виокремити плюси та мінуси їх реалізацій;
– дослідити існуючі фреймворки та вибрати найбільш вдалий;
– спроєктувати базу даних та систему і відобразити це на діаграмах;
– розробити план робот та реалізувати вебзастосунок;
– протестувати кінцевий сайт.

Об'єкт дослідження – створення вебзастосунків для читання графічних романів онлайн.

Предмет дослідження – створення вебзастосунків з використання фреймворку Symfony.

Метод дослідження – аналіз існуючих платформ для читання коміксів, проєктування вебзастосунку, розробка та впровадження основних функціональних елементів через використання технологій Symfony, HTML, CSS, JavaScript, а також практичне застосування бази даних MySQL.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, переліку посилань та одного додатку. Загальний обсяг роботи становить 53 сторінки. У переліку посилань 10 використаних джерел.

Перший розділ присвячений аналізу предметної області та постановці задачі. У ньому розглядається характеристика коміксів як медіа-продукту, їх популярність, а також проведено огляд існуючих платформ для читання коміксів, таких як Remanga та Mangalib, із їх порівнянням. Також наведено вимоги до вебзастосунку та сформульовано задачі розробки.

У другому розділі наведено огляд технологій і інструментів розробки. Проаналізовано можливості фреймворку Symfony, роботу з базами даних MySQL, створення структури бази даних, інтеграцію через Doctrine ORM, а також оптимізацію запитів. Описано використання сторонніх бібліотек та інструментів, таких як Jira, для управління проєктами.

У третьому розділі проілюстровано процес проєктування вебзастосунку. Наведено основні діаграми проєктування, включаючи ER-діаграми, діаграми класів та послідовності, що демонструють взаємодію основних компонентів системи.

У четвертому розділі описано реалізацію вебзастосунку. Деталізовано розробку бекенд-частини з використанням архітектури MVC у Symfony, створення моделей, контролерів, маршрутизації, а також впровадження авторизації та аутентифікації. Описано розробку фронтенд-частини, що забезпечує зручний інтерфейс користувача.

В додатку наведено скриншоти основних сторінок розробленого вебзастосунку.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Характеристика коміксів як медіа-продукту та їх популярність

Комікси є унікальним видом медіа-продукту, що поєднує в собі текстовий та візуальний контент. Завдяки цій особливості вони легко сприймаються різними аудиторіями, незалежно від віку чи культурного контексту. У коміксах текстові повідомлення, як правило, лаконічні, а основний акцент робиться на ілюстраціях, що передають емоції, атмосферу та динаміку сюжету.

Перші комікси виникли ще наприкінці XIX століття, але справжній «золотий вік» цього жанру припав на середину XX століття. У той час комікси стали популярними завдяки супергеройським історіям від таких компаній, як Marvel та DC. Згодом жанрове різноманіття коміксів значно розширилося, охоплюючи наукову фантастику, пригоди, драму, фентезі, гумор тощо.

Сьогодні комікси є не лише способом розваги, але й важливим культурним явищем. Вони слугують засобом для передачі соціальних ідей, політичних меседжів та навіть історичних подій. Наприклад, графічні новели, як більш серйозний піджанр коміксів, часто порушують складні теми, такі як війна, права людини чи екологічні проблеми.

Популярність коміксів зростає завдяки екранізаціям. Фільми та серіали, створені за мотивами коміксів, залучають нову аудиторію до цього виду мистецтва. Крім того, розвиток цифрових технологій дозволив зробити комікси доступними онлайн, що значно розширило їхню популярність у світі.

Таким чином, комікси є універсальним і багатофункціональним медіа-продуктом, який адаптується до сучасних умов і продовжує приваблювати мільйони людей у всьому світі. У рамках цієї роботи буде створено платформу, яка дозволить користувачам зручно читати комікси онлайн, популяризуючи цей жанр ще більше.

## 1.2 Огляд існуючих платформ для читання коміксів

На сьогоднішній день існує багато платформ для читання коміксів онлайн, які пропонують широкий функціонал і доступ до великої кількості контенту. Серед популярних платформ, які привертають увагу користувачів, можна виділити ReManga та MangaLib.

### 1.2.1 ReManga

ReManga – це популярна платформа для шанувальників манги, вебтунів та коміксів, яка вирізняється своїм зручним сучасним інтерфейсом і великим асортиментом контенту. Вона пропонує доступ до сотень тисяч назв, що охоплюють як відомі хіти, так і менш популярні твори, включаючи рідкісні видання. Завдяки потужній системі фільтрації, користувачі можуть легко знаходити твори за жанром, рейтингом чи популярністю, що забезпечує інтуїтивність і комфорт у використанні.

Особливістю платформи є її інтерактивність: користувачі можуть ділитися відгуками, оцінками, а також спілкуватися з іншими читачами у форумному форматі, створюючи живу спільноту. Крім цього, доступна функція персоналізації, яка дозволяє зберігати улюблені твори, формувати списки для читання та стежити за виходом нових глав. Завдяки адаптації для мобільних пристроїв, ReManga надає можливість читати з комфортом як на смартфоні, так і на планшеті, що робить її універсальною і зручною.

Попри численні переваги, платформа має деякі недоліки. Зокрема, реклама іноді може переривати процес читання, а частина контенту залишається недоступною без реєстрації. Однак ці аспекти компенсуються якістю наданих послуг і розмаїттям контенту, що робить ReManga популярною серед любителів манги та коміксів у різних куточках світу.

### 1.2.2 MangaLib

MangaLib – це одна з найпопулярніших платформ для шанувальників манги та коміксів, яка привертає увагу активною спільнотою і багатим функціоналом. Платформа не тільки надає зручний доступ до численних серій, але й стимулює читачів до взаємодії: користувачі можуть брати участь у створенні перекладів, обговорювати улюблені серії та ділитися враженнями. Це створює атмосферу, в якій кожен може відчути себе частиною великої і згуртованої спільноти.

Навігація на сайті побудована таким чином, щоб пошук необхідного контенту був максимально швидким і комфортним. Користувачі можуть знаходити твори за іменем автора, жанром або статусом (наприклад, завершений чи такий, що ще оновлюється). Система рекомендацій платформи автоматично підбирає нові комікси на основі вподобань користувача, що значно розширює можливості відкриття нового контенту. Важливу роль відіграють соціальні функції, які дозволяють додавати друзів, приєднуватися до тематичних груп і обговорювати сюжетні лінії, створюючи живе середовище для обміну думками.

Крім того, платформа пропонує цікаві інструменти для відстеження прогресу. Інтерактивні статистики показують, які серії були прочитані, а які ще залишаються в очікуванні. Це допомагає користувачам організовувати своє читання та не пропускати важливих оновлень. Завдяки такому підходу MangaLib є не просто місцем для читання, а справжнім цифровим простором для фанатів манги.

Проте є й деякі недоліки, які можуть вплинути на досвід користування. Наприклад, частина матеріалів недоступна для незареєстрованих користувачів, а велика популярність платформи іноді призводить до технічних збоїв або уповільнення роботи через високе навантаження на сервери. Незважаючи на це, MangaLib залишається одним із провідних ресурсів для тих, хто цінує якісний контент і активну взаємодію з однодумцями.

### 1.2.3 Порівняння платформ

Обидві платформи мають схожий функціонал, спрямований на зручність користувачів і багатий контент. Однак ReManga більше акцентує увагу на простоті й швидкості доступу до контенту, тоді як MangaLib створює атмосферу соціальної взаємодії та спільноти.

Обидва сайти демонструють приклад якісних рішень для читання коміксів онлайн, але їх функціонал і дизайн можуть бути ще більше вдосконалені. Це надихає на створення нової платформи, яка поєднуватиме кращі риси цих рішень і запропонує унікальний досвід для користувачів. Детальніше у таблиці 1.1.

Таблиця 1.1 – Порівняння платформ читання манги

| **Критерій** | **ReManga** | **MangaLib** |
|---|---|---|
| Аудиторія | Любителі манги, вебтунів та інших видів коміксів | Фанати манги, активна спільнота користувачів |
| Інтерфейс | Сучасний, простий, з фільтрами за жанрами, рейтингом і популярністю | Зручна навігація з пошуком за авторами, жанрами та статусом |
| Контент | Велика бібліотека, включаючи популярні й рідкісні видання | Широкий вибір манги з активною системою перекладів |
| Рекомендації | Немає розширених рекомендацій | Система рекомендацій на основі вподобань користувача |
| Соціальні функції | Обговорення через відгуки, оцінки | Дружні взаємодії: групи, друзі, обговорення |
| Мобільна адаптивність | Висока, зручна для читання на смартфонах і планшетах | Адаптована, але може бути менш стабільною через технічне навантаження |
| Персоналізація | Створення списків улюблених коміксів | Відстеження прогресу читання, персоналізовані списки |
| Реєстрація | Потрібна для доступу до деякого контенту | Необхідна для доступу до розширених функцій |

Продовження таблиці 1.1

| **Критерій** | **ReManga** | **MangaLib** |
|---|---|---|
| Реклама | Є, інколи заважає читанню | Менше реклами, але платформа залежить від технічного стану серверів |
| Особливості | Проста у використанні, акцент на швидкості доступу | Активна спільнота, інтерактивні статистики та соціальний елемент |
| Недоліки | Наявність реклами, обмежений доступ без реєстрації | Можливі технічні проблеми через велике навантаження |

Обидві платформи мають свої сильні сторони. **ReManga** більше орієнтована на швидкий доступ до контенту, тоді як **MangaLib** надає додаткові можливості для соціальної взаємодії та персоналізації. Це показує, що в новій платформі для читання коміксів можна поєднати переваги обох рішень, створивши ще більш комфортний і функціональний продукт.

### 1.3 Вимоги до вебзастосунку

Розробка вебзастосунку для читання коміксів – це складний процес, що потребує врахування численних аспектів, аби забезпечити зручність, ефективність і безпеку використання. Основними завданнями є створення функціоналу, який дозволить користувачам легко знаходити та читати комікси, а також розробка технічної бази, здатної підтримувати стабільну роботу системи при високому навантаженні.

Однією з ключових особливостей такого застосунку є організація контенту. Система повинна пропонувати інструменти для каталогізації коміксів за різними критеріями, як-от жанр, автор, популярність, що дозволить користувачам швидко знаходити те, що їх цікавить. Інтеграція функції пошуку

за ключовими словами та розширених фільтрів стане невід'ємною частиною зручного інтерфейсу. Важливою складовою є й підтримка різних режимів читання: від покрокового перегляду до безперервного, з можливістю масштабування сторінок для комфортного огляду дрібних деталей.

Ще одним важливим аспектом є взаємодія користувачів із системою через персоналізовані профілі. Реєстрація та авторизація через email або соціальні мережі відкривають можливість зберігати історію читання, налаштування та вподобані комікси. Система рекомендацій, що базується на вподобаннях користувача, допоможе відкривати новий цікавий контент, підвищуючи рівень залученості. Крім цього, інтерактивні функції, такі як залишення відгуків, оцінки та можливість ділитися коміксами, сприяють формуванню активної спільноти користувачів.

Технічна сторона також відіграє важливу роль. Продуктивність системи повинна забезпечувати швидке завантаження контенту незалежно від кількості одночасних користувачів. Вебзастосунок має бути кросплатформеним, працювати на ПК, планшетах і смартфонах, забезпечуючи адаптивність дизайну для різних розмірів екранів. Безпека даних – це обов'язковий компонент, що включає захист персональної інформації користувачів і механізми запобігання витоку даних чи хакерським атакам.

Крім того, важливо передбачити масштабованість платформи, щоб вона могла розширювати свій функціонал і обслуговувати дедалі більшу кількість користувачів без втрати продуктивності. Стабільність роботи є критичною умовою для будь-якого успішного застосунку, тому потрібно мінімізувати час простоїв і забезпечити безперебійний доступ до контенту.

З урахуванням усіх цих аспектів розробка вебзастосунку здатна не лише задовольнити потреби сучасних користувачів, але й сприяти популяризації коміксів як важливого медіа-продукту. Ефективна реалізація подібного проєкту допоможе створити платформу, яка стане комфортним цифровим середовищем для шанувальників цього виду мистецтва.

## 1.4 Постановка задачі розробки

Метою даної роботи є розробка сучасного вебзастосунку, який відкриватиме доступ до багатої бібліотеки коміксів, забезпечуватиме зручність їх читання та сприятиме інтерактивній взаємодії користувачів. Застосунок повинен об’єднувати функціонал, що дозволить зберігати улюблені твори, читати їх на будь-яких пристроях та обмінюватися враженнями з іншими шанувальниками.

Для створення такої системи необхідно збудувати надійну структуру бази даних, яка забезпечить збереження всіх необхідних даних, зокрема інформації про комікси, їх авторів, жанри та сторінки. Важливо також передбачити зберігання профілів користувачів із можливістю запису історії їх читання, відгуків і персоналізованих списків.

Інтерфейс користувача відіграє центральну роль у забезпеченні зручності використання. Він має бути адаптивним, що гарантує комфортну роботу як на настільних комп’ютерах, так і на мобільних пристроях. Інструменти пошуку, фільтрації та сортування повинні сприяти легкому доступу до контенту, тоді як функціональність для читання, зокрема перегляд сторінок у різних режимах і можливість масштабування, покликана створити максимально приємний користувацький досвід.

Не менш важливою складовою є система управління обліковими записами, що дозволяє користувачам реєструватися через електронну пошту чи соціальні мережі, зберігати персональні налаштування та улюблені твори. Адміністративна частина вебзастосунку повинна забезпечувати ефективне управління контентом, включаючи додавання нових коміксів, оновлення інформації та модерування відгуків і коментарів.

Для забезпечення надійності та захисту користувацьких даних необхідно впровадити комплексні механізми безпеки, які запобігатимуть несанкціонованому доступу до системи та атакам на сервер. Оптимізація продуктивності має включати методи швидкого завантаження сторінок і

зменшення навантаження на сервер, щоб гарантувати стабільну роботу навіть за умов високої відвідуваності.

Реалізація всіх зазначених аспектів дозволить створити якісний та функціональний продукт, який відповідатиме сучасним вимогам. Такий застосунок не лише забезпечить комфортне читання коміксів, а й сприятиме розвитку спільноти шанувальників цього виду мистецтва, пропонуючи їм зручний цифровий простір для обміну ідеями й враженнями

## 1.5 Висновки до розділу 1

Тож було проведено огляд сучасного стану ринку вебзастосунків для читання коміксів, визначено їх основні характеристики, переваги та недоліки. Аналіз таких платформ, як ReManga та MangaLib, дозволив виділити ключові функції, які необхідно реалізувати у вебзастосунку, а також виявити аспекти, які можна покращити для створення конкурентного продукту.

На основі аналізу було сформульовано основні вимоги до системи, що охоплюють функціональні та нефункціональні аспекти. Зокрема, увага була приділена таким моментам, як зручність інтерфейсу, безпека персональних даних, стабільність роботи та оптимізація продуктивності.

Крім того, визначено чіткі завдання для розробки вебзастосунку, включаючи створення адаптивного інтерфейсу, системи управління контентом, функціоналу для збереження історії читання та інтерактивної взаємодії користувачів.

Таким чином, проведений аналіз предметної області та постановка задачі створюють міцний фундамент для подальшого проєктування та реалізації вебзастосунку, який відповідатиме потребам сучасних користувачів та сприятиме популяризації коміксів як медіа-продукт

# 2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ РОЗРОБКИ

## 2.1 Огляд фреймворку Symfony та його можливостей

Symfony – це потужний PHP-фреймворк, який використовується для створення масштабованих і надійних вебзастосунків. Його головна особливість полягає у модульності: розробник має можливість використовувати лише ті компоненти, які відповідають конкретним потребам проєкту. Такий підхід сприяє ефективності розробки та дозволяє зосередитись на вирішенні основних задач.

Однією з ключових переваг Symfony є його гнучкість. Завдяки підтримці сучасних стандартів веброзробки та інтеграції з іншими інструментами, цей фреймворк ідеально підходить для реалізації як невеликих вебзастосунків, так і складних корпоративних систем. Наприклад, інтеграція з Doctrine ORM забезпечує зручну роботу з базами даних, а використання Twig спрощує створення динамічних вебсторінок.

Symfony також відомий своїми вбудованими інструментами для забезпечення безпеки. У фреймворку реалізовано захист від найпоширеніших вебатак, таких як SQL-ін’єкції та XSS. Крім того, система контролю прав доступу дозволяє гнучко управляти рівнями доступу для різних типів користувачів.

Ще однією важливою рисою Symfony є його продуктивність. Завдяки механізмам кешування, оптимізованій роботі з HTTP-запитами та іншим вбудованим інструментам, розроблені застосунки демонструють високу швидкодію навіть при значному навантаженні [1].

Неможливо оминути увагою і підтримку великої спільноти, яка постійно розширює можливості фреймворку, додаючи нові пакети та модулі. Це робить Symfony не лише зручним інструментом для розробки, але й надійною платформою, яка має довгострокову перспективу розвитку.

Завдяки своїм можливостям Symfony є оптимальним вибором для реалізації вебзастосунку для читання коміксів, адже він дозволяє створити продуктивну, безпечну та масштабовану систему, яка відповідатиме сучасним вимогам користувачів.

## 2.2 Робота з базами даних (MySQL)

Одним із ключових елементів будь-якого вебзастосунку є база даних, яка забезпечує зберігання, управління та швидкий доступ до даних. У цьому проєкті для реалізації серверної частини використовується реляційна база даних MySQL. Цей вибір обумовлений її продуктивністю, широкою підтримкою спільноти та багатим набором функціональних можливостей [2].

MySQL дозволяє працювати з великими обсягами даних без втрати швидкодії, що є важливим аспектом для систем, які передбачають зберігання тисяч зображень коміксів, метаданих і користувацьких профілів. Її структура забезпечує гнучкість у створенні складних реляційних зв’язків між таблицями, що необхідно для розробки багатофункціонального вебзастосунку.

### 2.2.1 Створення структури бази даних

Під час проєктування бази даних було приділено увагу побудові чіткої та зрозумілої структури, яка враховує специфіку предметної області. Основні таблиці охоплюють зберігання інформації про комікси, глави, сторінки, користувачів, їхні налаштування та активність. Наприклад, таблиці для коміксів та їх глав мають реляційний зв’язок типу «один до багатьох», що дозволяє ефективно організувати зберігання даних про розділи коміксів та їхній вміст [4].

### 2.2.2 Інтеграція з Symfony через Doctrine ORM

Для роботи з MySQL у Symfony використовується Doctrine ORM, яка забезпечує зручність у написанні SQL-запитів і дозволяє працювати з базою даних на рівні об’єктів. Doctrine автоматизує багато рутинних процесів, таких як створення запитів, керування зв’язками між таблицями та підтримка змін у структурі бази даних. Наприклад, для створення нового запису в базі достатньо створити об’єкт певної сутності та викликати відповідний метод збереження [3].

### 2.2.3 Оптимізація запитів та продуктивність

Особливу увагу було приділено оптимізації запитів для забезпечення швидкого доступу до даних. Використання індексів на ключових полях значно прискорює виконання запитів, що є важливим для роботи із великим обсягом записів, таких як дані про сторінки коміксів або історію читання користувачів.

MySQL забезпечує надійний фундамент для побудови вебзастосунку, надаючи можливість створити продуктивну та масштабовану систему. Інтеграція з Symfony через Doctrine ORM робить процес взаємодії з базою даних більш ефективним, зменшуючи складність розробки та підтримки застосунку [5].

## 2.3 Бібліотеки та сторонні інструменти

Для розробки вебзастосунку для читання коміксів важливу роль відіграють бібліотеки та сторонні інструменти, які допомагають полегшити процес розробки, прискорити реалізацію функцій і підвищити продуктивність системи. Використання таких інструментів дозволяє зосередитися на

розв’язанні основних завдань, зменшуючи витрати часу на реалізацію типових функцій [6].

Однією з таких бібліотек є Twig – шаблонізатор, який інтегрується з Symfony. Twig дозволяє розділяти логіку і вигляд вебзастосунку, що робить код більш зрозумілим і підтримуваним [8]. Використання шаблонізатора дозволяє створювати гнучкі й масштабовані інтерфейси, адже він надає можливість зручно маніпулювати даними, що передаються в шаблони, і забезпечує безпеку, зменшуючи ризик SQL-ін’єкцій та інших вразливостей. Twig також дозволяє ефективно працювати з різними типами контенту, такими як зображення, текст і відео, що є важливим для такого проєкту, як платформа для читання коміксів.

Для рендерингу зображень та роботи з медіафайлами можна використовувати сторонні API. Наприклад, для забезпечення швидкого завантаження і коректного відображення коміксів у різних форматах використовуються інструменти, що підтримують стиснення зображень та їх оптимізацію. Це дозволяє значно покращити швидкість завантаження сторінок та зменшити навантаження на сервер. У разі потреби можна інтегрувати сторонні сервіси для обробки зображень, такі як ImageMagick або TinyPNG, що дозволяє користувачам комфортно переглядати комікси, навіть коли вони мають великий розмір або високу роздільну здатність [7].

Ще однією важливою складовою є використання бібліотек для анімацій та інтерактивності, таких як GreenSock (GSAP). Вона дозволяє створювати плавні анімації та переходи, що підвищують зручність користувацького досвіду, особливо при перегляді великих зображень коміксів. Анімації можуть бути використані для відображення елементів інтерфейсу, таких як кнопки або переходи між розділами, що створює більш інтерактивний та динамічний вигляд вебзастосунку.

Система керування версіями Git є незамінним інструментом для організації спільної роботи над проєктом. Вона дозволяє тримати всі зміни в коді під контролем, вести історію змін, а також забезпечує безпеку при

розробці, оскільки дозволяє повертатись до попередніх версій у разі помилок або непередбачуваних ситуацій. Git також дозволяє зручно інтегруватися з сервісами хостингу, такими як GitHub або GitLab, що дає можливість здійснювати код-рев'ю, працювати з гілками та тестувати нові функції без ризику для основної версії проєкту [9].

Використання бібліотек та сторонніх інструментів значно спрощує розробку вебзастосунку, дозволяючи фокусуватись на реалізації основних бізнес-логік і знижуючи витрати часу на створення базових функцій. Ці інструменти допомагають забезпечити високу ефективність, продуктивність і зручність використання вебзастосунку для кінцевих користувачів.

### 2.4 Системи управління проєктами (Jira)

У процесі розробки вебзастосунку для читання коміксів важливим є ефективне управління проєктом, що дозволяє уникнути плутанини в задачах та забезпечує чіткий контроль над усіма етапами роботи. Хоча системи управління проєктами зазвичай асоціюються з координацією командної роботи, вони також надзвичайно корисні для індивідуальних розробників. Навіть працюючи самостійно, використання таких інструментів, як Jira, допомагає структуровано підходити до вирішення задач, не пропускати важливі деталі та дотримуватися термінів.

Jira дозволяє створювати різні типи задач, такі як баги, фічі або технічні завдання, що допомагає розділити великий проєкт на невеликі кроки. Ця деталізація не тільки дозволяє зосередитися на окремих аспектах розробки, але й спрощує контроль за прогресом. Інструмент також дає змогу налаштувати робочі процеси під специфіку проєкту, що є корисним як для команди, так і для одного розробника, оскільки дозволяє швидко адаптуватися до змін або нових пріоритетів.

Для індивідуальної роботи Jira також може бути корисною завдяки інтеграції з іншими інструментами, такими як GitHub або GitLab. Наприклад,

зміни у репозиторії можуть автоматично оновлювати статус задач, що дозволяє фіксувати всі зміни у проєкті в одному місці. Це полегшує повернення до попередніх етапів або перегляд історії задач у разі виникнення проблем.

Важливим аспектом є можливість зберігати в Jira коментарі, замітки та записи про виконання задач. Це допомагає створити прозору систему, яка дозволяє не забути про виконані кроки чи контекст прийнятих рішень. Для індивідуальних розробників це зручно, оскільки завдяки чіткій документації можна легко повернутися до задач через деякий час або передати проєкт іншій людині.

Таким чином, використання Jira забезпечує структурованість і організованість навіть у проєктах, де працює одна людина. Це дозволяє уникати плутанини, ефективно планувати свою роботу та вчасно досягати поставлених цілей. У випадку розробки вебзастосунку для читання коміксів, це особливо важливо для збереження високої якості роботи та виконання проєкту у встановлені терміни.

## 2.5 Висновки до розділу 2

У цьому розділі були розглянуті ключові технології та інструменти, які будуть використовуватись для розробки вебзастосунку для читання коміксів. Вибір фреймворку Symfony обумовлений його можливостями для ефективної розробки серверної частини застосунку, забезпечення безпеки, масштабованості та простоти в інтеграції з іншими технологіями. Symfony надає потужні засоби для побудови API, управління базами даних, а також реалізації бізнес-логіки, що дозволяє значно спростити процес розробки і зменшити час на реалізацію.

Що стосується роботи з базами даних, вибір MySQL є обґрунтованим, оскільки ця система керування базами даних має високу продуктивність,

надійність і добре підтримується в контексті веброзробки. Вона дозволяє зберігати великі об’єми даних, таких як комікси, користувачі та відгуки, і забезпечує швидкий доступ до інформації завдяки оптимізації запитів і збереженню структурованих даних.

Використання бібліотек та сторонніх інструментів, таких як Twig для рендеринга шаблонів, дозволяє забезпечити чистий і зручний для розробки код на стороні сервера. Інтеграція з API для роботи з зображеннями також важлива для забезпечення ефективної роботи з медіа-контентом, що є основною частиною вебзастосунку. Інструменти для рендерингу зображень та їх обробки дозволяють оптимізувати час завантаження і зробити перегляд коміксів зручнішим для користувачів.

Використання Jira як системи управління проєктами дасть змогу організувати роботу команди, відстежувати виконання завдань, управляти спринтами та зберігати прозорість у процесі розробки. Це дозволить зберігати ефективність і дисципліну, а також полегшить процес співпраці між учасниками проєкту. Водночас, система контролю версій Git забезпечить зручне управління кодом і дозволить уникати помилок при розробці.

Таким чином, вибір цих технологій і інструментів дозволяє створити потужний і зручний вебзастосунок для читання коміксів, який буде простим у використанні для кінцевих користувачів і ефективним у підтримці для розробників. Всі інструменти забезпечують високу продуктивність і стабільність роботи системи, що важливо для успіху продукту.

# 3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

## 3.1 Основні діаграми проєктування

Основні діаграми проєктування є ключовим елементом у процесі розробки вебзастосунку, оскільки вони дозволяють наочно представити структуру та взаємодію компонентів системи. Вони є важливими для візуалізації того, як різні частини програми взаємодіють між собою і як їх правильно організувати. Створення діаграм допомагає зрозуміти логіку системи ще на етапі проєктування, що дає можливість виправити потенційні помилки до початку реалізації. Діаграми також важливі для узгодження вимог до проєкту, адже вони дозволяють виявити недоліки на ранніх етапах, знижуючи ризик непорозумінь і помилок під час розробки.

Проєктування системи з допомогою діаграм полегшує командну роботу, оскільки надає чітке уявлення про структуру та взаємодію різних компонентів, що сприяє кращій комунікації між розробниками, тестувальниками та іншими учасниками проєкту. Це також дозволяє зекономити час, оскільки дає змогу виявити та усунути потенційні проблеми ще до їх виникнення. Діаграми стають основою для подальшої розробки коду та бази даних, забезпечуючи правильність і ефективність роботи системи.

Таким чином, діаграми проєктування не лише допомагають візуалізувати і оптимізувати структуру системи, але й сприяють зниженню ризиків при її реалізації, що робить їх важливим інструментом для успішного завершення розробки вебзастосунку.

### 3.1.1 ER-діаграма

ER-діаграма (Entity-Relationship Diagram) (див. рис. 3.1) є важливим інструментом для проєктування структури бази даних вебзастосунку, оскільки

вона дозволяє візуалізувати сутності, їх атрибути та взаємозв’язки між ними. У нашому випадку, для вебзастосунку, що надає можливість читання коміксів, ця діаграма допомагає чітко визначити, які дані будуть зберігатися в базі та як вони взаємодіятимуть між собою.

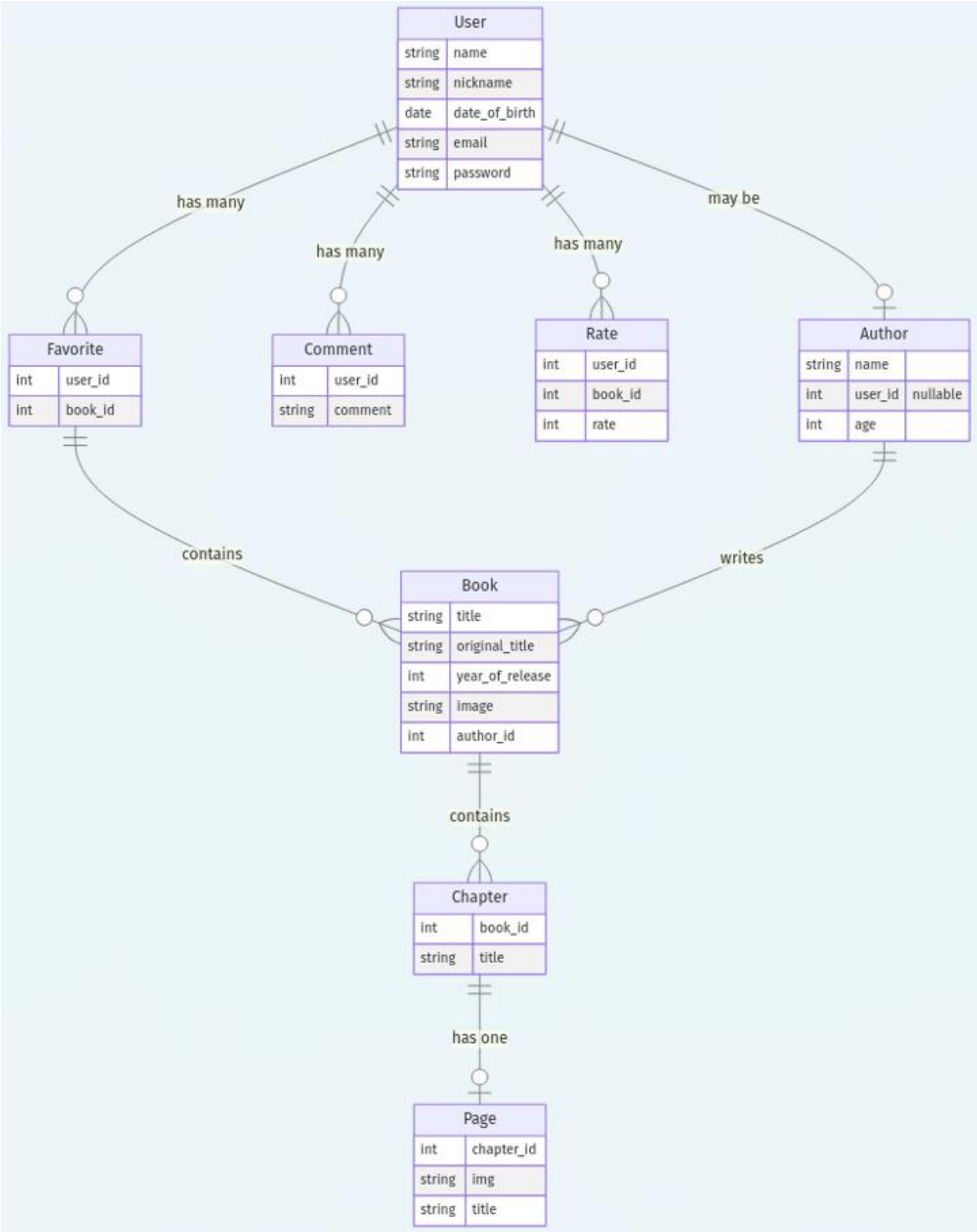



Рисунок 3.1 – ER-діаграма

Основними сутностями, що відображаються в ER-діаграмі, є користувач, автор, книга, розділ, сторінка, коментар, оцінка та улюблені книги. Користувач є однією з центральних сутностей, оскільки він може залишати коментарі, оцінювати книги та додавати їх у список улюблених. Автори коміксів також є важливою частиною системи, оскільки кожен автор може мати кілька книг, які, в свою чергу, складаються з розділів і сторінок.

Кожен коментар і оцінка користувача пов’язані з певною книгою, що дозволяє відслідковувати відгуки та реакції на різні види контенту. Оцінки надаються лише один раз для кожної книги, а коментарі можуть залишатися на будь-яку сторінку або книгу. Користувач може створювати список улюблених книг, що дозволяє йому організовувати контент за власними вподобаннями.

Зв’язки між сутностями також чітко визначені в діаграмі. Користувач може мати кілька коментарів, оцінок і улюблених книг, що створює зв’язки один до багатьох між користувачем і іншими сутностями. Кожна книга складається з кількох розділів, а кожен розділ включає кілька сторінок. Така структура дозволяє забезпечити правильну організацію даних у базі, а також дозволяє зручно працювати з ними у рамках вебзастосунку.

Завдяки цій ER-діаграмі стає зрозумілою структура бази даних, що є основою для ефективного проєктування та реалізації функціоналу вебзастосунку. Використовуючи фреймворк Symfony і Doctrine ORM, ми можемо зручно організувати сутності та їх зв’язки в базі даних, забезпечивши правильну роботу системи.

### 3.1.2 Діаграма класів

Діаграма класів є одним із ключових інструментів в об’єктно-орієнтованому проєктуванні, оскільки вона дозволяє візуалізувати структуру і взаємозв’язки між класами системи. У контексті розробки вебзастосунку для читання коміксів діаграма класів використовується для моделювання логіки

програми, яка базується на взаємодії користувачів, авторів, книг, розділів, сторінок, коментарів, оцінок і списків улюбленого контенту (див. рис. 3.2, 3.3).

Кожен клас представляє сутність у програмі, що має атрибути, які описують властивості цієї сутності, та методи, що визначають її поведінку. Наприклад, клас User містить атрибути, що описують профіль користувача, такі як ім’я, електронна пошта і пароль, а також методи для створення, оновлення та видалення профілю. Клас Book включає властивості, які характеризують книгу, зокрема назву, рік випуску та ідентифікатор автора, а також методи для додавання, редагування чи видалення книг.

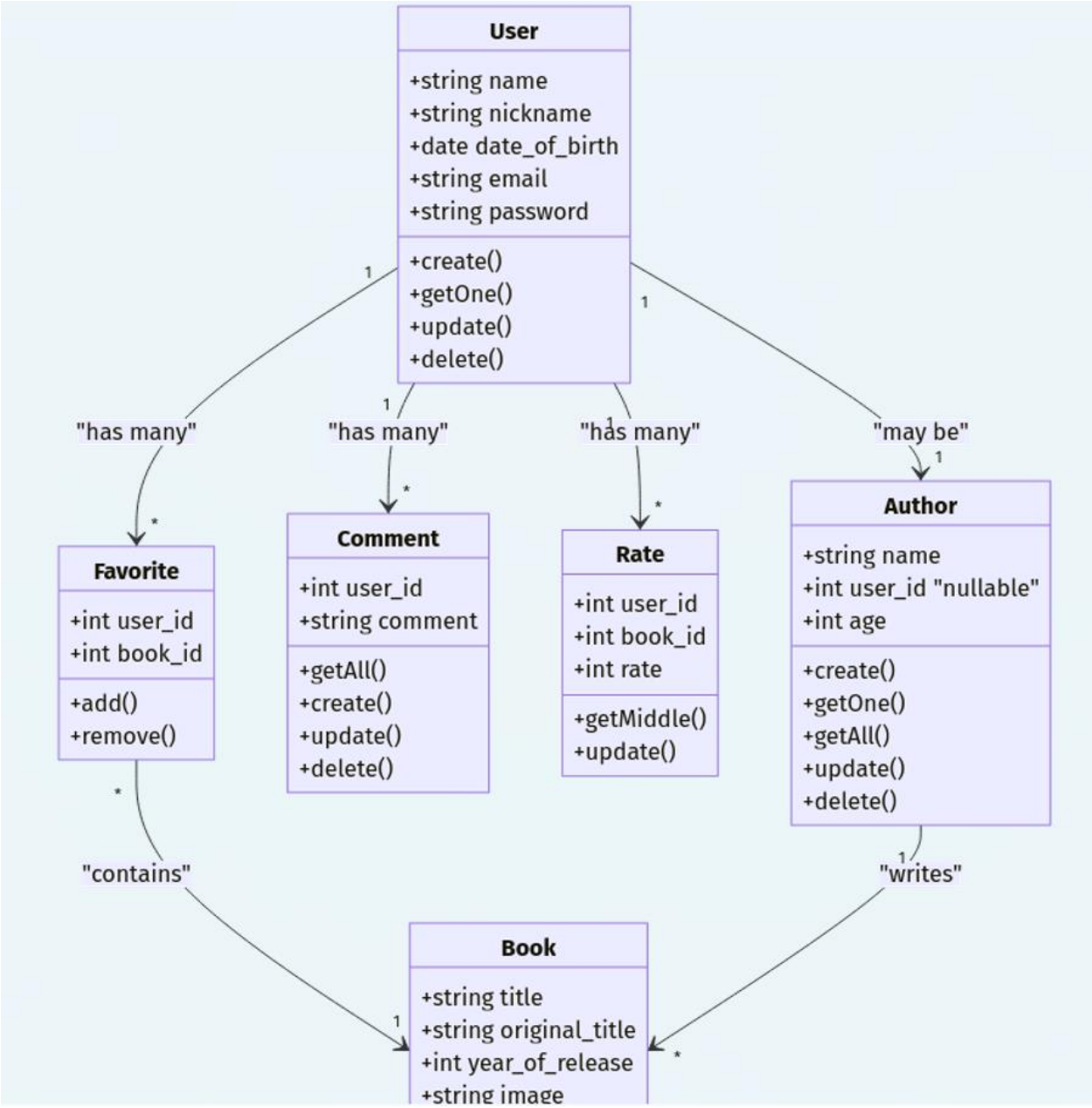



Рисунок 3.2 – Діаграма класів (1)




Рисунок 3.3 – Діаграма класів (2)

Зв’язки між класами відображають взаємодію між об’єктами системи. Наприклад, клас Author пов’язаний із класом Book через зв’язок «один до багатьох», що відображає можливість автора створювати кілька книг. Клас User взаємодіє з класом Comment, що моделює коментарі користувачів до книг, а також з класом Rate, який зберігає оцінки, поставлені користувачами. Клас Favorite відображає зв’язок між користувачем і списком улюблених книг.

Діаграма класів дозволяє розробникам ще на етапі проєктування чітко визначити структуру взаємодій між об’єктами системи. Це забезпечує ефективну організацію коду, легкість у супроводі та розширенні функціоналу. У рамках розробки за допомогою фреймворку Symfony, діаграма класів служить основою для налаштування моделей і їх взаємозв’язків у Doctrine ORM. Вона сприяє автоматизації створення таблиць у базі даних, дозволяючи уникнути помилок у визначенні їх структури.

Таким чином, діаграма класів є важливим елементом процесу проєктування, що не лише формує базу для реалізації програмного забезпечення, але й забезпечує прозорість і зручність у роботі з системою на всіх етапах її життєвого циклу.

### 3.1.3 Діаграма послідовності

Діаграма послідовності (див. рис. 3.4) є важливим інструментом для візуалізації динамічної поведінки системи, зокрема порядку взаємодії між об’єктами та компонентами. Вона дозволяє побачити, як функції чи процеси виконуються у системі з точки зору послідовності дій, які вони викликають. У вебзастосунку для читання коміксів діаграма послідовності допомагає моделювати сценарії взаємодії користувачів із системою, такі як реєстрація, авторизація, пошук коміксів або додавання їх до улюблених.

Основна цінність діаграми послідовності полягає у демонстрації часової залежності дій. Наприклад, у процесі авторизації користувач спочатку вводить свої облікові дані, які передаються до серверу. Сервер перевіряє їх дійсність, після чого повертає відповідь, яка визначає, чи був доступ до системи наданий. Аналогічно, під час додавання книги до списку улюблених, спочатку відправляється запит до сервера для оновлення бази даних, а потім користувачу надається підтвердження успішної операції.

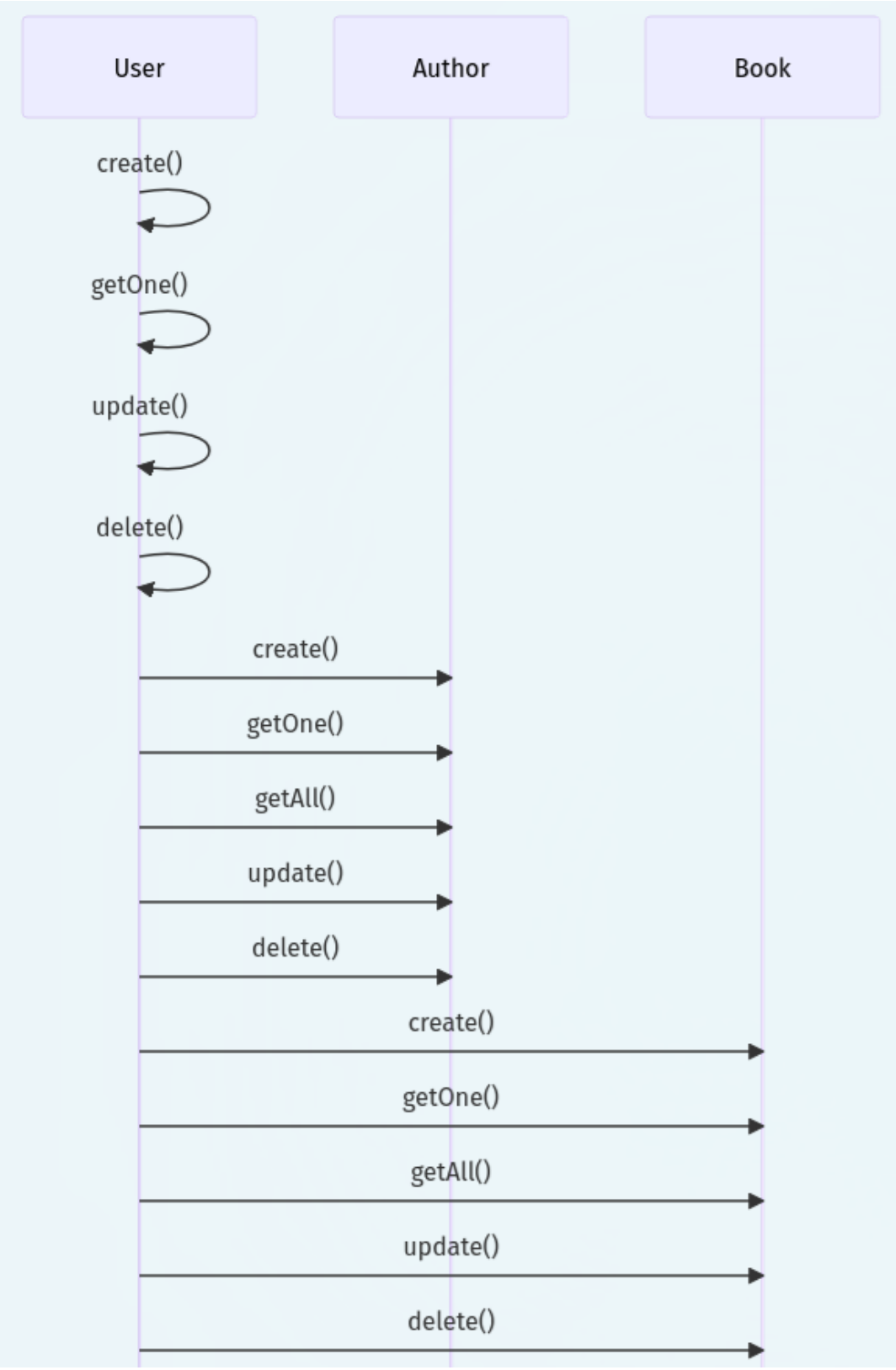



Рисунок 3.4 – Фрагмент діаграми послідовностей

Діаграма послідовності є корисною для виявлення можливих проблем у логіці взаємодії компонентів системи. Вона дозволяє деталізувати кожен крок процесу, визначити, де можуть виникнути потенційні помилки або затримки,

і запропонувати способи їх усунення. У випадку використання Symfony така діаграма полегшує проєктування маршрутів, контролерів та сервісів, забезпечуючи чітке розуміння ролі кожного елемента у виконанні бізнес-логіки.

Важливість цієї діаграми також полягає у тому, що вона допомагає синхронізувати очікування всіх членів команди. Розробники, тестувальники та інші учасники проєкту можуть використовувати її як орієнтир для перевірки правильності реалізації функціоналу, забезпечуючи відповідність між проєктною документацією та кінцевим продуктом.

Таким чином, діаграма послідовності не лише візуалізує взаємодії у системі, але й слугує основою для проєктування й впровадження ключових бізнес-процесів у вебзастосунку.

## 3.2 Висновки до розділу 3

Таким чином було проведено проєктування вебзастосунку для читання коміксів, що є одним із ключових етапів у розробці програмного забезпечення. Розглянуті основні діаграми дозволили сформувати чітке уявлення про структуру даних та динаміку взаємодії компонентів системи.

Розробка ER-діаграми допомогла ефективно спроєктувати базу даних, забезпечуючи коректне зберігання, обробку та доступ до інформації про користувачів, авторів, книги, глави та сторінки. Завдяки цьому вдалося створити логічну та структуровану модель даних, яка відповідає вимогам функціоналу вебзастосунку.

Аналіз діаграми класів дозволив сформувати основні програмні сутності та взаємозв’язки між ними, що забезпечує чіткість і зрозумілість архітектури проєкту. Враховуючи особливості Symfony, діаграма відображає не лише статичну структуру системи, але й допомагає при створенні моделей, сервісів і контролерів.

Розробка діаграми послідовності дала змогу моделювати ключові бізнес-процеси, такі як авторизація користувачів чи додавання коміксів до улюблених. Це забезпечило чітке уявлення про логіку виконання дій і дозволило виявити потенційні точки збоїв чи оптимізації.

Проєктування основних діаграм не лише допомогло створити структурну та функціональну модель системи, але й сприяло зниженню ризику помилок у подальших етапах розробки. Отримані результати забезпечують стабільність, масштабованість і простоту розширення функціоналу вебзастосунку, що є важливим для успішного впровадження проєкту.

# 4 РОЗРОБКА ВЕБЗАСТОСУНКУ

Розділ «Розробка вебзастосунку» можна поділити на дві основні частини: реалізацію бекенд-частини та фронтенд-частини. Оскільки проєкт базується на Symfony, основна увага буде приділена опису принципів роботи бекенду. Для фронтенду, навіть якщо він є мінімальним, слід виділити окремий підрозділ для документування його функціоналу.

## 4.1 Розробка бекенд-частини

Розробка бекенд-частини є ключовим етапом у створенні будь-якого вебзастосунку, оскільки саме тут відбувається обробка даних, логіка взаємодії з базою даних, авторизація користувачів та інші важливі функції, що забезпечують правильну роботу системи. У цьому розділі розглядається процес створення бекенду для вебзастосунку, який включає в себе використання фреймворку Symfony, реалізацію моделей, контролерів та інтеграцію з базою даних.

Основною метою розробки бекенд-частини є забезпечення безпечного, стабільного та ефективного функціонування застосунку. Для цього було обрано фреймворк Symfony, який дозволяє швидко розробляти високоякісні вебзастосунки, зберігаючи високу продуктивність і масштабованість. Symfony забезпечує багатий набір інструментів для роботи з маршрутизацією, авторизацією, обробкою запитів та ін.

У рамках розділу буде докладно описано, як реалізувати основні елементи бекенду, зокрема моделі даних, контролери, а також механізми взаємодії з базою даних за допомогою Doctrine ORM. Крім того, особливу увагу буде приділено процесу авторизації користувачів та механізмам безпеки, що є критично важливими для захисту даних і забезпечення стабільної роботи платформи.

Цей розділ дозволить зрозуміти, як правильно побудувати бекенд-частину застосунку, враховуючи усі важливі аспекти, що впливають на ефективність і безпеку системи.

### 4.1.1 Основи архітектури MVC у Symfony

Архітектура Model-View-Controller (MVC) є однією з найпоширеніших у розробці сучасних вебзастосунків. Вона забезпечує чітке розділення логіки додатку, інтерфейсу користувача та роботи з даними. Це сприяє кращій підтримуваності, масштабованості та зрозумілості коду. Модель відповідає за управління даними та їх обробку, подання відповідає за візуалізацію інформації, а контролер забезпечує зв’язок між ними, обробляючи запити від користувача та передаючи дані до подання.

Symfony впроваджує архітектуру MVC як основний принцип розробки, що дозволяє ефективно організовувати код і зменшувати його залежності. Моделі у Symfony реалізуються через Doctrine ORM, яка дозволяє працювати з базою даних як з об’єктами. За допомогою спеціальних анотацій у класах визначаються зв’язки між таблицями, їхні поля та інші аспекти роботи з базою даних. Це значно спрощує доступ до даних і забезпечує узгодженість структури програми [10].

Контролери у Symfony є основними точками обробки запитів. Вони отримують HTTP-запити, виконують відповідну логіку і повертають відповіді у вигляді HTML, JSON або інших форматів. Symfony надає гнучкий механізм маршрутизації, що дозволяє налаштовувати маршрути для кожного контролера, а також використовувати динамічні параметри у запитах.

Подання у Symfony, як правило, реалізуються за допомогою шаблонізатора Twig. Це дозволяє зручно створювати шаблони з використанням змінних, циклів, умовних операторів та інших інструментів. Twig сприяє розділенню логіки програми та її представлення, забезпечуючи чистоту і зрозумілість коду.

Таким чином, використання архітектури MVC у Symfony надає розробникам потужні інструменти для створення організованих і зручних у підтримці вебзастосунків. Завдяки чіткому розділенню компонентів забезпечується узгодженість структури, спрощується робота над проєктом і покращується якість коду.

### 4.1.2 Розробка моделей

У Symfony моделі реалізуються за допомогою Doctrine ORM, що забезпечує взаємодію з базою даних через об’єктно-реляційне відображення (ORM). Це дозволяє працювати з базою даних у вигляді об’єктів, уникаючи прямого написання SQL-запитів. Для кожної сутності бази даних створюється відповідний клас, який визначає її структуру, властивості та зв’язки з іншими сутностями.

Наприклад, розглянемо модель для сутності User. Клас User включає атрибути, що відповідають колонкам у таблиці бази даних: ім’я, електронну пошту, пароль тощо. Використовуючи Doctrine, можна легко визначити ці атрибути за допомогою анотацій (на рисунку вказані не всі поля) (рис. 4.1).

```
namespace App\Entity;
use Doctrine\ORM\Mapping as ORM;
#[ORM\Entity]
class User
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    private int $id;
    #[ORM\Column(type: 'string', length: 100)]
    private string $name;
    #[ORM\Column(type: 'string', unique: true)]
    private string $email;
    #[ORM\Column(type: 'string')]
```

```
    private string $password;
    public function getName(): string {
        return $this->name;
    }
    public function setName(string $name): self {
        $this->name = $name;
        return $this;
    }
    public function getEmail(): string {
        return $this->email;
    }
```

Рисунок 4.1 – Приклад моделі

Ця модель визначає таблицю User у базі даних, включаючи її поля та типи даних. Вона також забезпечує гетери та сетери для доступу та зміни значень атрибутів.

Для роботи із зв’язками між таблицями використовуються відповідні анотації. Наприклад, модель Book може мати зв’язок «багато до одного» з автором (рис. 4.2).

```
#[ORM\ManyToOne(targetEntity: Author::class, inversedBy: 'books')]
#[ORM\JoinColumn(nullable: false)]
private Author $author;
```

Рисунок 4.2 – Приклад зв’язку

Цей підхід дозволяє легко описувати відносини між сутностями, зберігаючи логічну структуру даних.

Моделі у Symfony не лише описують структуру бази даних, але й виступають як частина бізнес-логіки, оскільки можуть включати методи для виконання специфічних завдань, наприклад, перевірку унікальності даних або обчислення середнього рейтингу. Таким чином, розробка моделей є фундаментальною частиною створення додатку, яка визначає не лише дані, але й їх поведінку.

### 4.1.3 Розробка контролерів

Контролери у Symfony відповідають за обробку HTTP-запитів і підготовку відповідей. Вони є центральним елементом логіки застосунку, отримуючи дані від користувача, передаючи їх у моделі для обробки та формуючи відповідь для відображення. Кожен контролер є PHP-класом, методи якого обробляють конкретні маршрути.

Контролер, який відповідає за роботу з книгами, виглядає так (рис. 4.3).

```
namespace App\Controller;
use …
#[Route('/books')]
class BookController extends AbstractController
{
   #[Route('/', name: 'book_index', methods: ['GET'])]
   public function index(BookRepository $bookRepository): Response {
      $books = $bookRepository->findAll();
      return $this->render('book/index.html.twig', [
         'books' => $books,
      ]);
   }
   #[Route('/new', name: 'book_new', methods: ['GET', 'POST'])]
   public function create(Request $request, EntityManagerInterface $entityManager):
Response {
      $book = new Book();
      $form = $this->createForm(BookType::class, $book);
      $form->handleRequest($request);
      if ($form->isSubmitted() && $form->isValid()) {
         $entityManager->persist($book);
         $entityManager->flush();
         return $this->redirectToRoute('book_index');
      }
      return $this->render('book/new.html.twig', [
         'form' => $form->createView(),
      ]);
   }
```

Рисунок 4.3 – Приклад контроллеру

У цьому прикладі контролер реалізує CRUD-операції для сутності Book. Метод index отримує список усіх книг із репозиторію та передає їх у Twig-шаблон для відображення. Метод create відповідає за створення нової книги, обробляючи форму, яка заповнюється користувачем. Подібним чином методи edit і delete обробляють редагування та видалення відповідно.

Контролери забезпечують простий механізм маршрутизації за допомогою атрибутів, таких як #[Route]. Це дозволяє чітко вказати, який метод обробляє певний маршрут і які HTTP-методи підтримуються. Завдяки Symfony механізмам контролери стають потужним інструментом для організації бізнес-логіки та взаємодії з моделями й представленням.

### 4.1.4 Маршрутизація

Маршрутизація в Symfony відповідає за визначення відповідності між запитами HTTP і контролерами, які їх обробляють. Вона є ключовою складовою для організації роботи вебзастосунку, дозволяючи направляти запити до відповідних методів залежно від URL-адреси, методу запиту (GET, POST тощо) і параметрів. Symfony надає декілька способів налаштування маршрутів: за допомогою YAML-файлів, XML, PHP або через атрибути безпосередньо у контролерах.

У контролері маршрут можна визначити через атрибути. Э контролер для роботи з книгами, його методи можуть мати такі маршрути (рис. 4.4).

```
#[Route('/books', name: 'book_index', methods: ['GET'])]
public function index(BookRepository $bookRepository): Response
{
    $books = $bookRepository->findAll();
    return $this->render('book/index.html.twig', ['books' => $books]);
}
```

```
#[Route('/books/new', name: 'book_new', methods: ['GET', 'POST'])]
public function create(Request $request, EntityManagerInterface $entityManager): Response
{
    $book = new Book();
    $form = $this->createForm(BookType::class, $book);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $entityManager->persist($book);
        $entityManager->flush();

        return $this->redirectToRoute('book_index');
    }
    return $this->render('book/new.html.twig', ['form' => $form->createView()]);
}
```

Рисунок 4.4 – Приклад роутінгу в котроллері

Маршрути визначаються через атрибут #[Route], де вказується шлях, ім’я маршруту та HTTP-методи, які підтримуються. Це дозволяє однозначно пов’язати кожен маршрут із відповідним методом контролера.

Іншим способом є використання YAML-файлів. Наприклад, маршрут для відображення всіх книг можна описати так (рис. 4.5).

```
book_index:
    path: /books
    controller: App\Controller\BookController::index
    methods: GET
```

Рисунок 4.5 – Приклад роутінгу у YAML-файлах

Тут кожен маршрут має ключ path, який вказує на URL, і controller, що вказує на метод, який його обробляє.

Symfony також підтримує параметризовані маршрути. Це дозволяє використовувати динамічні частини в URL, наприклад, для перегляду книги за її ідентифікатором (рис. 4.6).

```
#[Route('/books/{id}', name: 'book_show', methods: ['GET'])]
public function show(Book $book): Response
{
    return $this->render('book/show.html.twig', ['book' => $book]);
}
```

Рисунок 4.6 – Приклад параметроризованого роута

Тут id є параметром, який автоматично передається у метод контролера.

Гнучкість маршрутизації дозволяє Symfony легко адаптуватися до вимог будь-якого проєкту. Використання параметрів, строгих HTTP-методів і перевірки дозволяє будувати безпечні та оптимальні маршрути. Це створює основу для ефективного оброблення запитів і правильної організації логіки у вебзастосунку.

### 4.1.5 Реалізація авторизації та аутентифікації

Авторизація та аутентифікація є критичними аспектами будь-якого вебзастосунку, що працює з персональними даними користувачів. Вебзастосунки на Symfony мають потужний інструмент для реалізації цих процесів за допомогою компонента Security, який підтримує різні способи аутентифікації, включаючи базову аутентифікацію, форму входу та аутентифікацію через токени.

Аутентифікація користувача полягає в перевірці його ідентичності, що може бути досягнуто за допомогою введення логіна та пароля або через інші методи, такі як токени. Авторизація, з іншого боку, визначає права користувача на доступ до конкретних ресурсів і виконання певних дій.

В Symfony аутентифікація та авторизація налаштовуються через конфігураційний файл security.yaml, де вказуються методи аутентифікації та правила доступу для різних частин застосунку.

Конфігурації аутентифікації за допомогою форми (рис. 4.7).

```
security:
    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
    firewalls:
        main:
            pattern: ^/
            form_login:
                login_path: login
                check_path: login
            logout:
                path: logout
            secure_all_login: true
            roles:
                - ROLE_USER
```

Рисунок 4.7 – Конфігурації аутентифікації за допомогою форми

Використовується провайдер, який зв’язує користувачів з базою даних через клас User та поле email.

Контролер для входу (рис. 4.8 ).

```
// src/Controller/SecurityController.php
namespace App\Controller;
use …
class SecurityController extends AbstractController
{
    #[Route('/login', name: 'login')]
    public function login(Request $request)
    {
        return $this->render('security/login.html.twig');
    }
}
```

Рисунок 4.8 – Приклад контроллеру для авторизації

При використанні токенів для аутентифікації, конфігурація в security.yaml виглядатиме наступним чином (рис. 4.9).

```
security:
   firewalls:
      api:
         pattern: ^/api/
         stateless: true
         jwt: ~
```

Рисунок 4.9 – Конфігурація в security.yaml для токенів аутентифікації

Тут ми вказуємо, що для всіх запитів до API шляхом використовується токен. Використання токенів дозволяє створювати безпечні механізми аутентифікації для API-запитів, що робить застосунок більш гнучким і масштабованим.

Контролер для аутентифікації через токен (рис. 4.10).

```
// src/Controller/ApiController.php
namespace App\Controller;
use …
class ApiController extends AbstractController
{
   #[Route('/api/data', name: 'api_data', methods: ['GET'])]
   public function getData()
   {
        return new JsonResponse(['message' => 'Authorized']);
   }
}
```

Рисунок 4.10 – Приклад контроллеру для аутентифікації

Для правильного налаштування авторизації потрібно використовувати ролі та права доступу (рис. 4.11).

```
security:
   access_control:
      - { path: ^/admin, roles: ROLE_ADMIN }
```

Рисунок 4.11 – Приклад налаштування ролей

Це означає, що всі запити до маршруту admin будуть доступні лише користувачам з роллю ROLE_ADMIN.

В цілому, Symfony надає гнучкі інструменти для реалізації безпечної аутентифікації та авторизації користувачів, що дозволяє легко налаштовувати різні рівні доступу до ресурсів вебзастосунку, забезпечуючи конфіденційність та безпеку даних користувачів.

### 4.2 Розробка фронтенд-частини

Фронтенд-частина вебзастосунку є важливою складовою для забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів. Хоча фронтенд цього проєкту не передбачає складних інтерактивних елементів чи використання великих фреймворків, він додається для наглядності та покращення користувацького досвіду. Завдання полягає в тому, щоб забезпечити простий, але ефективний інтерфейс для відображення інформації, отриманої з бекенд-частини, і взаємодії з користувачем.

У цьому випадку буде використано базові технології: HTML, CSS та JavaScript для створення статичних сторінок і динамічного оновлення контенту без перезавантаження.

Для зручності та структурування коду можна застосовувати прості скрипти на JavaScript, а також використовувати базові принципи CSS для оформлення елементів інтерфейсу.

Хоча фронтенд-частина в цьому проєкті не є складною, вона повинна бути функціональною та зрозумілою для користувача.

Наприклад, сторінки будуть включати основні елементи взаємодії, такі як форми для авторизації, відображення списків книг чи коміксів, а також кнопки для взаємодії з контентом, забезпечуючи таким чином простоту використання системи.

Основні сторінки застосунку можна побачити в Додатку А.

### 4.3 Тестування

У розробці вебзастосунку для читання коміксів особливу увагу приділено функціональним тестам, оскільки важливо перевірити правильність роботи основних компонентів системи, зокрема авторизації, взаємодії з базою даних та обробки запитів.

*Тестування з використанням Postman*. Postman є потужним інструментом для тестування API, який широко використовується для перевірки серверної частини вебзастосунку. Цей інструмент дає змогу тестувати правильність роботи різних HTTP-запитів, що дозволяє розробникам перевірити, чи коректно сервер обробляє запити.

У контексті розробки вебзастосунку для читання коміксів, Postman допомагає перевіряти авторизацію та аутентифікацію користувачів, взаємодію з базою даних, а також перевірку статусів відповідей. Крім того, Postman дозволяє тестувати різні HTTP-методи, такі як GET, POST (див. рис. 4.12), PUT і DELETE, що є важливим для належної обробки запитів на різних етапах роботи з сервером. Завдяки своїм можливостям, Postman спрощує процес інтеграційного тестування, дозволяючи автоматизувати перевірку кожного запиту і швидко виявляти потенційні проблеми в API.

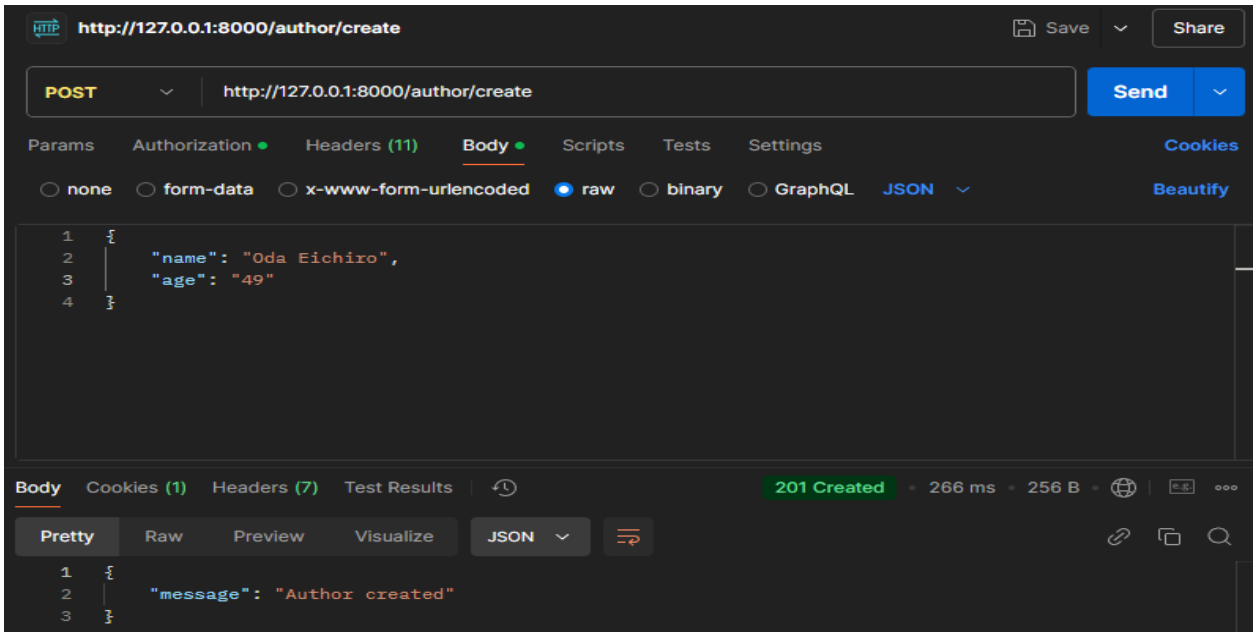



Рисунок 4.12 – Приклад Post-запита

Тестування є важливим етапом розробки вебзастосунку, оскільки дозволяє виявити помилки на ранніх етапах і забезпечити належну якість кінцевого продукту. Воно охоплює різні типи тестів, такі як функціональні, інтеграційні, юзабіліті-тести та тести на безпеку і продуктивність.

### 4.4 Висновки до розділу 4

Тож були детально розглянуті ключові етапи створення як бекенд, так і фронтенд частин системи. Використання архітектури MVC у Symfony стало основою для чіткого поділу обов’язків між моделями, контролерами та представленнями, що дозволяє забезпечити структурованість, масштабованість і зручність у розробці. Моделі, контролери та представлення організовані так, щоб кожна частина відповідала за свою функцію, забезпечуючи правильний потік даних і логіку роботи системи. Використання Doctrine ORM для взаємодії з базою даних дозволило створити ефективну систему для обробки запитів і збереження інформації.

Розробка контролерів і моделей здійснювалася таким чином, щоб забезпечити максимальну гнучкість і зручність в обробці запитів користувачів. Контролери виконують роль «мосту» між запитами та відповідями, направляючи дані до відповідних моделей і представлень. Крім того, реалізація маршрутизації в Symfony дозволила чітко організувати обробку запитів, що забезпечує зручне налаштування і масштабованість маршруту для нових функціональностей.

Реалізація авторизації та аутентифікації через токен дозволила створити надійний механізм захисту, що забезпечує безпечний доступ до ресурсів вебзастосунку. Цей підхід дозволяє уникнути необхідності зберігання даних користувачів на клієнтському боці та забезпечує зручність роботи з авторизацією в рамках масштабованих вебсистем.

Фронтенд частина, хоча й була реалізована в мінімалістичному вигляді, надає користувачам простий, але функціональний інтерфейс для взаємодії з

вебзастосунком. Створення базових сторінок на HTML і CSS дозволило забезпечити гарний візуальний вигляд, а додавання мінімальної інтерактивності за допомогою JavaScript підвищило зручність використання сайту.

Узагальнюючи, цей розділ показує, як різні компоненти системи взаємодіють між собою та створюють ефективний та зручний вебзастосунок. Всі етапи розробки, починаючи від архітектурних рішень і закінчуючи реалізацією простого інтерфейсу, були організовані таким чином, щоб забезпечити стабільну та безпечну роботу вебзастосунку, який відповідає вимогам користувачів і розробників.

# ВИСНОВКИ

У процесі розробки вебзастосунку для читання коміксів була реалізована комплексна система, яка охоплює всі етапи розробки, починаючи від проєктування та вибору технологій до реалізації основних функціональних можливостей. Під час роботи над бекенд-частиною застосунку було обрано фреймворк Symfony, який надав потужний інструментарій для розробки стабільної, масштабованої та безпечної системи. Архітектура MVC дозволила чітко розділити логіку роботи з даними, обробку запитів і відображення результатів, що сприяло підвищенню ефективності та зручності роботи з кодом.

Одним із основних завдань було забезпечення правильного функціонування бази даних, для чого було використано Doctrine ORM. Це дозволило оптимізувати взаємодію з базою даних, спростити створення та виконання запитів, а також забезпечити збереження даних у зручному для подальшого використання вигляді. Завдяки цьому система працює швидко, а дані зберігаються в належному вигляді, що є важливим для подальшої експансії проєкту.

Особливу увагу було приділено безпеці та авторизації користувачів. Для цього була реалізована система аутентифікації через токен, що дозволяє захистити користувацькі дані і забезпечити зручний доступ до платформи. Використання сучасних стандартів авторизації дозволяє забезпечити необхідний рівень захисту, зберігаючи при цьому зручність для користувачів. У поєднанні з ефективною маршрутизацією це забезпечує коректну і безпечну обробку запитів.

Фронтенд-частина застосунку, хоча і була розроблена без великих складнощів, також відіграє важливу роль у забезпеченні зручності використання платформи. Використання базових технологій HTML, CSS та JavaScript дозволило створити зрозумілий та інтуїтивно зрозумілий інтерфейс.

Завдяки цьому користувачі можуть легко орієнтуватися на платформі, переглядати комікси, авторів та розділи, що суттєво покращує їх досвід користування. Базова інтерактивність на клієнтській частині дає можливість взаємодіяти з контентом, при цьому не перевантажуючи інтерфейс зайвими елементами.

Крім того, робота з інтерфейсом забезпечує не тільки доступ до функціональних можливостей, але й позитивне враження від використання платформи. Легкість у навігації, зручність у перегляді контенту, а також спрощений процес взаємодії з вебзастосунком робить його доступним для широкої аудиторії.

У результаті розробки вебзастосунку для читання коміксів була створена стабільна, зручна та функціональна платформа, яка відповідає вимогам як користувачів, так і сучасним стандартам у розробці вебзастосунків. Вибір технологій та інструментів, ефективне використання архітектурних принципів та інтеграція бази даних дозволили створити рішення, яке можна без проблем масштабувати, додаючи нові функції та можливості для користувачів. Додатково, на основі отриманих знань, проєкт можна доопрацьовувати, покращуючи його функціональність та підвищуючи ефективність роботи.

# ПЕРЕЛІК ПОСИЛАНЬ

1. SensioLabs. Symfony – The Ultimate PHP Framework for High-Performance Web Applications. *SensioLabs Blog*. 2023. URL: https://sensiolabs.com/symfony (дата звернення: 08.09.2024).

2. Netgen. Building a scalable web application using Symfony and eZ Platform. *Netgen Blog*. 2022. URL: https://netgen.io/blog/building-scalable-web-applications (дата звернення: 08.09.2024).

3. SymfonyCasts. Best practices for building a Symfony web application. *SymfonyCasts Blog*. 2023. Vol. 12. P. 15–32. URL: https://symfonycasts.com (дата звернення: 09.09.2024).

4. Dots United. Leveraging Symfony for high-performance media-rich platforms. *Dots United Technical Papers*. 2023. Vol. 8, Issue 2. P. 47–58. URL: https://dotsunited.com (дата звернення: 09.09.2024).

5. Acilia. Advanced routing and templating with Symfony: A guide for developers. *Acilia Tech Papers*. 2023. Vol. 10, Issue 4. P. 67–81. URL: https://acilia.io/blog/symfony-routing (дата звернення: 11.09.2024).

6. Chao Y. et al. A comparative study of PHP frameworks: Evaluation criteria and implementation techniques. *International Journal of Soft Computing and Engineering*. 2019. Vol. 9, Issue 3. P. 110–115. DOI: 10.35940/ijrte.C1020.1083S19.

7. Prokofyeva O., Boltunova L. Comparing PHP frameworks for modern web application development. *Proceedings of the International Conference on Web Technologies*. 2018. P. 34–45.

8. IEEE Xplore. Design process for web applications with emphasis on Symfony. *IEEE Journals & Magazine*. 2020. DOI: 10.1109/1041948.

9. Milos P., Zurkiewicz T. Weighting and ranking criteria for PHP frameworks in academic projects. *Software Technologies Journal*. 2021. Vol. 15, Issue 2. P. 60–72.

10. Nirmal J. et al. Comparative analysis of open-source frameworks for web development. *International Journal of Advanced Research in Computer Science*. 2022. Vol. 13, Issue 4. P. 44–50.

# ДОДАТОК А

## Сторінки застосунку

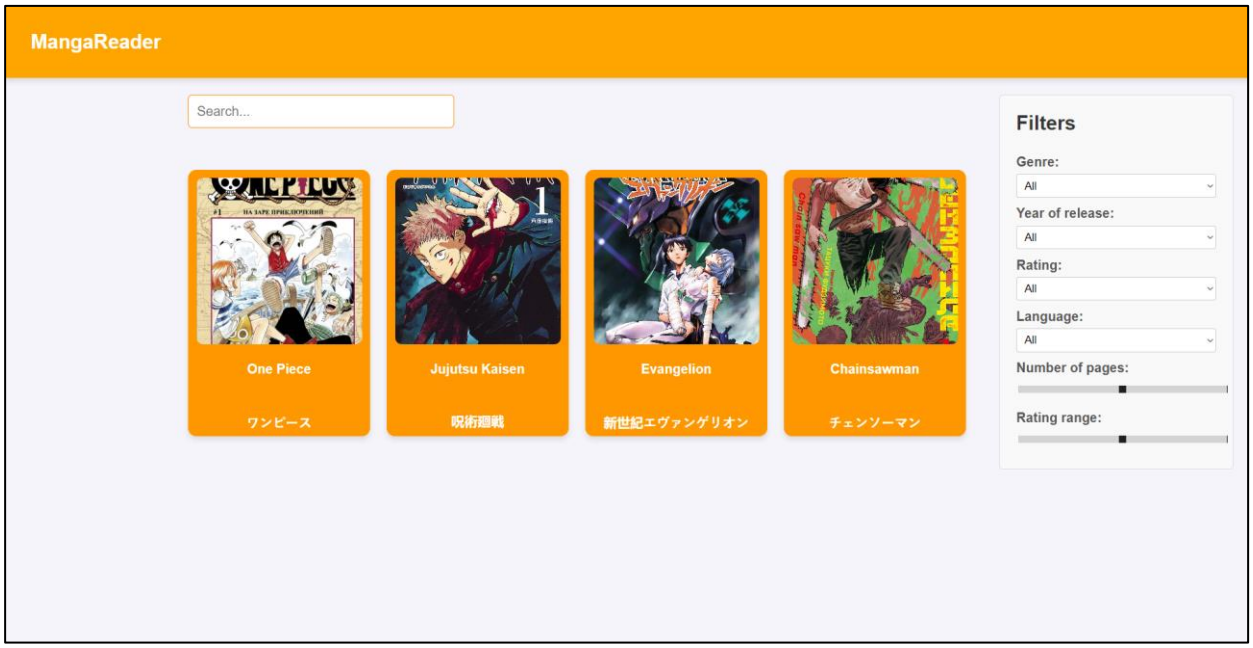



Рисунок А.1 – Головна сторінка

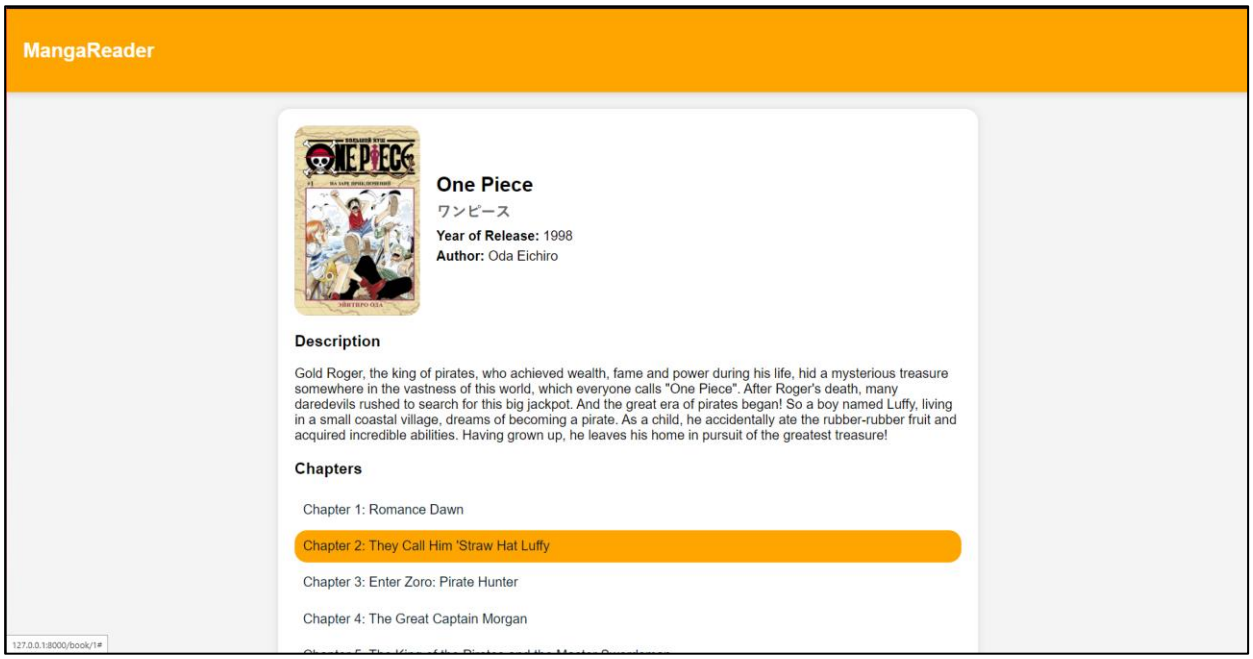



Рисунок А.2 – Сторінка книги

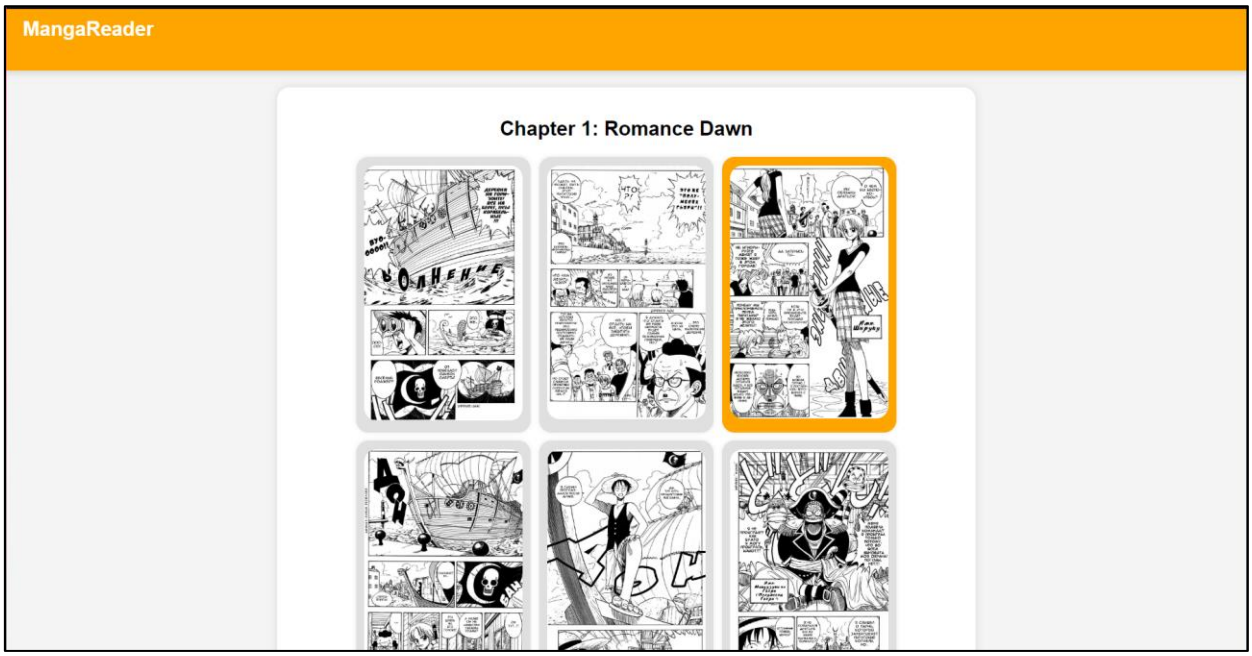



Рисунок А.3 – Пре перегляд глави




Рисунок А.4 – Блок коментарів