Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до курсової роботи
з дисципліни «Організація баз даних та знань»

за темою
Розробка інформаційної системи «Система онлайн-каталогу манґи»
з використанням СУБД SQL Server та мови запитів SQL

Виконала:
студентка 2 курсу, групи ІТІНФ-24 -1

Лазарєва Дарина Євгенівна
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник Яковлева О.В.
(посада, прізвище, ініціали)

2025

Харківський національний університет радіоелектроніки

(найменування закладу вищої освіти)

Факультет Інформаційно-аналітичних технологій та менеджменту

(повна назва)

Рівень вищої освіти перший (бакалаврський)

Кафедра Інформатики

(повна назва)

Спеціальність 122 Комп'ютерні науки

(код і повна назва)

Освітня програма Інформатика

(повна назва освітньої програми)

Дисципліна Організація баз даних та знань

(повна назва освітньої програми)

Курс 2 Група ІТІНФ-24-1 Семестр 3 .

ЗАВДАННЯ

на курсовий проект (роботу) студента

Лазарєви Дарини Євгенівни

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка інформаційної системи «Система онлайн-каталогу манґи» з використанням СУБД SQL Server та мови запитів SQL.

2. Строк здачі студентом закінченого проекту (роботи) 27.12.2025 .

3. Вихідні данні проекту (роботи) *мова запитів SQL, СУБД MS SQL Server, середовище MS SQL Server Management Studio, CASE – засіб візуального проектування даних Erwin, середовище розробки програмних систем MS Visual Studio, мова програмування C#, технологія ADO.NET, відомості про предметну область «Система онлайн-каталогу манґи»*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1. Реляційні бази даних: принципи проектування та використання

2. Проектування бази даних для предметної області «Система онлайн-каталогу манґи»

3. Проектування та розробка інформаційної системи для предметної області «Онлайн-каталог манґи»

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

6. Дата видачі завдання 03.10.2023.

КАЛЕНДАРНИЙ ПЛАН

№	Найменування етапів курсового проекту	Термін виконання етапів проекту	Примітки
1	*Отримання завдання*	*3.10.2023*	*Виконано*
2	*Вивчення теоретичного матеріалу, ознайомлення з програмним забезпеченням*	*3.10.23 – 30.10.23*	*Виконано*
3	*Розробка структури РБД*	*5.11.23 – 20.11.23*	*Виконано*
4	*Розробка функціональності ІС*	*21.11.23 – 17.12.23*	*Виконано*
5	*Тестування ІС*	*17.12.23*	*Виконано*
6	*Оформлення пояснювальної записки*	*18.12.23 – 21.12.23*	*Виконано*
7	*Захист курсової роботи*	*27.12.23*	*Виконано*

Студент_____ Керівник _____
(підпис) (підпис)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка: 87 с., 52 рис., 16 табл., 2 додатки, 19 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, ВЕБ-ДОДАТОК, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, РЕЛЯЦІЙНА БАЗА ДАНИХ, СУТНІСТЬ, ER-МОДЕЛЬ, МОДЕЛЬ «СУТНІСТЬ–ЗВ'ЯЗОК», МОВА ЗАПИТІВ SQL, MS SQL SERVER, ADO.NET, ASP.NET CORE MVC, C#, HTML, CSS, JAVASCRIPT, VISUAL STUDIO.

Курсова робота присвячена розробці інформаційної системи для онлайн-каталогу манґи, яка призначена для автоматизації процесів зберігання, керування та відображення цифрового контенту. Розроблена система забезпечує зручний доступ користувачів до каталогу манґи, глав і сторінок, а також підтримує функції пошуку, фільтрації, реєстрації та авторизації користувачів.

Інформаційна система дозволяє зберігати дані про манґу, жанри, статуси, типи, глави та сторінки, а також забезпечує зв'язки між ними на основі реляційної бази даних. Для збереження даних використовується СУБД MS SQL Server, а взаємодія із базою даних реалізована за допомогою технології ADO.NET та SQL-запитів.

Метою курсової роботи є розробка веб-орієнтованої інформаційної системи з використанням сучасних технологій для забезпечення швидкої та ефективної взаємодії між клієнтською частиною застосунку та базою даних. Для реалізації серверної логіки використано мову програмування C# та фреймворк ASP.NET Core MVC, для роботи з базою даних — MS SQL Server і мову запитів SQL, а для розробки інтерфейсу користувача — HTML, CSS та JavaScript. Середовищем розробки є Visual Studio 2022.

INFORMATION SYSTEM, WEB APPLICATION, DATABASE MANAGEMENT SYSTEM, RELATIONAL DATABASE, ENTITY,

ENTITY-RELATIONSHIP MODEL, SQL QUERY LANGUAGE, MS SQL SERVER, ADO.NET, ASP.NET CORE MVC, C#, HTML, CSS, JAVASCRIPT, VISUAL STUDIO.

The coursework is dedicated to the development of an information system called “Mankura”, which serves as an online manga catalog designed to automate the processes of storing, managing and displaying digital content. The developed system provides convenient access for users to manga titles, chapters and pages, and also supports functionality for search, filtering, user registration and authorization.

The system stores information about manga, genres, statuses, types, chapters and pages, and maintains relationships between these entities using a relational database. Data storage is implemented using MS SQL Server, while interaction with the database is carried out via ADO.NET and SQL queries.

The purpose of the work is to develop a web-oriented information system using modern technologies to ensure fast and efficient interaction between the client side of the application and the database. The server-side logic is implemented using the C# programming language and the ASP.NET Core MVC framework, database operations are performed using MS SQL Server, and the user interface is implemented with HTML, CSS and JavaScript. The development environment used is Visual Studio 2022.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

РБД – реляційна база даних

СУБД – система управління базами даних

РСУБД – реляційна система управління базами даних

ПК – первинний ключ

ЗК – зовнішній ключ

НФ – нормальна форма

DDL – Data Definition Language

DML – Data Modification Language

SQL – Structured Query Language

ВСТУП

У сучасному світі наше повсякденне життя тісно пов'язане з інтернет-ресурсами, і взаємодія з даними стала необхідністю для багатьох сфер діяльності, включаючи медичні установи. Створення і впровадження інформаційної системи для стоматологічної клініки виявляється ключовим етапом у поліпшенні надання стоматологічних послуг та оптимізації управління клінічною діяльністю.

Бази даних, як структуровані сукупності даних, відіграють важливу роль у забезпеченні ефективного зберігання і обробки інформації. У випадку стоматологічної клініки, вони дозволяють зберігати дані про пацієнтів, медичні історії, терміни прийому, а також автоматизувати процеси запису на прийом та контролю над формуванням планів лікування та фінансового обліку.

Наявність структурованої бази даних сприяє покращенню якості медичного обслуговування пацієнтів, дозволяючи зберігати і отримувати дані про проведені процедури та лікування з врахуванням індивідуальних потреб кожного пацієнта.

Метою даної роботи є розробка та впровадження інформаційної системи каталогу манґи — веб-платформи для розміщення, керування та читання манґи, яка дозволяє оптимізувати процеси роботи з контентом, спростити адміністрування каталогу та забезпечити зручний доступ користувачів до електронних матеріалів.

Для досягнення поставленої мети, використовуватимуться засоби систем управління базами даних, а саме мова програмування C# та платформа ASP.NET Core. Використання цих інструментів дозволить створити зручну та ефективну інформаційну систему для стоматологічної клініки, що сприятиме покращенню якості надання медичних послуг та спрощенню управління клінічними процесами.

1 РЕЛЯЦІЙНІ БАЗИ ДАНИХ ТА ПРИНЦИПИ ЇХ ПРОЕКТУВАННЯ

1.1 Виникнення реляційних баз даних

У червні 1970 року вчений-інформатик з IBM Едгар Ф. Кодд опублікував наукову статтю під назвою «Реляційна модель даних для великих спільних банків». Ця стаття представила новий спосіб моделювання даних. Він розробив спосіб побудови групи перехресно зв’язаних таблиць, який дозволить вам зберігати будь-яку частину даних лише один раз. База даних із такою структурою може відповісти на будь-яке запитання, якщо відповідь десь у ній зберігається. Дисковий простір використовувався б ефективно в той час, коли зберігання було дорогим. Ця стаття запустила бази даних у майбутнє [1].

Oracle випустила на ринок першу комерційну реляційну базу даних у 1979 році, а потім DB2, SAP Sysbase ASE та Informix.

У 1980-х і 90-х роках реляційні бази даних ставали все більш домінуючими, надаючи багаті індекси, щоб зробити будь-який запит ефективним. Об’єднання таблиць, термін для операцій читання, які об’єднують окремі записи в один, і транзакції, що означає комбінацію читань і особливо записів, розподілених по базі даних, були важливими. SQL, мова структурованих запитів, стала мовою даних, і розробники програмного забезпечення навчилися використовувати її, щоб запитувати те, що їм потрібно, і дозволяти базі даних вирішувати, як це доставити. Для запобігання сюрпризів у базу даних було впроваджено суворі гарантії.

1.2 Властивості реляційної моделі даних

Реляційна база даних — це тип бази даних, яка зберігає та надає доступ до точок даних, пов’язаних одна з одною. Реляційні бази даних базуються на реляційній моделі, інтуїтивно зрозумілому, простому способі представлення даних у таблицях. У реляційній базі даних кожен рядок у таблиці є записом з унікальним ідентифікатором, який називається ключем. Стовпці таблиці містять атрибути даних, і кожен запис зазвичай має значення для кожного атрибута, що полегшує встановлення зв’язків між точками даних [2].

Реляційна модель даних забезпечувала стандартний спосіб представлення та запиту даних, який міг використовувати будь-який додаток. З самого початку розробники визнали, що головною перевагою моделі реляційної бази даних є використання таблиць, які є інтуїтивно зрозумілим, ефективним і гнучким способом зберігання структурованої інформації та доступу до неї.

Згодом з’явилася ще одна сильна сторона реляційної моделі, коли розробники почали використовувати мову структурованих запитів (SQL) для запису та запиту даних у базі даних. Протягом багатьох років SQL широко використовувався як мова для запитів до бази даних. Базуючись на реляційній алгебрі, SQL забезпечує внутрішньо послідовну математичну мову, яка полегшує покращення продуктивності всіх запитів до бази даних. Для порівняння, інші підходи повинні визначати окремі запити.

Для забезпечення цілісності даних при розробці реляційних баз даних досить часто звертаються до такого поняття як нормалізація.

Нормалізація — це процес організації даних у базі даних. Він включає створення таблиць і встановлення зв’язків між цими таблицями відповідно до правил, розроблених як для захисту даних, так і для того, щоб зробити базу даних більш гнучкою за рахунок усунення надмірності та непослідовних залежностей [3].

Надлишкові дані витрачають дисковий простір і створюють проблеми з обслуговуванням. Якщо потрібно змінити дані, які існують у кількох місцях, дані мають бути змінені точно так само в усіх місцях. Зміну адреси клієнта легше реалізувати, якщо ці дані зберігаються лише в таблиці «Клієнти» і більше ніде в базі даних.

Що таке «непослідовна залежність»? Хоча користувачеві інтуїтивно зрозуміло шукати адресу конкретного клієнта в таблиці «Клієнти», може не мати сенсу шукати там зарплату працівника, який звертається до цього клієнта. Заробітна плата працівника пов’язана з працівником або залежить від нього, тому її слід перемістити до таблиці Employees. Неузгоджені залежності можуть ускладнити доступ до даних, оскільки шлях пошуку даних може бути відсутнім або пошкодженим.

Існує кілька правил нормалізації бази даних. Кожне правило називається «нормальною формою». Якщо перше правило дотримується, то кажуть, що база даних знаходиться в «першій нормальній формі». При дотриманні перших трьох правил вважається, що база даних знаходиться в «третій нормальній формі». Хоча можливі інші рівні нормалізації, третя нормальна форма вважається найвищим рівнем, необхідним для більшості програм.

Не використовуйте кілька полів в одній таблиці для зберігання схожих даних. Наприклад, для відстеження товарного запасу, який може надходити з двох можливих джерел, інвентарний запис може містити поля для коду постачальника 1 і коду постачальника 2.

Записи не повинні залежати ні від чого, крім первинного ключа таблиці (складеного ключа, якщо необхідно). Наприклад, розглянемо адресу клієнта в системі бухгалтерського обліку. Адреса потрібна для таблиці «Клієнти», а також для таблиць «Замовлення», «Доставка», «Рахунки-фактури», «Дебіторська заборгованість» і «Інформація». Замість того, щоб зберігати адресу клієнта як окремий запис у кожній із цих таблиць, збережіть її в

одному місці, або в таблиці «Клієнти», або в окремій таблиці «Адреси».

Значення в записі, які не є частиною ключа цього запису, не належать до таблиці. Загалом, щоразу, коли вміст групи полів може стосуватися кількох записів у таблиці, подумайте про розміщення цих полів в окремій таблиці.

Наприклад, у таблиці «Набір співробітників» можна включити назву та адресу університету кандидата. Але для групових розсилок потрібен повний список ВНЗ. Якщо інформація про університет зберігається в таблиці кандидатів, неможливо вказати університети без поточних кандидатів. Створіть окрему таблицю «Університети» та зв'яжіть її з таблицею «Кандидати» за допомогою ключа коду університету [3].

1.3 SQL як невід'ємна частина баз даних

SQL – це стандартно комп'ютерна мова для управління базами даних та для обробки даних. SQL використовується для запиту, вставки, оновлення та зміни даних. Ви можете розглядати SQL як засіб зв'язку між користувачем та СУБД (система управління БД) [5].

Мовою SQL описуються набори даних, які можуть допомогти відповісти на запитання. Використовуючи SQL потрібно застосовувати правильний синтаксис. Синтаксис – це набір правил, які забезпечують правильне поєднання елементів мови. Синтаксис SQL базується на синтаксисі англійської мови та має багато спільних елементів із синтаксисом Visual Basic for Applications (VBA) [6].

SQL є потужним інструментом для роботи з реляційними базами даних, і його використання дозволяє ефективно управляти та взаємодіяти з великим обсягом даних у сучасних інформаційних системах.

SQL використовується для визначення структури таблиць у базі даних.

Кожна таблиця представляє собою колекцію даних у вигляді рядків та стовпців.

Також дана мова дозволяє додавати нові рядки даних, оновлювати існуючі та видаляти непотрібні. SQL дозволяє визначати, хто і як може отримувати доступ до даних та які операції вони можуть виконувати.

1.4 ADO.NET як технологія для розробки інформаційних систем

ADO.NET — це технологія доступу до даних у середовищі .NET, яка забезпечує взаємодію додатків із реляційними базами даних за допомогою SQL-запитів. Вона входить до стандартної бібліотеки .NET і надає розробнику повний контроль над процесами підключення, виконання запитів, отримання та обробки даних.

На відміну від ORM-рішень, таких як Entity Framework або Dapper, ADO.NET не приховує процес роботи з базою даних, а дозволяє напряму керувати SQL-операціями, структурами таблиць та параметрами запитів. Такий підхід відповідає концепції **Database-First**, коли логіка роботи додатку адаптується до вже спроєктованої бази даних.

Основними перевагами використання ADO.NET є висока продуктивність, передбачуваність виконання запитів та мінімальні накладні витрати на обробку даних. Дана технологія не створює проміжних шарів абстракції, тому операції читання та запису виконуються швидше, що є важливим для веб-додатків з великим обсягом даних.

Робота з ADO.NET базується на використанні ключових компонентів, таких як *SqlConnection, SqlCommand, SqlDataReader*, які забезпечують встановлення з’єднання, виконання команд і отримання результатів запитів. Такий підхід вимагає від розробника знань SQL та принципів роботи

реляційних баз даних, проте надає більшу гнучкість і контроль у порівнянні з ORM-фреймворками.

ADO.NET також дозволяє легко працювати зі збереженими процедурами, параметризованими запитами та транзакціями, що є важливим для гарантування цілісності та безпеки даних. Завдяки прямому доступу до механізмів бази даних забезпечується стабільність та надійність програмного продукту.

Таким чином, вибір ADO.NET у даному проєкті є обґрунтованим, оскільки дана технологія забезпечує оптимальне поєднання продуктивності, керованості та сумісності із СУБД **MS SQL Server**, що повністю відповідає вимогам до розробки інформаційної системи онлайн-каталогу манґи.

1.5 Постановка задачі

Метою курсової роботи є розробка реляційної бази даних та веб-орієнтованої інформаційної системи для предметної області «Система онлайн-каталогу манґи». Розроблена система призначена для зберігання, керування та відображення інформації про манґу, розділи, глави та сторінки, а також для організації взаємодії користувачів із цифровим контентом.

Інформаційна система забезпечує можливість перегляду каталогу манґи, виконання пошуку та фільтрації за різними параметрами (жанр, тип, статус, рік випуску), а також перегляд окремих глав у режимі онлайн-читання. Система підтримує механізм авторизації та реєстрації користувачів, що дозволяє реалізувати персоналізовані функції, такі як коментування розділів та додавання творів до списку читання.

З боку адміністратора система надає засоби керування контентом: додавання, редагування та видалення манґи, жанрів, глав і сторінок, а також управління користувачами та коментарями. Це забезпечує повноцінне адміністрування каталогу та підтримку актуальності даних.

Зберігання структурованих відомостей про об’єкти предметної області здійснюється у реляційній базі даних MS SQL Server. Система реалізує зв’язки між сутностями «Манґа», «Жанр», «Автор», «Глава», «Сторінка», «Користувач», «Коментар», що забезпечує логічну цілісність та узгодженість даних. Для зберігання зображень (обкладинок та сторінок) використовується зовнішнє файлове сховище.

Інформаційна система орієнтована на дві основні категорії користувачів:

— **Кінцевий користувач (читач)** — переглядає каталог манґи, читає глави, залишає коментарі, керує власним списком читання.
— **Адміністратор** — здійснює модерацію даних, додає новий контент, редагує існуючий, керує користувачами та структурою бази даних.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати предметну область та визначити основні об’єкти та зв’язки між ними;
– розробити бізнес-правила для предметної області «Система онлайн-каталогу манґи»;
– побудувати ER-модель даних у нотації Чена;
– розробити логічну модель бази даних за допомогою CASE-засобу;
– реалізувати модель даних у СУБД MS SQL Server;
– розробити веб-додаток для взаємодії користувача з інформаційною системою;
– реалізувати авторизацію, перегляд каталогу, фільтрацію, читання глав та коментування;

– реалізувати адміністративний інтерфейс для керування контентом;
– провести тестування функціональності та працездатності системи;
– оформити пояснювальну записку відповідно до вимог.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ ДЛЯ ПРЕДМЕТНОЇ ОБЛАСТІ «ОНЛАЙН-КАТАЛОГ МАНҐИ»

2.1 Специфікація вимог

Інформаційна система «Mankura» призначена для організації, зберігання та керування електронним каталогом манґи, а також для забезпечення користувачам можливості перегляду та читання глав у веб-середовищі. Система орієнтована на роботу з великими наборами графічних та текстових даних і автоматизує процеси адміністрування контенту, підтримання структури творів, управління користувацькою активністю та взаємодії з базою даних.

Функціональне призначення системи полягає у наданні засобів для зручного доступу до каталогу манґи, перегляду інформації про твори, їх жанрову приналежність, тип, статус публікації, рік випуску, наявні глави та сторінки. Користувач має можливість переглядати вміст твору у вигляді послідовності сторінок, керувати списком читання, залишати коментарі, а також переходити між розділами та главами безпосередньо у веб-інтерфейсі. Система підтримує пошук та фільтрацію творів за різними параметрами, що підвищує зручність навігації та скорочує час доступу до потрібної інформації.

Важливою особливістю системи є наявність механізму розмежування прав доступу. У системі передбачено два основні типи користувачів — звичайний користувач (читач) та адміністратор. Звичайний користувач має доступ до перегляду каталогу, читання манґи, роботи зі списком читання, а також створення і видалення власних коментарів. Він взаємодіє лише з тими даними, які стосуються процесу перегляду контенту, та не має можливості модифікувати

структуру або склад системних даних.

Адміністратор системи, на відміну від звичайного користувача, відповідає за наповнення та підтримку бази даних. До його повноважень належать додавання, редагування та видалення записів про манґу, керування інформацією про жанри, типи та статуси, завантаження обкладинок, створення і структурування глав та сторінок, а також адміністрування коментарів і контролювання користувацької діяльності. Окремим напрямом відповідальності адміністратора є підтримка цілісності та узгодженості даних, оскільки від коректності зв’язків між сутностями залежить стабільність роботи всієї системи.

До системи висуваються також нефункціональні вимоги. Інтерфейс має бути інтуїтивно зрозумілим, логічно структурованим і придатним для використання як рядовими користувачами, так і адміністраторами. Система повинна забезпечувати стабільність роботи при збільшенні кількості творів, сторінок і користувачів, а також підтримувати можливість подальшого масштабування. Зберігання даних здійснюється у реляційній базі даних MS SQL Server, що гарантує структурованість даних і можливість виконання складних запитів, пов’язаних з навігацією, фільтрацією та логічною обробкою інформації.

Окрему увагу приділено роботі з графічними файлами сторінок та обкладинок. Система повинна забезпечувати коректне збереження, прив’язку та відображення відповідних ресурсів, а також мінімізувати дублювання даних та ризик втрати прив’язок між сторінками, главами та відповідними творами. Для цього передбачено структуровану організацію зберігання даних і чітку систему ідентифікації елементів.

Подальший розвиток системи може передбачати вдосконалення механізмів оптимізації роботи з великими обсягами графічних даних, розширення інструментів аналітики користувацької активності,

інтеграцію додаткових сервісів та впровадження нових функціональних можливостей, що сприятимуть підвищенню ефективності використання інформаційної системи.

2.2 Розробка бізнес-правил для предметної області «Онлайн каталог манґи»

У контексті проектування бази даних для онлайн-платформи читання манґи важливим етапом є формування бізнес-правил, які визначають структуру зберігання контенту, даних про користувачів, авторів та взаємодію читачів із твором. Ці правила повинні враховувати особливості публікації манґи, організацію глав і сторінок, механізм оцінювання та коментування, а також процес відстеження читання користувачами. Платформа має забезпечувати прозорість структури даних, підтримку багатьох жанрів, авторів і ролей, а також зручне керування публікаціями. Нижче наведено сформовані бізнес-правила, відповідно до яких у подальшому проєктуватиметься база даних цієї предметної області:

– у системі зберігається список користувачів, для кожного з яких фіксуються електронна пошта, ім’я користувача, пароль, дата реєстрації та роль у системі;

– кожен користувач має одну роль, а ролі зберігаються у окремому довіднику;

– у базі зберігається список манґи, для кожного твору фіксуються назва, опис, обкладинка, тип, статус публікації, видавництво та дата релізу;

– одна манґа може належати до одного видавництва та мати один статус і один тип, але може бути пов’язана з кількома жанрами;

– одна манґа може мати декілька авторів, при цьому кожен автор може брати участь у створенні кількох творів;

– манґа складається з глав, кожна глава належить лише одній манзі, свій номер, та номер тому;

– кожна глава містить кілька сторінок, кожна сторінка має URL сторінки та її порядковий номер;

– користувачі мають можливість залишати коментарі до манґи; кожен коментар пов’язаний з одним користувачем та однією манґою і містить текст та дату публікації;

– у системі зберігається інформація про процес читання: тип читання та мангу;

– один користувач може читати декілька манґ одночасно, а для кожної манґи зберігається окремий запис процесу читання.

2.3 Глосарій бази даних

Глосарій баз даних є ключовим інструментом для розкриття та розуміння термінології, яка оточує цю складну та важливу галузь інформаційних технологій. Цей глосарій створено з метою уточнення основних понять та термінів, пов'язаних із базами даних, для полегшення вивчення та впровадження концепцій у сфері керування інформацією [12].

Сприяючи кращому розумінню призначення основних об'єктів баз даних, глосарій визначає ключові терміни, що стосуються структури, функціональності та оптимізації баз даних. Це важливий ресурс, який надає чіткі визначення, спрощує комунікацію між фахівцями і забезпечує єдність термінології в контексті проектів баз даних.

Наведемо перелік первинних ключів та назв атрибутів, що використовуються в структурі бази даних каталогу манґи.

ID_Manga – унікальний ідентифікаційний номер манґи в базі даних.

NameManga – назва манґи.

Description – текстовий опис твору.

Cover – шлях до зображення обкладинки манґи.

ReleaseDate – дата виходу або початку публікації манґи.

ID_Publish – унікальний номер видавництва.

NamePublish – назва видавництва.

ID_Status – унікальний ідентифікатор статусу манґи.

NameStatus – назва статусу публікації (наприклад, активна, завершена, призупинена).

ID_Type – унікальний номер типу манґи.

NameType – назва типу твору (манґа, манхва, вебкомікс тощо).

ID_Genre – унікальний номер жанру.

NameGenre – назва жанру манґи.

ID_Author – унікальний номер автора.

NameAuthor – ім’я або псевдонім автора.

ID_Role – унікальний номер ролі.

NameRole – назва ролі (для авторів або користувачів).

ID_Chapter – унікальний номер глави манґи.

ChapterNumber – порядковий номер глави.

ChapterURL – посилання або шлях до перегляду глави.

VolumeNumber – номер тому, до якого належить глава.

ID_Page – унікальний номер сторінки глави.

PageNumber – порядковий номер сторінки.

ImagePath – шлях до зображення сторінки.

ID_User – унікальний номер користувача платформи.

Email – електронна адреса користувача.

UserName – ім’я користувача в системі.

Password – пароль облікового запису.

DescriptionUser – інформація профілю користувача / опис «про себе».

Avatar – шлях до зображення аватара.

RegistrationDate – дата реєстрації користувача.

ID_Comment – унікальний номер коментаря.

CommentText – зміст коментаря.

CommentDate – дата та час публікації коментаря.

Type – стан процесу читання (у процесі, прочитано, відкладено тощо).

2.4 Проектування структури реляційної бази даних

2.4.1 Логічна ER-діаграма предметної області

ER-діаграма (Entity-Relationship diagram) – це графічний інструмент, який використовується для моделювання взаємозв'язків між різними сутностями в інформаційних системах. ER-діаграми часто використовуються при проектуванні баз даних для представлення структури даних та взаємозв'язків між ними [13].

Основні елементи ER-діаграми включають:

– сутності (Entities) – представляють об'єкти чи концепції, які зберігають дані, наприклад, може бути сутність «Користувач» чи «Манга»;

– відносини (Relationships) – показують, як одна сутність пов'язана з іншою, наприклад, відношення «Манга містить кілька глав»;

– атрибути (Attributes) – представляють властивості сутностей, наприклад, у сутності «Користувач» можуть бути атрибути, такі як ім'я, електронна пошта, тощо;

– ключі (Keys) – ідентифікують унікальність записів в сутностях, наприклад, у сутності «Користувач» ключем може бути ідентифікатор користувача.

ER-діаграми надають зрозумілий спосіб відображення структури бази даних та її взаємозв'язків, що допомагає розробникам та аналітикам краще розуміти вимоги до системи і планувати структуру даних для оптимального зберігання та обробки інформації.

У нотації Чена сутності зображуються прямокутником, у середині якого є ім’я сутності. Слабка сутність позначається прямокутником з подвійною рамкою. Атрибути зображуються у вигляді овалу, з’єднаного з відповідним прямокутником. Первинні ключі виділяються підкресленням. Зв’язок позначається ромбом. Зв’язок між слабкою сутністю та тією, від якої вона залежить, позначається ромбом з подвійною рамкою. Учасники зв’язку приєднані до ромбу лінією. Для позначення типу зв’язку використовуються символи «1» та «M». Подвійна лінія означає повну участь сутності у зв’язку [14].

На рисунку 2.1 наведена логічна ER – діаграма предметної області «Онлайн каталог манґи» у синтаксисі Чена.

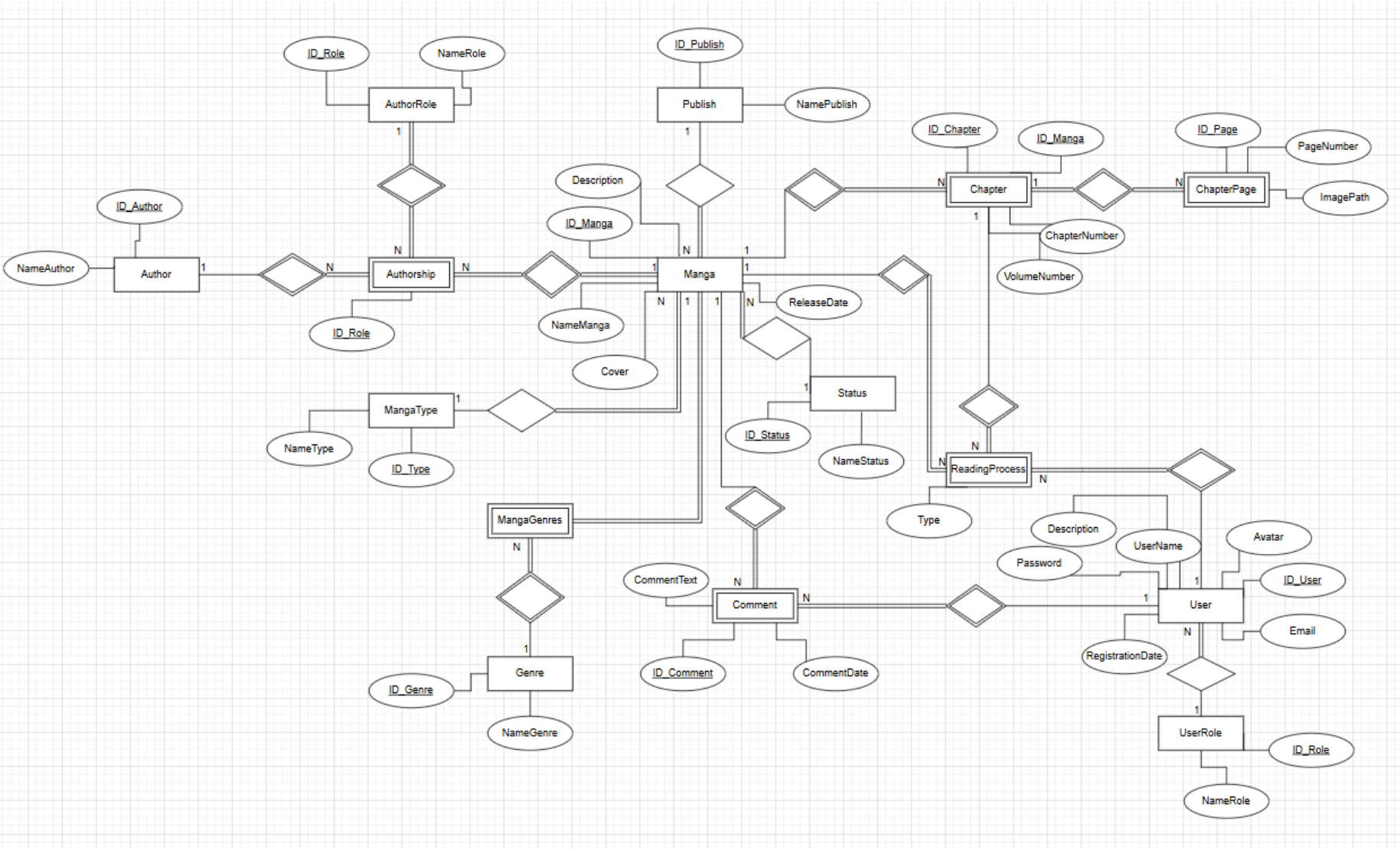

Рисунок 2.1 – Логічна ER-діаграма в нотації Чена

2.4.2 Перевірка на відповідність вимогам реляційної моделі даних

У даному підрозділі ми зосередимося на процесі перевірки на відповідність вимогам реляційної моделі даних. Цей етап є важливим у розробці бази даних, оскільки дозволяє переконатися, що структура даних відповідає визначеним вимогам та стандартам реляційної моделі. Ми розглянемо критерії та методи, які використовуються під час перевірки цілісності та оптимальності реляційної бази даних, забезпечуючи високу якість та ефективність взаємодії з інформацією в системі.

Таким чином, структура БД ПО «Каталог манґи», що представлена на рисунку 2.1, складається із таких сутностей: пацієнт, співробітники клініки, лікарі, розклад лікарів, спеціалізація, діагнози, діагнози по спеціалізації, ліки та медикаменти, послуги, розрахунок ліків на послугу, запис до лікаря, медична картка пацієнта, рентген, надані послуги, медикаменти для наданої послуги задля розрахунку вартості за індивідуальний випадок.

Розкриття ER-діаграми по Чену наведено на рисунку 2.2.

Для систематизації та визначення ступеня відповідності до встановлених стандартів, введена таблиця 2.1. У цій таблиці фіксуються результати перевірки, що дозволяє виявити та коригувати будь-які відхилення, забезпечуючи надійність та ефективність бази даних відповідно до вимог РМД. Це і є головною відмінністю логічного проектуванні від концептуального. Логічне проектування бази даних – процес створення моделі ПО на основі обраної моделі організації даних, але без урахування типу цільової СУБД та інших фізичних аспектів реалізації.

Обрана модель даних у цільовій СУБД – реляційна.

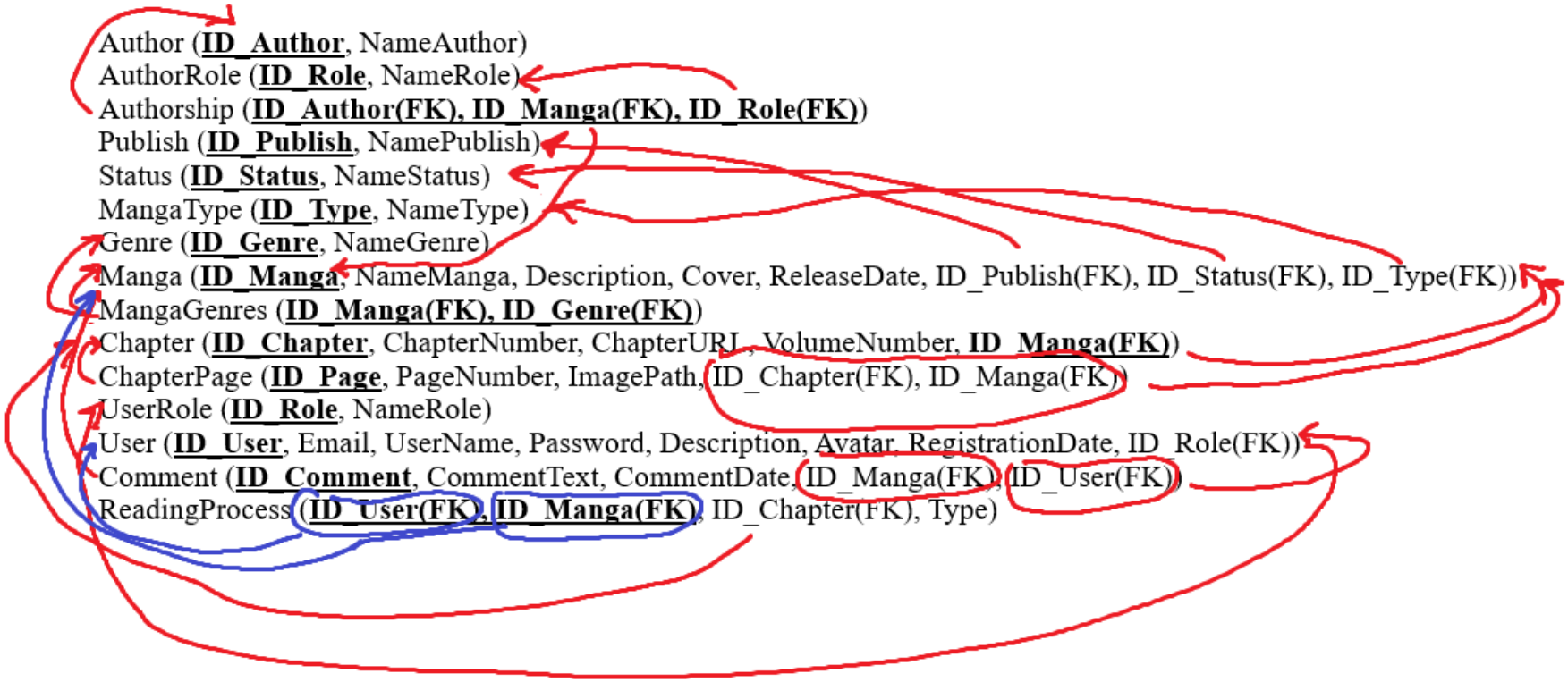

Рисунок 2.2 – Розкриття ER-діаграми по Чену

Таблиця 2.1 – Перевірка розробленої БД на відповідність вимогам РМД

Опис проблеми	Результат перевірки
Наявність двосторонніх зв’язків типу M:N	Відсутні
Наявність зв’язків із атрибутами	Відсутні
Наявність складних зв’язків	Відсутні
Наявність багатозначних атрибутів	Відсутні
Наявність рекурсивних зв’язків M:N	відсутні
Перевірка на відповідність вимогам 1НФ	Всі атрибути атомарні, тобто БД відповідає вимогам 1НФ
Перевірка на відповідність вимогам 2НФ	Наявні сутності зі складеними ПК, однак всі атрибути залежать виключно від складеного ПК, тому БД відповідає вимогам 2НФ
Перевірка на відповідність вимогам 3НФ	Всі сутності перевірено на наявність транзитивних залежностей, транзитивні залежності відсутні, тобто БД відповідає вимогам 3НФ
Переперевірка зв’язків типу 1:1	Виконано. Дані перетікають з однієї сутності до іншої відповідно до бізнес-правил. Порушень принципів РМД не виявлено
Аналіз рекурсивних зв’язків 1:1	Відсутні

2.4.3 Опис структури бази даних у документації

Таким чином, на фізичному рівні необхідно створити такі таблиці (табл. 2.2 – 2.16).

Таблиця 2.2 – Структура сутності Author

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Author	Числовий (INT) + IDENTITY(1,1)	PK	Унікальний ідентифікатор автора.

Продовження таблиці 2.2

NameAuthor	Текстовий (VARCHAR(100))		Ім’я або псевдонім автора. Not null

Таблиця 2.3 – Структура сутності AuthorRole

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Role	Числовий (INT) + IDENTITY(1,1)	PK	Унікальний номер ролі автора.
NameRole	Текстовий (VARCHAR(50))		Назва ролі (сценарист, художник тощо).

Таблиця 2.4 – Структура сутності Authorship

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Author	Числовий (INT)	Частина складеного ПК + FK references Author(ID_Author)	Посилання на автора.
ID_Manga	Числовий (INT)	Частина складеного ПК + FK references Manga(ID_Manga)	Посилання на манґу.
ID_Role	Числовий (INT)	FK references AuthorRole(ID_Role)	Вказує роль автора у творі.

Таблиця 2.5 – Структура сутності Publish

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Publish	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер видавництва.
NamePublish	Текстовий (VARCHAR(100))		Назва видавництва.

Таблиця 2.6 – Структура сутності Status

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Status	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер статусу.
NameStatus	Текстовий (VARCHAR(50))		Статус публікації (активна, завершена тощо).

Таблиця 2.7 – Структура сутності MangaType

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Type	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний ідентифікатор типу манґи.
NameType	Текстовий (VARCHAR(50))		Тип твору (манґа, манхва, вебкомікс).

Таблиця 2.8 – Структура сутності Manga

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Manga	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер манґи.
NameManga	Текстовий (VARCHAR(150))		Назва манґи.
Description	Текстовий (TEXT)		Опис твору.
Cover	Текстовий (VARCHAR(255))		Посилання на обкладинку.
ReleaseDate	Дата (DATE)		Дата публікації.
ID_Publish	Числовий (INT)	FK references Publish(ID_Publish)	Посилання на видавництво.
ID_Status	Числовий (INT)	FK references Status(ID_Status)	Статус публікації.
ID_Type	Числовий (INT)	FK references MangaType(ID_Type)	Тип манґи.

Таблиця 2.9 – Структура сутності Genre

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Genre	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер жанру.
NameGenre	Текстовий (VARCHAR(50))		Назва жанру.

Таблиця 2.10 – Структура сутності Genre

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Genre	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер жанру.
NameGenre	Текстовий (VARCHAR(50))		Назва жанру.

Таблиця 2.11 – Структура сутності MangaGenres

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Manga	Числовий (INT)	Частина складеного ПК + FK references Manga(ID_Manga)	Посилання на мангу.
ID_Genre	Числовий (INT)	Частина складеного ПК + FK references Genre(ID_Genre)	Посилання на жанр.

Таблиця 2.12 – Структура сутності Chapter

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Chapter	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер глави.
ChapterNumber	Числовий (INT)		Номер глави.
ChapterURL	Текстовий (VARCHAR(255))		Посилання на главу.
VolumeNumber	Числовий (INT)		Номер тому.

Продовження таблиці 2.12

ID_Manga	Числовий (INT)	FK references Manga(ID_Manga)	Посилання на манґу.

Таблиця 2.13 – Структура сутності ChapterPage

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Page	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер сторінки.
PageNumber	Числовий (INT)		Номер сторінки.
ImagePath	Текстовий (VARCHAR(255))		Посилання на зображення сторінки.
ID_Chapter	Числовий (INT)	FK references Chapter(ID_Chapter)	Посилання на главу.
ID_Manga	Числовий (INT)	FK references Manga(ID_Manga)	Посилання на манґу.

Таблиця 2.14 – Структура сутності UserRole

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Role	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер ролі користувача.
NameRole	Текстовий (VARCHAR(50))	—	Назва ролі.

Таблиця 2.15 – Структура сутності User

Назва атрибуту	Тип даних	Індекси	Зміст
ID_User	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер користувача.
Email	Текстовий (VARCHAR(100))	Унікальний індекс	Електронна адреса користувача.

Продовження таблиці 2.15

UserName	Текстовий (VARCHAR(50))		Логін користувача.
Password	Текстовий (VARCHAR(100))		Пароль.
Description	Текстовий (TEXT)		Опис профілю.
Avatar	Текстовий (VARCHAR(255))		Посилання на аватар.
RegistrationDate	Дата (DATE)		Дата реєстрації.
ID_Role	Числовий (INT)	FK references UserRole(ID_Role)	Роль користувача.

Таблиця 2.16 – Структура сутності Comment

Назва атрибуту	Тип даних	Індекси	Зміст
ID_Comment	Числовий (INT) + IDENTITY(1,1)	ПК	Унікальний номер коментаря.
CommentText	Текстовий (TEXT)		Текст коментаря.
CommentDate	Дата/час (DATETIME)		Дата публікації.
ID_Manga	Числовий (INT)	FK references Manga(ID_Manga)	Манґа, до якої належить коментар.
ID_User	Числовий (INT)	FK references User(ID_User)	Автор коментаря.

Таблиця 2.17 – Структура сутності ReadingProcess

Назва атрибуту	Тип даних	Індекси	Зміст
ID_User	Числовий (INT)	Частина складеного ПК + FK references User(ID_User)	Користувач.

Продовження таблиці 2.17

ID_Manga	Числовий (INT)	Частина складеного ПК + FK references Manga(ID_Manga)	Манґа.
ID_Chapter	Числовий (INT)	FK references Chapter(ID_Chapter)	Глава, на якій зупинився користувач.
Type	Текстовий (VARCHAR(30))		Статус читання (у процесі, прочитано тощо).

2.5 Побудова моделі даних за допомогою Erwin

Наступним етапом є побудова моделі даних з використанням CASE-засобу візуального проектування Erwin [15]. Цей етап є важливою частиною процесу розробки бази даних, оскільки надає можливість створення логічної та фізичної моделей, які визначають структуру та організацію даних в системі. На рисунках 2.3 та 2.4 подані відповідно логічна та фізична моделі, що відображають процес проектування бази даних за допомогою Erwin.

Логічна модель зорієнтована на концептуальне визначення сутностей, їхніх взаємозв'язків та атрибутів. У свою чергу, фізична модель визначає, як саме ці концепції будуть реалізовані в термінах таблиць, ключів, індексів та інших об'єктів реляційної бази даних.

2.6 Фізичне проектування моделі даних в СУБД MS SQL SERVER

У контексті системи управління базами даних (СУБД), можна аналізувати створену базу даних через діаграму, яка ілюструє відношення між таблицями та їх атрибутами. Ця діаграма представляє собою таблицю, де кожен стовпець відповідає атрибутам, рядки визначають записи, а назва таблиці знаходиться у верхній частині. При цьому атрибути, які є первинними ключами, позначаються символом ключа [16].

У випадку існування зовнішніх ключів, таблиці з'єднуються зв'язками, де значок ключа вказує, яке відношення є батьківським, а інші є дочірніми. Такий підхід до відображення структури бази даних дозволяє з легкістю розуміти взаємозв'язки між таблицями та їх ключовими атрибутами.

На рисунку 2.5 наведена схема бази даних, яка була згенерована за допомогою СУБД SQL Server 19. Ця схема надає візуальне представлення структури бази даних і вказує на зв'язки між різними таблицями, визначаючи

їхню взаємодію та структурні особливості.

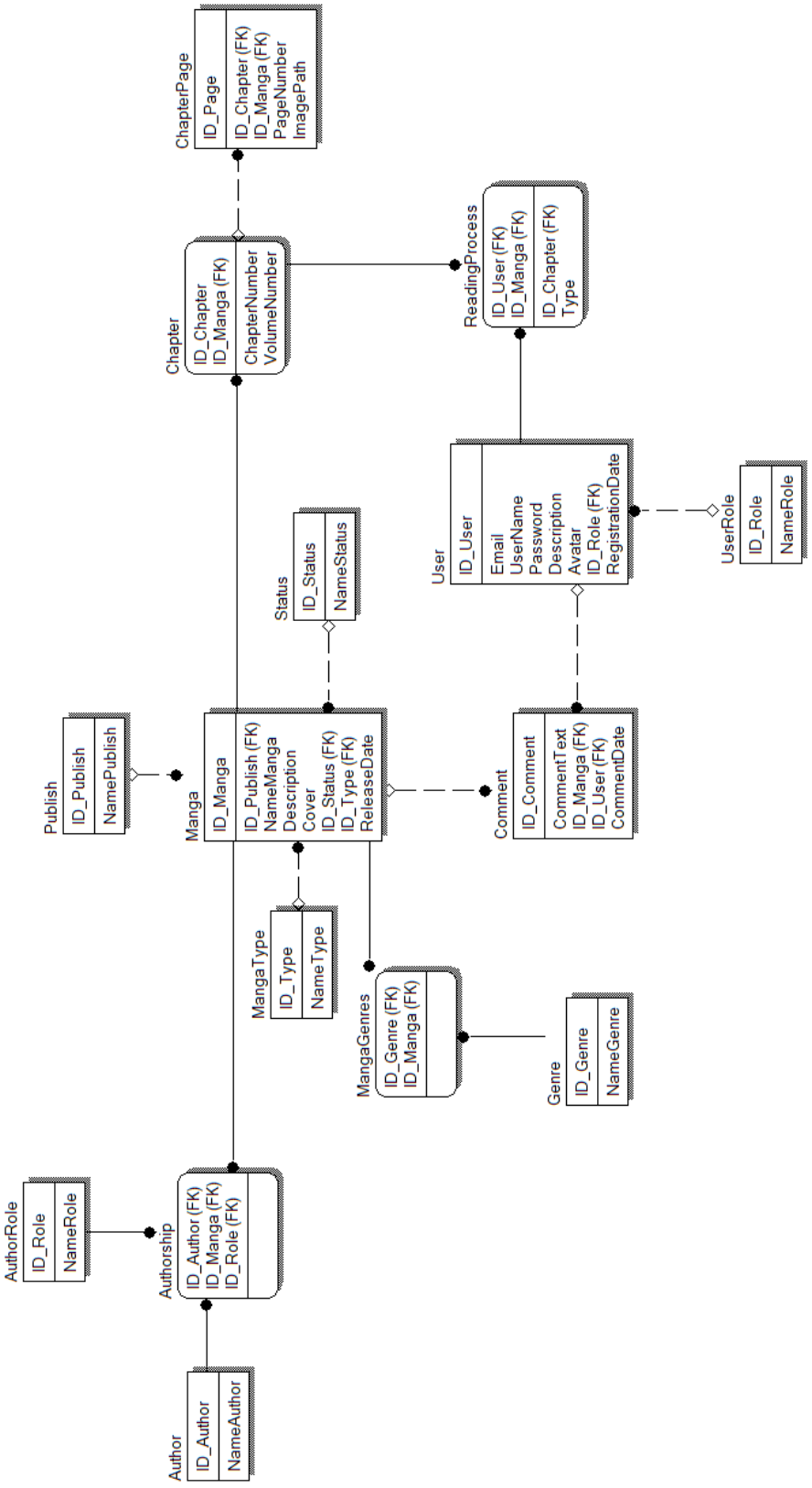

Рисунок 2.3 – Логічна модель даних

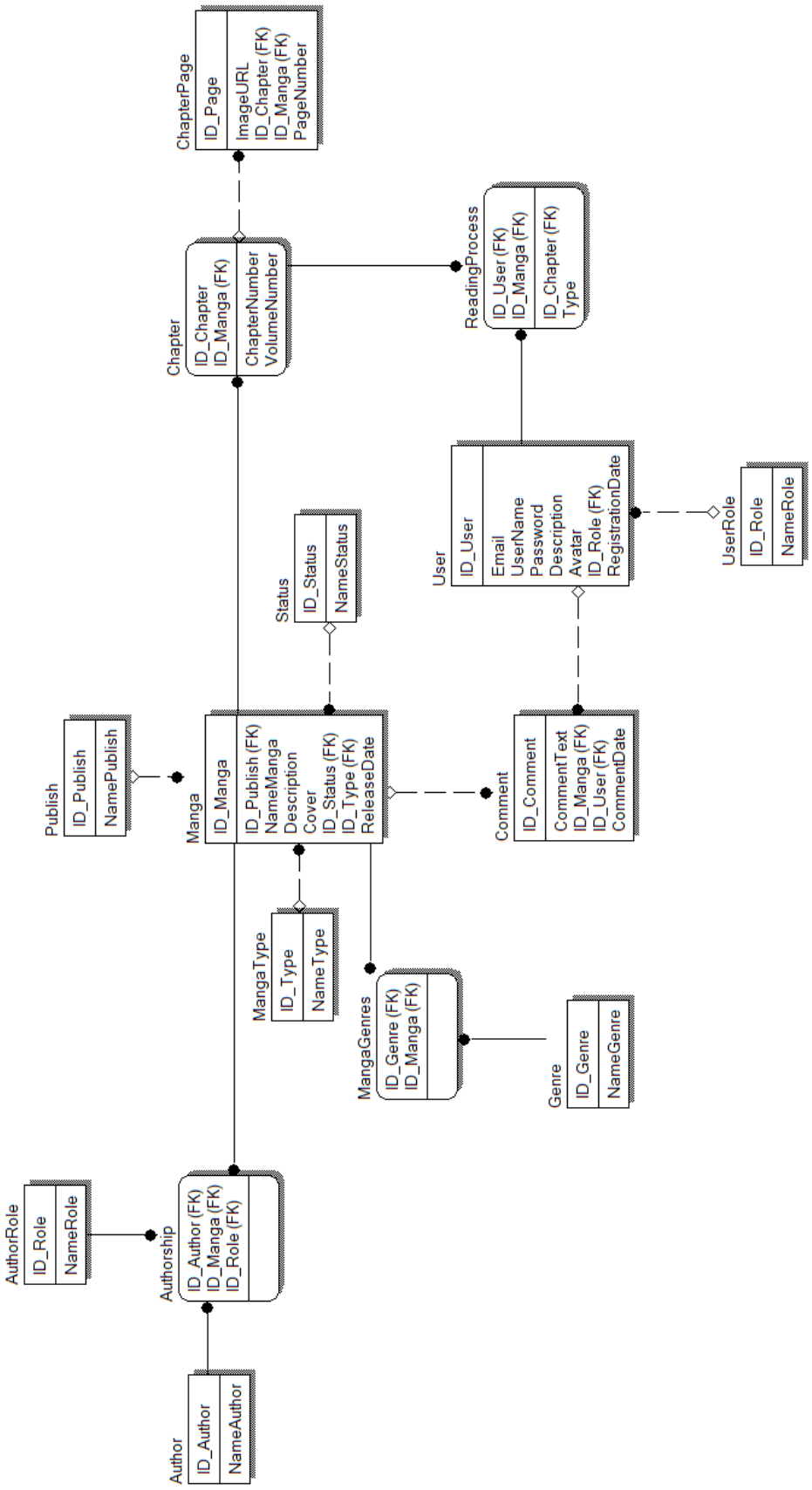

Рисунок 2.4 – Фізична модель даних

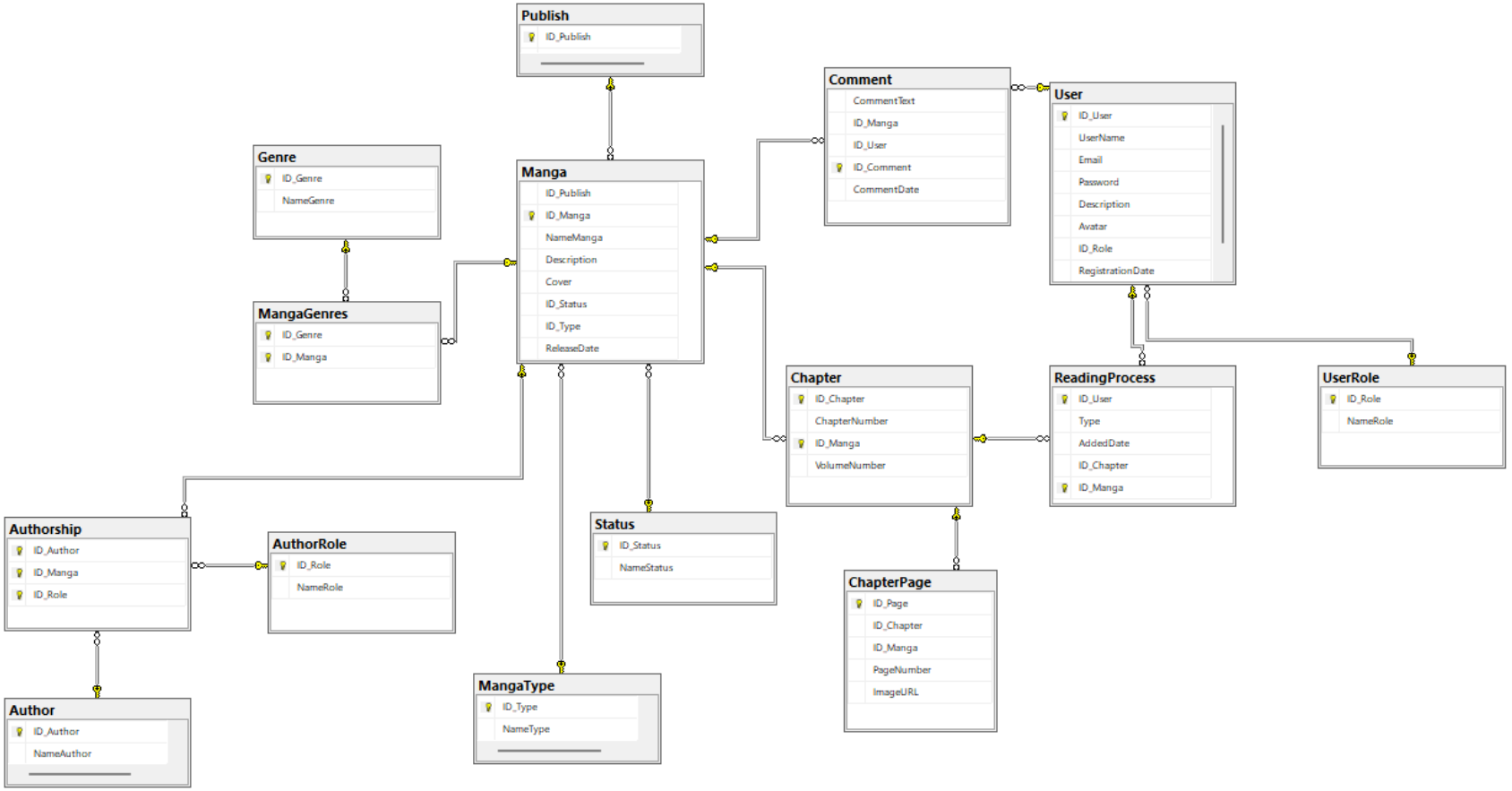

Рисунок 2.5 – Згенерована схема даних у СУБД MS SQL SER

2.1 Запити DDL SQL для створення структури БД та запити DML SQL для заповнення БД

У лістингу 2.1 наведено DDL скрипт для створення структури БД у SQL Server.

Лістинг 2.1 – Запити DDL SQL

```
CREATE TABLE Publish
(
    ID_Publish          integer IDENTITY ( 1,1 ) ,
    NamePublish         varchar(100)  NOT NULL ,
    CONSTRAINT XPKPublish PRIMARY KEY (ID_Publish ASC)
)
go

CREATE TABLE Status
(
    ID_Status           integer IDENTITY ( 1,1 ) ,
    NameStatus          varchar(20)  NOT NULL ,
    CONSTRAINT XPKStatus PRIMARY KEY (ID_Status ASC)
)
go

CREATE TABLE MangaType
(
    ID_Type             integer IDENTITY ( 1,1 ) ,
    NameType            varchar(50)  NOT NULL ,
    CONSTRAINT XPKMangaType PRIMARY KEY (ID_Type ASC)
)
go
```

```
CREATE TABLE Manga
(
    ID_Publish          integer  NOT NULL ,
    ID_Manga            integer IDENTITY ( 1,1 ) ,
    NameManga           varchar(200)  NOT NULL ,
    Description         text  NULL ,
    Cover               varchar(250)  NULL ,
    ID_Status           integer  NOT NULL ,
    ID_Type             integer  NOT NULL ,
    ReleaseDate         smallint NULL ,
    CONSTRAINT XPKManga PRIMARY KEY (ID_Manga ASC),
    CONSTRAINT R_3 FOREIGN KEY (ID_Publish) REFERENCES
Publish(ID_Publish)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION,
    CONSTRAINT R_26 FOREIGN KEY (ID_Status) REFERENCES Status(ID_Status)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION,
    CONSTRAINT R_31 FOREIGN KEY (ID_Type) REFERENCES
MangaType(ID_Type)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
)
go

CREATE TABLE UserRole
(
    ID_Role             integer IDENTITY ( 1,1 ) ,
    NameRole            varchar(20)  NOT NULL ,
    CONSTRAINT XPKUserRole PRIMARY KEY (ID_Role ASC)
)
go
```

```
CREATE TABLE [User]
(
    ID_User           integer IDENTITY ( 1,1 ) ,
    UserName          varchar(20)  NOT NULL UNIQUE,
    Email             varchar(255)  NOT NULL UNIQUE,
    Password          varchar(255)  NOT NULL ,
    Description       varchar(255)  NULL ,
    Avatar            varchar(255)  NULL ,
    ID_Role           integer  NOT NULL ,
    RegistrationDate  datetime  NOT NULL ,
    CONSTRAINT XPKUser PRIMARY KEY (ID_User ASC),
    CONSTRAINT R_13 FOREIGN KEY (ID_Role) REFERENCES
UserRole(ID_Role)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
)
go

CREATE TABLE MangaRating
(
    ID_Manga          integer  NOT NULL ,
    ID_User           integer  NOT NULL ,
    Rating            smallint  NULL ,
    RatingDate        datetime  NULL ,
    CONSTRAINT XPKMangaRating PRIMARY KEY (ID_Manga ASC,ID_User
ASC),
    CONSTRAINT R_29 FOREIGN KEY (ID_Manga) REFERENCES
Manga(ID_Manga)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION,
    CONSTRAINT R_30 FOREIGN KEY (ID_User) REFERENCES [User](ID_User)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
)
```

```
go

CREATE TABLE Author
(
    ID_Author          integer IDENTITY ( 1,1 ) ,
    NameAuthor          varchar(100)  NOT NULL ,
    CONSTRAINT XPKAuthor PRIMARY KEY (ID_Author ASC)
)
go

CREATE TABLE AuthorRole
(
    ID_Role            integer IDENTITY ( 1,1 ) ,
    NameRole            varchar(20)  NOT NULL ,
    CONSTRAINT XPKAuthorRole PRIMARY KEY (ID_Role ASC)
)
go

CREATE TABLE Authorship
(
    ID_Author          integer  NOT NULL ,
    ID_Manga           integer  NOT NULL ,
    ID_Role            integer  NOT NULL ,
    CONSTRAINT XPKAuthorship PRIMARY KEY (ID_Author ASC,ID_Manga
ASC,ID_Role ASC),
    CONSTRAINT R_4 FOREIGN KEY (ID_Author) REFERENCES
Author(ID_Author)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION,
    CONSTRAINT R_5 FOREIGN KEY (ID_Manga) REFERENCES Manga(ID_Manga)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION,
    CONSTRAINT R_28 FOREIGN KEY (ID_Role) REFERENCES AuthorRole(ID_Role)
                ON DELETE NO ACTION
```

```
                    ON UPDATE NO ACTION
)
go

CREATE TABLE Genre
(
    ID_Genre          integer IDENTITY ( 1,1 ) ,
    NameGenre         varchar(50)  NOT NULL ,
    CONSTRAINT XPKGenre PRIMARY KEY (ID_Genre ASC)
)
go

CREATE TABLE MangaGenres
(
    ID_Genre          integer  NOT NULL ,
    ID_Manga          integer  NOT NULL ,
    CONSTRAINT XPKMangaGenres PRIMARY KEY (ID_Genre ASC,ID_Manga
ASC),
    CONSTRAINT R_8 FOREIGN KEY (ID_Genre) REFERENCES Genre(ID_Genre)
                    ON DELETE NO ACTION
                    ON UPDATE NO ACTION,
CONSTRAINT R_9 FOREIGN KEY (ID_Manga) REFERENCES Manga(ID_Manga)
                    ON DELETE NO ACTION
                    ON UPDATE NO ACTION
)
go

CREATE TABLE Comment
(
    CommentText       text  NULL ,
    ID_Manga          integer  NOT NULL ,
    ID_User           integer  NOT NULL ,
    ID_Comment        integer IDENTITY ( 1,1 ) ,
    CommentDate       datetime  NOT NULL ,
```

```
        CONSTRAINT XPKComment PRIMARY KEY (ID_Comment ASC),
        CONSTRAINT R_21 FOREIGN KEY (ID_Manga) REFERENCES
Manga(ID_Manga)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION,
    CONSTRAINT R_22 FOREIGN KEY (ID_User) REFERENCES [User](ID_User)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
    )
    go

    CREATE TABLE Chapter
    (
        ID_Chapter          integer IDENTITY ( 1,1 ) ,
        ChapterNumber       integer  NULL ,
        ChapterURL          varchar(255)  NULL ,
        ID_Manga            integer  NOT NULL ,
        VolumeNumber        integer  NULL ,
        CONSTRAINT XPKChapter PRIMARY KEY (ID_Chapter ASC,ID_Manga ASC),
        CONSTRAINT R_35 FOREIGN KEY (ID_Manga) REFERENCES
Manga(ID_Manga)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
    )
    go

    CREATE TABLE ChapterPage
    (
        ID_Page             integer IDENTITY ( 1,1 ) ,
        ID_Chapter          integer  NOT NULL ,
        ID_Manga            integer  NULL ,
        PageNumber          integer  NULL ,
        ImagePath           varchar(255)  NOT NULL ,
        CONSTRAINT XPKChapterPage PRIMARY KEY (ID_Page ASC),
```

```
        CONSTRAINT R_37 FOREIGN KEY (ID_Chapter,ID_Manga) REFERENCES
Chapter(ID_Chapter,ID_Manga)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
    )
    go

    CREATE TABLE ReadingProcess
    (
        ID_User          integer  NOT NULL ,
        Type             varchar(50)  NULL
        CONSTRAINT Check_ReadingProcess_1534466357
                CHECK  ( [Type]='Read' OR [Type]='Readable' OR [Type]='Will be
read' OR [Type]='Favorites' ),
        AddedDate        datetime  NULL ,
        ID_Chapter       integer  NOT NULL ,
        ID_Manga         integer  NOT NULL ,
        CONSTRAINT XPKReadingProcess PRIMARY KEY (ID_User ASC,ID_Manga
ASC),
        CONSTRAINT R_14 FOREIGN KEY (ID_User) REFERENCES [User](ID_User)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION,
    CONSTRAINT R_36 FOREIGN KEY (ID_Chapter,ID_Manga) REFERENCES
Chapter(ID_Chapter,ID_Manga)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
    )
    go
```

У лістингу 2.2 наведено DML скрипт для створення структури БД у SQL Server.

Лістинг 2.2 – Запити DML SQL

```
INSERT INTO Author (NameAuthor) VALUES
('Shinichi Fukuda'),('Lina Lim'),('Kim Carnby'),('Jung Bomsik'),
('Jing Zhang'),('Hajime Isayama'),('Natsu Hyuuga'),('Nekokurage'),
('Eiichiro Oda'),('Hwang Youngchan'),('Toshio Sako'),('zizai_orangebrush'),
('Yongseok Jo'),('Masashi Kishimoto'),('Riichiro Inagaki'),('Boichi'),
('Tatsuya Endo'),('Aka Akasaka'),('Mengo Yokoyari');

INSERT INTO AuthorRole (NameRole)
VALUES('Writer'),('Artist');

INSERT INTO Authorship (ID_Author, ID_Manga, ID_Role)
VALUES(1, 1, 1), (1, 1, 2),
(2, 2, 1), (2, 2, 2),
(3, 3, 1), (4, 3, 2),
(5, 4, 1), (5, 4, 2),
(6, 5, 1), (6, 5, 2),
(7, 6, 1), (8, 6, 2),
(9, 7, 1), (9, 7, 2),
(3, 8, 1), (10, 8, 2),
(11, 9, 1), (11, 9, 2),
(12, 10, 1), (12, 10, 2),
(3, 11, 1), (3, 11, 2),
(13, 12, 1), (13, 12, 2),
(14, 13, 1), (14, 13, 2),
(15, 14, 1), (16, 14, 2),
(17, 15, 1), (17, 15, 2),
```

```
(18, 16, 1), (19, 16, 2);

    INSERT INTO Genre (NameGenre)
    VALUES
('Action'),('Adventure'),('Comedy'),('Drama'),('Josei'),('Fantasy'),
    ('Slice of Life'),('Sci-Fi'),('Horror'),('Mystery'),('Psychological'),('Romance'),
    ('Supernatural'),('Sports'),('Thriller'),('Historical'),('Mecha'),('Isekai'),('Cyberpunk'),
('Shonen'),('Shojo'),('Josei'), ('School'),('Seinen'),('Survival'),('Martial Arts');

    UPDATE Manga SET Cover = '/img/covers/1.jpg' WHERE ID_Manga = 1;
    UPDATE Manga SET Cover = '/img/covers/2.jpg' WHERE ID_Manga = 2;
    UPDATE Manga SET Cover = '/img/covers/3.jpg' WHERE ID_Manga = 3;
    UPDATE Manga SET Cover = '/img/covers/4.jpg' WHERE ID_Manga = 4;
    UPDATE Manga SET Cover = '/img/covers/5.jpg' WHERE ID_Manga = 5;
    UPDATE Manga SET Cover = '/img/covers/6.jpg' WHERE ID_Manga = 6;
    UPDATE Manga SET Cover = '/img/covers/7.jpg' WHERE ID_Manga = 7;
    UPDATE Manga SET Cover = '/img/covers/8.jpg' WHERE ID_Manga = 8;
    UPDATE Manga SET Cover = '/img/covers/9.jpg' WHERE ID_Manga = 9;
```

UPDATE Manga SET Cover = '/img/covers/10.jpg' WHERE ID_Manga = 10;

UPDATE Manga SET Cover = '/img/covers/11.jpg' WHERE ID_Manga = 11;

UPDATE Manga SET Cover = '/img/covers/12.jpg' WHERE ID_Manga = 12;

UPDATE Manga SET Cover = '/img/covers/13.jpg' WHERE ID_Manga = 13;

UPDATE Manga SET Cover = '/img/covers/14.jpg' WHERE ID_Manga = 14;

UPDATE Manga SET Cover = '/img/covers/15.jpg' WHERE ID_Manga = 15;

UPDATE Manga SET Cover = '/img/covers/16.jpg' WHERE ID_Manga = 16;

INSERT INTO Manga (ID_Publish, NameManga, ReleaseDate, Description, ID_Status, ID_Type)

VALUES

(1, 'My Dress-Up Darling', 2018, 'Wakana Gojo, a high school boy, spends his days perfecting the art of making traditional Hina dolls. His peaceful world is turned upside down when Marin Kitagawa, a popular, beautiful girl, discovers his secret passion—and demands he use his expert sewing skills to help her with her own ultimate passion: cosplay. This is a sparkling romantic comedy about finding common ground in the most unexpected places.', 1, 1),

(2, 'White Blood', 2019, 'Hayan Park has successfully concealed her true identity as a pure-blood vampire, living a quiet life as a normal student. But when a terrifying wave of human-vampire hybrids begins to threaten the city, she has no choice but to unleash the full power of her blood to protect her fellow humans. This thrilling fantasy blends action, romance, and dark

secrets.', 2, 2),

(2, 'Pigpen', 2018, 'A man wakes up on a remote, deserted island with no memory of his past, covered in blood. His only clue is a strange family living nearby, running a cannibalistic pig farm. Trapped in a nightmare of psychological terror, he must uncover the horrifying truth about this isolated family and his own identity before he becomes the next meal.', 2, 2),

(3, 'Ayeshah"s Secret', 2011, 'A poignant romantic drama centered around Ayeshah, a young woman struggling with her emotional distance and the dark secrets surrounding her past. As she navigates complex relationships and personal discoveries, she must face the hidden traumas that define her identity and ultimately find her path to love and acceptance.', 2, 2),

(4, 'Attack on Titan', 2009, 'Humanity lives confined within three enormous walls to protect themselves from the gargantuan, humanoid Titans that devour people seemingly without reason. After a colossal Titan breaks through the outermost wall, Eren Yeager vows revenge and joins the Survey Corps, determined to wipe out the creatures and uncover the dark mysteries hidden both within and beyond the walls.', 3, 1),

(1, 'The Apothecary Diaries', 2017, 'Maomao, a young woman trained as a pharmacist, is unexpectedly sold into service in the Emperor"s inner court. Though she tries to keep a low profile, her vast knowledge of medicine and toxins quickly earns the attention of the handsome chief eunuch, Jinshi. Now, she must use her sharp intellect to solve medical mysteries and palace intrigues in the Forbidden City.', 1, 1),

(5, 'One Piece', 1997, 'Monkey D. Luffy, a boy whose body gained the properties of rubber after unintentionally eating a Devil Fruit, sets out on the high seas. His ultimate goal is to become the King of the Pirates by finding the legendary treasure, the One Piece. He gathers a loyal crew, the Straw Hat Pirates, and faces global navies, rival pirates, and epic supernatural

challenges.', 1, 1),

(2, 'Sweet Home', 2017, 'After a personal tragedy, a recluse high school student moves into an old apartment complex. His solitude is violently interrupted when a sudden, terrifying phenomenon turns humans into grotesque monsters based on their deepest desires. Trapped inside with his neighbors, he must battle the monsters, both outside and within himself, to survive the apocalypse.', 2, 2),

(5, 'Usogui', 2005, 'Baku Madarame, known by the nickname "The Lie Eater," is a genius gambler who fears nothing. He challenges the deadliest gamblers in the world, wagering massive amounts—and often his own life—in extreme, high-stakes games. He is backed by the Kakerou, a clandestine organization that oversees the underworld"s most dangerous gambling rings.', 2, 1),

(6,'Devil"s Love', 2019, 'This fantasy romance manhua explores the complex relationships between angels and demons across multiple interwoven storylines. The central conflict revolves around the forbidden love and tension that arises when supernatural beings defy the cosmic order for affection. It is a gripping collection of dramatic stories about destiny and choice.', 3, 3),

(2, 'Shotgun Boy', 2021, 'The chilling sequel to "Bastard," this horror-thriller follows a character struggling to escape the traumatic legacy of his serial killer father. Set in a world where escape seems impossible, the story is a dark descent into the psychological battle against one"s own violent nature and the relentless pursuit of a fresh start.', 2, 2),

(2, 'Wind Breaker', 2013, 'Jo Ja-hyun is a model student and the school"s top academic star, but he secretly harbors a passion for cycling. When he is reluctantly pulled into the school"s amateur street cycling crew, he discovers the adrenaline of speed, the intensity of racing, and the enduring bonds of friendship that change his rigid, structured life forever.',

1, 2),

(5, 'Naruto', 1999, 'Orphaned and ostracized by his village, Naruto Uzumaki is an energetic and mischievous young ninja who carries a powerful, dark secret. Driven by the dream of becoming the Hokage (the strongest leader of his village), he trains relentlessly to earn respect, build unbreakable bonds, and face overwhelming challenges.', 2, 1),

(5, 'Dr. Stone', 2017, 'In a terrifying instant, every person on Earth is petrified into stone. Thousands of years later, the brilliant and scientifically minded Senku Ishigami awakens. Determined to rebuild human civilization from scratch, he embarks on an ambitious mission to rediscover and apply all of science, from the stone age forward, in a world ruled by nature.', 2, 1),

(5, 'Spy x Family', 2019, 'To carry out a crucial undercover mission, the master spy codenamed "Twilight" must hastily assemble a fake family. Unbeknownst to him, his "wife" is a deadly assassin, and his adopted "daughter" is a mind-reading telepath. This hilarious action-comedy follows their daily lives as they try to maintain their separate, explosive secrets while pretending to be a perfectly normal family.', 1, 1),

(5, 'Oshi no Ko', 2020, 'What if the path to the entertainment industry was opened to you from the very beginning? This story plunges into the dark side of the idol world: a doctor is unexpectedly reborn as the baby of his favorite idol, Ai Hoshino. As he grows up surrounded by secrets and ambition, he witnesses the glamour, lies, and dangerous realities of show business.', 1, 1);

```
INSERT INTO MangaGenres (ID_Genre, ID_Manga)
VALUES
(7,1), (3,1),(12,1),(23,1),(24,1),
(1,2),(12,2),(13,2),(4,2),
(9,3),(15,3),(11,3),(4,3),(2,3),(25,3),
(4,4),(10,4),(15,4),(11,4),(16,4),
```

```
(1,5),(4,5),(6,5),(20,5),
(10,6),(16,6),(4,6),(13,6),(5,6),
(2,7),(3,7),(6,7),(20,7),(1,7),
(15,8),(9,8),(6,8),(25,8),
(1,9),(11,9),(15,9),(24,9),
(12,10),(6,10),(13,10),(4,10),
(1,11),(9,11),(15,11),(11,11),(25,11),
(4,12),(14,12),(1,12),
(26,13), (2,13),(6,13),(3,13),(20,13),
(2,14),(8,14),(3,14),(20,14),
(3,15),(12,15),(7,15),(1,15),(8,15),(20,15),(10,15),
(11,16),(4,16),(10,16),(13,16),(5,16);

INSERT INTO MangaType (NameType)
VALUES('Manga'),('Manhwa'),('Manhua');

INSERT INTO Publish(NamePublish)
VALUES ('Square Enix'),('Naver Webtoon'),('Doki Doki'),('Kodansha'),('Shueisha'),('zizai_orangebrush');

INSERT INTO Status (NameStatus)
VALUES('Ongoing'),('Completed'),
('Break'),('Cancelled');

INSERT INTO UserRole (NameRole)
VALUES('Admin'), ('User');
```

Таким чином, спроектовано логічну схему реляційної бази даних для предметної області «Каталог манґи», яка чітко відображає взаємозв'язки між

різними сутностями. Логічна схема, виражена у синтаксисі Чена та спроектована у ERWIN, враховує основні аспекти клінічної діяльності, такі як облік пацієнтів, медичні історії, та терміни прийому. Фізична схема бази даних враховує оптимізацію для швидкого та ефективного доступу до інформації, забезпечуючи високий рівень продуктивності системи.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Розробка дизайну користувача

Програма розроблена на мові програмування C# з використанням фреймворку ASP.NET Core MVC. C# є сучасною, потужною та кросплатформною мовою програмування, яка дозволяє створювати високопродуктивні веб-додатки, що можуть працювати як на операційних системах Windows, так і на Linux та macOS завдяки .NET Core/.NET [17].

Для створення структури та інтерфейсу користувача застосовано архітектуру Model-View-Controller (MVC) у складі ASP.NET Core. Цей підхід забезпечує чіткий поділ логіки (контролери), даних (моделі) та представлення (вигляди Razor), що сприяє підтримуваність коду, його масштабованості та зручності тестування. Razor Views (.cshtml) дозволяють легко комбінувати HTML-размітку з кодом C#, створюючи динамічні, інтуїтивно зрозумілі та адаптивні веб-сторінки. Використання ASP.NET Core MVC допомагає розробляти сучасні, безпечні та ефективні веб-додатки з мінімальними зусиллями на організацію інтерфейсу.

Для зручності при розробці дизайну програми використовувалася програма Figma – хмарний сервіс для створення інтерфейсів користувача, прототипів та спільної роботи в реальному часі. Figma стала важливим інструментом для планування зовнішнього вигляду веб-сторінок, створення інтерактивних прототипів та узгодження концепцій ще на ранніх етапах розробки.

На рисунку 3.1 подано макет програми, створений за допомогою Figma. Цей макет дозволяє візуально оцінити структуру сторінок, компонування елементів інтерфейсу та потік взаємодії користувача, що значно полегшує уточнення дизайну та його подальшу реалізацію в коді.

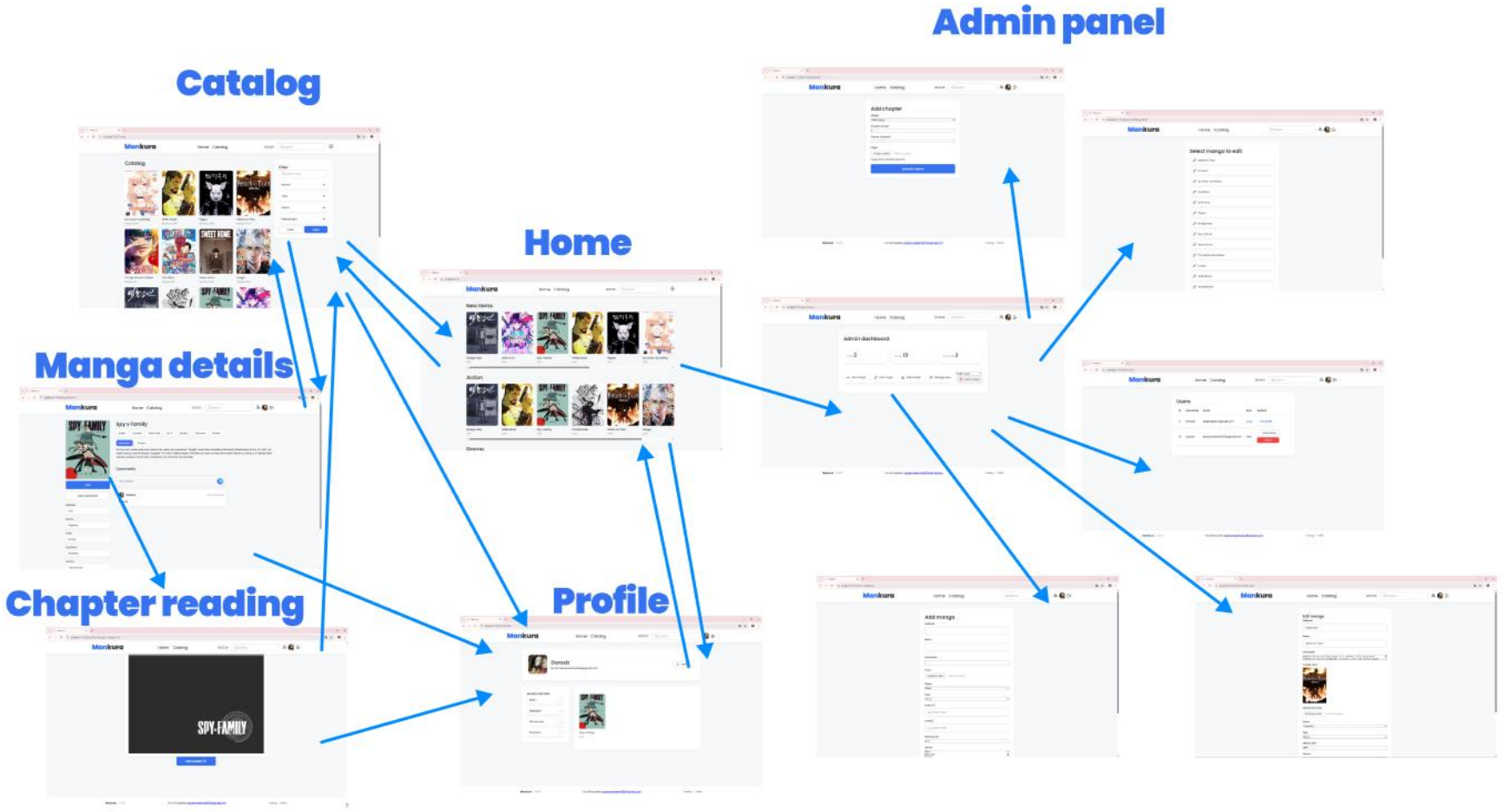

Рисунок 3.1 – Макет програми

3.2 Реалізація необхідної функціональності для інформаційної системи

3.2.1 Програмне забезпечення для розробки функціоналу інформаційної системи

Розробка функціоналу інформаційної системи «Mankura» здійснювалася з використанням сучасних програмних засобів, технологій та бібліотек, що забезпечують надійність, масштабованість та зручність підтримки програмного продукту. Основою системи є веб-орієнтована архітектура, що передбачає взаємодію клієнтської та серверної частин через браузерне середовище.

У процесі реалізації системи було використано такі програмні засоби та технології:

- мова програмування C#. Основна мова серверної частини застосунку, яка забезпечує реалізацію бізнес-логіки, обробку запитів користувачів, роботу з базою даних та керування структурою даних;
- фреймворк ASP.NET Core MVC. Використовується як основна платформа для створення веб-додатка за шаблоном Model–View–Controller, що забезпечує розділення логіки представлення, обробки запитів і роботи з даними, підвищуючи читабельність коду та зручність підтримки системи;
- .NET 8. Платформа виконання застосунку, яка забезпечує кросплатформеність, високу продуктивність та підтримку сучасних можливостей середовища розробки;
- ADO.NET та Microsoft.Data.SqlClient. Бібліотеки для виконання SQL-запитів, взаємодії з реляційною базою даних, отримання та обробки результатів, що дозволяють безпосередньо контролювати процес обробки даних та оптимізувати роботу із запитами;
- MS SQL Server. Система управління базами даних, що

використовується для зберігання структурованої інформації про манґу, жанри, статуси, типи, глави, сторінки, користувачів та коментарі; забезпечує цілісність, узгодженість та надійність даних;

- Razor-шаблони (Views). Використовуються для формування динамічних веб-сторінок та відображення даних, отриманих із контролерів;
- HTML, CSS та JavaScript. Застосовуються для розробки клієнтської частини, створення інтерфейсу користувача, стилізації елементів сторінок та реалізації інтерактивної взаємодії;
- Visual Studio. Середовище розробки, що використовувалося для написання коду, відлагодження, керування структурою проєкту та збіркою застосунку;
- система керування файлами у каталозі wwwroot. Призначена для зберігання графічних ресурсів (обкладинок та сторінок манґи), що забезпечує організовану структуру та доступність зображень у процесі роботи користувачів із системою.

3.2.2 Підключення до БД

Підключення до бази даних є одним з ключових етапів при розробці інформаційних систем, оскільки саме від нього залежить коректність збереження, обробки та доступу до даних користувачів. У веб-додатку «Mankura» для роботи з базою даних використовується підхід на основі технології ADO.NET та провайдера Microsoft.Data.SqlClient, що забезпечує прямий контроль над виконанням SQL-запитів і високу продуктивність доступу до даних.

Для зберігання параметрів підключення застосовується конфігураційний файл appsettings.json (рис. 3.2), у якому міститься рядок

підключення (connection string) до бази даних. Такий підхід дозволяє відокремити налаштування середовища від програмного коду та спрощує переналаштування застосунку у разі зміни сервера або структури бази даних.

Під час запуску веб-додатка рядок підключення зчитується автоматично через механізм конфігурації ASP.NET Core і передається у відповідні репозиторії та сервіси, що взаємодіють з базою даних. Отримання підключення відбувається шляхом створення об’єкта SqlConnection, після чого відкривається з’єднання з БД і виконуються SQL-запити на запис або читання даних.

```
{
    "ConnectionStrings": {
        "DefaultConnection": "Server= DESKTOP-HREFMFG\\MSSQLSERVER01;Database=MankuraDB;Trusted_Connection=True;TrustServerCertificate=True"
    },
```

Рисунок 3.2 – Вміст файлу конфігурації з рядком підключення до БД

Наступним кроком є створення класу Db, призначеного для отримання об’єкта підключення до бази даних (рис. 3.3).

```
using Microsoft.Data.SqlClient;

Ссылок: 4
public class Db
{
    private readonly string _cs;

    Ссылок: 0
    public Db(IConfiguration config)
    {
        _cs = config.GetConnectionString("DefaultConnection");
    }

    Ссылок: 6
    public SqlConnection GetConnection()
    {
        return new SqlConnection(_cs);
    }
}
```

Рисунок 3.3 – Розроблений клас Db для управління підключенням до бази

даних

Клас Db є сервісом, який реєструється в контейнері залежностей ASP.NET Core і використовує механізм Dependency Injection. Конструктор класу отримує об’єкт IConfiguration, за допомогою якого зчитується рядок підключення з файлу appsettings.json. Метод GetConnection() повертає новий екземпляр SqlConnection з уже налаштованим рядком підключення.

Використання такого класу дозволяє уніфікувати доступ до підключення в усіх частинах програми (контролерах, репозиторіях, сервісах). Завдяки Dependency Injection ASP.NET Core автоматично створює та передає екземпляр Db туди, де він потрібен, що є сучасним і рекомендованим підходом замість ручного управління синглтонами.

Екземпляр Db створюється в межах одного HTTP-запиту (scoped), що забезпечує безпечну та ефективну роботу з підключеннями. Автоматичне зчитування конфігурації та ін’єкція залежностей дозволяють уникнути жорсткого зв’язування коду з конкретним рядком підключення та полегшують тестування. Перевірка наявності рядка підключення відбувається на рівні конфігурації ASP.NET Core – якщо його немає, додаток видасть виняток на етапі запуску, що допомагає виявити проблему на ранній стадії.

Цей клас допомагає впорядкувати та забезпечити однаковий і безпечний доступ до бази даних в усьому додатку, спрощуючи управління підключенням та полегшуючи можливі зміни конфігурації.

Останнім кроком є реєстрація сервісу в контейнері залежностей у файлі Program.cs (рис. 3.4).

```
using Mankura.Data;
using Mankura.Services;
using Microsoft.AspNetCore.Authentication.Cookies;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddScoped<Db>();
```

Рисунок 3.4 – Реєстрація сервісу Db у файлі Program.cs

Подібний підхід дозволяє зручно та централізовано отримувати конфігураційні дані з appsettings.json, зокрема рядок підключення до бази даних, і використовувати його для подальшої роботи у програмі. Це полегшує управління налаштуваннями та їхню зміну без зміни вихідного коду.

ConfigurationBuilder (вбудований в ASP.NET Core) автоматично завантажує appsettings.json як джерело конфігурації. Метод GetConnectionString отримує рядок підключення з розділу "ConnectionStrings" за іменем "DefaultConnection". Якщо рядок підключення не знайдено або порожній, ASP.NET Core видасть виняток на етапі конфігурації додатка. Реєстрація сервісу виконується за допомогою builder.Services.AddScoped<Db>(), що робить клас доступним для ін’єкції в контролери та інші компоненти.

Такий підхід дозволяє підвищити рівень безпеки при зверненні до бази даних при розробці застосунку, а також забезпечити досить просте підключення до інших баз даних (з іншого сервера) шляхом зміни рядка підключення у файлі appsettings.json.

3.2.3 Реалізація перегляду об’єктів

Для реалізації перегляду інформації, що міститься у реляційних таблицях бази даних, було вирішено використати Razor Views у фреймворку ASP.NET Core MVC для динамічного відображення даних у зручному та сучасному веб-інтерфейсі (рис. 3.5 – 3.7).

Razor Views (.cshtml) дозволяють комбінувати HTML-розмітку з кодом C#, що дає можливість легко виводити списки та детальну інформацію з бази даних у вигляді адаптивних карток, сіток або каруселей. У веб-додатку «Mankura» дані про мангу, глави та коментарі відображаються саме за допомогою Razor Views: на головній сторінці – у горизонтальних секціях за жанрами та новинками, у каталозі – у вигляді сітки карток з обкладинками, а на сторінці детального перегляду – з повним описом, жанрами, характеристиками та блоком коментарів.

Використання Razor Views є оптимальним рішенням для веб-додатку, оскільки забезпечує кращу продуктивність, масштабування та користувацький досвід у браузері.

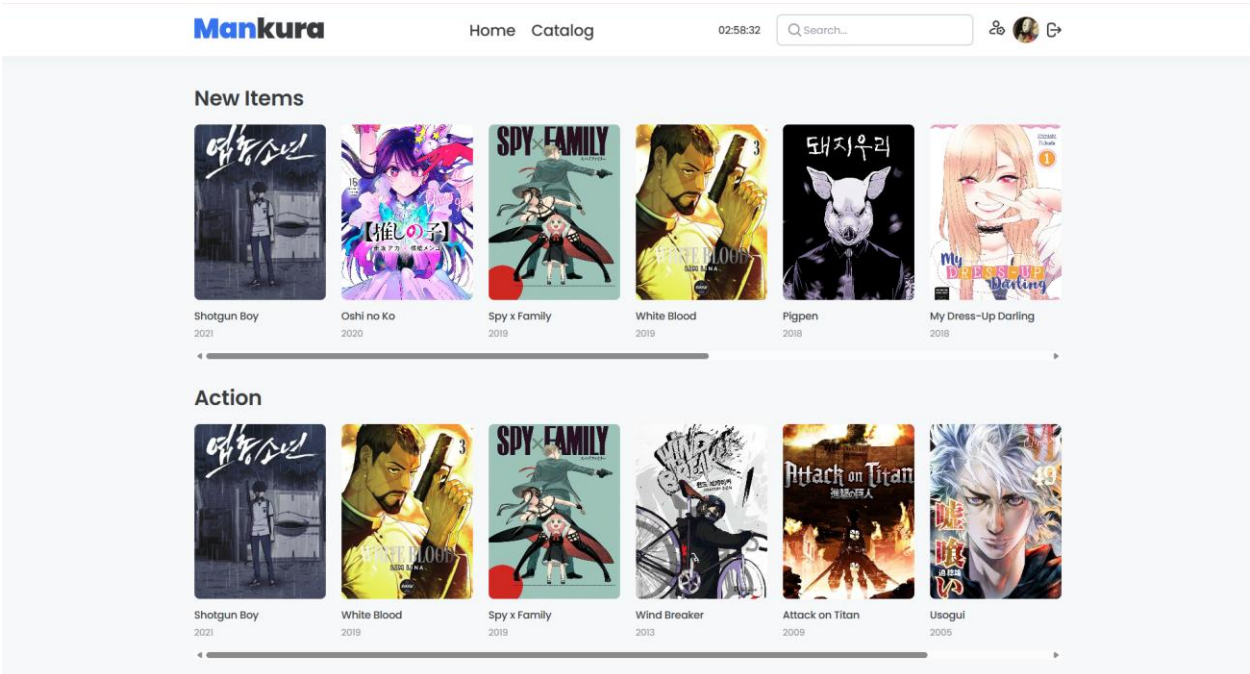

Рисунок 3.5 – Головна сторінка з відображенням новинок та тематичних секцій за жанрами

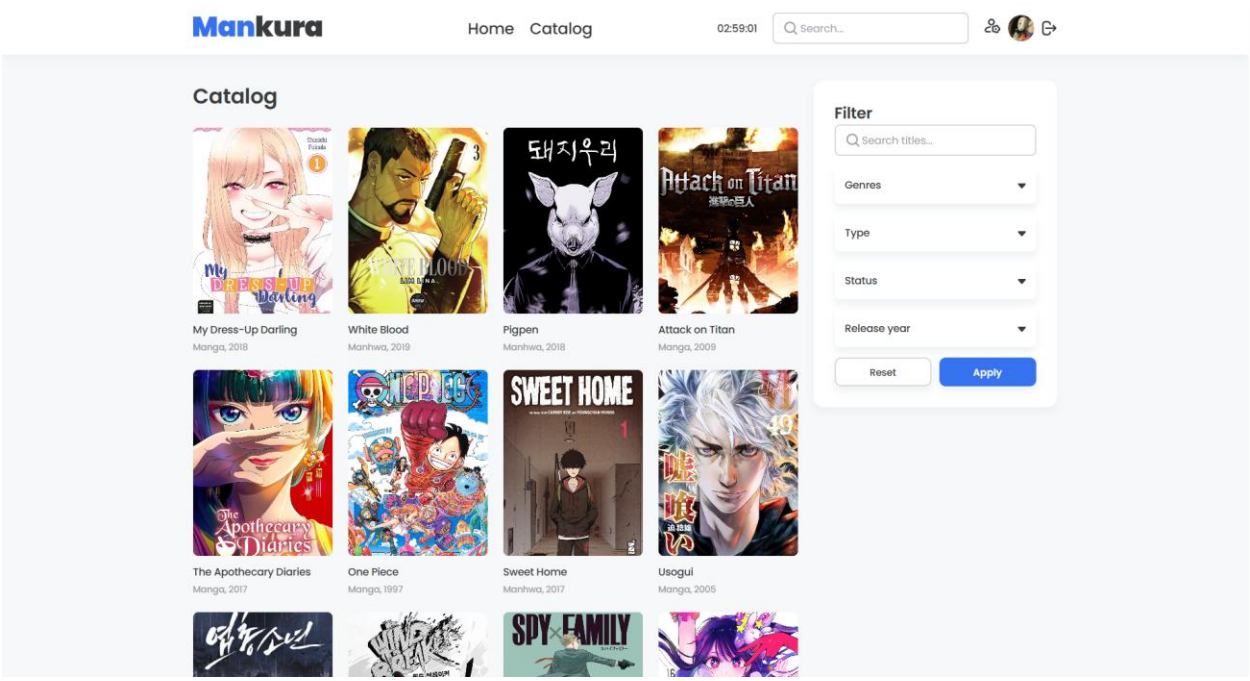

Рисунок 3.6 – Сторінка каталогу з сіткою карток манги та панеллю фільтрів

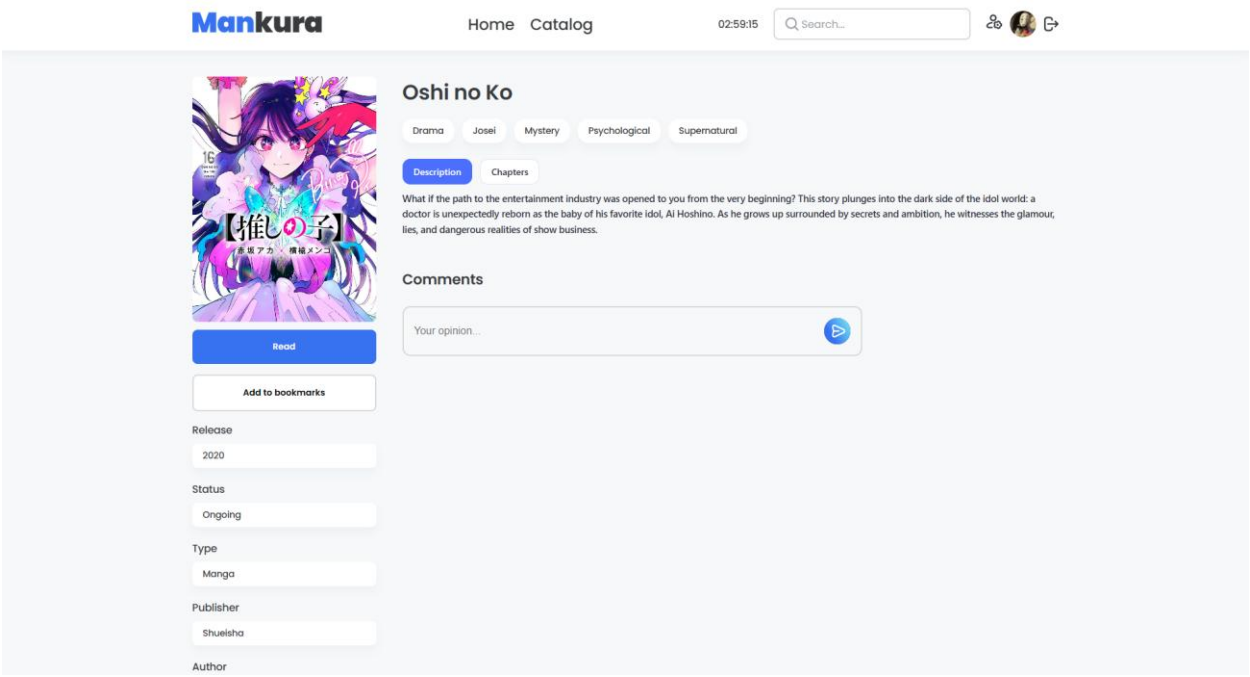

Рисунок 3.7 – Сторінка детального перегляду манги з описом, характеристиками та коментарями

У репозиторії MangaRepository реалізовано методи для отримання даних: метод GetAll() (рис. 3.8) повертає список усіх манг з пов’язаними сутностями (статус, тип, видавець) через JOIN-запити, сортуючи за назвою. Метод GetById(int id)(рис. 3.9) повертає детальну інформацію про одну мангу, включаючи жанри, авторів та художників за допомогою додаткових методів GetGenresForManga, GetAuthorsForManga, GetArtistsForManga. Дані передаються з контролера до view через модель або ViewBag, де відображаються за допомогою @foreach.

Такий підхід забезпечує високу гнучкість представлення даних, адаптивність інтерфейсу для різних пристроїв, чіткий поділ логіки від представлення та інтеграцію з CSS для привабливого вигляду.

```
public List<Manga> GetAll()
{
    var mangas = new List<Manga>();

    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = @"
    SELECT
        m.ID_Manga,
        m.NameManga,
        m.Description,
        m.Cover,
        m.ReleaseDate AS ReleaseYear,
        s.NameStatus,
        t.NameType,
        p.NamePublish
    FROM Manga m
    JOIN Status s ON s.ID_Status = m.ID_Status
    JOIN MangaType t ON t.ID_Type = m.ID_Type
    JOIN Publish p ON p.ID_Publish = m.ID_Publish
    ORDER BY m.NameManga ASC
    ";

    using var cmd = new SqlCommand(sql, conn);
    using var reader = cmd.ExecuteReader();

    while (reader.Read())
    {
        mangas.Add(new Manga
        {
            Id = reader.GetInt32(reader.GetOrdinal("ID_Manga")),
            Name = reader.GetString(reader.GetOrdinal("NameManga")),
            Description = reader["Description"] as string,
            Cover = reader["Cover"] as string,

            ReleaseYear = reader.GetInt16(reader.GetOrdinal("ReleaseYear")),
            Status = reader.GetString(reader.GetOrdinal("NameStatus")),
            Type = reader.GetString(reader.GetOrdinal("NameType")),
            Publisher = reader.GetString(reader.GetOrdinal("NamePublish"))
        });
    }


    return mangas;
}
```

Рисунок 3.8 – метод GetAll(), який повертає список усіх манг

```
public Manga? GetById(int id)
{
    Manga? manga = null;

    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = @"
        SELECT
            m.ID_Manga,
            m.NameManga,
            m.Description,
            m.Cover,
            m.ReleaseDate AS ReleaseYear,
            s.NameStatus,
            t.NameType,
            p.NamePublish

        FROM Manga m
        JOIN Status s ON s.ID_Status = m.ID_Status
        JOIN MangaType t ON t.ID_Type = m.ID_Type
        JOIN Publish p ON p.ID_Publish = m.ID_Publish

        WHERE m.ID_Manga = @id;
    ";

    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@id", id);

    using var reader = cmd.ExecuteReader();

    if (reader.Read())
    {
        manga = new Manga
        {
            Id = reader.GetInt32(reader.GetOrdinal("ID_Manga")),
            Name = reader.GetString(reader.GetOrdinal("NameManga")),
            Description = reader["Description"] as string,
            Cover = reader["Cover"] as string,
            ReleaseYear = reader.GetInt16(reader.GetOrdinal("ReleaseYear")),
            Status = reader.GetString(reader.GetOrdinal("NameStatus")),
            Type = reader.GetString(reader.GetOrdinal("NameType")),
            Publisher = reader.GetString(reader.GetOrdinal("NamePublish"))
        };
    }

    reader.Close();

    if (manga != null)
    {
        manga.Genres = GetGenresForManga(id); ;
        manga.Authors = GetAuthorsForManga(id);
        manga.Artists = GetArtistsForManga(id);
    }

    return manga;
}
```

Рисунок 3.9 – Метод GetById(int id), який повертає детальну

інформацію про одну манґу, включаючи жанри, авторів та художників за допомогою додаткових методів

3.2.4 Створення та керування записами манґи в каталозі

Операції вставки, редагування, видалення та пошуку є основними при взаємодії користувача з даними через застосунок. У проекті ці операції реалізовано в репозиторіях за допомогою ADO.NET з прямими SQL-запитами для максимального контролю та продуктивності.

Додавання нової манґи починається з того, що адміністратор заповнює форму у Razor-представленні, де на стороні клієнта відбувається первинна перевірка правильності введених значень. Після надсилання форма обробляється методом AddManga (рис. 3.10) в контролері AdminController, який приймає модель та виконує серверну валідацію через механізм ModelState. У разі помилки введення даних форма повертається з відповідними повідомленнями, а у випадку успішної валідації виконується передача даних у відповідні методи репозиторію.

На цьому етапі відбувається перевірка унікальності назви манґи. Для цього використовується метод ExistsByName (рис. 3.11). у класі MangaRepository, який виконує SQL-запит виду SELECT COUNT(*) FROM Manga WHERE NameManga = @name та повертає булеве значення, що вказує на наявність або відсутність дубліката. Якщо запис із такою назвою вже існує, адміністратор отримує повідомлення про помилку, а операція переривається. Даний механізм забезпечує цілісність каталогу та запобігає дублюванню інформації.

```
public int AddManga(
    int publisherId,
    string name,
    string description,
    string? cover,
    int statusId,
    int typeId,
    int releaseYear
)
{
    using var con = new SqlConnection(_connectionString);
    con.Open();

    var cmd = new SqlCommand(@"
        INSERT INTO Manga (ID_Publish, NameManga, Description, Cover, ID_Status, ID_Type, ReleaseDate)
        OUTPUT INSERTED.ID_Manga
        VALUES (@pid, @n, @d, @c, @s, @t, @y)", con);

    cmd.Parameters.AddWithValue("@pid", publisherId);
    cmd.Parameters.AddWithValue("@n", name);
    cmd.Parameters.AddWithValue("@d", description ?? (object)DBNull.Value);
    cmd.Parameters.AddWithValue("@c", cover ?? (object)DBNull.Value);
    cmd.Parameters.AddWithValue("@s", statusId);
    cmd.Parameters.AddWithValue("@t", typeId);
    cmd.Parameters.AddWithValue("@y", releaseYear);

    return (int)cmd.ExecuteScalar();
}
```

Рисунок 3.10 – Приклад реалізації методу для додавання нової манґи у реляційну таблицю БД

```
public bool ExistsByName(string name)
{
    using var con = new SqlConnection(_connectionString);
    con.Open();

    using var cmd = new SqlCommand(
        "SELECT COUNT(*) FROM Manga WHERE NameManga = @name", con);

    cmd.Parameters.AddWithValue("@name", name);

    return (int)cmd.ExecuteScalar() > 0;
}
```

Рисунок 3.11 – Приклад реалізації методу для перевірки унікальності назви манґи

Додатково система обробляє дані про видавця, авторів, художників та жанри. Якщо видавця ще не існує в базі, використовується метод GetOrCreatePublisher (рис. 3.12), який або повертає існуючий ідентифікатор, або створює новий запис у таблиці Publish. Аналогічним чином реалізовано додавання авторів та художників (рис. 3.13), де зв’язок між манґою та творцем формується через таблицю Authorship із записом відповідної ролі.

Паралельно вибрані жанри додаються у таблицю MangaGenres за допомогою .методу AddGenres(рис. 3.14).

```
public int GetOrCreatePublisher(string name)
{
    using var con = new SqlConnection(_connectionString);
    con.Open();

    using (var check = new SqlCommand(
        "SELECT ID_Publish FROM Publish WHERE NamePublish = @n", con))
    {
        check.Parameters.AddWithValue("@n", name);
        var result = check.ExecuteScalar();

        if (result != null)
            return (int)result;
    }

    using (var insert = new SqlCommand(
        "INSERT INTO Publish(NamePublish) OUTPUT INSERTED.ID_Publish VALUES(@n)", con))
    {
        insert.Parameters.AddWithValue("@n", name);
        return (int)insert.ExecuteScalar();
    }
}
```

Рисунок 3.12 – Метод для обробки даних про видавця

```
public int GetOrCreateAuthor(string name)
{
    using var con = new SqlConnection(_connectionString);
    con.Open();

    using (var check = new SqlCommand(
        "SELECT ID_Author FROM Author WHERE NameAuthor = @n", con))
    {
        check.Parameters.AddWithValue("@n", name);
        var res = check.ExecuteScalar();
        if (res != null)
            return (int)res;
    }

    using (var insert = new SqlCommand(
        "INSERT INTO Author(NameAuthor) OUTPUT INSERTED.ID_Author VALUES(@n)", con))
    {
        insert.Parameters.AddWithValue("@n", name);
        return (int)insert.ExecuteScalar();
    }
}
```

Рисунок 3.13 – Метод для обробки даних про авторів та художників

```
public void AddGenres(int mangaId, List<int> genreIds)
{
    using var con = new SqlConnection(_connectionString);
    con.Open();

    foreach (var genreId in genreIds)
    {
        using var cmd = new SqlCommand(
            "INSERT INTO MangaGenres (ID_Manga, ID_Genre) VALUES (@m, @g)", con);
        cmd.Parameters.AddWithValue("@m", mangaId);
        cmd.Parameters.AddWithValue("@g", genreId);
        cmd.ExecuteNonQuery();
    }
}
```

Рисунок 3.13 – Метод для обробки даних про жанри

Редагування існуючого запису відбувається за схожим механізмом. Після вибору манґи адміністратор переходить на сторінку редагування, де модель автоматично заповнюється актуальними даними, отриманими через метод GetById. Цей метод виконує SQL-запит із вибіркою повної інформації про конкретний об’єкт, включаючи жанри, авторів та інші пов’язані дані.

Після внесення змін адміністратор підтверджує оновлення, і контролер передає дані в метод Update (рис. 3.15), де за допомогою SQL-операції UPDATE змінюються значення відповідних полів таблиці Manga. У разі зміни жанрів додатково виконується очистка попередніх зв’язків і повторне створення нових у таблиці MangaGenres за допомогою методу UpdateGenres (рис. 3.16). Якщо адміністратор завантажує нову обкладинку, система оновлює шлях до неї в базі, а старий файл може бути замінений або видалений.

```
public void Update(int id, string name, string desc, int year,
           int statusId, int typeId, string? cover)
{
    using var con = new SqlConnection(_connectionString);
    con.Open();

    var sql = @"
UPDATE Manga SET
    NameManga = @name,
    Description = @desc,
    ReleaseDate = @year,
    ID_Status = @status,
    ID_Type = @type,
    Cover = @cover
WHERE ID_Manga = @id";

    using var cmd = new SqlCommand(sql, con);

    cmd.Parameters.AddWithValue("@id", id);
    cmd.Parameters.AddWithValue("@name", name);
    cmd.Parameters.AddWithValue("@desc", desc);
    cmd.Parameters.AddWithValue("@year", year);
    cmd.Parameters.AddWithValue("@status", statusId);
    cmd.Parameters.AddWithValue("@type", typeId);
    cmd.Parameters.AddWithValue("@cover", (object?)cover ?? DBNull.Value);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.15 – Приклад реалізації методу для оновлення даних про мангу

```
public void UpdateGenres(int mangaId, List<int> genreIds)
{
    using var con = new SqlConnection(_connectionString);
    con.Open();

    var del = new SqlCommand(
        "DELETE FROM MangaGenres WHERE ID_Manga = @m", con);
    del.Parameters.AddWithValue("@m", mangaId);
    del.ExecuteNonQuery();

    foreach (var gid in genreIds)
    {
        var ins = new SqlCommand(
            "INSERT INTO MangaGenres (ID_Manga, ID_Genre) VALUES (@m,@g)", con);
        ins.Parameters.AddWithValue("@m", mangaId);
        ins.Parameters.AddWithValue("@g", gid);
        ins.ExecuteNonQuery();
    }
}
```

Рисунок 3.16 – Приклад реалізації методу для оновлення даних про жанри

Операція видалення реалізована з урахуванням реляційних

залежностей. До того як запис манґи буде остаточно видалений із таблиці Manga, система здійснює каскадне очищення пов’язаних таблиць: глав, сторінок, жанрів, авторства, коментарів та записів читання, за допомогою метода DeleteManga. Всі дії виконуються в межах транзакції, що запобігає появі «осиротілих» записів у базі даних у разі помилки виконання. Після завершення операції додатково видаляється папка з пов’язаними медіафайлами у файловій системі.

```
[HttpPost]
[ValidateAntiForgeryToken]
Ссылок: 0
public IActionResult DeleteManga(int id)
{
    using var con = _db.GetConnection();
    con.Open();

    using var tr = con.BeginTransaction();

    try
    {
        ExecNonQuery(con, tr, @"
        DELETE FROM ChapterPage
        WHERE ID_Manga = @id
    ", id);

        ExecNonQuery(con, tr, @"
        DELETE FROM Chapter
        WHERE ID_Manga = @id
    ", id);

        ExecNonQuery(con, tr, @"
        DELETE FROM ReadingProcess
        WHERE ID_Manga = @id
    ", id);

        ExecNonQuery(con, tr, @"
        DELETE FROM Comment
        WHERE ID_Manga = @id
    ", id);

        ExecNonQuery(con, tr, @"
        DELETE FROM MangaGenres
        WHERE ID_Manga = @id
    ", id);

        ExecNonQuery(con, tr, @"
        DELETE FROM Authorship
        WHERE ID_Manga = @id
    ", id);

        ExecNonQuery(con, tr, @"
        DELETE FROM Manga
        WHERE ID_Manga = @id
    ", id);

        tr.Commit();
    }
    catch
    {
        tr.Rollback();
        throw;
    }


    var manga = _mangaRepository.GetById(id);
    if (manga != null)
    {
        var path = Path.Combine(
            Directory.GetCurrentDirectory(),
            "wwwroot",
            "uploads",
            "manga",
            manga.Name
        );

        if (Directory.Exists(path))
            Directory.Delete(path, true);
    }

    return RedirectToAction("Panel");
}
```

Рисунок 3.12 – Приклад реалізації методу для видалення манґи з реляційної таблиці БД

Операція пошуку використовується для вибору даних із таблиць, що задовольняють певні критерії. Реалізовано методом Search з SQL-запитом SELECT з оператором LIKE для пошуку за шаблоном (наприклад, за назвою манги). Користувачу достатньо ввести перші літери, щоб знайти відповідні записи (рис. 3.13).

```
public List<Manga> Search(string query)
{
    var list = new List<Manga>();

    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = @"
        SELECT
            m.ID_Manga,
            m.NameManga,
            m.Cover,
            m.ReleaseDate AS ReleaseYear
        FROM Manga m
        WHERE LOWER(m.NameManga) LIKE LOWER(@q)
        ORDER BY m.NameManga
    ";

    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@q", "%" + query.Trim() + "%");

    using var reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        list.Add(new Manga
        {
            Id = reader.GetInt32(0),
            Name = reader.GetString(1),
            Cover = reader["Cover"] as string,
            ReleaseYear = reader["ReleaseYear"] == DBNull.Value
                ? null
                : Convert.ToInt16(reader["ReleaseYear"])
        });
    }

    return list;
}
```

Рисунок 3.13 – Приклад реалізації методу для пошуку даних

3.2.5 Додавання глав до манги

Функціонал додавання глав до манги є ключовим для адміністрування каталогу в веб-додатку «Mankura». На сторінці адмін-панелі (Admin Dashboard) адміністратор бачить загальні статистики (кількість користувачів, манги, коментарів) та кнопки швидкого доступу: "Add manga", "Edit manga", "Add chapter", "Manage users", "Delete manga". Для додавання глави адміністратор обирає мангу з випадаючого списку, вводить номер глави (з автоматичною пропозицією наступного), назву, дату випуску та, за потребою, додає сторінки. Процес включає валідацію форми на стороні клієнта (JavaScript) та сервера (ModelState.IsValid в контролері), збереження в БД через репозиторій та обробку помилок. Це забезпечує послідовність контенту та уникнення помилок, таких як невалідні номери глав. Дані для випадаючих списків (select в Razor Views) завантажуються динамічно з бази даних через MangaRepository, наприклад, методом GetAll() для отримання списку назв манги з сортуванням. Приклад реалізації завантаження даних для списку та обробки форми наведено на рисунку 3.14.

```
[HttpGet]
Ссылок: 0
public IActionResult AddChapter()
{
    var mangas = _mangaRepository.GetAll()
        .Select(m => new SelectListItem
        {
            Value = m.Id.ToString(),
            Text = m.Name
        })
        .ToList();

    return View(new AddChapterViewModel
    {
        Mangas = mangas
    });
}

[HttpPost]
[ValidateAntiForgeryToken]
Ссылок: 0
public IActionResult AddChapter(AddChapterViewModel model)
{
    if (!ModelState.IsValid || model.Pages.Count == 0)
        return View(model);

    var manga = _mangaRepository.GetById(model.MangaId);
    if (manga == null)
        return NotFound();

    using var con = _db.GetConnection();
    con.Open();

    using (var check = new SqlCommand(@"
    SELECT ID_Chapter
    FROM Chapter
    WHERE ID_Manga = @manga AND ChapterNumber = @num
", con))
    {
        check.Parameters.AddWithValue("@manga", model.MangaId);
        check.Parameters.AddWithValue("@num", model.ChapterNumber);

        var exists = check.ExecuteScalar();
        if (exists != null)
        {
            ModelState.AddModelError("", "This chapter already exists.");

            model.Mangas = _mangaRepository.GetAll()
                .Select(m => new SelectListItem
                {
                    Value = m.Id.ToString(),
                    Text = m.Name
                })
                .ToList();

            return View(model);
        }
    }
```

Рисунок 3.14 – Метод AddChapter, де відбувається виклик GetAll() з MangaRepository для заповнення ViewBag.MangaList випадаючого списку

За допомогою методу GetById(int id) в MangaRepository програма отримує повну інформацію про мангу за ID, включаючи перевірку існування, для забезпечення цілісності даних при додаванні глави (наприклад, щоб пов'язати главу з правильним ID_Manga). Це дозволяє уникнути помилок при збереженні та зберігає реляційні зв'язки в БД.

Одним із ключових аспектів реалізації додавання глав є автоматичне визначення номеру нової глави та перевірка на наявність можливих логічних помилок. Система повинна гарантувати, що не виникне дублікатів номерів глав для однієї манги і що глави нумеруються послідовно. Для цього реалізовано метод GetNextChapter(int chapter Id) (рис. 3.15), який визначає наступний порядковий номер глави для конкретної манги. Цей метод виконує SQL-запит до таблиці Chapter, що обчислює максимальний номер глави для манги з заданим MangaId і повертає значення на 1 більше за знайдений максимум. Якщо для даної манги ще немає жодної глави, метод поверне значення 1. Таким чином, при виборі манги випадаючим списком, програма може автоматично запропонувати номер чергової глави (наприклад, якщо в БД найбільший номер – 5, то наступний буде 6).

```
public int? GetNextChapterId(int chapterId)
{
    using var conn = new SqlConnection(_cs);
    conn.Open();

    var sql = @"
        SELECT TOP 1 c2.ID_Chapter
        FROM Chapter c1
        JOIN Chapter c2
            ON c2.ID_Manga = c1.ID_Manga
        WHERE c1.ID_Chapter = @chapterId
          AND c2.ChapterNumber > c1.ChapterNumber
        ORDER BY c2.ChapterNumber ASC
    ";

    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@chapterId", chapterId);

    var res = cmd.ExecuteScalar();
    return res == null ? null : (int)res;
}
```

Рисунок 3.15 – Метод GetNextChapter для автоматичного визначення номеру нової глави

Особливу увагу при розробці було приділено механізму завантаження сторінок глави (рис. 3.16). Сторінки представляють собою окремі зображення, і кожне з них має свій порядковий номер, який визначає послідовність читання. Після вибору файлів із локального комп'ютера адміністратора система обробляє їх по одному, аналізуючи ім'я файлу для визначення номера сторінки. Номер сторінки отримується за допомогою спеціального методу GetPageNumberFromFileName(string fileName), який виділяє числовий індекс із назви файлу (наприклад, сторінка (5) стає сторінкою №6). Це дозволяє забезпечити коректну нумерацію навіть у випадках, коли файли додаються не в хронологічному порядку

```
int chapterId;
using (var cmd = new SqlCommand(@"
    INSERT INTO Chapter (ID_Manga, ChapterNumber, VolumeNumber)
    OUTPUT INSERTED.ID_Chapter
    VALUES (@manga, @chapter, @volume)
", con))
{
    cmd.Parameters.AddWithValue("@manga", model.MangaId);
    cmd.Parameters.AddWithValue("@chapter", model.ChapterNumber);
    cmd.Parameters.AddWithValue("@volume", (object?)model.VolumeNumber ?? DBNull.Value);

    chapterId = (int)cmd.ExecuteScalar();
}

int page = 1;

foreach (var file in model.Pages
.OrderBy(f => GetPageNumberFromFileName(f.FileName)))
{
    int pageNumber = GetPageNumberFromFileName(file.FileName);

    string fileName = $"{pageNumber:D3}{Path.GetExtension(file.FileName)}";
    string fullPath = Path.Combine(
        Directory.GetCurrentDirectory(),
        "wwwroot",
        "uploads",
        "manga",
        manga.Name,
        model.ChapterNumber.ToString(),
        fileName
    );

    Directory.CreateDirectory(Path.GetDirectoryName(fullPath)!);

    using var stream = new FileStream(fullPath, FileMode.Create);
    file.CopyTo(stream);

    string imageUrl = $"/uploads/manga/{manga.Name}/{model.ChapterNumber}/{fileName}";

    using var pageCmd = new SqlCommand(@"
        INSERT INTO ChapterPage
        (ID_Chapter, ID_Manga, PageNumber, ImageURL)
        VALUES (@ch, @m, @p, @url)
    ", con);

    pageCmd.Parameters.AddWithValue("@ch", chapterId);
    pageCmd.Parameters.AddWithValue("@m", model.MangaId);
    pageCmd.Parameters.AddWithValue("@p", pageNumber);
    pageCmd.Parameters.AddWithValue("@url", imageUrl);

    pageCmd.ExecuteNonQuery();
}
```

Рисунок 3.15 – Додавання сторінок

При відображенні глави в читачі сторінки завантажуються методом GetPages(int chapterId) (рис. 3.16) якщо сторінок немає — просто повертається порожній список, що дозволяє коректно обробляти ситуації, коли глава існує, але сторінки ще не були додані.

```
public List<ChapterPage> GetPages(int chapterId)
{
    var list = new List<ChapterPage>();

    using var conn = new SqlConnection(_cs);
    conn.Open();

    var sql = @"
        SELECT ID_Page, ID_Chapter, PageNumber, ImageURL
        FROM ChapterPage
        WHERE ID_Chapter = @chapterId
        ORDER BY PageNumber
    ";

    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@chapterId", chapterId);

    using var r = cmd.ExecuteReader();
    while (r.Read())
    {
        list.Add(new ChapterPage
        {
            Id = r.GetInt32(0),
            ChapterId = r.GetInt32(1),
            PageNumber = r.GetInt32(2),
            ImageURL = r.GetString(3)
        });
    }

    return list;
}
```

Рисунок 3.16 – Метод GetPages

Функціонал видалення глави реалізовано на сторінці манґи в списку глав. Кнопка видалення відображається тільки для користувачів з роллю Admin. Після підтвердження у модальному вікні запит надсилається через JavaScript на метод Delete(int chapterId, int mangaId) (рис. 3.17) у класі ChapterRepository.

```
public void Delete(int chapterId, int mangaId)
{
    using var con = new SqlConnection(_cs);
    con.Open();

    var cmd = new SqlCommand(@"
        DELETE FROM ReadingProcess WHERE ID_Chapter = @c;
        DELETE FROM ChapterPage   WHERE ID_Chapter = @c;
        DELETE FROM Chapter       WHERE ID_Chapter = @c AND ID_Manga = @m;
    ", con);

    cmd.Parameters.AddWithValue("@c", chapterId);
    cmd.Parameters.AddWithValue("@m", mangaId);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.17 – Метод Delete

3.2.6 Авторизація користувачів

Підсистема реєстрації та авторизації у веб-застосунку «Mankura» побудована на основі репозиторію UserRepository, який забезпечує безпосередню взаємодію з таблицею користувачів у базі даних та виконує основні операції додавання, пошуку та перевірки облікових записів. Усі операції доступу до даних реалізовані через параметризовані SQL-запити, що гарантує безпеку та цілісність інформації під час обробки облікових записів.

Процес реєстрації нового користувача починається з перевірки унікальності введених даних. Для цього використовуються методи GetByEmail (string email)(рис. 3.18) та GetByUserName (stringuser Name)(рис. 3. 19). Ці запити дозволяють визначити, чи існує в системі користувач з такою ж електронною адресою або іменем, що унеможливлює дублювання облікових записів. У випадку виявлення збігу система повертає повідомлення про помилку, а створення облікового запису не відбувається.

```
public User? GetByEmail(string email)
{
    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = @"
        SELECT
            ID_User,
            Email,
            UserName,
            Password,
            RegistrationDate,
            ID_Role,
            Avatar
        FROM [User]
        WHERE Email = @email
    ";

    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@email", email);

    using var r = cmd.ExecuteReader();
    if (!r.Read()) return null;

    return new User
    {
        Id = (int)r["ID_User"],
        Email = (string)r["Email"],
        UserName = (string)r["UserName"],
        PasswordHash = (string)r["Password"],
        RegistrationDate = (DateTime)r["RegistrationDate"],
        RoleId = (int)r["ID_Role"],
        Avatar = r["Avatar"] as string
    };
}
```

Рисунок 3.18 – Метод GetByEmail()

```
public User GetByUserName(string username)
{
    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = "SELECT * FROM [User] WHERE UserName = @username";
    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@username", username);

    using var r = cmd.ExecuteReader();
    if (!r.Read()) return null;

    return new User
    {
        Id = (int)r["ID_User"],
        Email = (string)r["Email"],
        UserName = (string)r["UserName"],
        PasswordHash = (string)r["Password"],
        RoleId = (int)r["ID_Role"],
        Avatar = r["Avatar"] as string
    };
}
```

Рисунок 3.19 – Метод GetByUserName()

Після успішної перевірки користувач створюється методом Create(User user). Перед збереженням пароль хешується за допомогою BCrypt, а в базу даних заносяться email, ім’я користувача, дата реєстрації, роль та інші значення, пов’язані з обліковим записом. Це забезпечує безпечне зберігання облікових даних та готовність користувача до подальшої авторизації.

```
public void Create(User user)
{
    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = @"
       INSERT INTO [User]
        (Email, UserName, Password, RegistrationDate, ID_Role)
        VALUES
        (@email, @username, @password, @regDate, @roleId)
    ";
    using var cmd = new SqlCommand(sql, conn);

    cmd.Parameters.AddWithValue("@email", user.Email);
    cmd.Parameters.AddWithValue("@username", user.UserName);
    cmd.Parameters.AddWithValue("@password", user.PasswordHash);
    cmd.Parameters.AddWithValue("@regDate", user.RegistrationDate);
    cmd.Parameters.AddWithValue("@roleId", 2);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.20 – Метод Create()

Авторизація в системі реалізована на основі поєднання пошуку користувача в базі даних та перевірки введеного пароля. Для пошуку облікового запису використовується метод GetByEmailOrUserName(string login) (рис. 3.21), який дозволяє виконувати вхід як за електронною поштою, так і за ім’ям користувача.

```
public User GetByEmailOrUserName(string login)
{
    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = @"
        SELECT *
        FROM [User]
        WHERE Email = @login OR UserName = @login
    ";

    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@login", login);

    using var r = cmd.ExecuteReader();
    if (!r.Read()) return null;

    return new User
    {
        Id = (int)r["ID_User"],
        Email = (string)r["Email"],
        UserName = (string)r["UserName"],
        PasswordHash = (string)r["Password"],
        RoleId = (int)r["ID_Role"]
    };
}
```

Рисунок 3.21 – Метод GetByEmailOrUserName

Після отримання запису з бази системою виконується звірка пароля із збереженим хешем. У випадку збігу формується набір claims, до якого входять ідентифікатор користувача, ім’я, електронна адреса та роль (Admin або User). Дані claims використовуються для керування рівнем доступу в системі та визначають, чи буде користувач перенаправлений до сторінки профілю, чи до адміністративної панелі.

Таким чином, процеси реєстрації та авторизації в системі побудовані на основі **прямих SQL-запитів у репозиторії**, чіткої перевірки унікальності облікових даних та безпечної обробки паролів. Це забезпечує надійну взаємодію з базою даних, захист користувачів та коректне розмежування доступу між звичайними користувачами та адміністраторами.

3.2.7 Керування обліковим записом користувача та профілем

Підсистема керування обліковим записом забезпечує можливість перегляду та редагування персональних даних користувача, включаючи ім’я та аватар профілю. Усі операції взаємодії з профілем реалізовано через репозиторій UserRepository, де виконуються параметризовані SQL-запити для читання та оновлення інформації про користувачів.

Такий підхід дозволяє точно ідентифікувати користувача за його унікальним ключем і завантажити актуальні дані для відображення на сторінці профілю. Результат запиту заповнює модель користувача, яка передається у Razor-представлення для відображення персональної інформації.

Окремо реалізовано функціонал оновлення аватара профілю. Завантажений файл зображення зберігається у файловому сховищі застосунку, після чого шлях до зображення записується у базу даних за допомогою того ж методу UpdateProfile()(рис. 3.22), де поле Avatar оновлюється шляхом виконання SQL-операції UPDATE. Завдяки цьому зміни моментально відображаються в інтерфейсі користувача, а шлях до зображення зберігається централізовано в БД.

Важливим аспектом реалізації є те, що всі операції керування профілем виконуються строго у межах облікового запису поточного користувача. Ідентифікатор користувача отримується з автентифікаційних claims, після чого використовується як параметр у запитах репозиторію. Це виключає можливість несанкціонованого редагування чужих профілів та гарантує відповідність даних авторизованому сеансу.

```
public void UpdateProfile(int userId, string userName, string? avatar)
{
    using var conn = new SqlConnection(_connectionString);
    conn.Open();

    var sql = @"
        UPDATE [User]
        SET UserName = @userName,
            Avatar = COALESCE(@avatar, Avatar)
        WHERE ID_User = @id
    ";

    using var cmd = new SqlCommand(sql, conn);
    cmd.Parameters.AddWithValue("@id", userId);
    cmd.Parameters.AddWithValue("@userName", userName);
    cmd.Parameters.AddWithValue("@avatar", (object?)avatar ?? DBNull.Value);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.22 – Метод UpdateProfile()

3.2.9 Робота зі списком читання та закладками користувача

Модуль керування закладками забезпечує можливість додавання творів до персонального списку читання користувача, зміни статусу читання (наприклад, *Will be read*, *Readable*, *Read*, *Favorites*), а також видалення закладок із каталогу. Усі операції роботи зі списком читання реалізовано в репозиторії ReadingProcessRepository.

Додавання манги до списку читання та оновлення статусу читання виконується методом AddOrUpdate (int userId, int mangaId, int chapterId, string type) (рис. 3.23). Перед виконанням вставки здійснюється додаткова перевірка на наявність запису в таблиці, щоб уникнути дублювання. Для цього використовується метод Exists(int userId, int mangaId) (рис. 3.24), якщо запис уже існує, замість створення нового здійснюється оновлення статусу.

```
public void AddOrUpdate(int userId, int mangaId, int chapterId, string type)
{
    using var con = new SqlConnection(_cs);
    con.Open();

    var cmd = new SqlCommand(@"
    IF EXISTS (
        SELECT 1 FROM ReadingProcess
        WHERE ID_User = @u AND ID_Manga = @m
    )
    BEGIN
        UPDATE ReadingProcess
        SET Type = @t,
            ID_Chapter = @ch,
            AddedDate = GETDATE()
        WHERE ID_User = @u AND ID_Manga = @m
    END
    ELSE
    BEGIN
        INSERT INTO ReadingProcess
        (ID_User, Type, AddedDate, ID_Chapter, ID_Manga)
        VALUES (@u, @t, GETDATE(), @ch, @m)
    END
    ", con);

    cmd.Parameters.AddWithValue("@u", userId);
    cmd.Parameters.AddWithValue("@m", mangaId);
    cmd.Parameters.AddWithValue("@ch", chapterId);
    cmd.Parameters.AddWithValue("@t", type);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.23 – Метод AddOrUpdate ()

```
public bool Exists(int userId, int mangaId, string type)
{
    using var con = new SqlConnection(_cs);
    con.Open();

    var cmd = new SqlCommand(@"
        SELECT 1
        FROM ReadingProcess
        WHERE ID_User = @u AND ID_Manga = @m
    ", con);

    cmd.Parameters.AddWithValue("@u", userId);
    cmd.Parameters.AddWithValue("@m", mangaId);

    return cmd.ExecuteScalar() != null;
}
```

Рисунок 3.24 – Метод Exists()

Видалення твору зі списку читання реалізовано методом

Remove(int userId, int mangaId) (рис. 3.25). Операція видалення також є параметризованою, що забезпечує захист від SQL-ін’єкцій і зберігає цілісність даних. Видалення не впливає на саму мангу в каталозі, а лише прибирає її з персонального списку поточного користувача.

```
public void Remove(int userId, int mangaId, string type)
{
    using var con = new SqlConnection(_cs);
    con.Open();

    using var cmd = new SqlCommand(@"
        DELETE FROM ReadingProcess
        WHERE ID_User = @u AND ID_Manga = @m AND Type = @t
    ", con);

    cmd.Parameters.AddWithValue("@u", userId);
    cmd.Parameters.AddWithValue("@m", mangaId);
    cmd.Parameters.AddWithValue("@t", type);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.25 – Метод Remove()

3.2.10 Робота з коментарями користувачів у системі

Функціонал коментування у веб-застосунку «Mankura» забезпечує можливість користувачам залишати відгуки до манґи, переглядати вже існуючі коментарі, а також видаляти власні записи. Усі операції взаємодії з таблицею Comment реалізовано в репозиторії CommentRepository, де виконуються параметризовані SQL-запити для створення, отримання та видалення коментарів.

Додавання нового коментаря виконується методом Add(Comment comment)(рис. 3.26).

Перед виконанням операції контролер перевіряє авторизацію користувача та коректність введених даних, після чого запис передається до репозиторію. Такий підхід гарантує, що кожен коментар пов’язаний з конкретним профілем користувача та конкретним твором у каталозі.

```
public void Add(int mangaId, int userId, string text)
{
    using var con = new SqlConnection(_cs);
    con.Open();

    var sql = @"
        INSERT INTO Comment (CommentText, ID_Manga, ID_User, CommentDate)
        VALUES (@text, @mangaId, @userId, GETDATE());
    ";

    using var cmd = new SqlCommand(sql, con);
    cmd.Parameters.AddWithValue("@text", text);
    cmd.Parameters.AddWithValue("@mangaId", mangaId);
    cmd.Parameters.AddWithValue("@userId", userId);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.25 – Метод Add() для додавання коментарів

Для відображення коментарів на сторінці манґи використовується метод GetByManga(int mangaId) (рис. 3.26).

Отримані записи додатково можуть об’єднуватися з таблицею користувачів для виведення імені, аватара та дати публікації. Параметризований підхід забезпечує безпечну фільтрацію даних та дозволяє відображати лише коментарі, що належать поточній манзі.

```
public List<Comment> GetByManga(int mangaId)
{
    var list = new List<Comment>();

    using var con = new SqlConnection(_cs);
    con.Open();

    var sql = @"
    SELECT
        c.ID_Comment,
        c.ID_User,
        c.CommentText,
        c.CommentDate,
        u.UserName,
        u.Avatar
    FROM Comment c
    JOIN [User] u ON u.ID_User = c.ID_User
    WHERE c.ID_Manga = @mangaId
    ORDER BY c.CommentDate DESC;
    ";

    using var cmd = new SqlCommand(sql, con);
    cmd.Parameters.AddWithValue("@mangaId", mangaId);

    using var r = cmd.ExecuteReader();
    while (r.Read())
    {
        list.Add(new Comment
        {
            Id = (int)r["ID_Comment"],
            UserId = (int)r["ID_User"],
            Text = r["CommentText"]?.ToString() ?? "",
            CommentDate = (DateTime)r["CommentDate"],
            UserName = r["UserName"]?.ToString() ?? "",
            Avatar = r["Avatar"]?.ToString() ?? "",
        });
    }

    return list;
}
```

Рисунок 3.25 – Метод GetByManga () для відображення коментарів на сторінці манґи

Видалення коментаря реалізовано методом Delete(int id, int userId). Таким чином, користувач може видаляти лише власні коментарі.

```
public void Delete(int id, int userId)
{
    using var con = new SqlConnection(_cs);
    con.Open();

    using var cmd = new SqlCommand(@"
        DELETE FROM Comment
        WHERE ID_Comment = @id AND ID_User = @user
    ", con);

    cmd.Parameters.AddWithValue("@id", id);
    cmd.Parameters.AddWithValue("@user", userId);

    cmd.ExecuteNonQuery();
}
```

Рисунок 3.26 – Метод Delete()

3.3 Ілюстрація роботи ІС

Для демонстрації роботи програми увійдемо під логіном та паролем адміністратора (рис. 3.27).

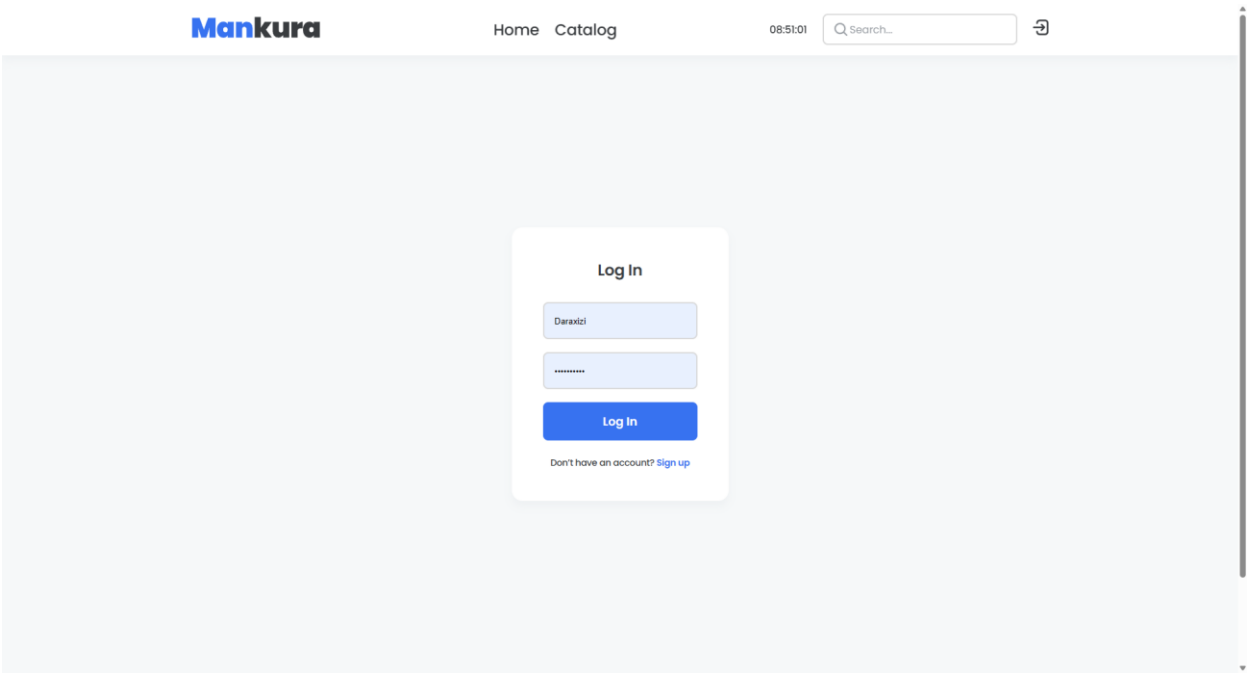

Рисунок 3.27 – Інтерфейс авторизації

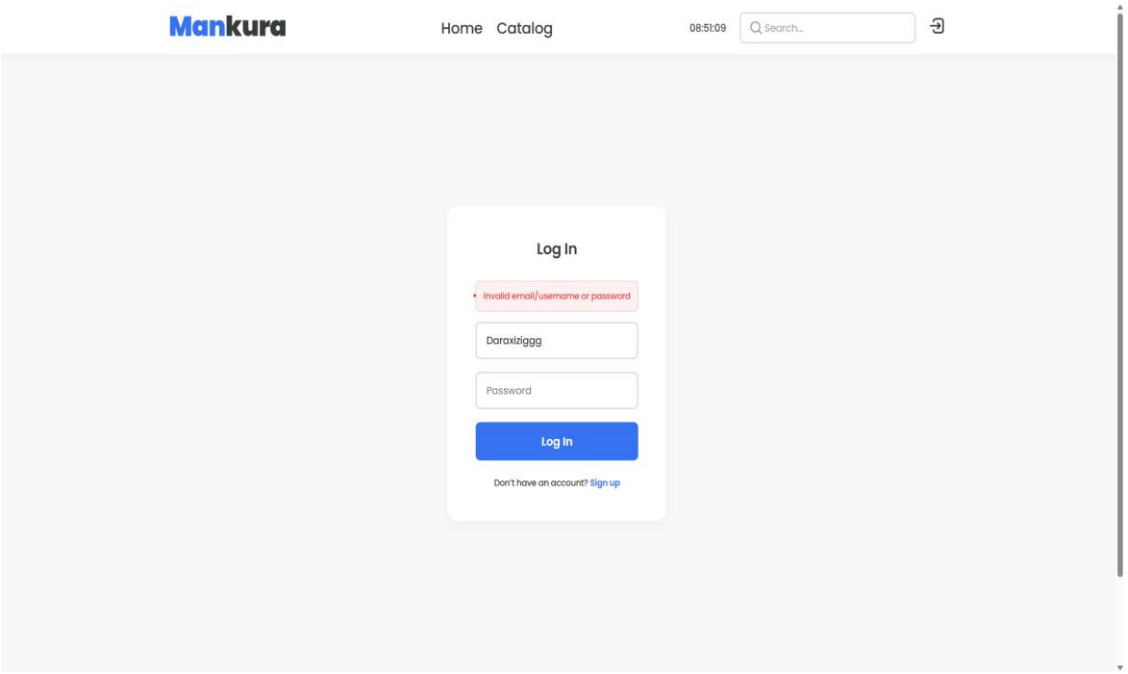

Рисунок 3.28 – Повідомлення щодо неправильного логіну або паролю

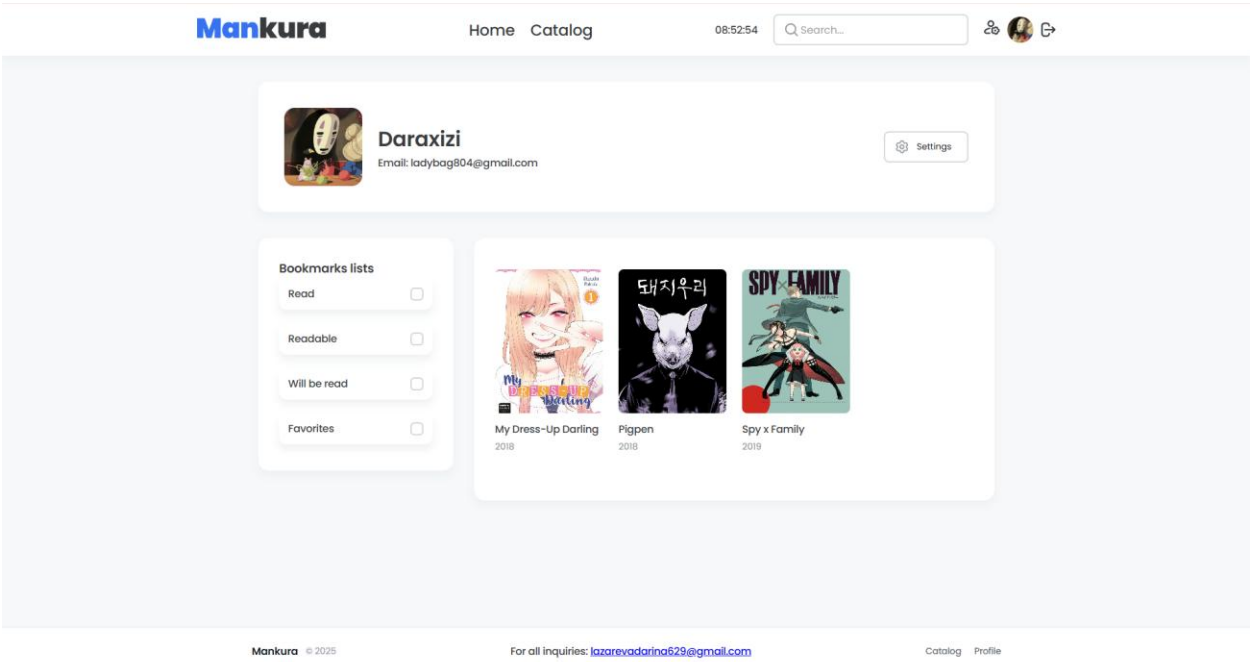

Рисунок 3.29 – Після успішної авторизації переходимо у профіль користувача

Після натискання кнопки Admin Panel адміністратор матиме змогу перейти на панель для управління даними. (рис. 3.30).

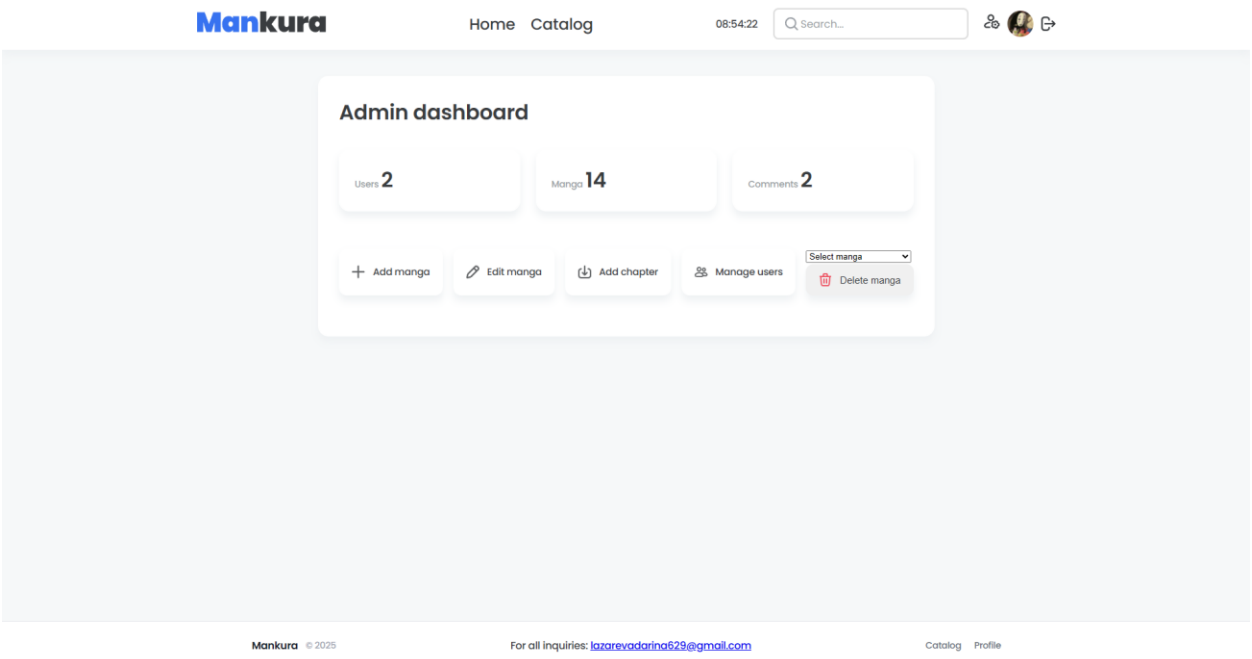

Рисунок 3.30 – Панель управління даними

Після натискання кнопки Add manga ми маємо змогу перейти до форми додавання манґи(рис. 3.31).

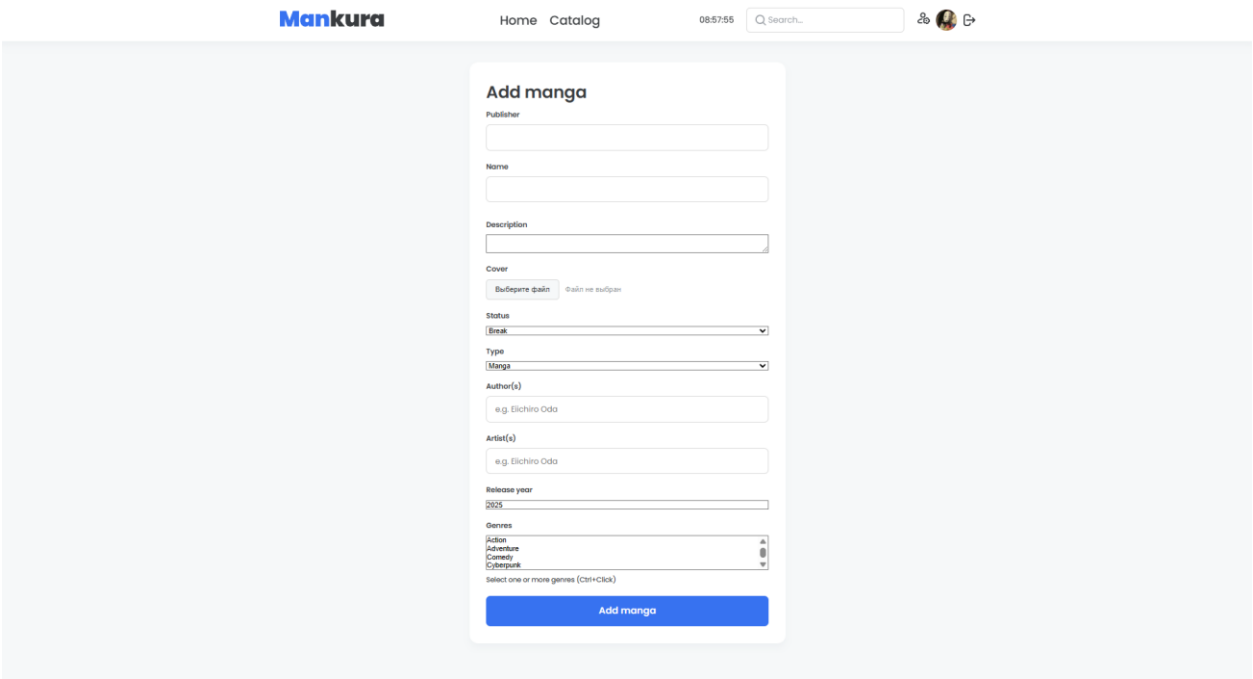

Рисунок 3.31 – Форма для додавання манги

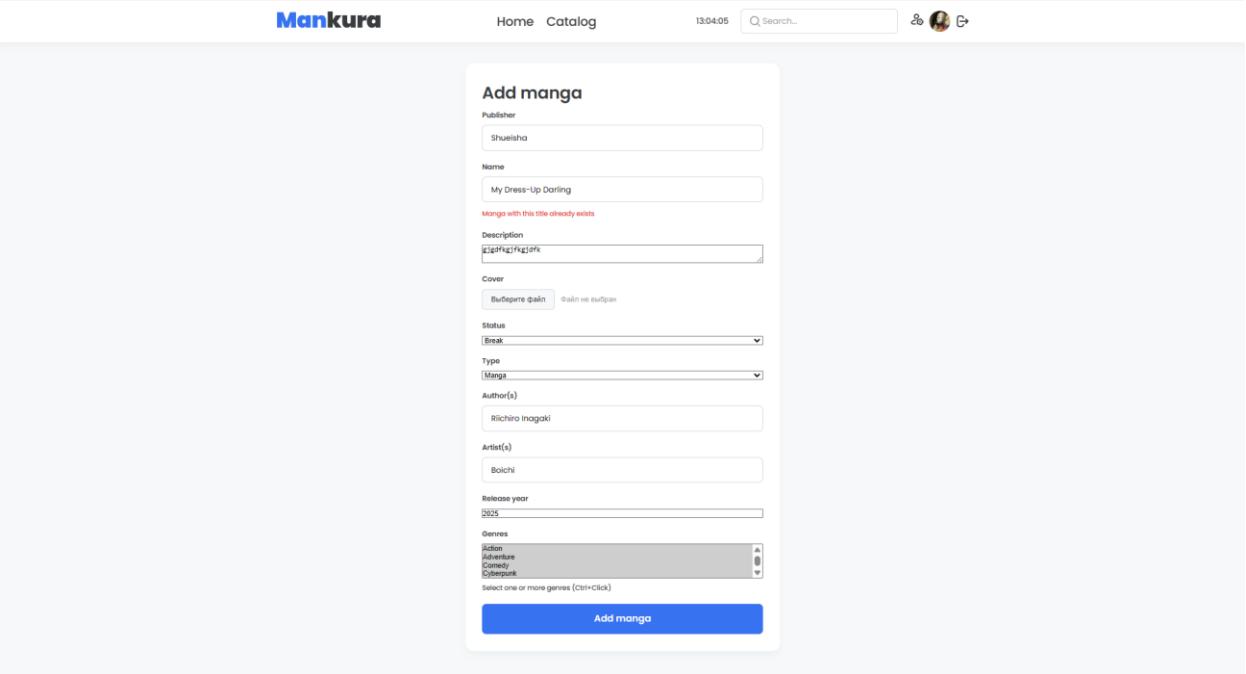

Рисунок 3.32 – Помилка при додаванні манґи с вже існуючою назвою

Після натискання кнопки Edit manga ми маємо змогу перейти до форми зміни даних. (рис. 3.33).

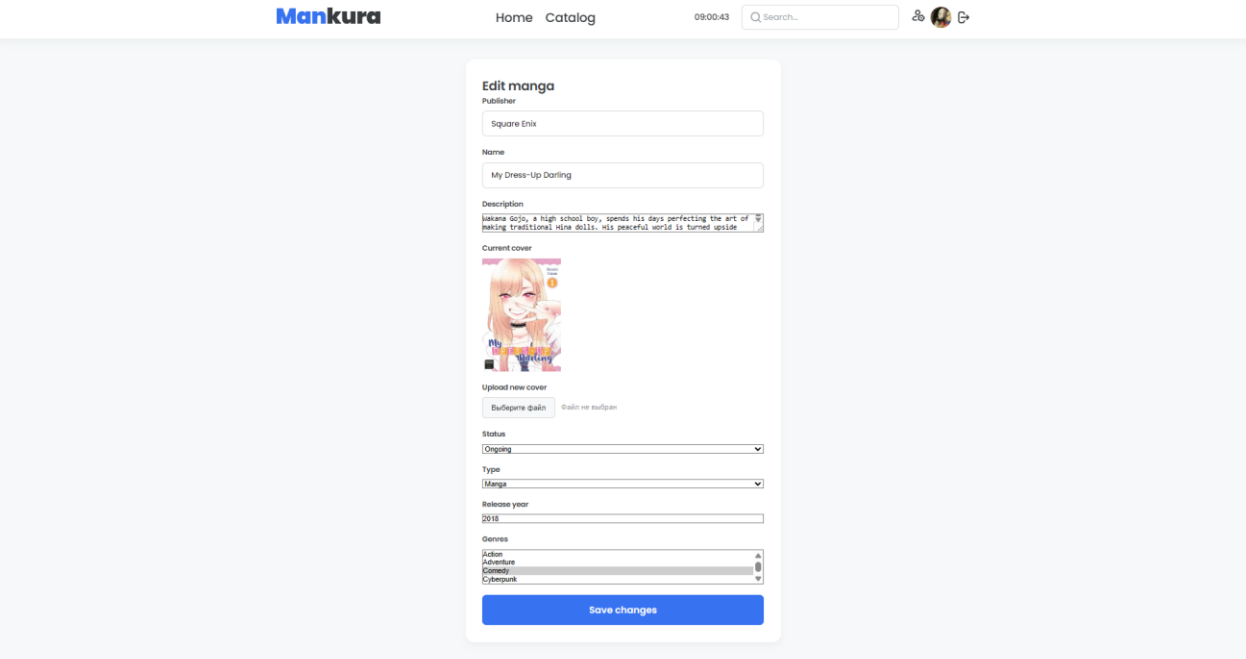

Рисунок 3.33 – Форма зміни даних

Після натискання кнопки Add chapter ми маємо змогу перейти до форми додавання глави. (рис. 3.34).

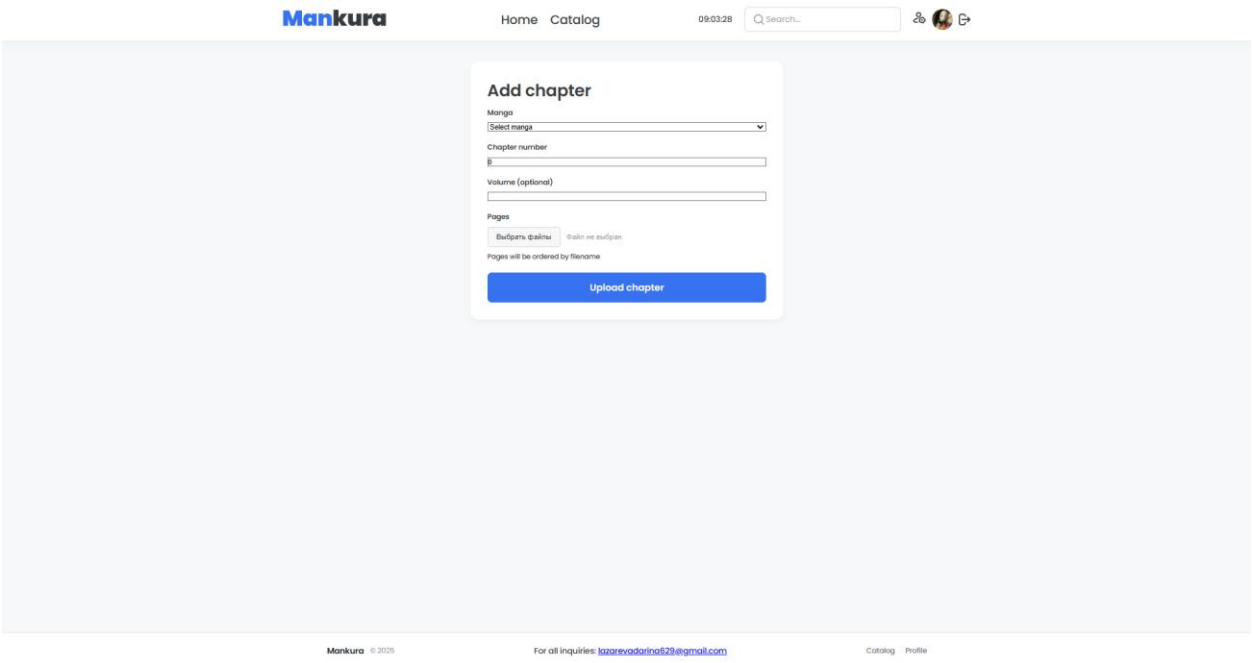

Рисунок 3.34 – Форма додавання глави

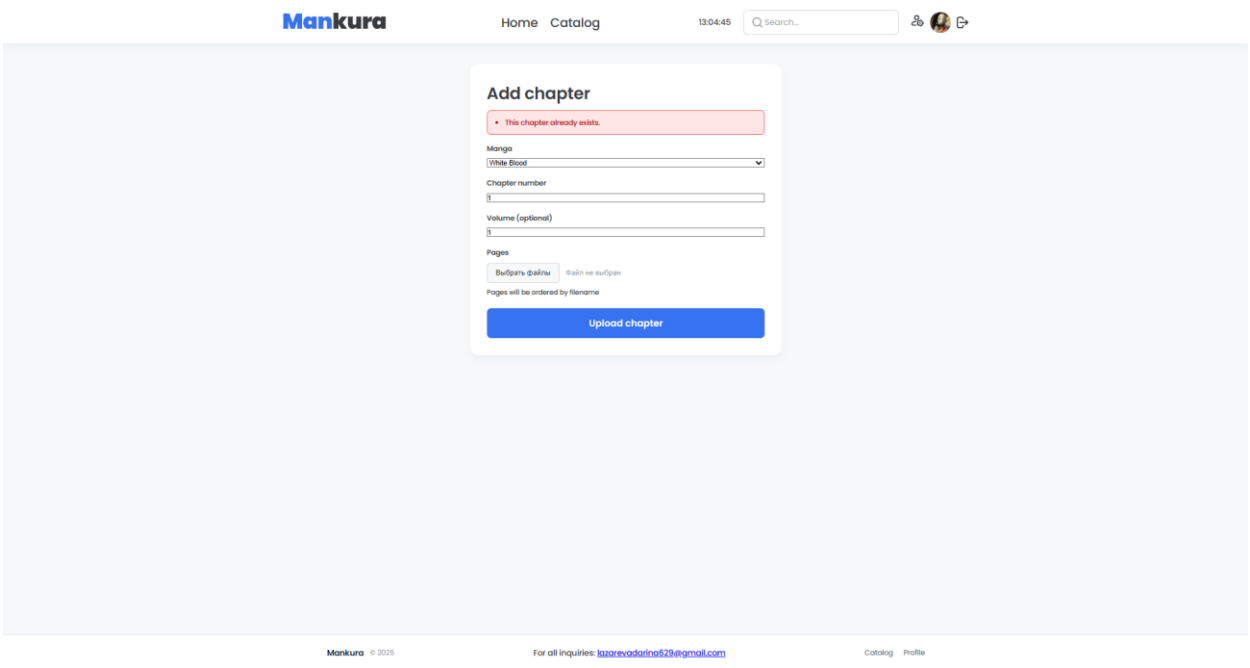

Рисунок 3.35 – Помилка при додаванні глави, яка вже існує

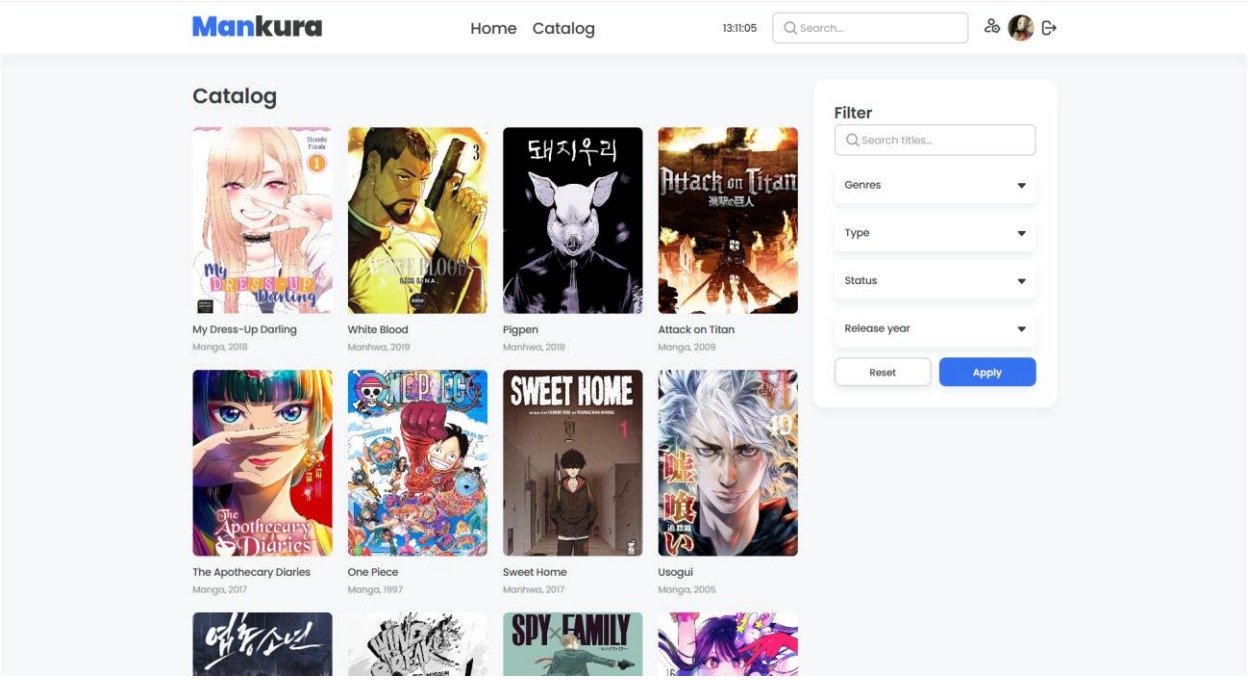

Рисунок 3.36 – Сторінка каталогу

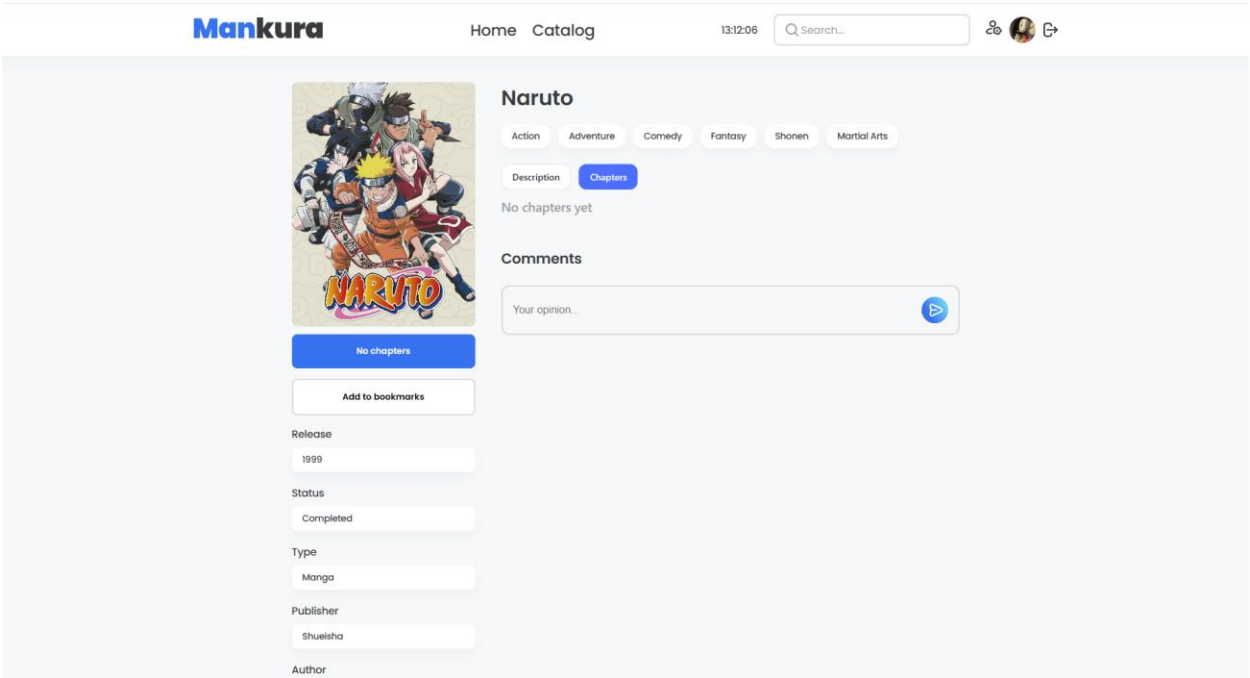

Рисунок 3.37 – Сторінка з відомостями про манґу (без глав)

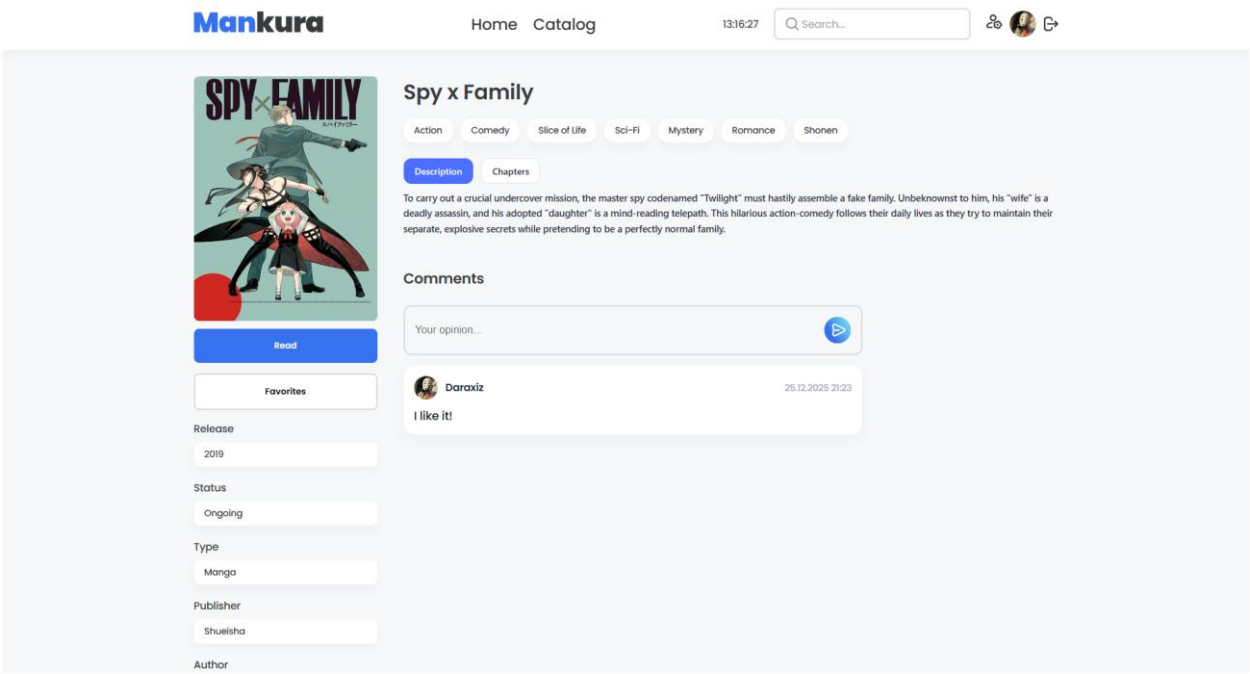

Рисунок 3.38 – Сторінка з відомостями про манґу (з главами)

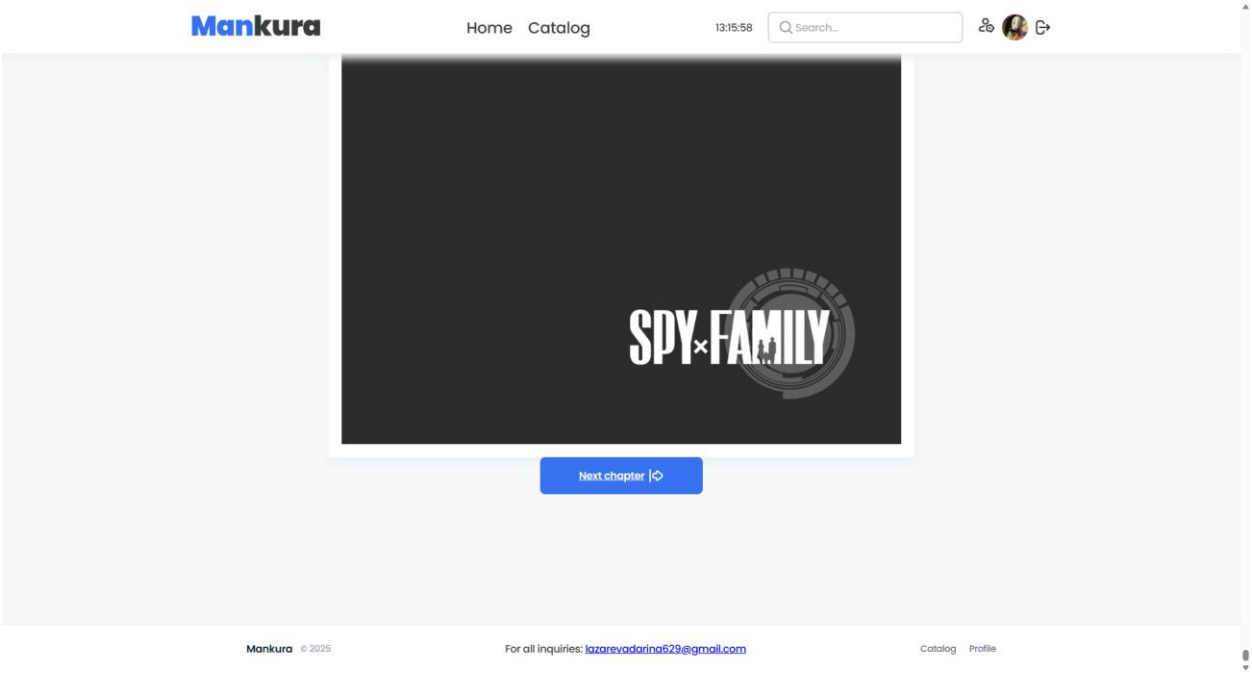

Рисунок 3.38 – Сторінка читання глави

ВИСНОВКИ

В результаті виконання курсової роботи було розроблено модель реляційної бази даних для предметної області «Система онлайн-каталогу манґи», створено її логічну та фізичну структуру, а також реалізовано програмне забезпечення для взаємодії з розробленою базою даних.

Під час виконання курсової роботи були вирішені такі завдання:

– вивчено основи реляційної моделі даних та принципи організації реляційних БД;

– досліджено особливості побудови структур даних для веб-систем контент-менеджменту;

– проаналізовано переваги застосування реляційних зв’язків при зберіганні об’єктів предметної області «манґа – глава – сторінка – користувач»;

– розглянуто принципи нормалізації даних та застосовано їх при проектуванні структури БД;

– спроєктовано ER-модель бази даних та визначено ключові сутності та зв’язки між ними;

– реалізовано фізичну модель бази даних у середовищі Microsoft SQL Server;

– опрацьовано механізми взаємодії Web-застосунку з БД за допомогою ADO.NET та SQL-запитів;

– розроблено програмне забезпечення, яке забезпечує роботу з каталогом манґи, главами, сторінками, користувачами, коментарями та списками читання;

– реалізовано функціонал авторизації користувачів, рольову модель (адміністратор / користувач) та елементи адміністрування контенту;

– проведено тестування роботи системи та перевірку коректності виконання CRUD-операцій;

– оформлено пояснювальну записку, яка відображає процес розробки бази

даних та програмного продукту.

Під час виконання курсової роботи було використано таке програмне забезпечення:

– Microsoft SQL Server — система керування базами даних;
– SQL Server Management Studio — середовище адміністрування та роботи з БД;
– Microsoft Visual Studio 2022 — середовище розробки програмного забезпечення;
– ASP.NET Core MVC — фреймворк для розробки веб-застосунку;
– C# — мова програмування для реалізації серверної логіки;
– HTML, CSS, JavaScript — технології клієнтського інтерфейсу.

Було спроєктовано логічну схему реляційної бази даних для предметної області «онлайн-каталог манґи», яка відображає взаємозв’язки між такими сутностями, як Manga, Chapter, ChapterPage, User, Comment, ReadingList, Genre та ін. Розроблена модель забезпечує структуроване зберігання інформації про контент, користувачів та історію взаємодії з системою.

Фізична модель бази даних побудована з урахуванням вимог до продуктивності, цілісності даних та можливості подальшого масштабування системи. Використання реляційних зв’язків забезпечило логічність структури, зручність виконання запитів та підтримку цілісності даних на рівні БД.

Розроблене програмне забезпечення «Система онлайн-каталогу манґи» реалізує повноцінну інформаційну систему для роботи з електронним каталогом манґи. Користувачі мають можливість переглядати каталог, читати глави, залишати коментарі та керувати списком читання. Адміністратор отримує інструменти для управління контентом, додавання та редагування манґи, глав і сторінок, а також модерації користувацької активності.

Таким чином, виконання курсової роботи дозволило отримати практичні навички проектування реляційних баз даних, реалізації веб-застосунків,

взаємодії з базою даних на рівні SQL-запитів та побудови інформаційних систем із реальним практичним застосуванням.

ПЕРЕЛІК ПОСИЛАНЬ

1. A brief history of databases: From relational, to NoSQL, to distributed SQL. *Cockroach Labs*. URL: https://www.cockroachlabs.com/blog/history-of-databases-distributed-sql/ (дата звернення: 20.12.2023).

2. What is a relational database?. Oracle | Cloud Applications and Cloud Platform. URL: https://www.oracle.com/database/what-is-a-relational-database/ (дата звернення: 20.12.2023).

3. Database normalization description - Microsoft 365 Apps. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description (дата звернення: 20.12.2023).

4. Simplify and optimize: how dapper, the micro orm, transforms data retrieval into a seamless and efficient process. *medium.com*. URL: https://medium.com/@nirajranasinghe/simplify-and-optimize-how-dapper-the-micro-orm-transforms-data-retrieval-into-a-seamless-and-d30b6f9799d1 (дата звернення: 20.12.2023).

5. Для чого потрібен SQL для тестування?. *Qalight*. URL: https://qalight.ua/baza-znaniy/dlya-chogo-potriben-sql-dlya-testuvannya/ (дата звернення: 20.12.2023).

6. SQL в Access: основні поняття, глосарій і синтаксис - Підтримка від Microsoft. *Microsoft Support*. URL: https://support.microsoft.com/uk-ua/topic/sql-в-access-основні-поняття-глосарій-і-синтаксис-444d0303-cde1-424e-9a74-e8dc3e460671 (дата звернення: 20.12.2023).

7. Dapper - micro ORM with epic possibilities. *MethodPoet*. URL: https://methodpoet.com/dapper/#Using_Dapper (date of access: 20.12.2023).Платформа для створення діаграм в синтаксисі Чена, «вороняча лапка». URL: https://app.diagrams.net/ (дата звернення: 05.12.2023).

8. Using Dapper with ASP.NET Core Web API - Code Maze. *Code Maze*. URL: https://code-maze.com/using-dapper-with-asp-net-core-web-api/ (дата звернення: 20.12.2023).

9. Pattankar M. What is Dapper? How to Use Dapper in ASP.Net Core?. Intensive Project Based Training | DotNetTricks. URL: https://www.dotnettricks.com/learn/aspnetcore/what-is-dapper-and-how-to-use-dapper-in-aspnet-core (дата звернення: 20.12.2023).

10. Khan S. M. A. CRUD In Asp.NET Core Using Dapper With Practical Example. *DEV Community*. URL: https://dev.to/sardarmudassaralikhan/crud-in-aspnet-core-using-dapper-with-practical-example-396n (дата звернення: 20.12.2023).

11. Zang A. Building a CRUD API With Dapper. *Telerik Blogs*. URL: https://www.telerik.com/blogs/building-crud-api-dapper (дата звернення: 20.12.2023).

12. Глосарій загальних термінів бази даних. *Tebapit*. URL: https://tebapit.com/глосарій-загальних-термінів-бази-дан/ (дата звернення: 21.12.2023).

13. Бази даних. Поняття ER-моделі. Поняття сутності (entity). Атрибути. Види атрибутів | BestProg. *BestProg | Програмування: теорія та практика*. URL: https://www.bestprog.net/uk/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ua/ (дата звернення: 21.12.2023).

14. Нотація пітера чена. *Stud*. URL: https://stud.com.ua/77222/informatika/notatsiya_pitera_chena (дата звернення: 21.12.2023).

15. CA ERWin Data Modeler (ERWin). *Stud*. URL: https://stud.com.ua/77235/informatika/erwin_data_modeler_erwin (дата звернення: 21.12.2023).

16. What is SQL Server? - SQL Server. *Microsoft Learn: Build skills that*

open doors in your career. URL: https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16 (дата звернення: 21.12.2023).

17. What is windows forms - windows forms .NET. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0 (дата звернення: 21.12.2023).

18. Що таке патерн Singleton: навіщо він потрібен та як його використовувати. *Highload.today - медіа для розробників*. URL: https://highload.today/uk/shho-take-patern-singleton-navishho-vin-potriben-ta-yak-jogo-vikoristovuvati/ (дата звернення: 21.12.2023).

19. DataGridView Control - Windows Forms .NET Framework. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/datagridview-control-windows-forms?view=netframeworkdesktop-4.8 (дата звернення: 21.12.2023).

ДОДАТОК А. ТЕХНІЧНЕ ЗАВДАННЯ

Метою курсової роботи є розробка реляційної бази даних для предметної області «Система онлайн каталогу манґи» та програмного забезпечення для роботи з нею. Розроблена інформаційна система «Онлайн каталогу манґи» призначена для організації, зберігання та керування цифровим контентом, а також для забезпечення зручної взаємодії користувачів із каталогом манґи у веб-середовищі.

Програмне забезпечення автоматизує процеси додавання, редагування та структуризації манґи, глав і сторінок, підтримує відображення каталогу на сайті, а також забезпечує можливість комунікації користувачів із системою через коментарі, закладки та персональні профілі. Інтерфейс системи орієнтований на простоту використання як для читачів, так і для адміністратора, що спрощує керування контентом та підтримку структури ресурсу.

Для адміністратора платформи система надає можливість додавати та редагувати манґу, створювати глави, завантажувати сторінки, керувати користувачами та контролювати структуру каталогу. Крім того, реалізовано функціонал перевірки коректності даних, що запобігає появі дубльованих записів і забезпечує цілісність бази даних.

Для звичайних користувачів система дозволяє переглядати манґу, переходити між главами, додавати твори до закладок, залишати коментарі, а також керувати власним профілем. Авторизація забезпечує персоналізований доступ і розмежування прав між ролями користувача та адміністратора.

Всі ці можливості в комплексі роблять інформаційну систему «Mankura» зручним інструментом для організації онлайн-каталогу манґи, управління цифровим контентом та створення комфортного середовища для читачів.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– ознайомитися з теоретичними основами проектування реляційних баз даних та побудови ER-моделей;
– сформувати бізнес-правила предметної області «онлайн-каталог манґи»;
– на основі бізнес-правил розробити ER-діаграму моделі даних у синтаксисі Чена;
– спроєктувати логічну та фізичну модель бази даних та реалізувати її у системі керування базами даних MS SQL Server;
– розробити веб-застосунок із використанням ASP.NET Core MVC для взаємодії з базою даних;
– реалізувати функціонал керування манґою, главами, сторінками, користувачами, коментарями та закладками;
– забезпечити авторизацію користувачів та розмежування прав доступу;
– провести тестування програмного забезпечення та перевірити коректність роботи основних функцій;
– оформити пояснювальну записку за результатами виконання роботи.

ДОДАТОК Б. ІНСТРУКЦІЯ КОРИСТУВАЧА

Усі користувачі можуть авторизуватися завдяки власним логінам та паролям, які відповідно до їх даних знаходяться у базі даних. При помилковому введені логіну або паролю програма відобразить помилку про некоректно введені дані.

Вхід до системи під типом користувача «адміністратор». Логін: admin, пароль: admin. Після введення даної інформації у відповідні комірки на формі авторизації, система зможе надати користувачеві доступ до повного функціоналу адміністратора, а саме – доступ до особистої інформації пацієнтів та співробітників клініки; до інформації про наявні медикаменти та послуги; до інтерфейсу запису пацієнтів до спеціалістів; до інформації з платежами; а також до генерації статистики. Перехід від однієї форми до іншої реалізовано завдяки натисканню на відповідну кнопку з лівої бічної панелі.

Щодо лікарів, то кожен спеціаліст матиме доступ тільки до свого електронного медичного кабінету. Лікарі мають особисті логіни та паролі задля авторизації під своїм акаунтом. Дані користувачі матимуть змогу переглядати тільки тут інформацію, що відповідно до БД буде стосуватися виключно їх – тільки їх пацієнти та медичні картки візитів. Лікарі можуть змінювати статус візиту в залежності від того, чи реально прийшла людина на прийом, чи ні. Задля додавання нової мед. картки візиту лікарю достатньо обрати кортеж на DataGridView, після чого натиснути кнопку Add. Якщо медична картка успішно додалася – програма відобразить відповідне повідомлення.

Якщо користувач натискає на бічній панелі кнопку «Disease card», програма дозволяє лікарю перейти на форму з медичними картками, де вже відповідно до ситуації за кожний візит встановлюється діагноз, додаються

надані послуги, встановлюється коефіцієнт складності.

Рентгенолог має логін «perez», пароль – «radiologist». Після успішної авторизації програма відображає робоче вікно рентгенолога. Даному користувачу достатньо обрати необхідний кортеж, після чого у текстбокс занести шлях до зображення рентгену. Це робиться шляхом натискання правої клавіші миші по зображенню, після чого рентгенологу необхідно перейти до пункту «Властивості». Останнім кроком є копіювання шляху до зображення на робочому комп'ютері, назви зображення, та його формату. Після натискання клавіші update дані успішно оновлюються і рентгенолог зможе побачити на екрані відповідний рентген.

Login	Password
admin	admin
martinenko	doctor
perez	radiologist
vasiliev	doctor
kravchuk	doctor
antonenko	doctor
shevchenco	doctor
zhaharchuk	doctor

Рисунок Б.1 – Перелік логінів та паролів для авторизації до ІС

Для того, щоб підключитися до бази даних зі свого серверу у СУБД необхідно знайти файл конфігурації у папці проекту, що має назву «appsettings.json». Можете відкрити його за допомогою програми «Блокнот», після чого замінити дефолтний рядок під'єднання до БД на ваш.