Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Створення веб-порталу ASP. NET продажу книжок»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ19130»

_____ /Андрій ТАНЦЮРА/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ /доц. Олександр ІВАНОВ/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Creating a web portal ASP.NET book sales»
according to educational curriculum «Software engineering»
in the Speciality: «121 Software engineering»

Done by the student of the group PZ19130: /Andrii TANTSIURA/

Scientific Supervisor: /Oleksandr IVANOV/

Normative controller: /Olena KUROPIATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «Інженерія програмного забезпечення»
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____ /Вадим ГОРЯЧКІН/
(підпис)
Дата 22.12.2021

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Танцюрі Андрію Андрійовичу
1. Тема роботи: «Створення веб-порталу ASP. NET продажу книжок»
Керівник роботи: Іванов Олександр Петрович, доцент
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: 29.05.2022 р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
збір та аналіз вимог, зовнішнє проєктування, внутрішнє проєктування, проєктування архітектури системи, розробка програми, тестування та налагодження, висновки та рекомендації

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): відео роботи програми, презентація.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задач, збір інформації, вибір та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	18.03.22	
2	Програмування та відлагодження програми	23.05.22	
3	Тестування програми	27.05.22	
4	Розробка, узгодження і затвердження програмної документації	13.06.22	
5	Подання кваліфікаційної роботи до кафедри	08.06.22	
6	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	23.06.22	

Студент _____ Андрій ТАНЦЮРА
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Олександр ІВАНОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Метою дипломного проекту є проектування та розробка веб-сайту продажу книжок на платформі ASP.NET Framework.

Об’єкт дослідження: підходи до розробки сайтів та веб-додатків за допомогою фреймворку ASP.NET MVC 5, який реалізує патерн Model-View-Controller.

Предметом дипломної роботи є веб-сайт, розроблений на платформі ASP.NET MVC 5.

Пояснювальна записка складається зі вступу, 6 розділів, висновків, бібліографічного списку та 4 додатків:

- вступ — описує актуальність розробки веб-сайту, мету та експлуатаційне призначення розробки. Складається з 1 сторінки;
- перший розділ — містить опис постановки задачі та вимоги до розроблюваного веб-сайту. Складається з 6 сторінок;
- другий розділ — містить опис експлуатаційного та функціонального призначення, функціональних вимог до веб-сайту. Складається з 6 сторінок;
- третій розділ — містить опис внутрішнього проектування системи. Складається з 16 сторінок;
- четвертий розділ — містить опис архітектури системи. Складається з 11 сторінок;
- п’ятий розділ — містить обґрунтування обраних засобів розробки, деталізацію задачі та алгоритм роботи модулів. Складається з 10 сторінок;
- шостий розділ — містить опис процесу відлагодження та тестування веб-сайту. Складається з 5 сторінок;
- додатки містять технічне завдання та робочий проект.

Усього робота містить: таблиць — 10, рисунків — 25, бібліографія — 7 джерел.

Ключові слова: BOOK CATALOG, ASP.NET WEB APPLICATION, .NET FRAMEWORK, ASP.NET MVC 5, MODEL-VIEW-CONTROLLER, ENTITY FRAMEWORK, C#.

ЗМІСТ

ВСТУП

У світі, де життя людей надзвичайно тісно пов’язане з комп’ютерними технологіями, все більше сфер комерційної діяльності переходять на автоматизовані засоби збору, збереження та обробки цифрових даних, адже автоматизація будь-якої системи — це насамперед підвищення продуктивності і ефективності праці, якості надання послуг, усунення рутинних та монотонних операцій, що позитивно позначається на розвитку бізнесу будь-якого масштабу — малого, середнього чи великого.

Створення інтернет-магазину дозволяє значно розширити коло потенціальних клієнтів, запропонувавши їм можливість дистанційного ознайомлення з запропонованими товарами та здійснення їх замовлення онлайн, що в значній мірі зменшує необхідний для цього час, адже не потрібно витрачати його на відвідування звичайних магазинів. Щоб цього досягти, сайт має бути інтуїтивно зрозумілим та звичним для більшості покупців, тобто бути «цифровим» аналогом реальних магазинів та торгівельних майданчиків — мати каталог, кошик, тощо.

Платформа .NET надає потужну бібліотеку класів, що дозволяє використовувати вже існуючу функціональність, а архітектурний шаблон MVC, який реалізує інфраструктура ASP.NET MVC — забезпечує розділення відповідальності між шарами програмного продукту, підвищуючи можливості його подальшого розширення.

ASP.NET MVC — це технологія від компанії Microsoft, надає можливість розробляти гнучкі та функціональні веб-сайти та веб-додатки, яка продовжує набувати все більшої популярності, адже може використовуватися побудови як простих, так і дуже складних сайтів.

1 ЗБІР ТА АНАЛІЗ ВИМОГ

1.1 Огляд існуючих аналогів

Основна функціональність веб-сайту kupichitay.com.ua [1]:

- перегляд каталогу книг;
- сортування книг в каталозі по популярності, рейтингу, по спаданню або зростанню ціни;
- пошук книг по заголовку, автору;
- перегляд детальної інформації про книгу;
- написання відгуків про книгу;
- оцінювання книги;
- кошик (додавання книги, перегляд кошика, зміна кількості книг у кошику);
- оформлення замовлення;
- реєстрація при оформленні першого замовлення.

Головну сторінку веб-сайту зображено на рисунку 1.1.

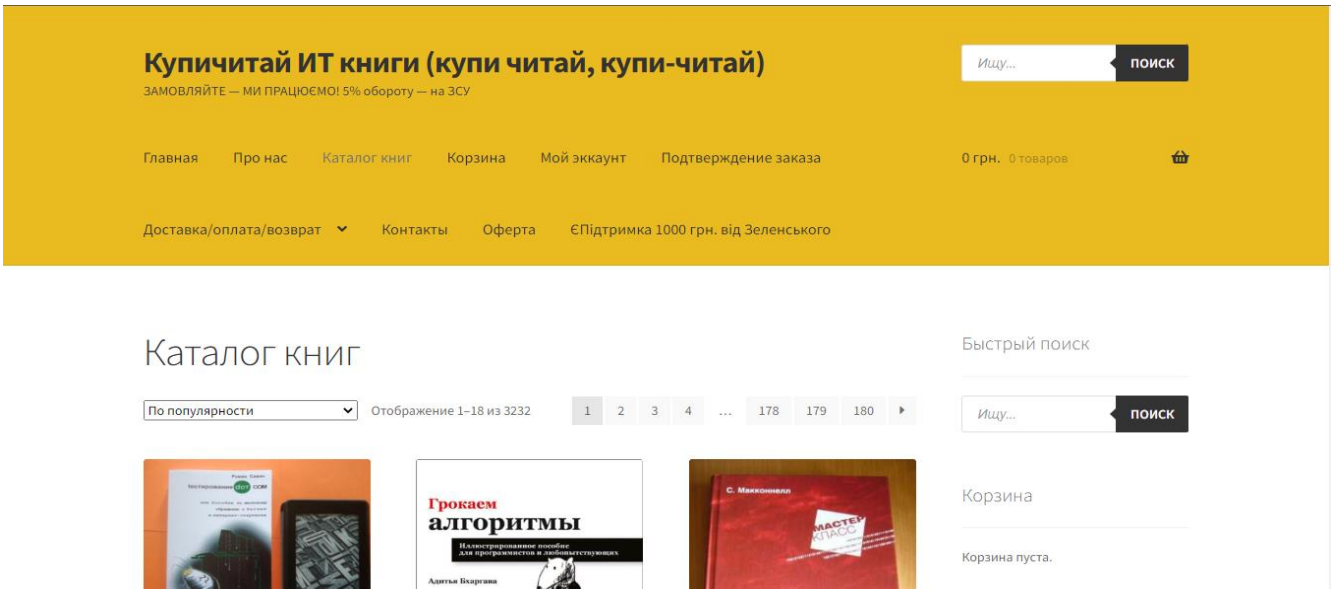

Рисунок 1.1 – Головна сторінка kupichitay.com.ua

openlibrary.org — це відкритий, редагований бібліотечний каталог [2], створений для створення веб-сторінки для кожної коли-небудь опублікованої книги.

Основна функціональність веб-сайту:

– перегляд добірок книг;
– пошук книг по заголовку, автору, ключовим словам;
– перегляд детальної інформації про книгу;
– прослуховування книги;
– написання відгуків про книгу;
– оцінювання книги;
– список бажаного;
– реєстрація та авторизація.

Головну сторінку веб-сайту зображено на рисунку 1.2.

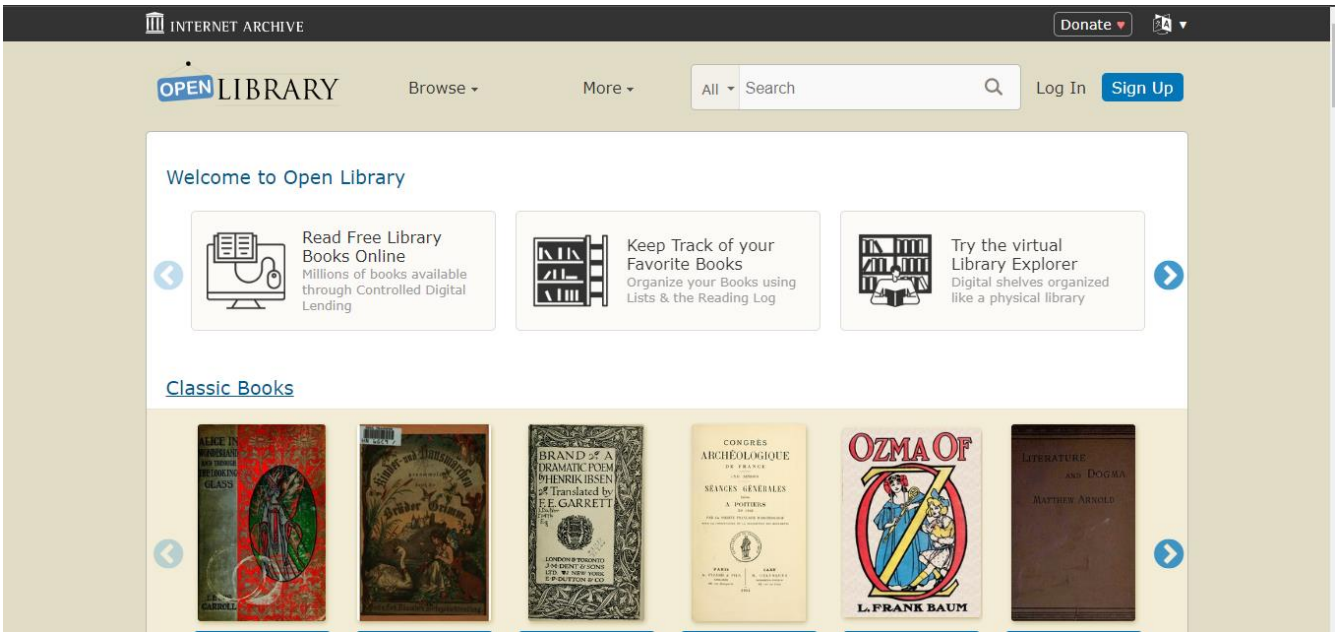

Рисунок 1.2 – Головна сторінка openlibrary.org

manybooks.net [3] — надає велику бібліотеку книг в цифровому форматі безкоштовно в Інтернеті.

Основна функціональність веб-сайту manybooks.net:

– виведення добірок книг;
– знижки;
– пошук книг по заголовку, автору, ключовим словам;
– сортування по заголовкам, авторам, популярності, рейтингу;
– фільтри по жанрам, мовам, рейтингу;

– перегляд детальної інформації про книгу;
– написання відгуків про книгу;
– оцінювання книги;
– реєстрація та авторизація.

Головну сторінку веб-сайту зображено на рисунку 1.3.

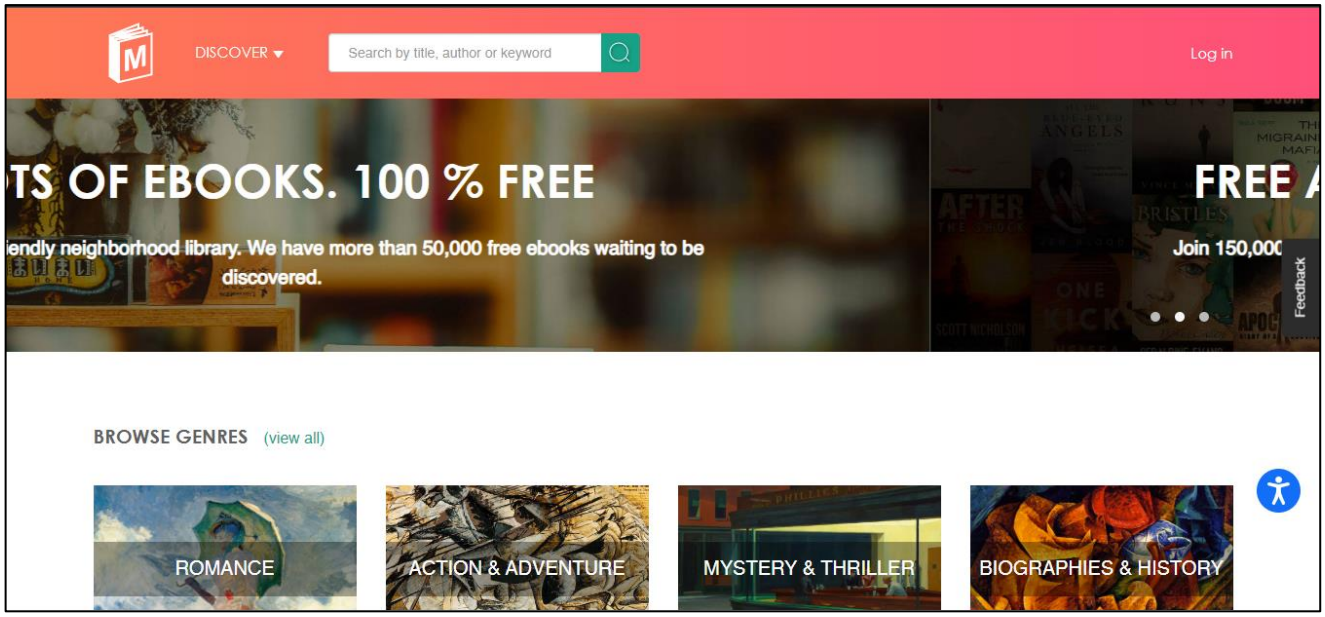

Рисунок 1.3 – Головна сторінка manybooks.net

1.2 Постановка задачі

На основі результатів дослідження існуючих аналогів інтернет-магазинів було сформульовано функціональні вимоги до веб-сайту.

Необхідно розробити веб-сайт для пошуку книг та можливістю їх замовлення.

Для наповнення сайту контентом та керування замовленнями сайт повинен мати панель адміністратора.

Для розширення потенціальних користувачів сайту вся текстова інформація повинна бути представлена на англійській мові.

Кожен користувач веб-сайту може мати одну з наступних ролей: адміністратор, гість, читач.

Можливості гостя (не авторизованого користувача):

– перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;

- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд книг у кошику;
- додавання та видалення книг з кошика;
- зміна кількості книг в кошику;
- створення облікового запису;
- авторизація.

Можливості читача:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд книг у кошику;
- додавання та видалення книг з кошика;
- зміна кількості книг в кошику;
- створення замовлення з введенням адресу пошти;
- перегляд списку замовлень;
- перегляд детальної інформації про замовлення;
- редагування облікового запису;
- вихід з облікового запису.

Можливості адміністратора:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд всіх замовлень;
- перегляд деталей замовлення;
- редагування статусу замовлення та дати його доставки;

- перегляд книг з можливістю фільтрації по жанрам;
- додавання нової книги;
- редагування книги;
- видалення книги;
- робота з авторами, жанрами, мовами та видавництвами, передбачає наступні операції: перегляд, додавання, редагування, видалення.

Висновки до розділу 1

В даному розділі було проведено дослідження основної функціональності існуючих аналогічних розробок інтернет-магазинів та на його основі було сформульовано функціональні вимоги до розроблюваного веб-сайту.

2 ЗОВНІШНЄ ПРОЄКТУВАННЯ

Функціональне призначення ПЗ — розробка інформаційного ресурсу для здійснення пошуку книг в каталозі магазину та оформлення замовлення.

Експлуатаційне призначення ПЗ — на даному інформаційному ресурсі продавець може розміщувати книги та обробляти замовлення, що поступають від клієнтів. Клієнти, в свою чергу, можуть здійснювати пошук книг в каталозі магазину по різним критеріям, що його цікавлять, переглядати повну інформацію про книги, замовляти їх та переглядати власну історію замовлень.

Функціональні вимоги до ПЗ — в залежності від ролі користувача, система повинна надавати користувачу наступні можливості:

2.1 Можливості гостя

Всі можливості, що надає система гостю:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд книг у кошику;
- додавання та видалення книг з кошика;
- зміна кількості книг в кошику;
- створення облікового запису;
- авторизація.

2.2 Можливості читача

Всі можливості, що надає система читачеві:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд книг у кошику;
- додавання та видалення книг з кошика;

- зміна кількості книг в кошику;
- створення замовлення з введенням адресу пошти;
- перегляд списку замовлень;
- перегляд детальної інформації про замовлення;
- редагування облікового запису;
- вихід з облікового запису.

2.3 Можливості адміністратора

Всі можливості, що надає система адміністратору:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд всіх замовлень;
- перегляд деталей замовлення;
- редагування статусу замовлення та дати його доставки;
- перегляд книг з можливістю фільтрації по жанрам;
- додавання нової книги;
- редагування книги;
- видалення книги;
- робота з авторами, жанрами, мовами та видавництвами, передбачає наступні операції: перегляд, додавання, редагування, видалення.

Для опису взаємодії різних користувачів з програмною системою було розроблено специфікацію функціональних вимог та представлено у вигляді діаграм прецедентів для таких акторів, як: гість, читач, адміністратор. Розроблені діаграми прецедентів наведено на рисунках 2.1 – 2.3.

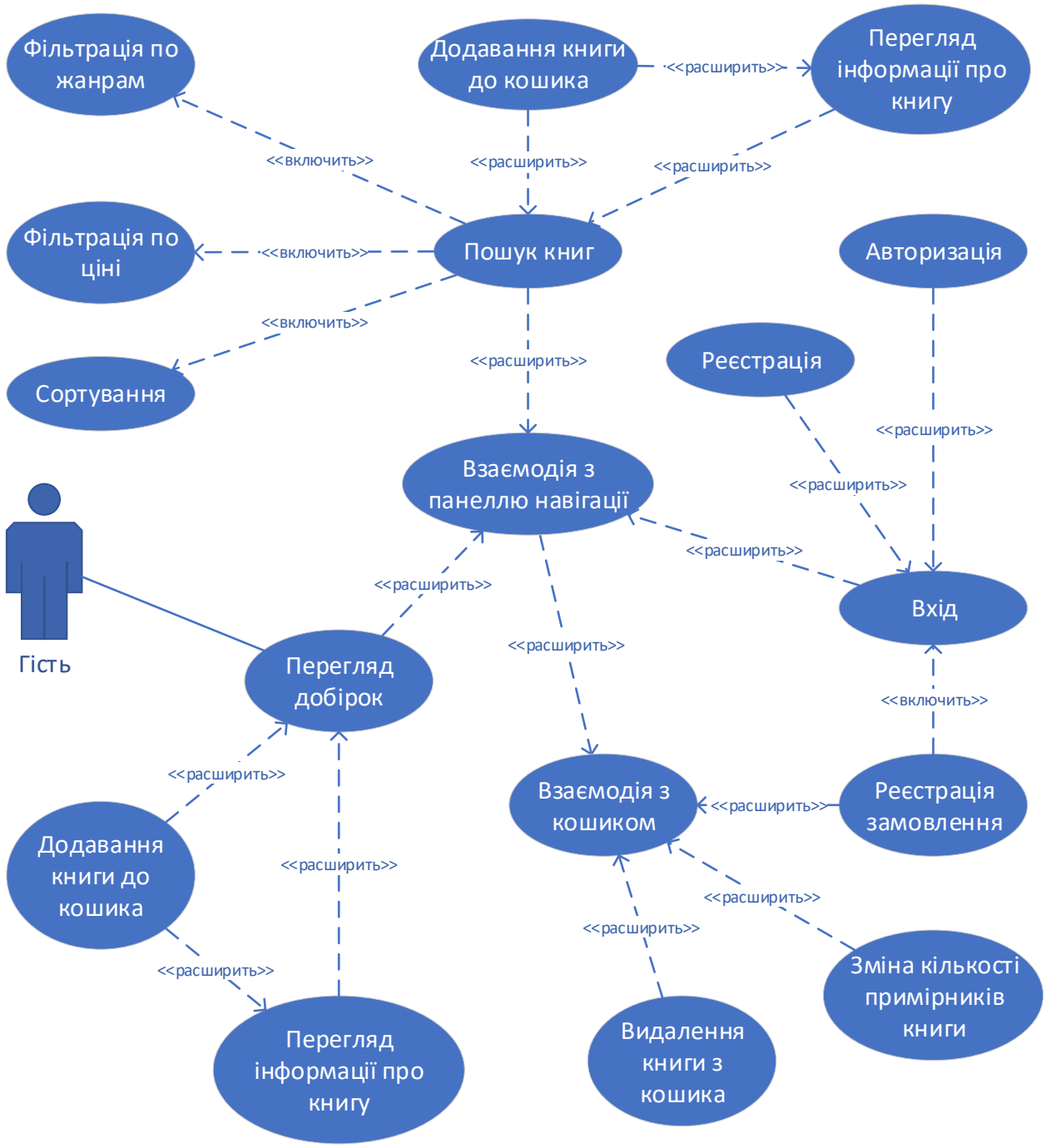

Рисунок 2.1 – Діаграма прецедентів для користувача в ролі «гість»

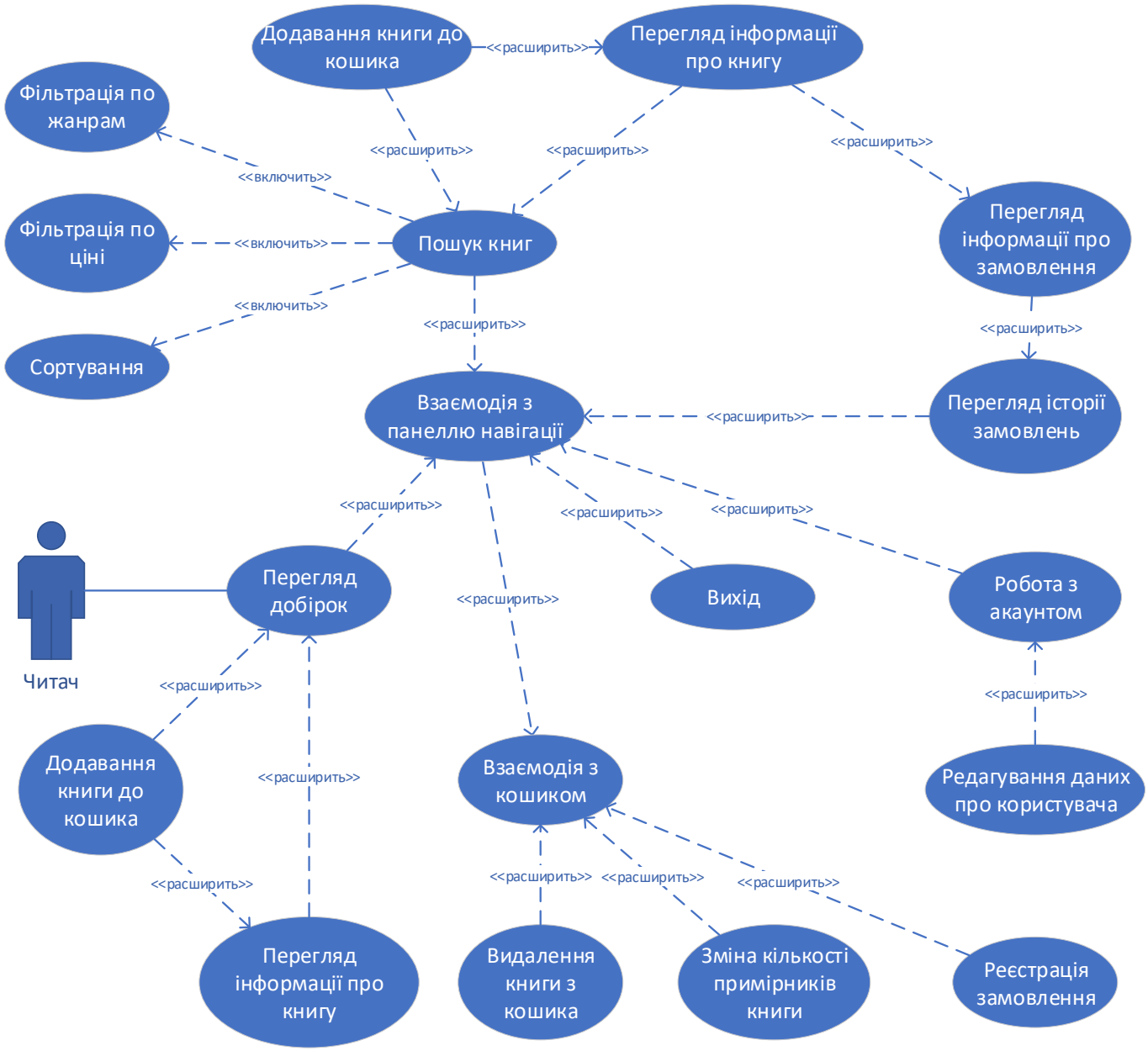

Рисунок 2.2 – Діаграма прецедентів для користувача в ролі «читач»

Рисунок 2.3 – Діаграма прецедентів для користувача в ролі «адміністратор»

Висновки до розділу 2

В даному розділі було описано експлуатаційне та функціональне призначення розроблюваної системи. Розроблено специфікацію функціональних вимог та її представлення у вигляді діаграм прецедентів, що в достатній мірі описує взаємодію користувачів з програмною системою.

3 ВНУТРІШНЄ ПРОЄКТУВАННЯ

3.1 Модулі та їх структура

Для розробки веб-сайту за допомогою ASP.NET MVC 5 використовується патерн MVC — Model-View-Controller. Даний патерн розділяє дані додатку та логіку їх обробки на 3 окремі шари:

- Model (модель) — представляє собою дані та методи роботи з ними, при цьому вона не залежить від представлення (не знає як вони відображаються);
- View (представлення) — відповідає за відображення даних моделі користувачу, та слідкує за змінами у ній. В ASP.NET представлення використовує обробник представлень Razor для впровадження коду .NET в розмітку HTML;
- Controller (контролер) — забезпечує зв'язок між представленням та моделлю, здійснюючи обробку введених даних, формування відповіді і взаємодії з користувачем.

Діаграми класів розроблюваних модулів з урахуванням архітектури MVC зображено на рисунках 3.1 – 3.10.

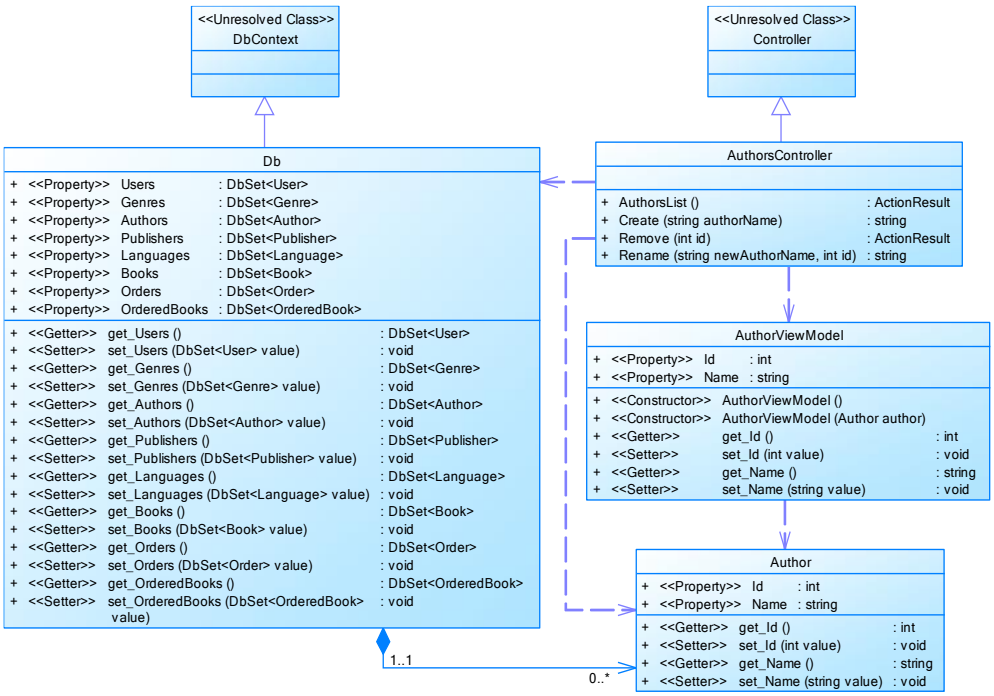

Рисунок 3.1 – Діаграма класів, в основі якої контролер таблиці авторів

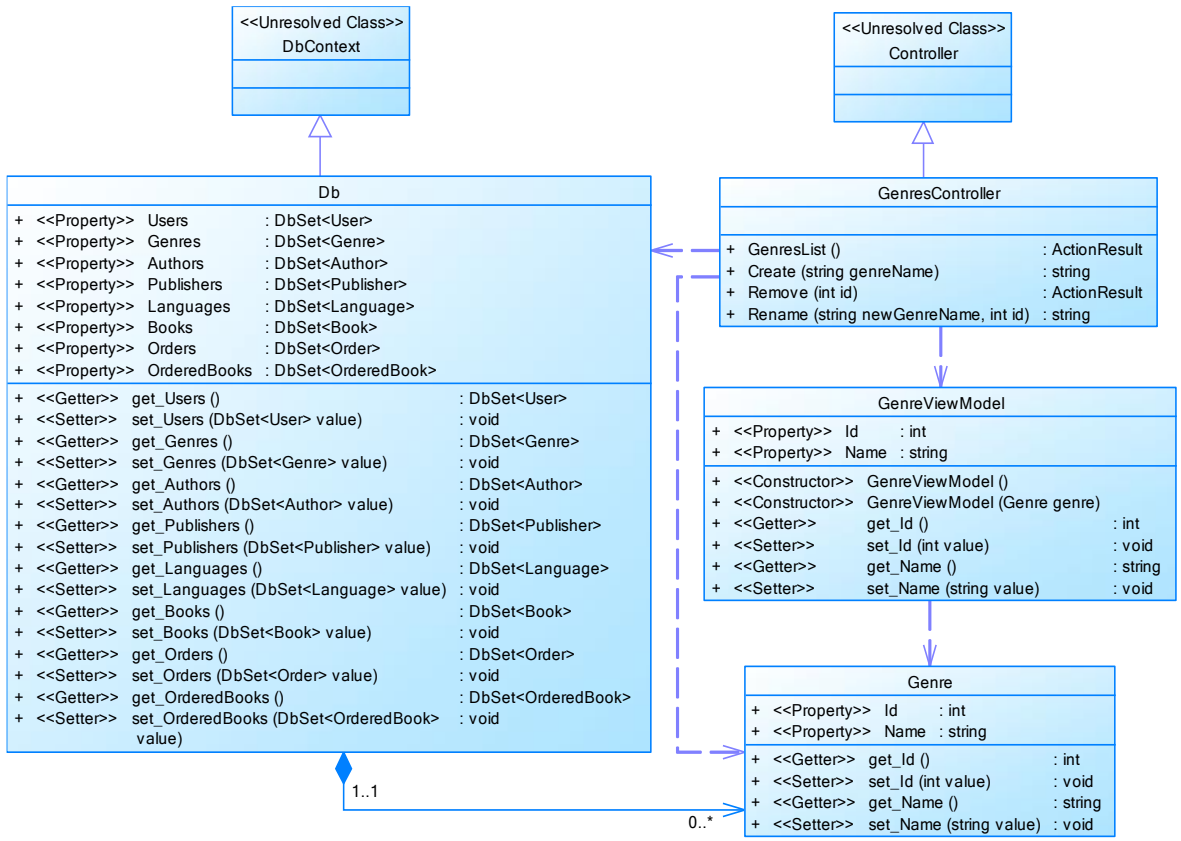

Рисунок 3.2 – Діаграма класів, в основі якої контролер таблиці жанрів

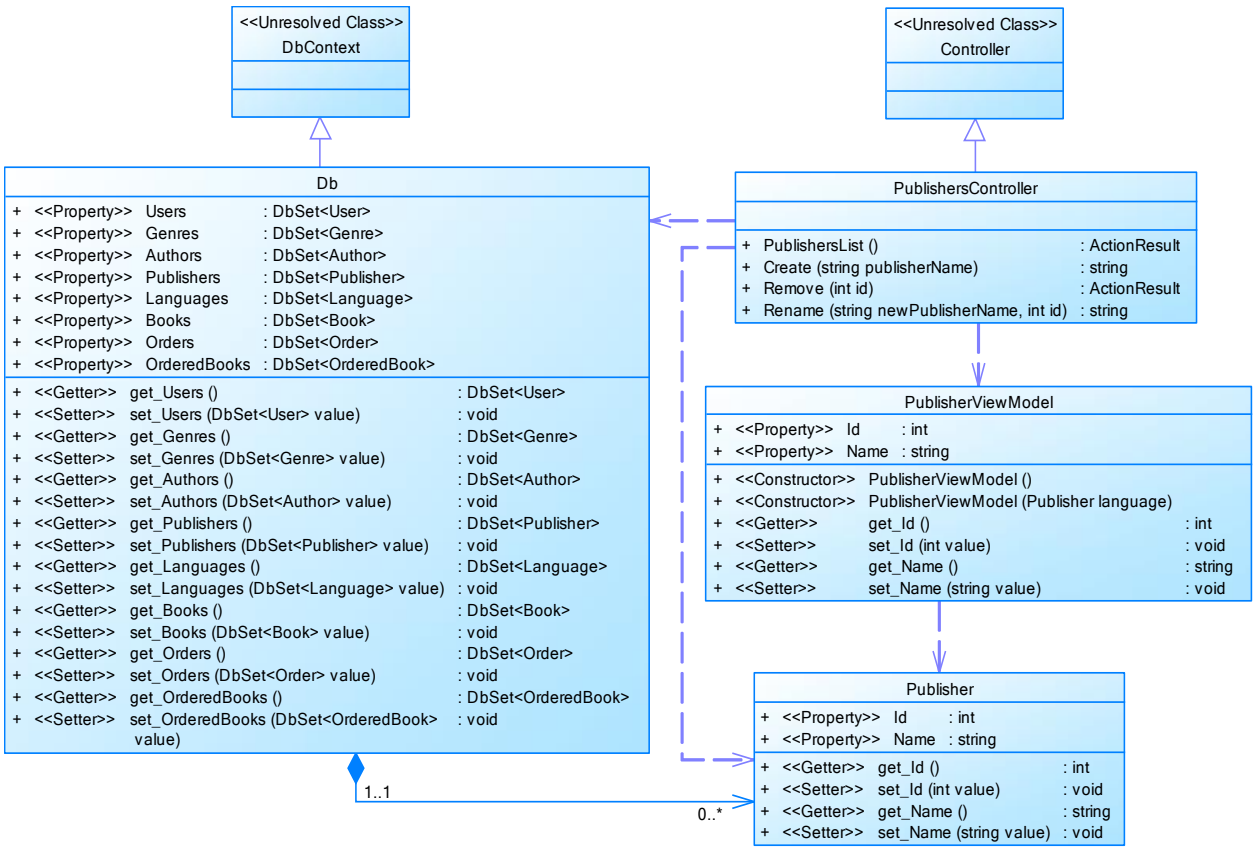

Рисунок 3.3 – Діаграма класів, в основі якої контролер таблиці видавництв

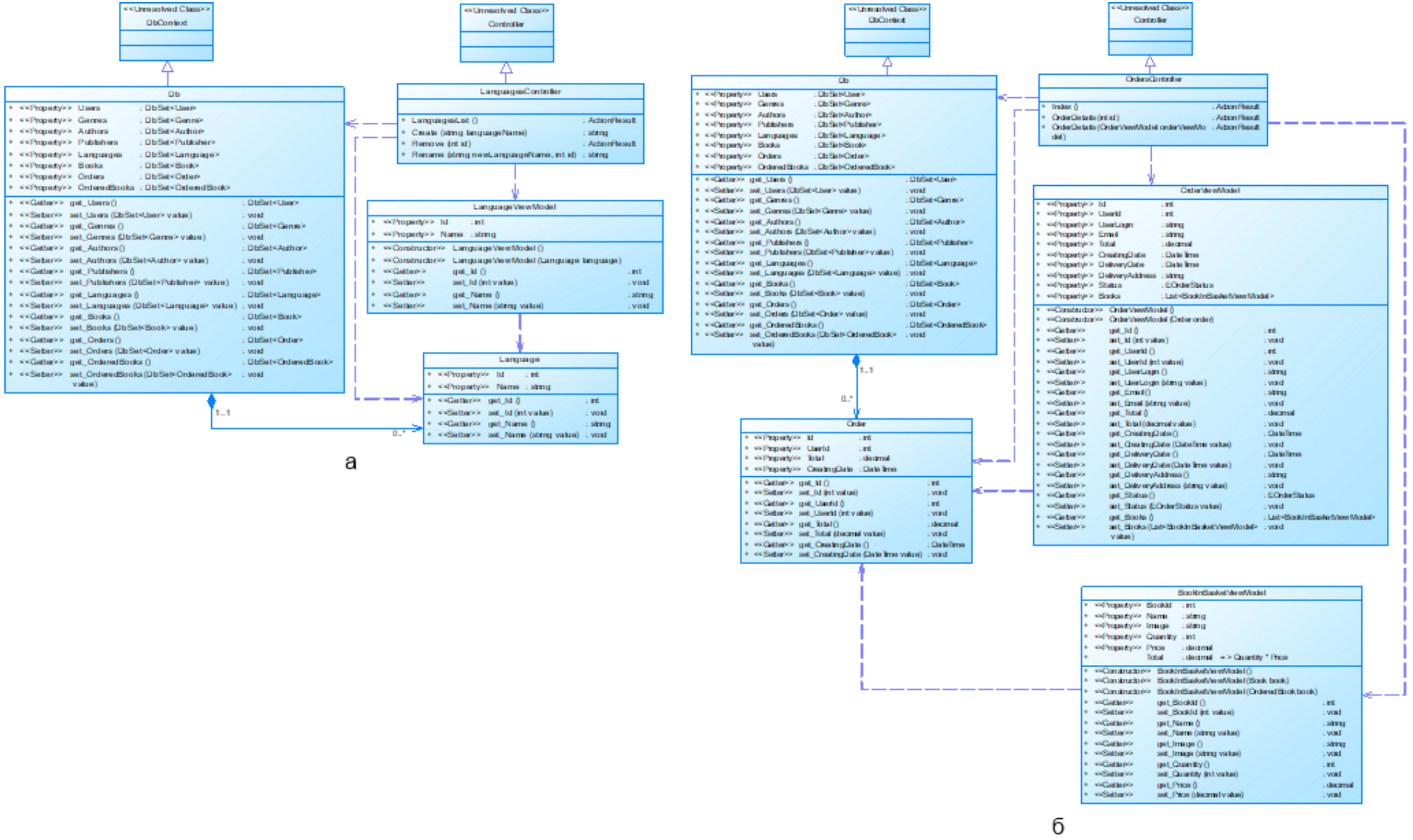

Рисунок 3.4 – Діаграми класів:

а — в основі якої контролер таблиці мов; б — в основі якої контролер таблиці замовлень (в області адміністратора)

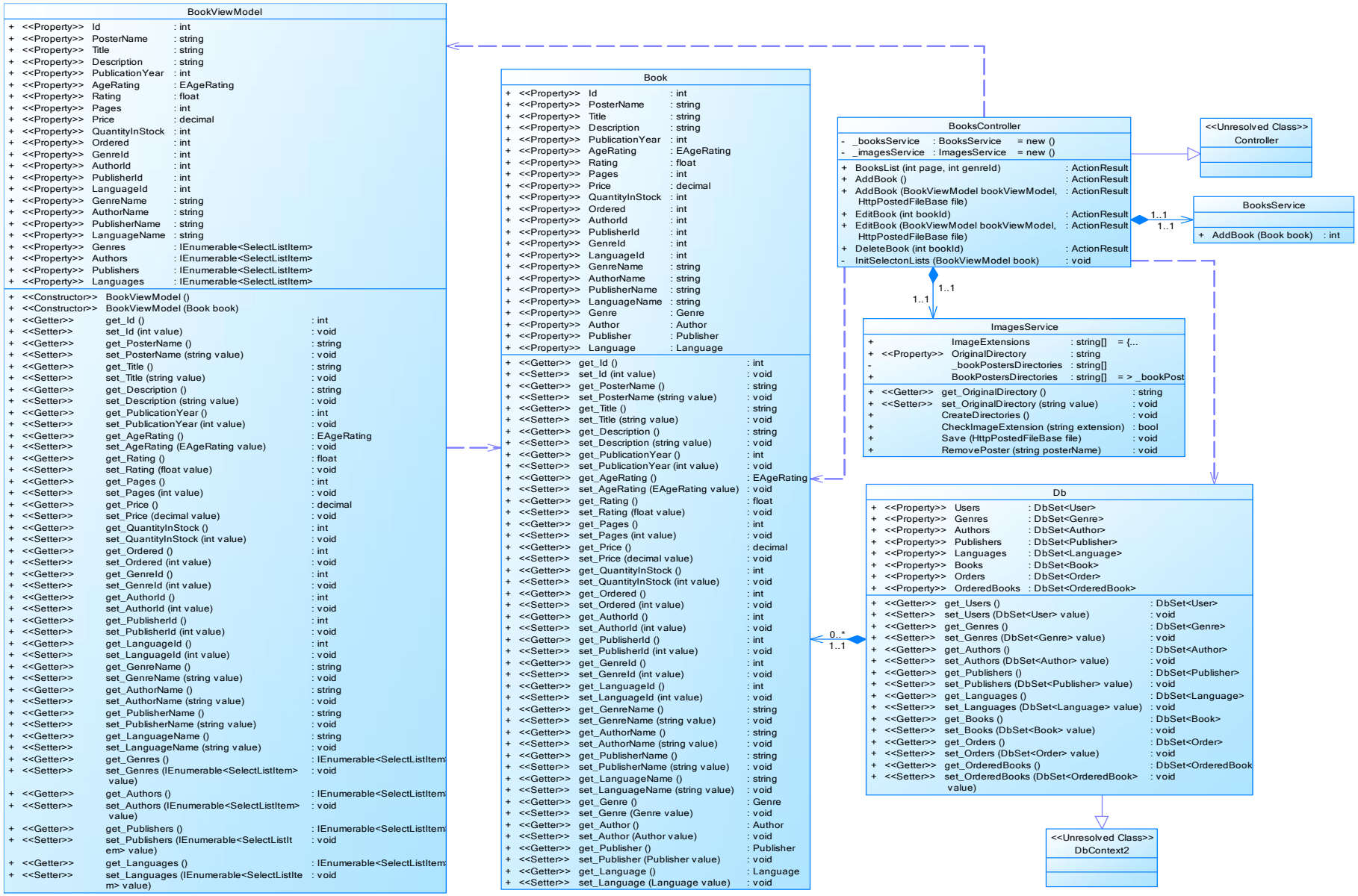

Рисунок 3.5 – Діаграма класів, в основі якої контролер таблиці книг

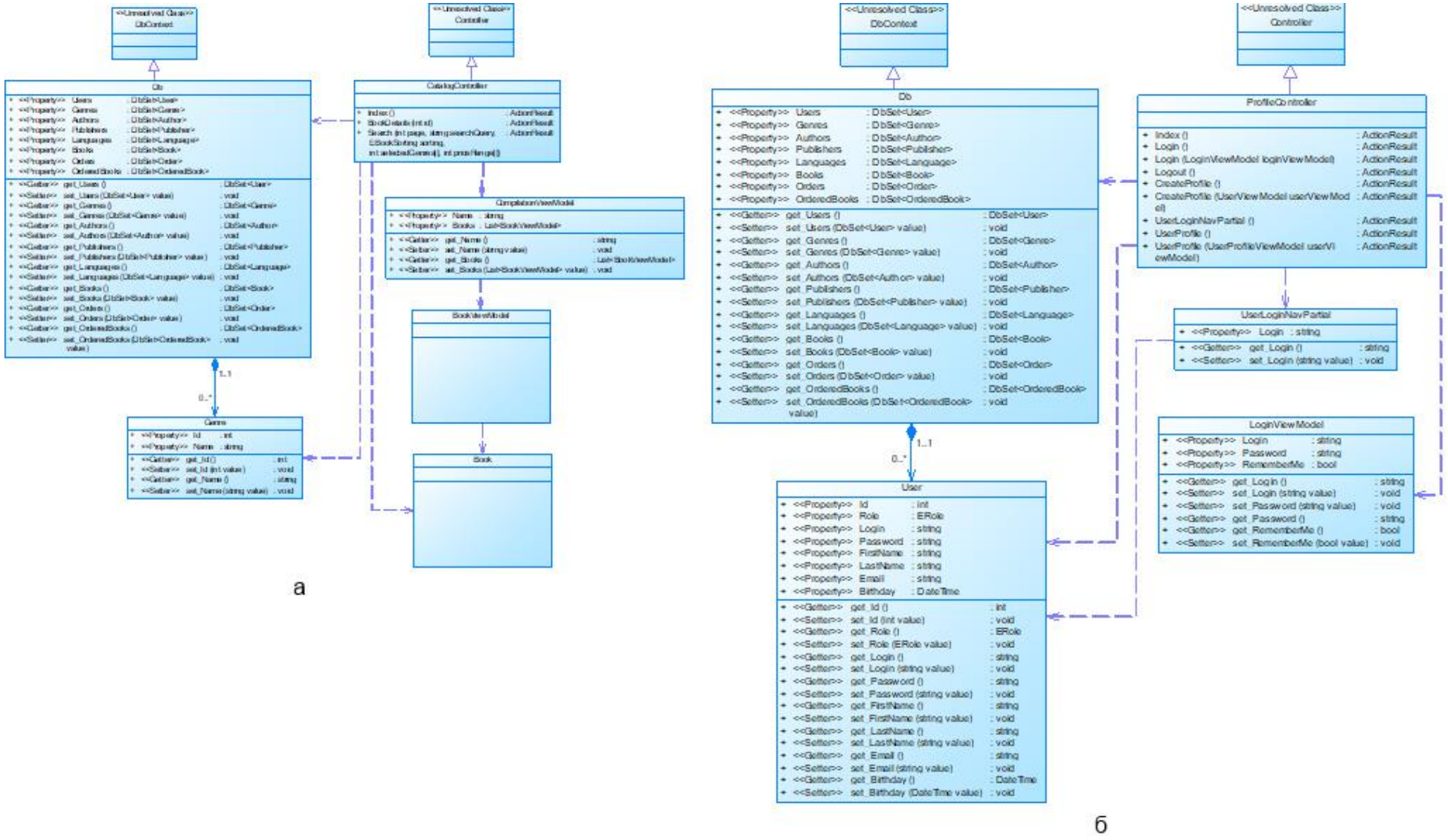

Рисунок 3.6 – Діаграми класів:

а — в основі якої контролер книг в каталозі; б — в основі якої контролер облікового запису користувача

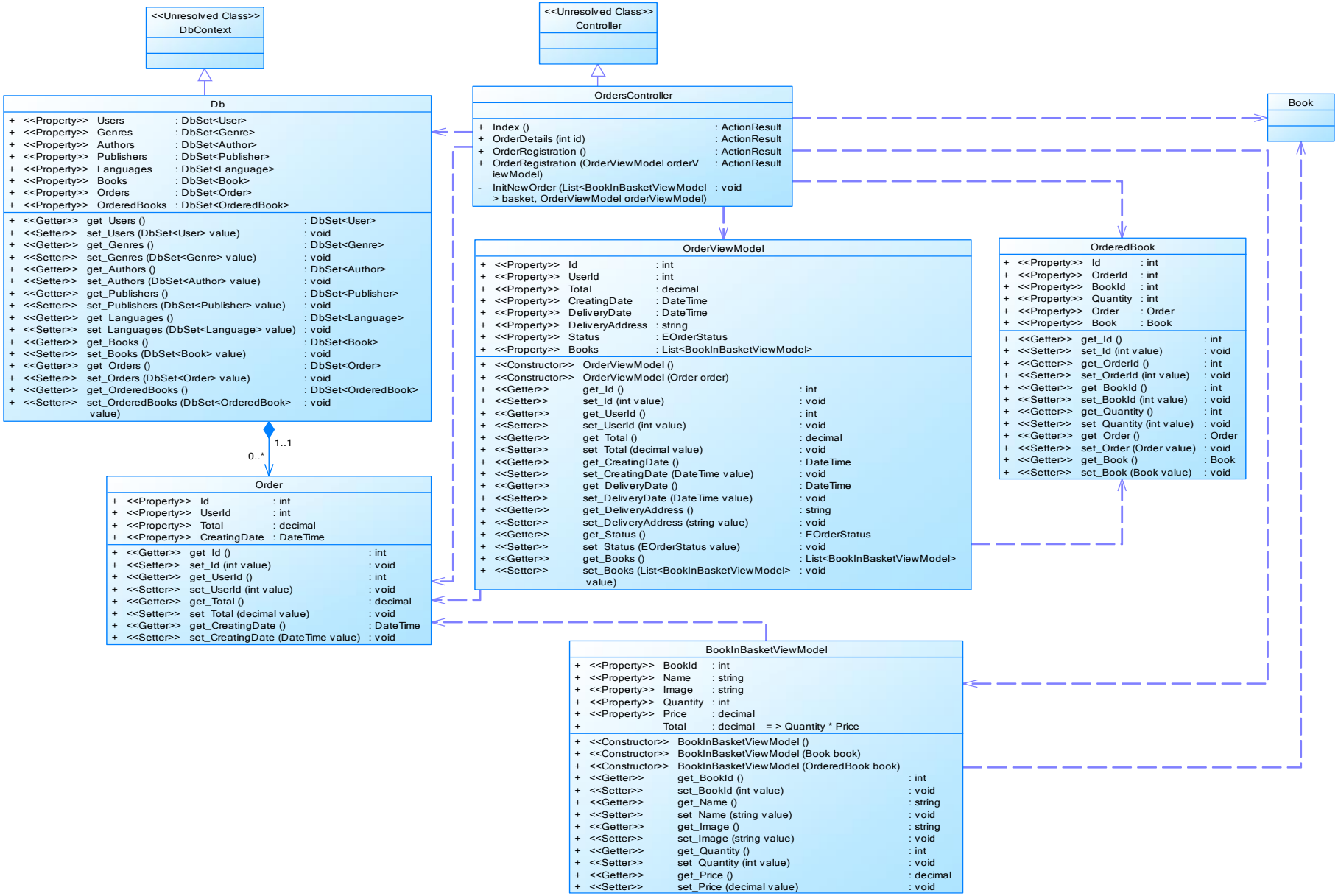

Рисунок 3.7 – Контролер таблиці замовлень (область користувача)

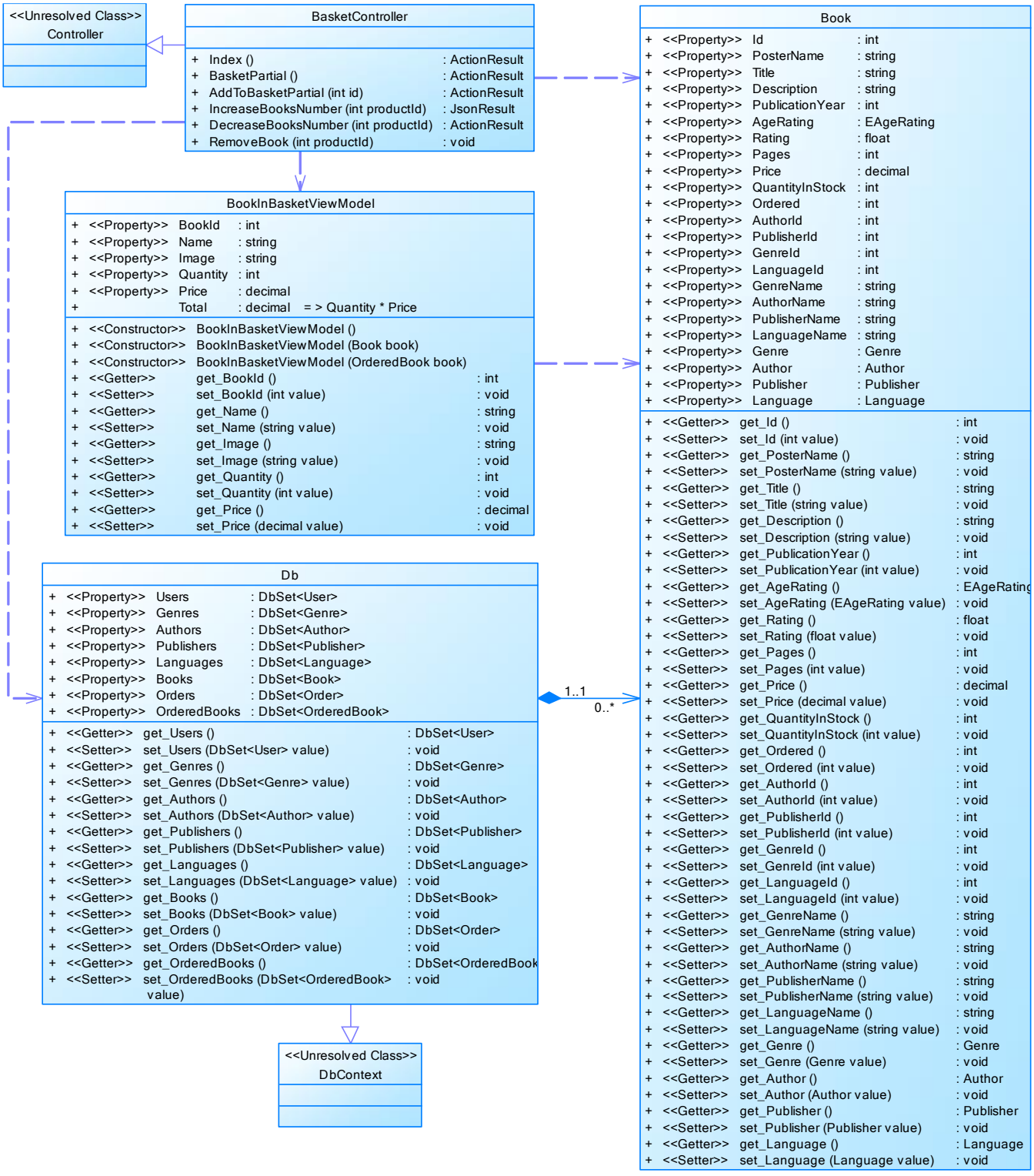

Рисунок 3.8 – Діаграма класів, в основі якої контролер книг у кошику покупця

3.2 Особливості інтерфейсу користувача

Інтерфейс веб-сайту повинен бути простим та інтуїтивно зрозумілим для користувача, при цьому не суперечити його минулому досвіду роботи з веб-сайтами та не вводити його в оману.

Основні класи Bootstrap, використані для задання кольорової палітри тексту: text-body, text-info, text-dark text-light, text-success, text-danger.

Основні класи Bootstrap, використані для задання кольорової палітри повідомлень: alert-info, alert-success, alert-danger.

Основні класи Bootstrap, використані для задання кольорової палітри кнопок: btn-success, btn-info, btn-danger, btn-outline-info, btn-secondary, btn-dark.

Для візуального виділення таких елементів як навігаційна панель та постери було використано клас бутстрапу shadow, який додає до них тіні.

Для візуального «вимкнення» елементів використано клас disabled.

Для операцій, на які потрібен деякий час, додано візуальний завантажувач.

Видалення записів супроводжується підтвердженням користувачем за допомогою відповідного діалогового вікна.

Кольорова палітра сайту:

- навігаційна панель: #343A40;
- текст навігаційної панелі: #FFFFFF;
- кнопка пошуку: #343A40;
- інформаційні кнопки: #a17A2B8;
- кнопки «To basket», «Login»: #343A40;
- кнопки додавання: #28A745;
- кнопки видалення: #DC3545;
- неактивні кнопки: #A0A6AB;
- головний фон: #FFFFFF;
- основний текст: #212529;
- автор книги: #89A745;
- фон футера: #343a40;
- текст футера: #6C757D;
- інформаційне повідомлення: фон — #D1ECF1, текст — #0C5460;
- повідомлення про успіх: фон — #D4EDDA, текст — #525724;
- повідомлення про невдачу: фон — #F8D7DA, текст — #F8D7DA.

Можливості веб-сайт було вичерпно описано в діаграмах прецедентів 2.1 – 2.3.

3.3 Повідомлення системи

Для оповіщення користувача про різні ситуації, що можуть виникнути під час його взаємодії з веб-сайтом, використовуються текстові повідомлення, наведені в таблиці 3.1.

Таблиця 3.1 – Повідомлення системи

Текст	Адресат	Ситуація	Рекомендовані дії
1	2	3	4
The author already exist!	Адміністратор	Автор вже існує	Ввести іншу назву жанру
The author has been added	Адміністратор	Додано нового автора	Переглянути
The author name has been changed	Адміністратор	Автора відредаговано	Переглянути
You have deleted a author	Адміністратор	Автора видалено	Переглянути
The genre already exist!	Адміністратор	Автор вже існує	Ввести інше ім’я автора
The genre has been added	Адміністратор	Додано новий жанр	Переглянути
The genre name has been changed	Адміністратор	Жанр відредаговано	Переглянути
You have deleted a genre	Адміністратор	Жанр видалено	Переглянути
The publisher already exist!	Адміністратор	Видавництво вже існує	Ввести іншу назву видавництва
The publisher has been added	Адміністратор	Додано нове видавництво	Переглянути
The publisher name has been changed	Адміністратор	Видавництво відредаговано	Переглянути
You have deleted a publisher	Адміністратор	Видавництво видалено	Переглянути
The language already exist!	Адміністратор	Мова вже існує	Ввести іншу назву мови
The language has been added	Адміністратор	Додано нову мову	Переглянути
The language name has been changed	Адміністратор	Мову відредаговано	Переглянути
You have deleted a language	Адміністратор	Мову видалено	Переглянути

Продовження таблиці 3.1

1	2	3	4
This poster is already used in another book!	Адміністратор	Обрано постер існуючої книги	Обрати інший постер
The poster was not uploaded - incorrect image extension!	Адміністратор	Обрано зображення з некоректним розширенням	Обрати зображенння з розширенням jpg, jpeg, pjpeg, gif, x-png, png
You have added a book!	Адміністратор	Додано нову книгу	Переглянути
The poster is a required field!	Адміністратор	Постер не обрано	Обрати постер
You edited the book!	Адміністратор	Книгу відредаговано	Переглянути
You have edited the order	Адміністратор	Замовлення відредаговано	Переглянути
Your basket is empty.	Читач, гість	Кошик порожній	Додати книгу до кошика
Available in stock:	Читач, гість	Недостатньо книг на складі	Зменшити кількість книг у кошику або обрати іншу книгу
Nothing found.	Адміністратор, читач, гість	Пошук книг не дав результатів	Виконати пошук з іншими критеріями
Your order is being processed	Читач	Оформлено нове замовлення	Переглянути
Invalid login or password.	Адміністратор, читач	Логін або пароль введене не правильно	Ввести коректний логін та пароль
Login ### is taken	Гість	Логін вже зареєстровано	Ввести інший логін
Now you are registered and can log in.	Гість	Реєстрацію завершено	Переглянути
Passwords do not match.	Читач	Паролі не співпадають	Ввести співпадаючі паролі
You have edited your profile	Читач	Обліковий запис відредаговано	Переглянути
The field is required	Гість читач, адміністратор,	Користувач не заповнив необхідне поле	Заповнити поле

3.4 Програмні засоби

Для взаємодії з веб-сайтом необхідно мати один з наступних веб-браузерів:

- Microsoft Edge v.102.0.1245.33+;
- Google Chrome v.102.0.5005.63+;
- Opera v.87.0+.

Висновки до розділу 3

В даному розділі було здійснено опис модулів та їх структури, особливостей інтерфейсу, включаючи кольорову палітру компонентів та їх призначення, і повідомлення системи для користувача.

4 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ

4.1 Компоненти системи на фізичному рівні

Щоб змоделювати компоненти розроблюваного веб-сайту, залежності та зв’язки між ними на фізичному рівні, доцільно скористатися UML-діаграмою компонентів. Створену діаграму компонентів зображено на рисунках 4.1 – 4.10.

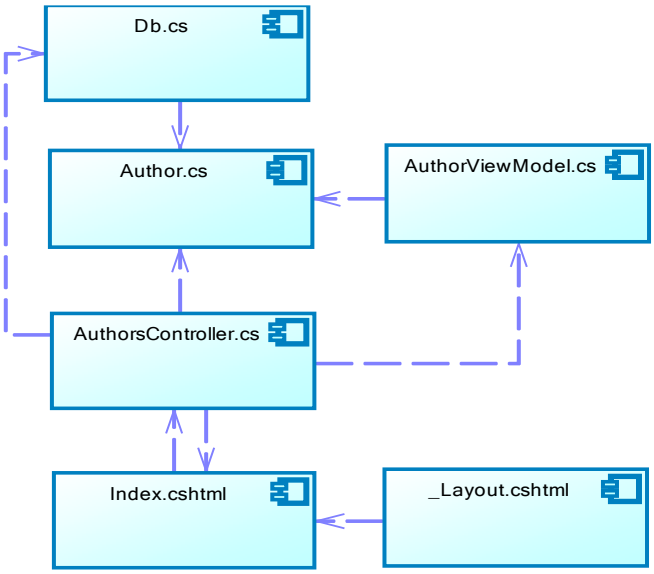

Рисунок 4.1 – Діаграма компонентів, в основі якої лежить контролер таблиці авторів

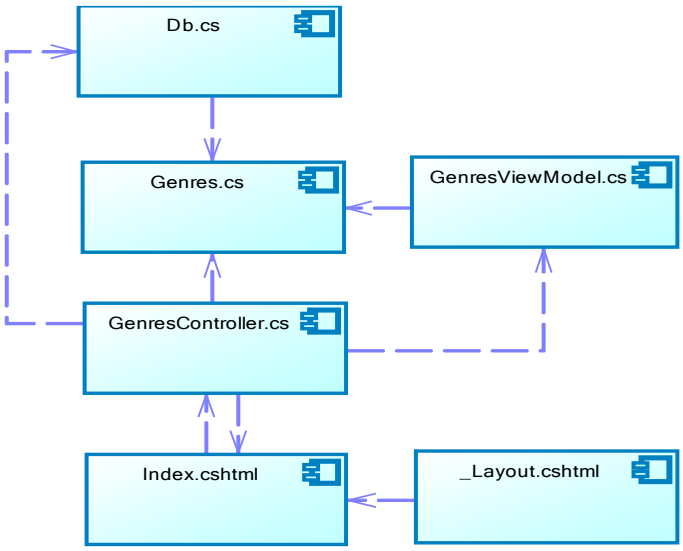

Рисунок 4.2 – Діаграма компонентів, в основі якої лежить контролер таблиці жанрів

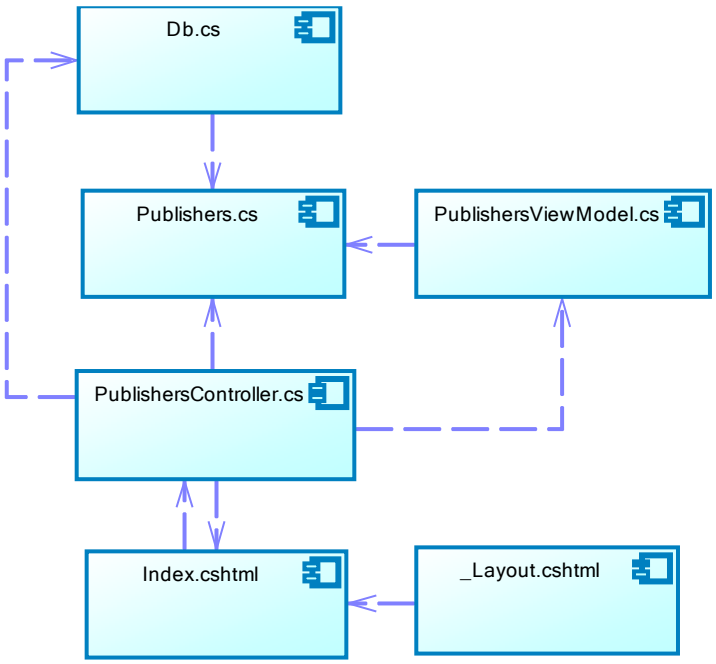

Рисунок 4.3 – Діаграма компонентів, в основі якої лежить контролер таблиці видавництв

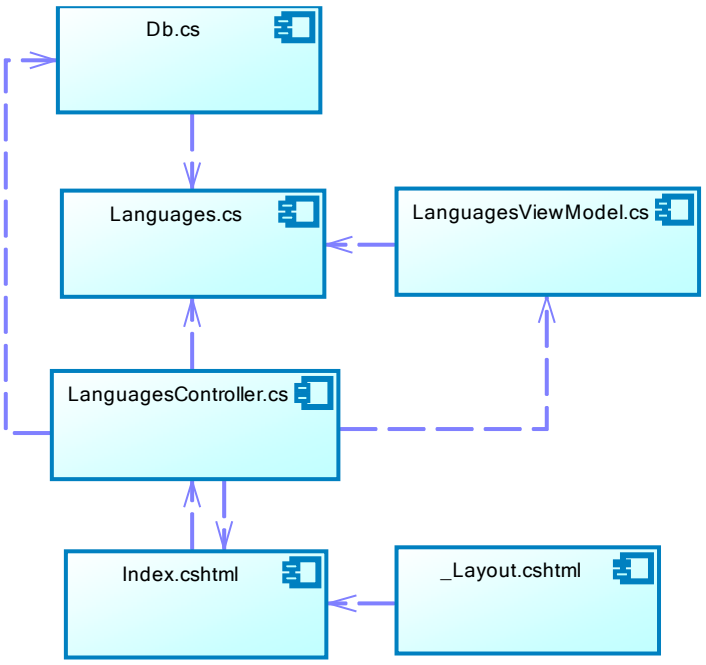

Рисунок 4.4 – Діаграма компонентів, в основі якої лежить контролер таблиці мов

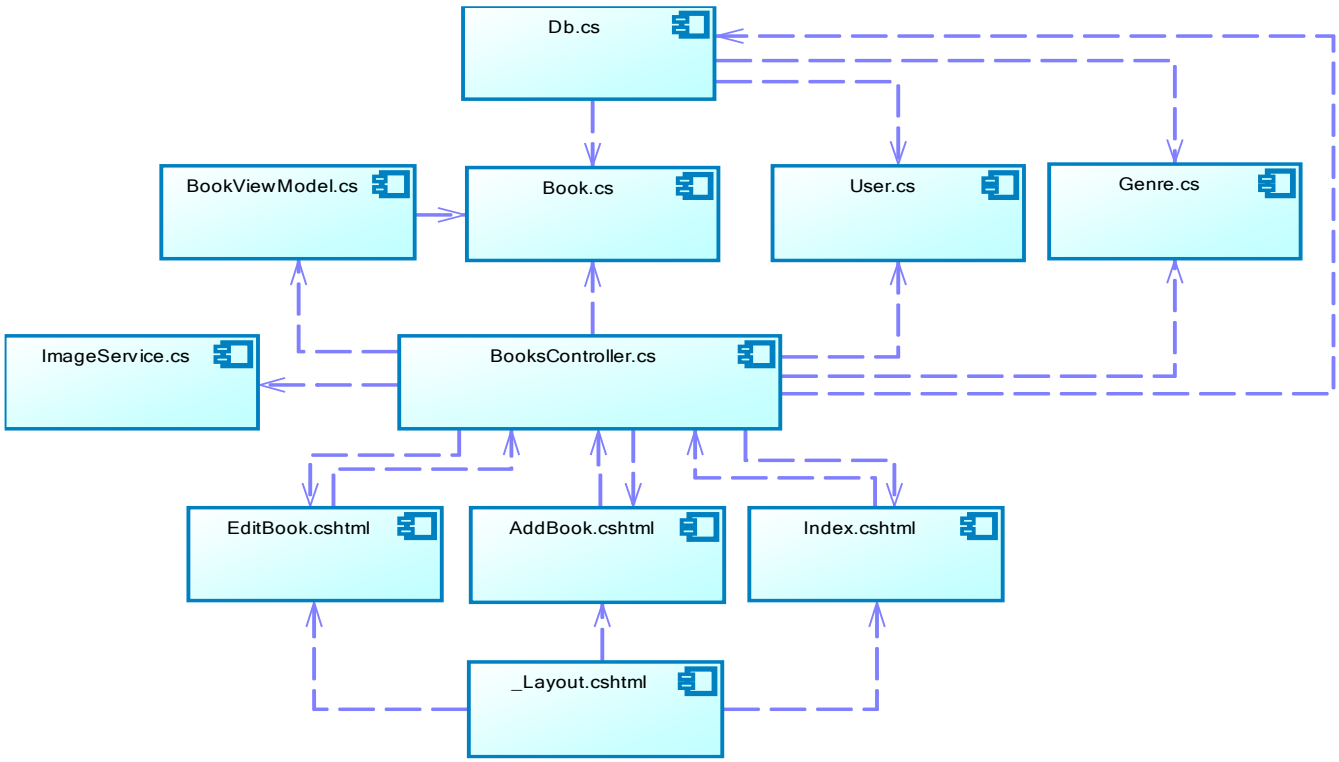

Рисунок 4.5 – Діаграма компонентів, в основі якої лежить контролер таблиці книг

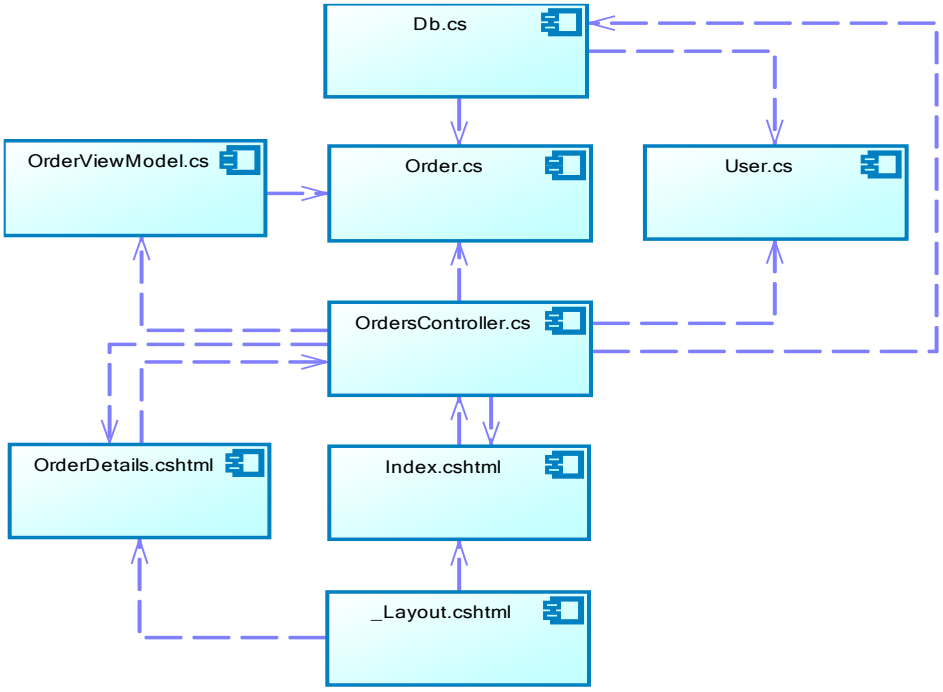

Рисунок 4.6 – Діаграма компонентів, в основі якої лежить контролер таблиці замовлень (в області адміністратора)

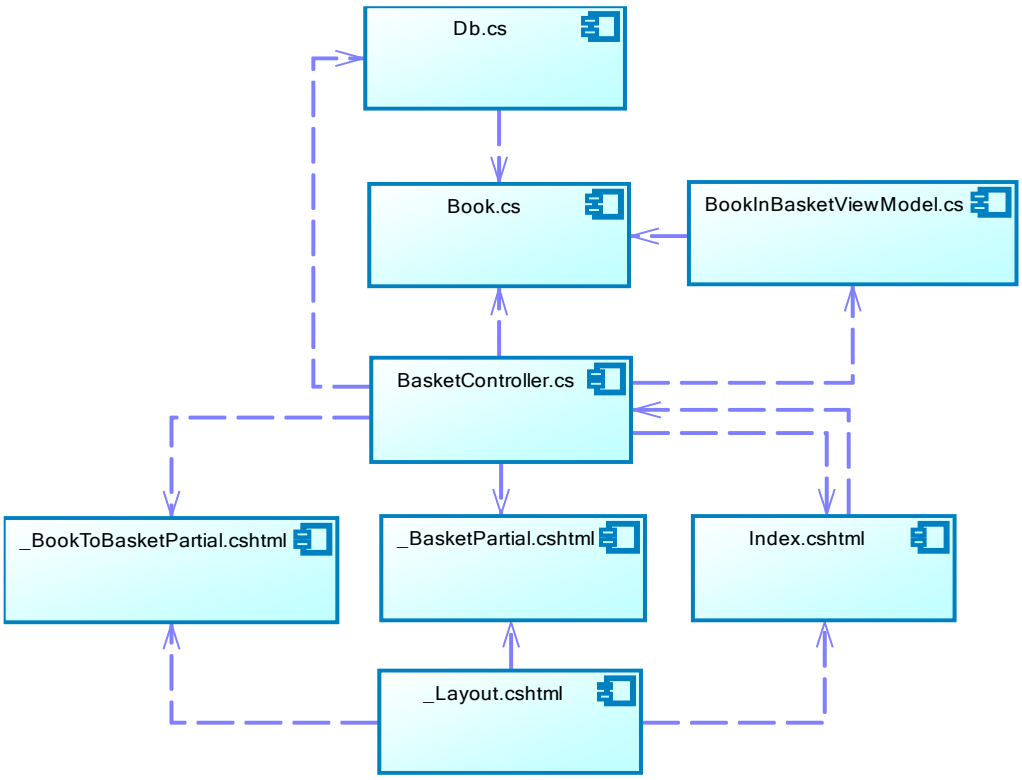

Рисунок 4.7 – Діаграма компонентів, в основі якої лежить контролер кошика покупця

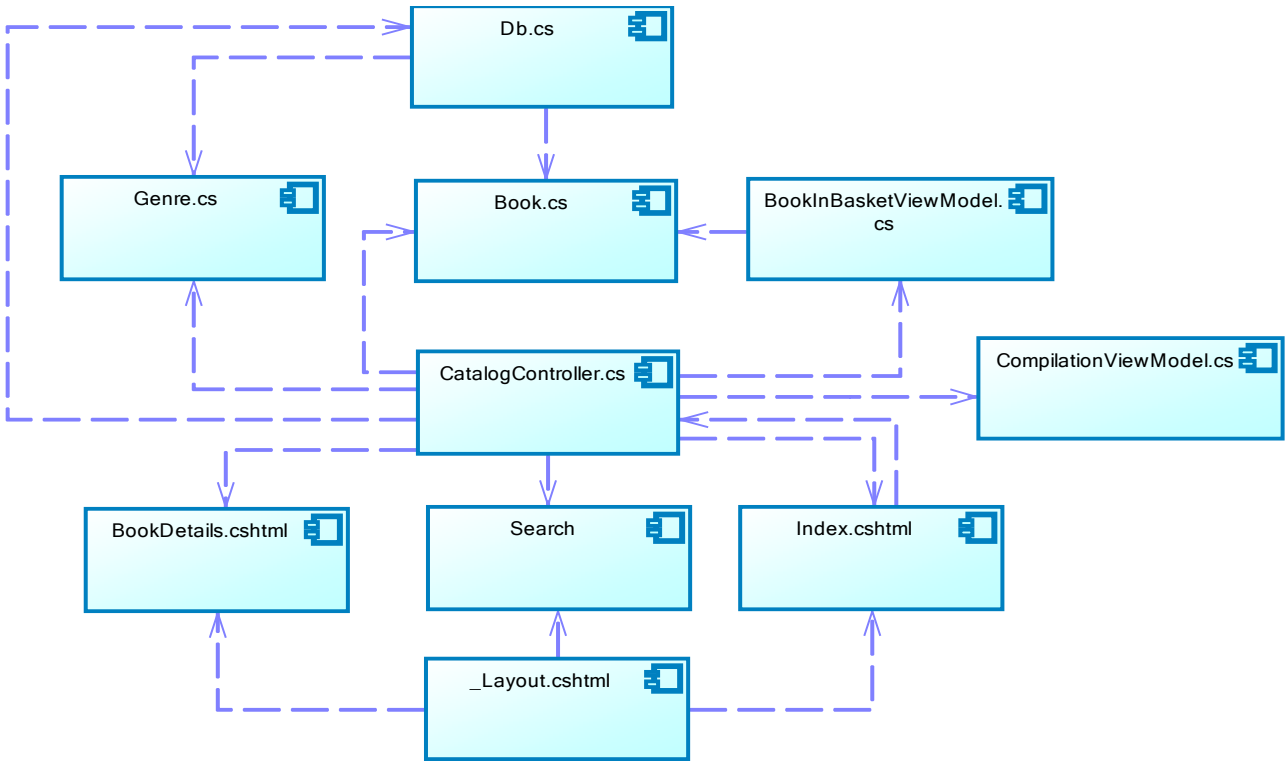

Рисунок 4.8 – Діаграма компонентів, в основі якої лежить контролер книг в каталозі

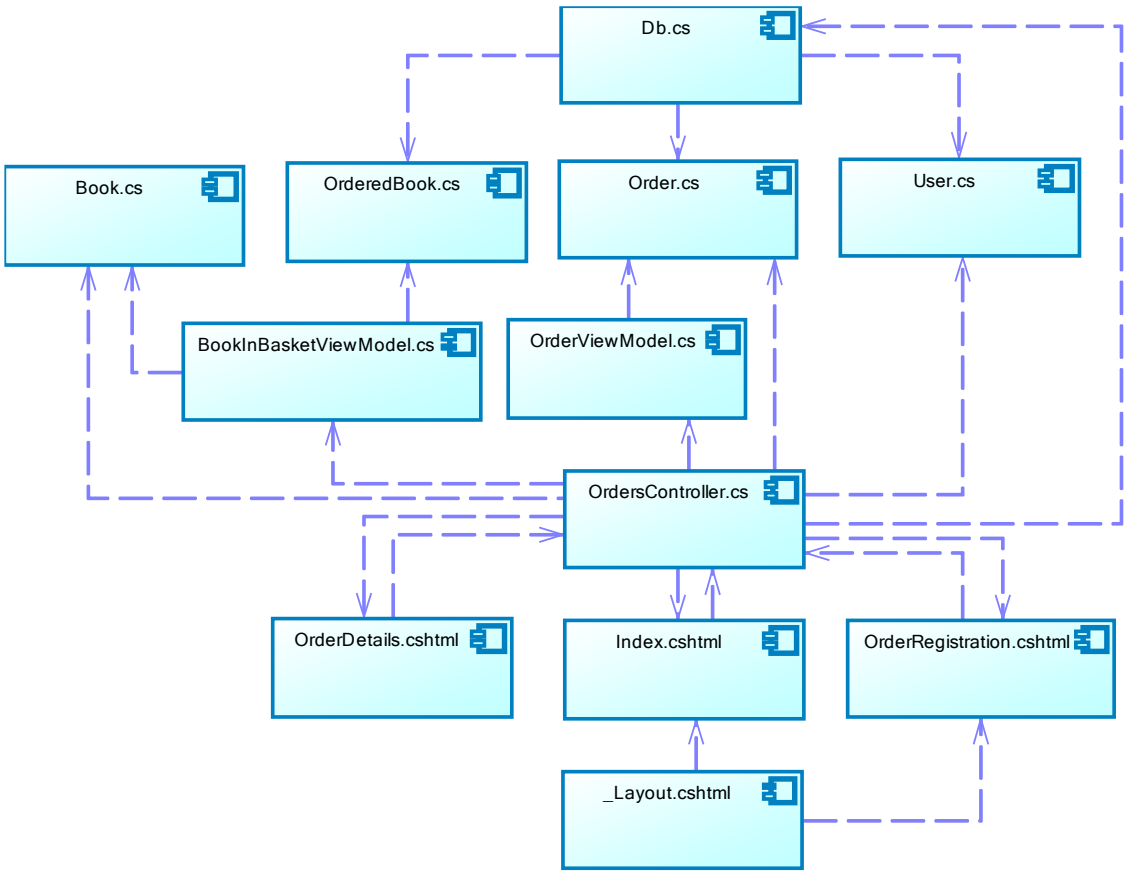

Рисунок 4.9 – Діаграма компонентів, в основі якої лежить контролер таблиці замовлень (в області користувача)

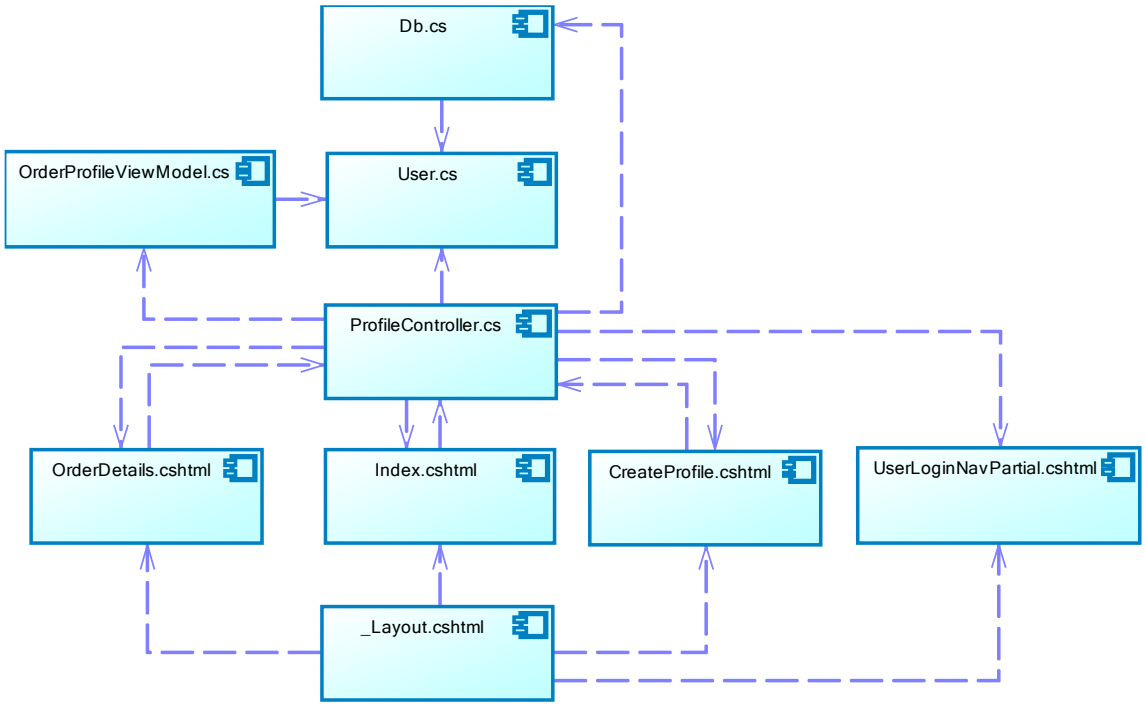

Рисунок 4.10 – Діаграма компонентів, в основі якої лежить контролер профілю користувача

4.2 Проєктування бази даних

Для реалізації всіх функціональних вимог до веб-сайту, було розроблено базу даних та проведено її нормалізацію. Фізичну схему БД у нотації Crow’s Foot зображено на рисунку 4.11.

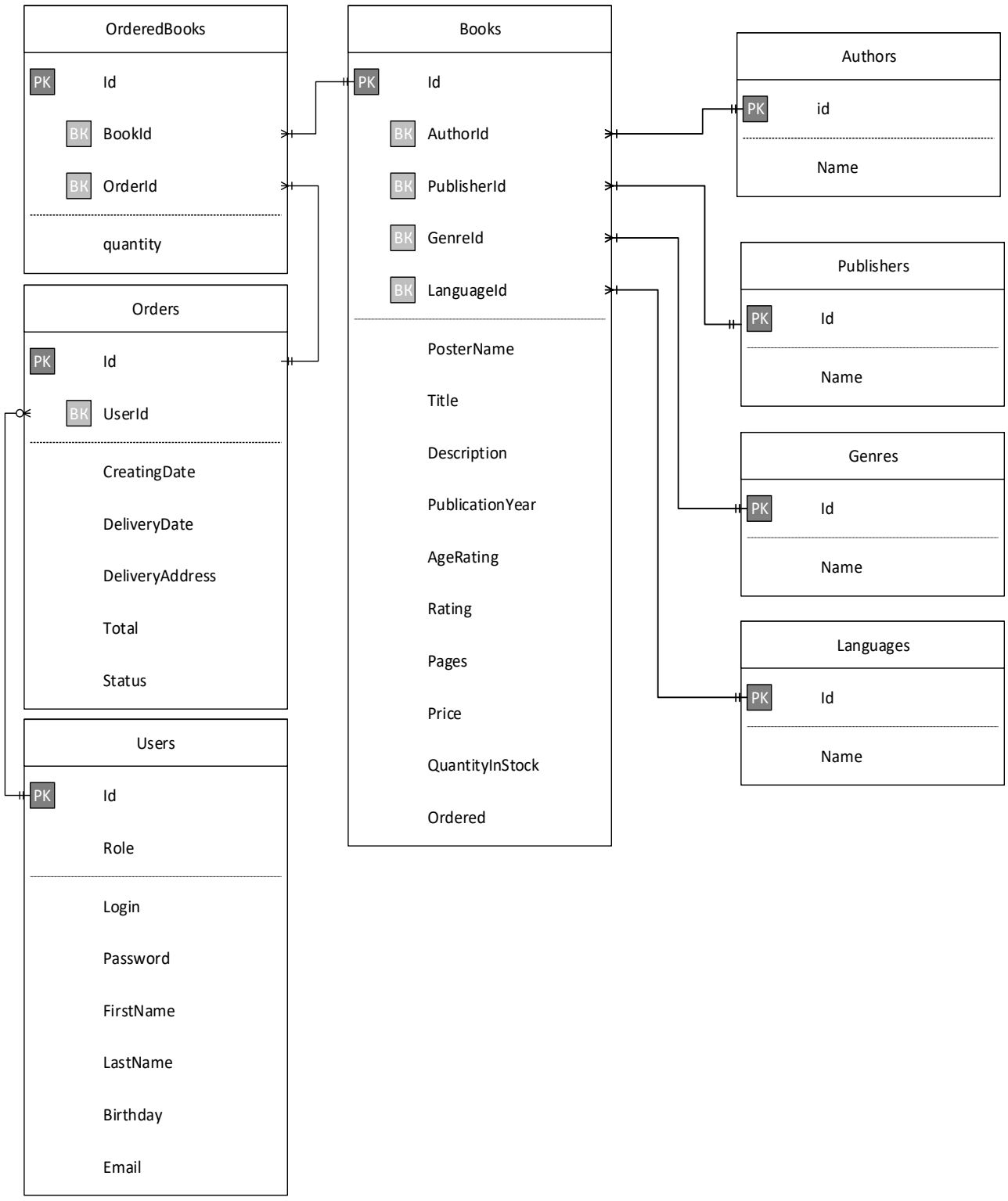

Рисунок 4.11 – Фізична схема бази даних у нотації Crow’s Foot

Для атрибуту Role таблиці Users було створено перелік ERole, який може містити наступні значення: Reader, Admin.

Для атрибуту Status таблиці Orders було створено перелік EOrderStatus, який може містити наступні значення: InProcessing, InDelivery, Delivered, Received, Cancelled.

Для атрибуту AgeRating таблиці Books було створено перелік EAgeRating, який може містити наступні значення: Any, Six, Twelve, Sixteen, Eighteen.

На основі спроектованої бази даних було розроблено таблиці 4.1 – 4.8.

Таблиця 4.1 – Таблиця «tableAuthors»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор автору	int	4	+
2	Name	Ім’я автора	NVARCHAR	70	-

Таблиця 4.2 – Таблиця «tableGenres»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор жанру	int	4	+
2	Name	Ім’я жанру	NVARCHAR	60	-

Таблиця 4.3 – Таблиця «tableLanguages»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор мови	INT	4	+
2	Name	Ім’я мови	NVARCHAR	60	-

Таблиця 4.4 – Таблиця «tablePublishers»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор видавництва	INT	4	+
2	Name	Назва видавництва	NVARCHAR	60	-

Таблиця 4.5 – Таблиця «tableUsers»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	2	3	4	5	6
1	Id	Ідентифікатор користувача	INT	4	+
2	Role	Ідентифікатор ролі	INT	4	-
3	Login	Ім’я для входу в обліковий запис	NVARCHAR	16	-
4	Password	Пароль	NVARCHAR	16	-

Продовження таблиці 4.5

1	2	3	4	5	6
5	FirstName	Ім'я	NVARCHAR	30	-
6	LastName	Прізвище	NVARCHAR	30	-
7	Email	Пошта	NVARCHAR	254	-

Таблиця 4.6 – Таблиця «tableBooks»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор користувача	INT	4	+
2	AuthorId	Ідентифікатор автора	INT	4	-
3	PublisherId	Ідентифікатор видавництва	INT	4	-
4	GenreId	Ідентифікатор жанру	INT	4	-
5	LanguageId	Ідентифікатор мови	INT	4	-
6	PosterName	Ідентифікатор ролі	INT	4	-
7	Title	Назва книги	NVARCHAR	150	-
8	Description	Опис книги	NVARCHAR	100	-
9	PublicationYear	Рік видавництва	INT	4	-
10	AgeRating	Вікове обмеження	INT	4	-
11	Rating	Рейтинг	FLOAT	4	-
12	Pages	Кількість сторінок	INT	4	-
13	Price	Вартість	DECIMAL	16	-
14	QuantityInStock	Кількість на складі	INT	4	-
15	Ordered	Кількість замовлень	INT	4	-

Таблиця 4.7 – Таблиця «tableOrders»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	2	3	4	5	6
1	Id	Ідентифікатор видавництва	INT	4	+
2	UserId	Ідентифікатор читача	INT	4	-
3	CreatingDate	Дата створення	DATETIME	16	-
4	DeliveryDate	Дата доставки	DATETIME	16	-
5	DeliveryAddress	Адреса доставки	NVARCHAR	2147483646	-
6	Total	Вартість замовлення	DECIMAL	16	-
7	Status	Статус замовлення	INT	4	-

Таблиця 4.8 – Таблиця «tableOrderedBooks»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор видавництва	INT	4	+
2	OrderId	Ідентифікатор замовлення	INT	4	-
3	BookId	Ідентифікатор книги	INT	4	-
4	Quantity	Кількість екземплярів книги	INT	4	-

Висновки до розділу 4

В даному розділі було змодельовано компоненти розроблюваного веб-сайту, залежності та зв’язки між ними на фізичному рівні, розроблено базу даних та здійснено її нормалізацію та представлено у вигляді діаграми в нотації Crow’s Foot та на її основі було описано таблиці бази даних.

5 РОЗРОБКА ПРОГРАМИ

Розробляти веб-сайти з допомогою фреймворку ASP.NET MVC можна на різних мовах програмування, що входять в комплект .NET Framework (C#, Visual Basic, J# та інші) [4]. Для розробки даного проекту буде використовуватись C#, так як він є сучасною об'єктно-орієнтованою універсальною мовою програмування, яка підтримує багато можливостей CLR.

На даний момент ASP.NET є найсучаснішою технологією розробки веб-сайтів, за допомогою якої у світі реалізується більша частина веб-проектів.

Можливості фреймворку:

- засоби кешування, що збільшують продуктивність додатків;
- використання користувацьких елементів керування;
- вбудовані засоби зберігання даних сесії на сервері;
- технологія доступу до даних ADO.NET;
- відділення коду від візуальної частини;
- підтримка веб-сервісів;
- інше.

За замовчуванням представлення в ASP.NET MVC використовує сторінки Razor (.cshtml, .vbhtml) або Web Forms для розробки макету сторінок користувацького інтерфейсу. Razor дозволяє створювати динамічні веб-сторінки, інтегруючи серверний код в html-розмітку [5].

Веб-сайт створений за допомогою ASP.NET можна розмістити на хості в Windows за допомогою IIS та IIS Express. IIS — це гнучкий, безпечний і керований веб-сервер для розміщення веб-додатків, включаючи ASP.NET Core.

Декомпозицію задачі за допомогою методу покрокової деталізації представлено у вигляді нумерованого списку.

1. Режими доступу до веб-сайту.
    1.1. Доступ гостя.
        1.1.1. Layout.
            1.1.1.1. Панель навігації.
                1.1.1.1.1. Перехід на сторінку з підбірками книг.
                1.1.1.1.2. Введення пошукового запиту.
                1.1.1.1.3. Перехід на сторінку пошуку книг.
                1.1.1.1.4. Відображення кількості книг в кошику та їхньої вартості.
                1.1.1.1.5. Перехід до кошику користувача.
                1.1.1.1.6. Перехід на сторінку входу.
            1.1.1.2. Підвал сайту.
                1.1.1.2.1. Інформація про сайт.
                1.1.1.2.2. Контактні дані.
        1.1.2. Body.
            1.1.2.1. Сторінка підбірок книг.
                1.1.2.1.1. Перегляд підбірок.
                    1.1.2.1.1.1. Топ.
                    1.1.2.1.1.2. Найкращі рейтинги.
                    1.1.2.1.1.3. Нові.
                    1.1.2.1.1.4. Для дітей.
                1.1.2.1.2. Перехід на сторінку перегляду інформації про книгу.
                1.1.2.1.3. Додавання книги до кошика.
            1.1.2.2. Сторінка пошуку.
                1.1.2.2.1. Перегляд списку знайдених книг.
                1.1.2.2.2. Додавання книги до кошика.
                1.1.2.2.3. Перехід на сторінку інформації про книгу.
                1.1.2.2.4. Фільтрація знайдених книг.
                    1.1.2.2.4.1. За жанрами.
                    1.1.2.2.4.2. За ціновим діапазоном.
                1.1.2.2.5. Сортування знайдених книг.

1.1.2.2.5.1. За популярністю.

1.1.2.2.5.2. По рейтингу .

1.1.2.2.5.3. За роком видавництва.

1.1.2.2.5.4. Від дорожчих до дешевших.

1.1.2.2.5.5. Від дешевших до дорожчих.

1.1.2.2.6. Завантаження інших знайдених книг.

1.1.2.3. Сторінка інформації про книгу.

1.1.2.3.1. Перегляд повної інформації про книгу.

1.1.2.3.2. Додавання книги до кошика.

1.1.2.4. Сторінка кошика.

1.1.2.4.1. Відображення доданих книг.

1.1.2.4.2. Збільшення та зменшення кількості примірників книги.

1.1.2.4.3. Видалення книги з кошика.

1.1.2.4.4. Відображення загальної вартості замовлення.

1.1.2.4.5. Перехід до сторінки створення замовлення.

1.1.2.5. Сторінка входу.

1.1.2.5.1. Заповнення даних для входу.

1.1.2.5.1.1. Введення логіну.

1.1.2.5.1.2. Введення паролю.

1.1.2.5.2. Кнопка «Запам’ятати мене».

1.1.2.5.3. Вхід з переадресацією на сторінку підбірок книг.

1.1.2.5.4. Перехід на сторінку створення облікового запису.

1.1.2.6. Сторінка створення облікового запису.

1.1.2.6.1. Введення даних користувача.

1.1.2.6.2. Створення облікового запису.

1.2. Доступ читача.

1.2.1. Layout.

1.2.1.1. Панель навігації.

1.2.1.1.1. Перехід на сторінку з підбірками книг.

1.2.1.1.2. Введення пошукового запиту.

1.2.1.1.3. Перехід на сторінку пошуку книг.

1.2.1.1.4. Відображення кількості книг в кошику та їхньої вартості.

1.2.1.1.5. Перехід на сторінку історії замовлень користувача.

1.2.1.1.6. Перехід на сторінку роботи з обліковим записом користувача.

1.2.1.1.7. Перехід до кошику користувача.

1.2.1.1.8. Вихід із системи.

1.2.1.2. Підвал сайту.

1.2.1.2.1. Інформація про сайт.

1.2.1.2.2. Контактні дані.

1.2.2. Body.

1.2.2.1. Сторінка підбірок книг.

1.2.2.1.1. Перегляд підбірок.

1.2.2.1.1.1. Топ.

1.2.2.1.1.2. Найкращі рейтинги.

1.2.2.1.1.3. Нові.

1.2.2.1.1.4. Для дітей.

1.2.2.1.2. Перехід на сторінку перегляду інформації про книгу.

1.2.2.1.3. Додавання книги до кошика.

1.2.2.2. Сторінка пошуку.

1.2.2.2.1. Перегляд списку знайдених книг.

1.2.2.2.2. Додавання книги до кошика.

1.2.2.2.3. Перехід на сторінку інформації про книгу.

1.2.2.2.4. Фільтрація знайдених книг.

1.2.2.2.4.1. За жанрами.

1.2.2.2.4.2. За ціновим діапазоном.

1.2.2.2.5. Сортування знайдених книг.

1.2.2.2.5.1. За популярністю.

1.2.2.2.5.2. По рейтингу .

1.2.2.2.5.3. За роком видавництва.

1.2.2.2.5.4. Від дорожчих до дешевших.

1.2.2.2.5.5. Від дешевших до дорожчих.

1.2.2.2.6. Завантаження інших знайдених книг.

1.2.2.3. Сторінка інформації про книгу.

1.2.2.3.1. Перегляд повної інформації про книгу.

1.2.2.3.2. Додавання книги до кошика.

1.2.2.4. Сторінка кошика.

1.2.2.4.1. Відображення доданих книг.

1.2.2.4.2. Збільшення та зменшення кількості примірників книги.

1.2.2.4.3. Видалення книги з кошика.

1.2.2.4.4. Відображення загальної вартості замовлення.

1.2.2.4.5. Перехід до сторінки створення замовлення.

1.2.2.5. Сторінка створення замовлення.

1.2.2.5.1. Введення адреси доставки.

1.2.2.5.2. Відображення підсумків для кожної замовленої книги.

1.2.2.5.3. Відображення загальної вартості замовлених книг.

1.2.2.5.4. Створення замовлення.

1.2.2.6. Сторінка замовлень користувача.

1.2.2.6.1. Перегляд історії замовлень.

1.2.2.6.2. Перехід на сторінку інформації про замовлення.

1.2.2.7. Сторінка інформації про замовлення.

1.2.2.7.1. Перегляд інформації про замовлення.

1.2.2.7.2. Перегляд списку замовлених книг.

1.2.2.7.3. Перехід на сторінку інформації про книгу.

1.2.2.8. Сторінка роботи з профілем користувача.

1.2.2.8.1. Редагування профілю.

1.2.2.8.2. Збереження змін.

1.3. Доступ адміністратора (звичайний режим).

1.3.1. Layout.

1.3.1.1. Панель навігації.

1.3.1.1.1. Перехід до режиму адміністративної панелі.

1.3.1.1.2. Перехід на сторінку з підбірками книг.

1.3.1.1.3. Введення пошукового запиту.

1.3.1.1.4. Перехід на сторінку пошуку книг.

1.3.1.1.5. Вихід із системи.

1.3.2. Body.

1.3.2.1. Сторінка підбірок книг.

1.3.2.1.1. Перегляд підбірок.

1.3.2.1.1.1. Топ.

1.3.2.1.1.2. Найкращі рейтинги.

1.3.2.1.1.3. Нові.

1.3.2.1.1.4. Для дітей.

1.3.2.1.2. Перехід на сторінку перегляду інформації про книгу.

1.3.2.2. Сторінка пошуку.

1.3.2.2.1. Перегляд списку знайдених книг.

1.3.2.2.2. Перехід на сторінку інформації про книгу.

1.3.2.2.3. Фільтрація знайдених книг.

1.3.2.2.3.1. За жанрами.

1.3.2.2.3.2. За ціновим діапазоном.

1.3.2.2.4. Сортування знайдених книг.

1.3.2.2.4.1. За популярністю.

1.3.2.2.4.2. По рейтингу.

1.3.2.2.4.3. За роком видавництва.

1.3.2.2.4.4. Від дорожчих до дешевших.

1.3.2.2.4.5. Від дешевших до дорожчих.

1.3.2.2.5. Завантаження інших знайдених книг.

1.3.2.3. Сторінка інформації про книгу.

1.3.2.3.1. Перегляд повної інформації про книгу.

1.4. Доступ адміністратора (адміністративний режим).

1.4.1. Layout.

1.1.1.1. Перехід на сторінку адміністративної панелі.

1.1.1.2. Перехід на сторінку з підбірками книг (звичайний режим).

1.1.1.3. Перехід на сторінку роботи з замовленнями.

1.1.1.4. Перехід на сторінку роботи з книгами.

1.1.1.5. Перехід на сторінку роботи з авторами.

1.1.1.6. Перехід на сторінку роботи з жанрами.

1.1.1.7. Перехід на сторінку роботи з видавництвами.

1.1.1.8. Перехід на сторінку роботи з мовами.

1.1.2. Body.

1.1.2.1. Сторінка роботи з замовленнями.

1.1.2.1.1. Перегляд.

1.1.2.1.2. Перехід на сторінку перегляду замовлення.

1.1.2.2. Сторінка перегляду замовлення.

1.1.2.2.1. Перегляд відомостей про замовлення.

1.1.2.2.2. Перегляд списку замовлених книг.

1.1.2.2.3. Редагування статусу замовлення.

1.1.2.2.4. Редагування дати доставки.

1.1.2.3. Сторінка роботи з книгами.

1.1.2.3.1. Перегляд.

1.1.2.3.2. Фільтрація по жанрам.

1.1.2.3.3. Перехід на сторінку додавання книги.

1.1.2.3.4. Перехід на сторінку редагування книги.

1.1.2.3.5. Видалення.

1.1.2.3.6. Перехід на іншу сторінку книг.

1.1.2.4. Сторінка роботи з авторами.

1.1.2.4.1. Перегляд.

1.1.2.4.2. Додавання.

1.1.2.4.3. Редагування.

1.1.2.4.4. Видалення.

1.1.2.5. Сторінка роботи з жанрами.

1.1.2.5.1. Перегляд.

1.1.2.5.2. Додавання.

1.1.2.5.3. Редагування.

1.1.2.5.4. Видалення.

1.1.2.6. Сторінка роботи з видавництвами.

1.1.2.6.1. Перегляд.

1.1.2.6.2. Додавання.

1.1.2.6.3. Редагування.

1.1.2.6.4. Видалення.

1.1.2.7. Сторінка роботи з мовами.

1.1.2.7.1. Перегляд.

1.1.2.7.2. Додавання.

1.1.2.7.3. Редагування.

1.1.2.7.4. Видалення.

Всі сторінки веб-сайту мають схожий алгоритм роботи:

- клієнт формує HTTP-запит веб-сторінки та відправляє його на сервер;
- система маршрутизації на сервері аналізує отриманий URL-запит та обирає відповідний маршрут;
- в залежності від обраного маршруту, сервер передає керування відповідному контролеру;
- контролер викликає один зі своїх action-методів, при необхідності обробляє дані, генерує представлення та відправляє його клієнту;
- клієнт отримує веб-сторінку, сформовану сервером, та працює з нею.

Висновки до розділу 5

В даному розділі було:

- обґрунтовано вибір фреймворку та мови програмування;
- проведено покрокову деталізацію розроблюваного проекту та її представлення у вигляді нумерованого списку;
- описано загальний алгоритм роботи сторінок веб-сайту.

6 ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

Для перевірки програми на відповідність встановленим вимогам, зазначеним у специфікації, і виявлення дефектів, доречним буде проведення тестування за допомогою стратегії чорної скриньки, а саме методом припущення про помилку.

При використанні методу припущення про помилку тестувальник складає список можливих помилок та ситуацій, в яких можуть з’явитися ці помилки. Більшою мірою, метод опирається на власний досвід та інтуїцію тестувальника програми. Для виявлення та локалізації знайдених помилок будуть використані методи індукції та дедукції.

Для налагодження програми будуть застосовуватися такі засоби середовища розробки, як:

- покрокове виконання програми;
- точки зупину;
- перегляд значень змінних.

Сформовано наступні припущення про помилку:

1. Чи пройде пошта валідацію, якщо вона вже зайнята іншим користувачем?
2. Чи пройде пароль валідацію, якщо він не відповідає наступним критеріям: наявність як мінімум 1 малої та 1 великої латинської літери, 1 цифри, кількість символів — від 6 до 16 символів?
3. Чи буде очищено кошик читача після виходу з системи?
4. Чи може користувач замовити більше книг ніж є в наявності?
5. Чи може адміністратор ввести значення дати доставки замовлення менше ніж значення поточної дати?
6. Чи може адміністратор обрати постер, який вже використовується іншою книгою?
7. Чи може адміністратор отримати доступ до роботи з кошиком?
8. Що буде з книгами, після видалення жанру, мови, автора чи видавництва, до якого вони відносяться?

Протокол налагодження програми наведено в таблиці 6.1.

Таблиця 6.1 – Протокол налагодження програми

Опис помилки	Опис ситуації	Способи усунення	Дії, що були застосовані для усунення
1	2	3	4
Можна використати одну пошту в декількох облікових записів	Гість створює обліковий запис, увівши пошту, зайняту існуючим користувачем	Перевірка існування облікових записів в базі даних з введеною поштою	При створенні облікового запису відбувається пошук користувачів в базі даних, що мають таку саму пошту. Якщо пошту знайдено, показати користувачу відповідне повідомлення.
Можна ввести слабкий пароль	Гість створює обліковий запис, увівши слабкий пароль	Перевірка наявності 1 малої та великої латинських літер, цифр та чи знаходиться довжина паролю в межах від 6 до 16 символів	При створенні облікового запису за допомогою регулярного виразу @"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{6,16}$" відбувається перевірка коректності паролю. Якщо пароль не пройшов валідацію, показати користувачу відповідне повідомлення.
Кошик користувача не очищується після виходу з системи	Користувач здійснює вихід із системи	Очистити сесію за допомогою Session.Clear()	В методі виходу з системи здійснити виклик Session.Clear()

Продовження таблиці 6.1

1	2	3	4
Читач може замовити більше книг, ніж є на складі	Читач вводить більшу кількість книг, ніж є на складі натискає «Register Order»	Перевіряти, чи не є введена кількість примірників книги більшою ніж є на складі, та виводити відповідне повідомлення читачу	При створенні замовлення перевірити, чи не є введена кількість примірників книги більшою за їх кількість на складі. Показати користувачу відповідне повідомлення
Адміністратор може ввести в якості дати замовлення дату, яка вже минула	При редагуванні дати замовлення та натискає «Save changes»	Задати мінімальне значення для вибору дати рівним DateTime.Now	Для тега <input type=local-time> в представленні, що відповідає за вибір дати, задати властивість min= DateTime.Now
Адміністратор може обрати постер, що використовується іншою книгою	При збереженні змін під час додавання або редагування книги	Перевірка існування книги в базі даних з такою ж назвою постеру	При збереженні книги здійснюється пошук книг в базі даних, що мають таку ж назву постера. У випадку збігу, показати користувачу відповідне повідомлення
Адміністратор може перейти на сторінку кошика за допомогою URL-запиту: /basket/index	Адміністратор вводить URL-запит /basket/index та перенаправляється на сторінку роботи з кошиком	Для класу BasketController встановити атрибут [Authorize(Roles = "Reader")], а для його методів — атрибут [AllowAnonymous]	Доступ до контроллера кошика обмежується за допомогою атрибутів [Authorize(Roles = "Reader")] та [AllowAnonymous].

Продовження таблиці 6.1

1	2	3	4
Відмова роботи програми — в БД не існує запису про жанр, автора, видавництво або мову	Адміністратор здійснив видалення жанру, автору, видавництва або мови, але книги продовжують на них посилатися через зовнішні ключі	Вказати каскадне видалення записів таблиці книг, додавши до обмежень таблиці ON DELETE CASCADE для зовнішніх ключів	Таблицю книг в базі даних оновлено за допомогою запиту, в якому вказане каскадне видалення записів по зовнішнім ключам

Висновки до розділу 6

В даному розділі було обрано методи тестування та відлагодження веб-сайту. Складено протокол налагодження програмної системи, що містить помилки та спосіб їх виправлення.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

В дипломній роботі було проведено вивчення методів розробки веб-сайтів та отримано навички з їх розробки за допомогою фреймворку ASP.NET MVC 5.

Під час виконання дипломної роботи було проведено аналіз аналогічних розробок, спроектовано та реалізовано веб-сайт для продажу книг. Завдяки використанню патерну Model-View-Controller, веб-сайт має широкі можливості розширення та легкості подальшої підтримки.

В проекті використано наступні NuGet-пакети:

- Bootstrap 4.1.3 — фреймворк в CSS. Включає в себе шрифти і JavaScript. Є найпопулярнішим front-end фреймворком для розробки адаптивних, мобільних проектів в Інтернеті [6];
- EntityFramework 6.2.0 — технологія доступу до даних для нових додатків;
- jQuery 3.6.0 — це швидка і лаконічна бібліотека JavaScript, яка спрощує перегляд HTML-документів, обробку подій, анімацію і взаємодію з Ajax для швидкої веб-розробки;
- Microsoft.AspNet.Mvc 5.2.8 — цей пакет містить збірки середовища виконання для ASP.Net MVC. ASP.NET MVC надає вам потужний, заснований на шаблонах спосіб створення динамічних веб-сайтів, який забезпечує чітке розділення обов’язків і надає повний контроль над розміткою сторінок;
- Microsoft.AspNet.Razor 3.2.8 — цей пакет містить збірки середовища виконання для ASP.NET Web Pages. ASP.NET Web Pages та новий синтаксис Razor забезпечують швидкий, лаконічний, чистий і легкий спосіб об'єднання серверного коду з HTML для створення динамічного веб-контенту;
- PagedList — спрощує для. NET-розробників написання коду підкачки. Це дозволяє використовуючи будь-який список типу IEnumerable<T> і, вказавши розмір сторінки і бажаний Індекс сторінки, вибрати тільки деяку підмножину цього списку.

Веб-сайт повністю відповідає вимогам, описаним в розділі 1 пояснювальної записки. Експлуатаційне призначення веб-сайту досягнуте шляхом реалізації таких можливостей, як:

- управління контентом каталогу та робота з замовленнями;
- генерація підбірок книг;
- пошук книг по різним критеріям та їх замовлення;
- перегляд історії замовлень читача.

Всі ці можливості надають читачу змогу в режимі онлайн знаходити книги, що його цікавлять, та приймати рішення про їх замовлення, а продавцеві — змогу розміщувати книги на інформаційному ресурсі та здійснювати їх продаж.

В подальшому, веб-сайт може бути покращений за рахунок реалізації таких можливостей:

- підбірка «Знижки»;
- підбірка «Рекомендації», що заснована на історії замовлень читача;
- список бажаних книг;
- система відгуків читачів, яка надає можливість оцінювати книги та писати до них коментарі;
- можливість блокування читачів, які поводяться неприйнятно.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Основний функціонал сайту kupichitay.com.ua [Електронний ресурс]. Спеціалізований магазин IT-літератури. – Режим доступу: https://kupichitay.com.ua/ (дата звернення: 01.06.2022).
2. Основний функціонал сайту openlibrary.org [Електронний ресурс]. Відкритий бібліотечний каталог. – Режим доступу: https://openlibrary.org/ (дата звернення: 01.06.2022).
3. Основний функціонал сайту manybooks.net [Електронний ресурс]. Безкоштовна бібліотека в цифровому форматі. – Режим доступу: https://manybooks.net/ (дата звернення: 01.06.2022).
4. Можливості ASP.NET [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://ru.wikipedia.org/wiki/Data_mining (дата звернення: 01.06.2022).
5. Представлення в ASP.NET [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://docs.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/introduction/adding-a-view (дата звернення: 01.06.2022).
6. Опис Bootstrap 4.1.3 [Електронний ресурс]. Матеріал з NuGet. Безкоштовний пакетний менеджер. – Режим доступу: https://www.nuget.org /packages/bootstrap/4.1.3 (дата звернення: 01.06.2022).
7. Сандерсон, С. ASP.NET MVC 4 Framework з примерами на C# для профессионалов, 4-е видання [Текст]. – М.: Вильямс, 2017. – 99 с.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ
18.02.2022

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИЖОК

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01255-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
18.02.22

Керівник розробки
Олександр ІВАНОВ
18.02.2022

Виконавець
Андрій ТАНЦЮРА
18.02.2022

Нормоконтролер
Олена КУРОП'ЯТНИК
18.02.2022

2022

ЗАТВЕРДЖЕНО
44165850.01255-01-ЛЗ

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИЖОК

Технічне завдання

44165850.01255-01

Листів 16

2022

АНОТАЦІЯ

Документ 44165850.01255-01 «ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИЖОК. Технічне завдання» входить до складу програмної документації на програму, що представляє собою веб-сайт для пошуку та замовлення книг.

В даному документі описано призначення та область застосування програмного продукту, основні вимоги, стадії та строки виконання проекту.

ЗМІСТ

ВСТУП

У світі, де життя людей надзвичайно тісно пов’язане з комп’ютерними технологіями, все більше сфер комерційної діяльності переходять на автоматизовані засоби збору, збереження та обробки цифрових даних, адже автоматизація будь-якої системи — це насамперед підвищення продуктивності і ефективності праці, якості надання послуг, усунення рутинних та монотонних операцій, що позитивно позначається на розвитку бізнесу будь-якого масштабу — малого, середнього чи великого.

Створення інтернет-магазину дозволяє значно розширити коло потенціальних клієнтів, запропонувавши їм можливість дистанційного ознайомлення з запропонованими товарами та здійснення їх замовлення онлайн, що в значній мірі зменшує необхідний для цього час, адже не потрібно витрачати його на відвідування звичайних магазинів. Щоб цього досягти, сайт має бути інтуїтивно зрозумілим та звичним для більшості покупців, тобто бути «цифровим» аналогом реальних магазинів та торгівельних майданчиків — мати каталог, кошик, тощо.

Платформа .NET надає потужну бібліотеку класів, що дозволяє використовувати вже існуючу функціональність, а архітектурний шаблон MVC, який реалізує інфраструктура ASP.NET MVC — забезпечує розділення відповідальності між шарами програмного продукту, підвищуючи можливості його подальшого розширення.

ASP.NET MVC — це технологія від компанії Microsoft, надає можливість розробляти гнучкі та функціональні веб-сайти та веб-додатки, яка продовжує набувати все більшої популярності, адже може використовуватися побудови як простих, так і дуже складних сайтів.

1 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 08.12.21 №77 ст. ректора Українського державного університету науки і технологій «Про призначення керівників та затвердження тем бакалаврських робіт» за спеціальністю 121 «Інженерія програмного забезпечення» факультету «Комп’ютерних технологій і систем» кафедри «Комп’ютерні інформаційні технології».

Тема дипломної роботи — «Веб-портал ASP.NET продажу книжок». Керівник — доцент Іванов О. П.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення ПЗ — розробка інформаційного ресурсу для здійснення пошуку книг в каталозі магазину та оформлення замовлення.

Експлуатаційне призначення ПЗ — на даному інформаційному ресурсі продавець може розміщувати книги та обробляти замовлення, що поступають від клієнтів. Клієнти, в свою чергу, можуть здійснювати пошук книг в каталозі магазину по різним критеріям, що його цікавлять, переглядати повну інформацію про книги, замовляти їх та переглядати власну історію замовлень.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

В залежності від ролі користувача, система повинна надавати йому доступ до різної функціональні можливості.

Гість має такі можливості:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд книг у кошику;
- додавання та видалення книг з кошика;
- зміна кількості книг в кошику;
- створення облікового запису;
- авторизація.

Читач має такі можливості:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд книг у кошику;
- додавання та видалення книг з кошика;
- зміна кількості книг в кошику;
- створення замовлення з введенням адресу пошти;
- перегляд списку замовлень;
- перегляд детальної інформації про замовлення;
- редагування облікового запису;
- вихід з облікового запису.

Адміністратор має такі можливості:

- перегляд добірок книг: топ, кращі рейтинги, нові, дитячі книги;
- перегляд детальної інформації про книгу;
- пошук книг за ключовими словами з фільтрацією по жанрам, ціновим діапазонам та сортуванням результатів;
- перегляд всіх замовлень;
- перегляд деталей замовлення;
- редагування статусу замовлення та дати його доставки;
- перегляд книг з можливістю фільтрації по жанрам;
- додавання нової книги;
- редагування книги;
- видалення книги;
- робота з авторами, жанрами, мовами та видавництвами, передбачає наступні операції: перегляд, додавання, редагування, видалення.

Для опису взаємодії різних користувачів з програмною системою було розроблено специфікацію функціональних вимог у вигляді діаграм прецедентів для таких акторів, як: гість, читач, адміністратор.

Розроблену діаграму прецедентів для користувача в ролі «гість» зображено на рисунку 3.1.

Розроблену діаграму для користувача в ролі «читач» зображено на рисунку 3.2.

Розроблену діаграму для користувача в ролі «адміністратор» зображено на рисунку 3.3.

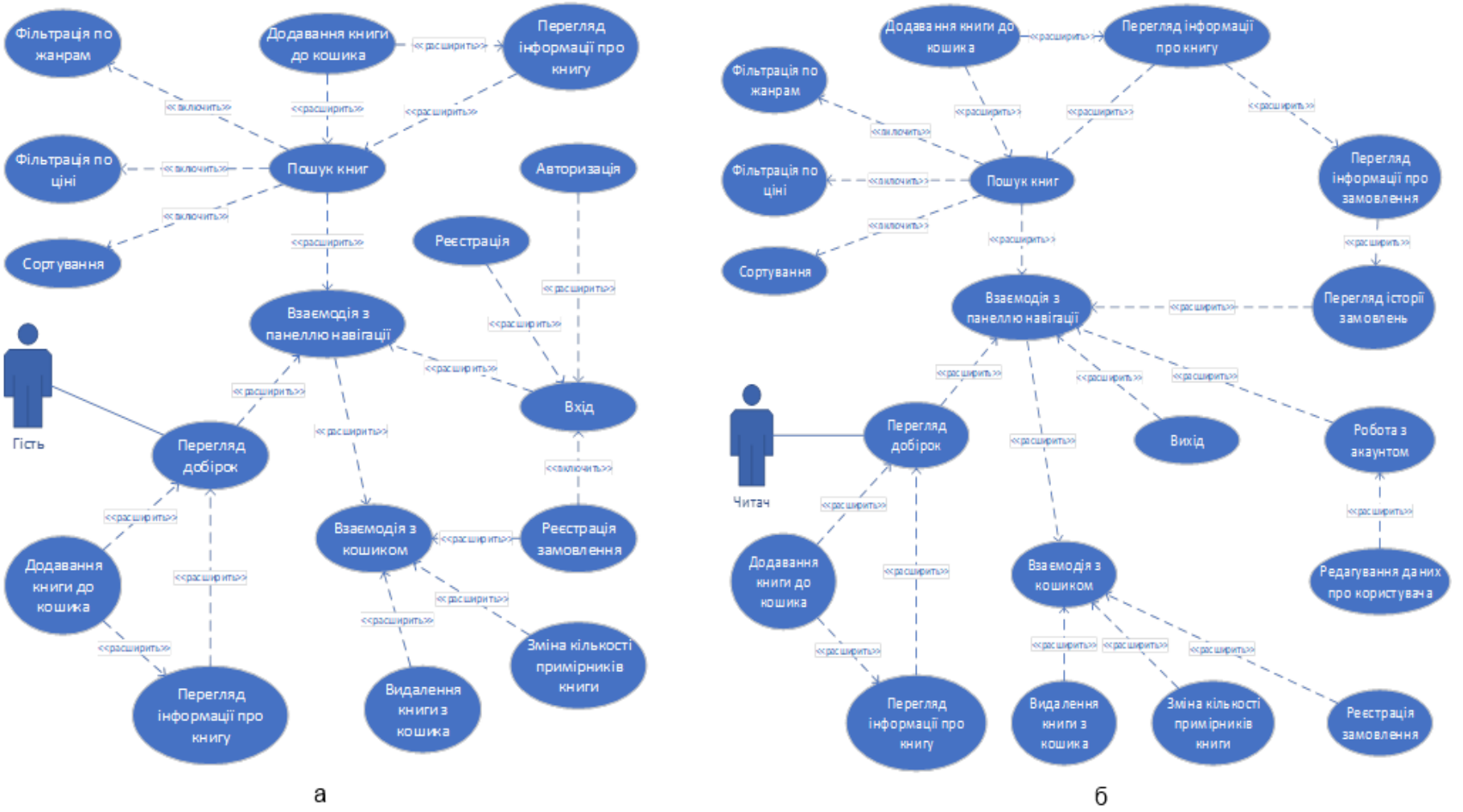

Рисунок 3.1 – Діаграма прецедентів:

а — для користувача в ролі «гість»; б — для користувача в ролі «читач»

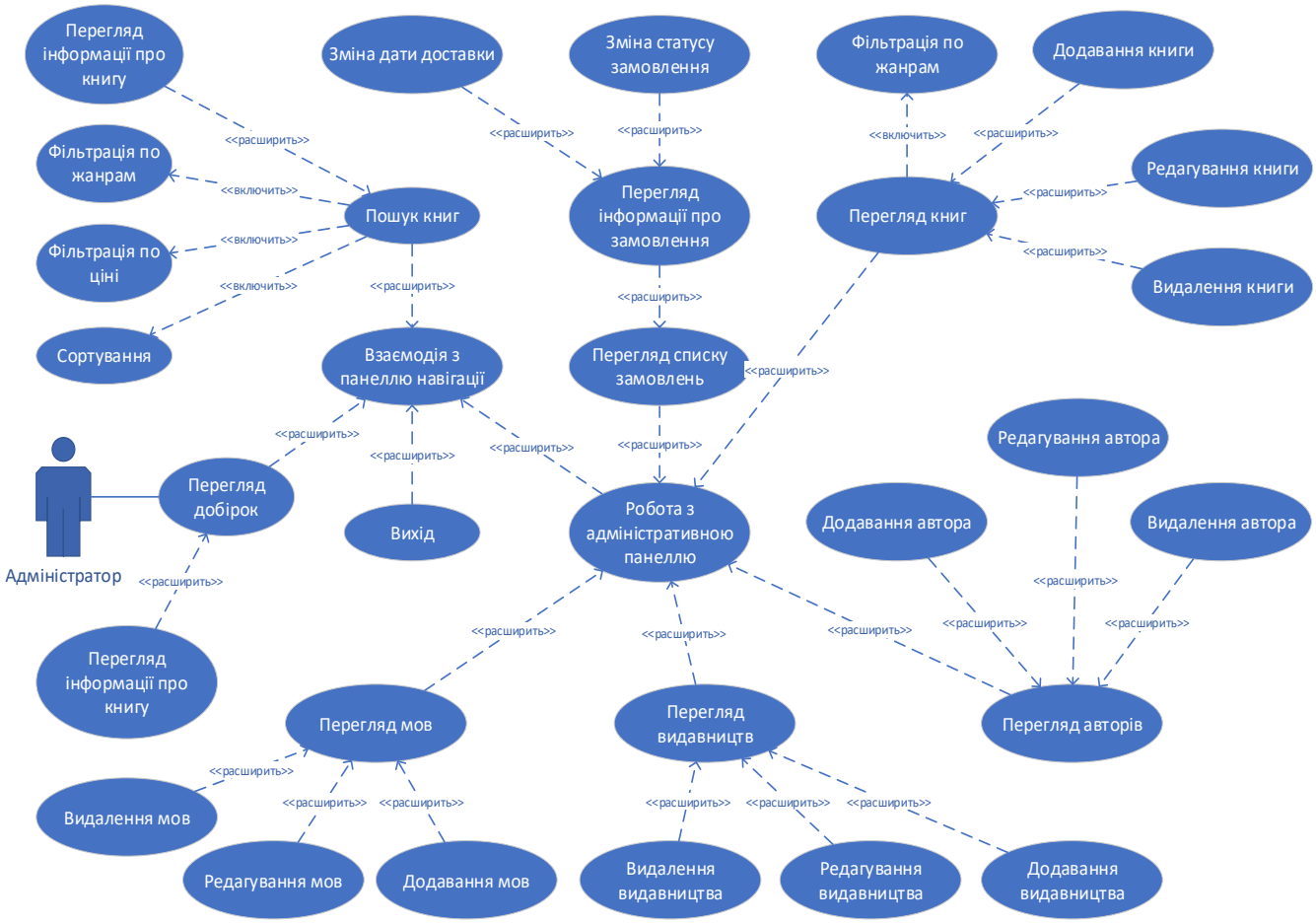

Рисунок 3.3 – Діаграма прецедентів для користувача в ролі «адміністратор»

3.2 Вимоги до надійності

Вимоги до надійності програмної системи наступні:

– система повинна здійснювати валідацію даних, введених користувачем;

– система повинна обмежувати доступ користувачів до бази даних в залежності від їх ролей;

– система повинна реагувати на запити користувача та надавати йому інформацію про результати їх виконання;

– наявність архівної копії вихідного тексту програми на зовнішньому носії;

– наявність резервної копії бази даних на зовнішньому носії.

3.3 Умови експлуатації

Для забезпечення стійкого функціонування програмного продукту на ЕОМ у приміщеннях, необхідно дотримуватись таких умов:

– допустима температура повітря — від +20° до +30° С,

– допустима відносна вологість повітря — від 40% до 60%;

– допустима швидкість руху повітря — до 0.1 м/с;

– забезпечення автономної роботи ЕОМ у випадку нестачі струму;

– користувач повинен мати базові навички роботи з ЕОМ та веб-браузером;

– користувач, що працює з ЕОМ, повинен пройти відповідний інструктаж по техніці безпеки.

3.4 Вимоги до складу та параметрів технічних засобів

Розроблюваний програмний продукт розрахований для використання на ЕОМ з наступними характеристики:

– наявність веб-браузеру;

– діагональ екрану — 15.5 дюймів;

– частота процесору — від 1 ГГц та вище;

– оперативна пам’ять — 512 Мб;

– пам'ять дискового накопичувача — 1 Гб.

Сервер, на якому буде розгорнуто програмний продукт, повинен мати наступні характеристики:

– частота процесору — від 2 ГГц та вище;

– оперативна пам’ять — 8 Гб;

– пам'ять дискового накопичувача — 2 Гб.

3.5 Вимоги до інформаційної та програмної сумісності

Програмний продукт розробляється всіх видів ОС, що підтримують роботу з веб-браузером.

Середовище розробки: Microsoft Visual Studio 2019 Community. Мова програмування: C#, платформа розробки: ASP.NET MVC 5, СУБД: Microsoft SQL Server, веб-браузер для відлагодження — Microsoft Edge v.102.0.1245.33.

3.6 Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від механічних, кліматичних та інших видів пошкоджень. На упаковці

повинна бути вказана назва продукту, номер версії (якщо вона змінювалась), мінімальні системні вимоги. На зворотній стороні упаковки вказується розробник та його юридична адреса.

Приклад маркування приведений на рисунку 3.4.

Програмний додаток «ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИЖОК» Мінімальні системні вимоги ЕОМ: – дігональ екрану: 15.5 дюймів; – частота процессору: від 1 ГГц; – ОЗУ: 512 Мб; – пам'ять на диску 1 Гб. Мінімальні системні вимоги серверу: – частота процессору: від 2 ГГц та вище; – ОЗУ: 8 Гб; – пам'ять на диску: 2 Гб.	Розробник: Танцюра А. А. Кафедра «КІТ», УДУНТ м. Дніпро, вул. Лазаряна 2 2022

Рисунок 3.4 – Приклад маркування

3.7. Вимоги до транспортування та зберігання

Транспортування повинне забезпечувати збереження програмного продукту, його цілісність і запобігання несанкціонованого доступу до нього. Програмний виріб міститься на оптичному носії даних типу CD-R і повинно мати відповідну упаковку для захисту від механічних ушкоджень та атмосферного впливу (пластиковий футляр або паперовий конверт).

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Документація до програмного продукту повинна задовольняти вимогам державного стандарту до оформлення програмної документації [1].

До складу програмної документації має входити технічне завдання та робочий проект.

До складу робочого проекту мають входити:

– специфікація;

– текст програми;

– опис програми.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки програмного продукту приведені в таблиці 5.1.

Таблиця 5.1 – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задач, збір інформації, вибір та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	24.02.22 – 18.03.22
Робочий проект	Програмування та відлагодження програми.	21.03.22 – 23.05.22
	Тестування програми.	20.05.22 – 27.05.22
	Розробка, узгодження і затвердження програмної документації.	28.05.22 – 13.06.22

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Контроль здійснюється шляхом тестування програмного продукту з метою виявлення помилок, відмов та їх усунення у процесі відлагодження програми.

Контроль за виконання роботи здійснює керівник розробки.

Прийом програмного продукту здійснюється уповноваженою університетом комісією.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ
10.06.2022

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИЖОК

Специфікація
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01255-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
10.06.22

Керівник розробки
Олександр ІВАНОВ
10.06.22

Виконавець
Андрій ТАНЦЮРА
10.06.22

Нормоконтролер
Олена КУРОП'ЯТНИК
10.06.22

2022

ЗАТВЕРДЖЕНО
44165850.01255-01-ЛЗ

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИЖОК

Специфікація

44165850.01255-01

Листів 2

2022

Специфікацію приведено в таблиці 1.

Таблиця 1 – Специфікація

Позначення	Найменування	Примітка
44165850.01255-01-ЛЗ	Лист затвердження	
44165850.01255-01 12 01-ЛЗ	Лист затвердження	
44165850.01255-01 12 01	Текст програми	
44165850.01255-01 13 01-ЛЗ	Лист затвердження	
44165850.01255-01 13 01	Опис програми	

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ
10.06.2022

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИГ

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01255-01 12 01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
10.06.2022

Керівник розробки
Олександр ІВАНОВ
10.06.2022

Виконавець
Андрій ТАНЦЮРА
10.06.2022

Нормоконтролер
Олена КУРОП'ЯТНИК
10.06.2022

2022

ЗАТВЕРДЖЕНО
44165850.01255-01 12 01-ЛЗ

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИГ

Текст програми

44165850.01255-01 12 01

Листів 29

2022

АНОТАЦІЯ

Документ 44165850.01255-01 12 01 «ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИГ. Текст програми» входить до складу програмної документації на програму, що представляє собою веб-сайт для пошуку та замовлення книг.

В даному документі описано взаємодію модулів програми та її вихідний код на мові C#.

ЗМІСТ

1 СХЕМА ВЗАЄМОДІЇ МОДУЛІВ

На рисунках 1.1 – 1.10 приведені схеми взаємодії модулів програми.

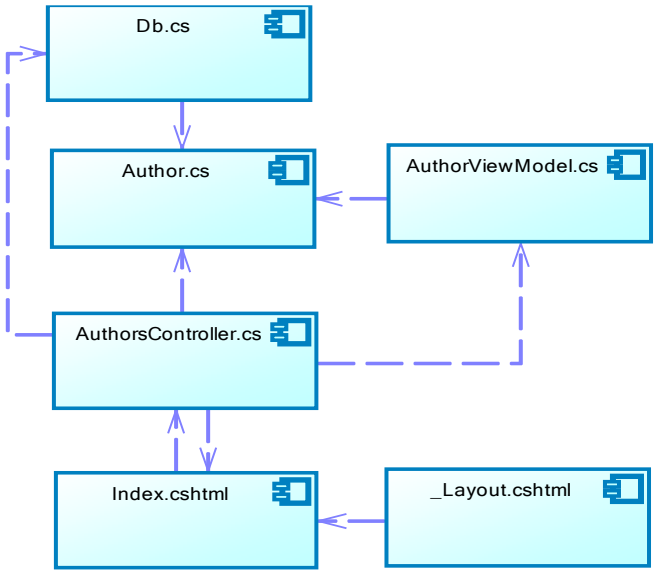

Рисунок 1.1 — Діаграма компонентів, в основі якої лежить контролер таблиці авторів

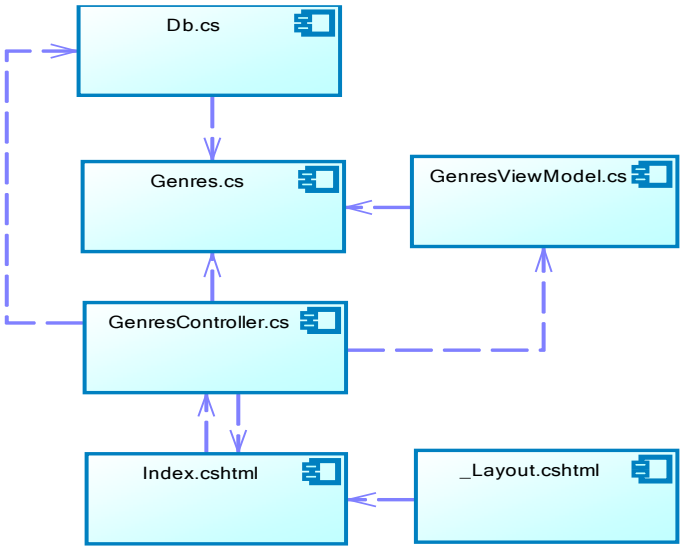

Рисунок 1.2 — Діаграма компонентів, в основі якої лежить контролер таблиці жанрів

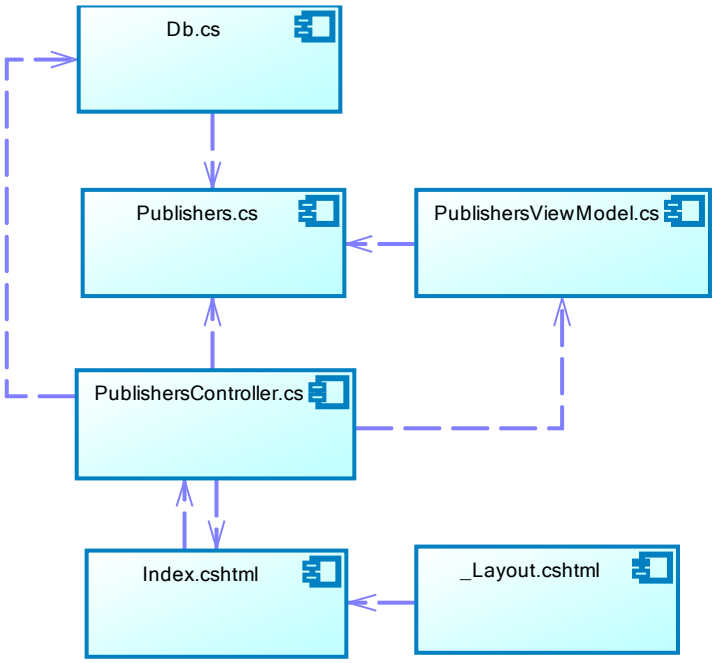

Рисунок 1.3 — Діаграма компонентів, в основі якої лежить контролер таблиці видавництв

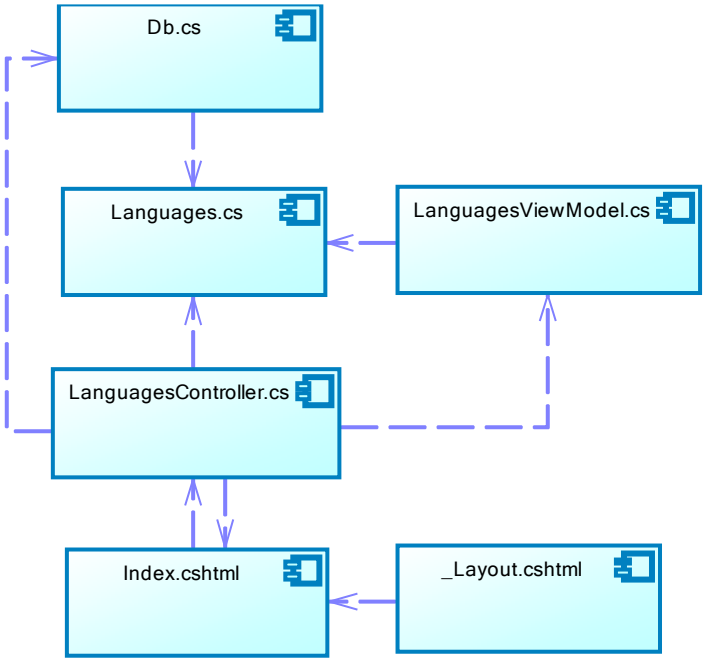

Рисунок 1.4 — Діаграма компонентів, в основі якої лежить контролер таблиці мов

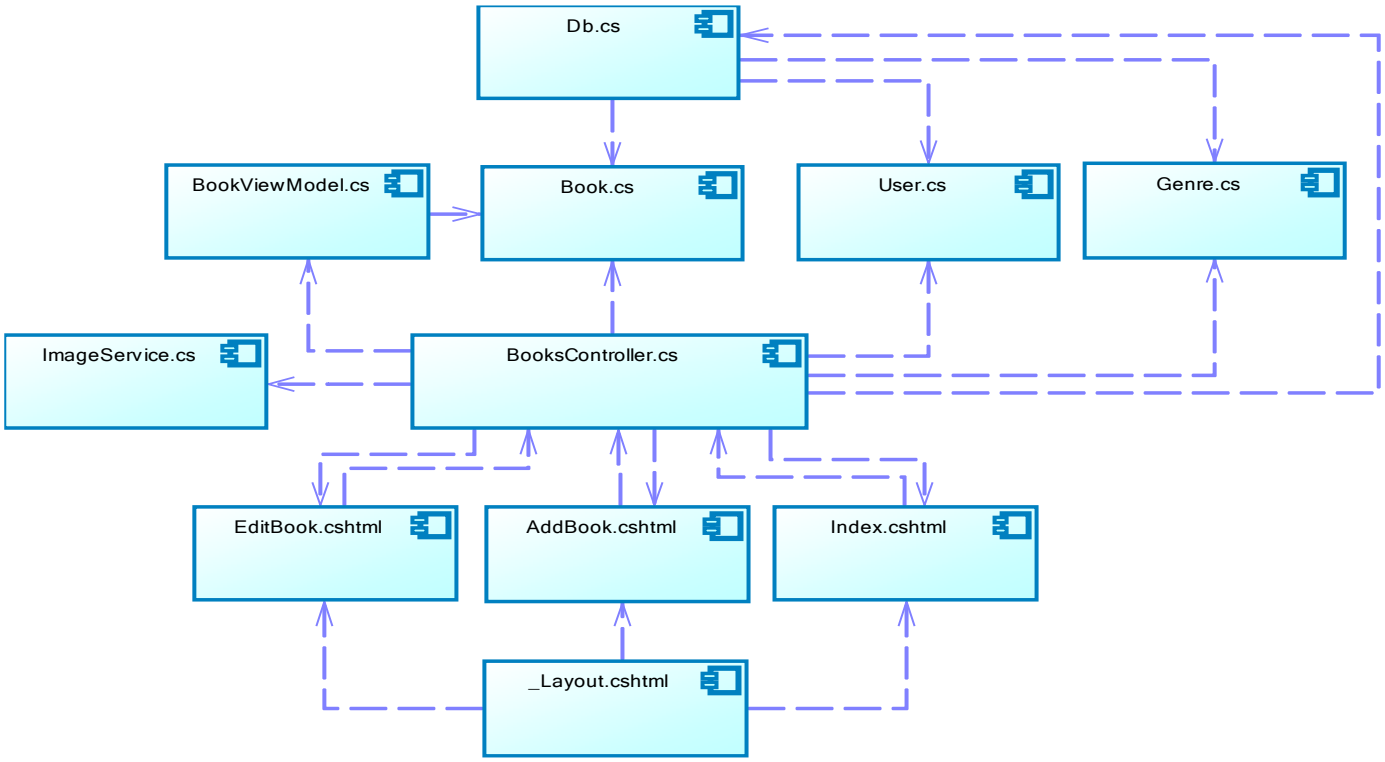

Рисунок 1.5 — Діаграма компонентів, в основі якої лежить контролер таблиці книг

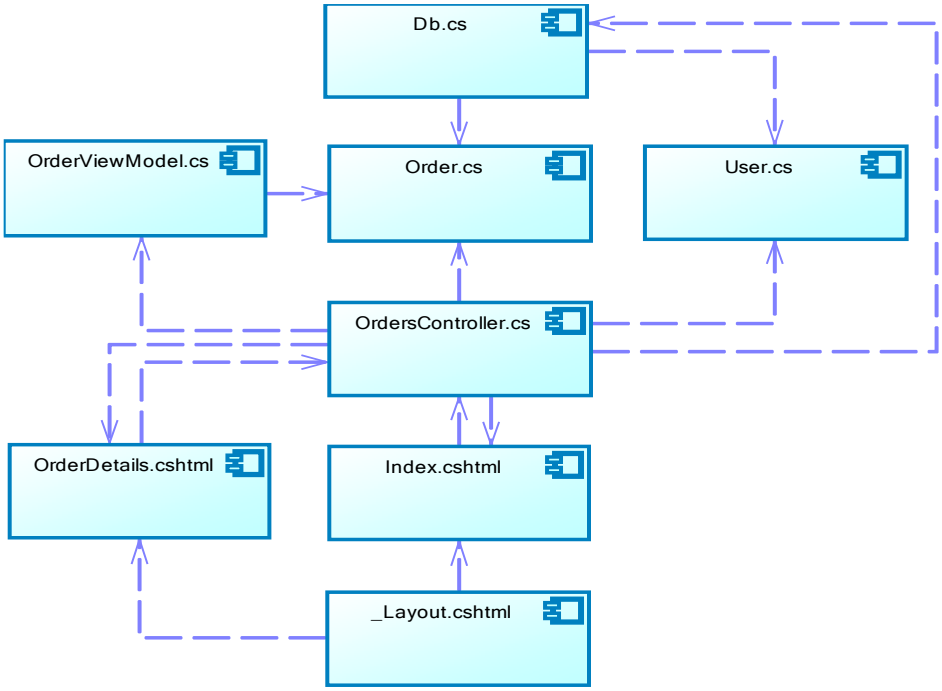

Рисунок 1.6 — Діаграма компонентів, в основі якої лежить контролер таблиці замовлень (в області адміністратора)

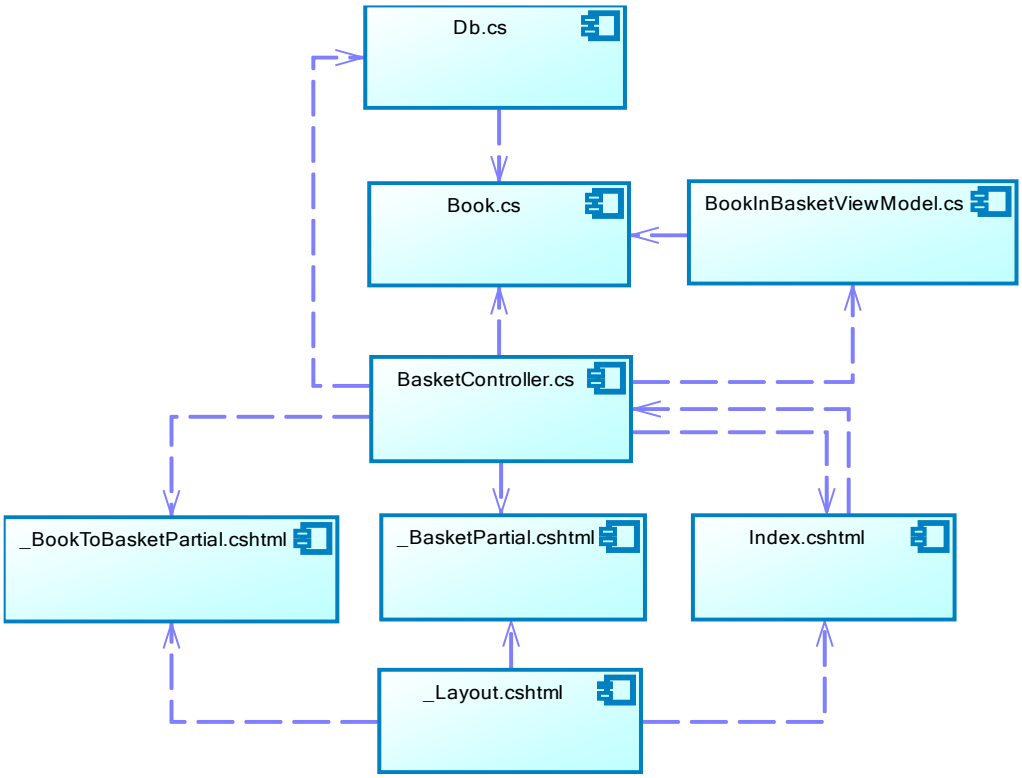

Рисунок 1.7 — Діаграма компонентів, в основі якої лежить контролер кошика покупця

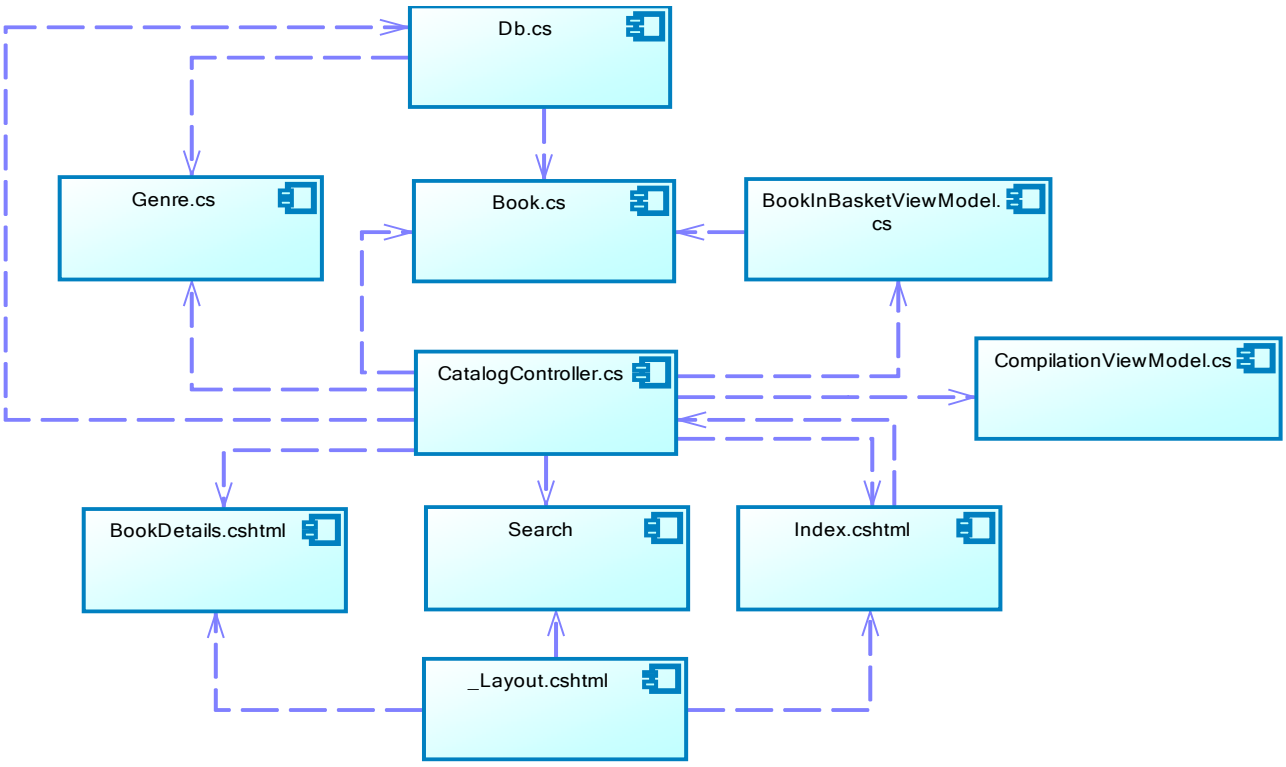

Рисунок 1.8 — Діаграма компонентів, в основі якої лежить контролер книг в каталозі

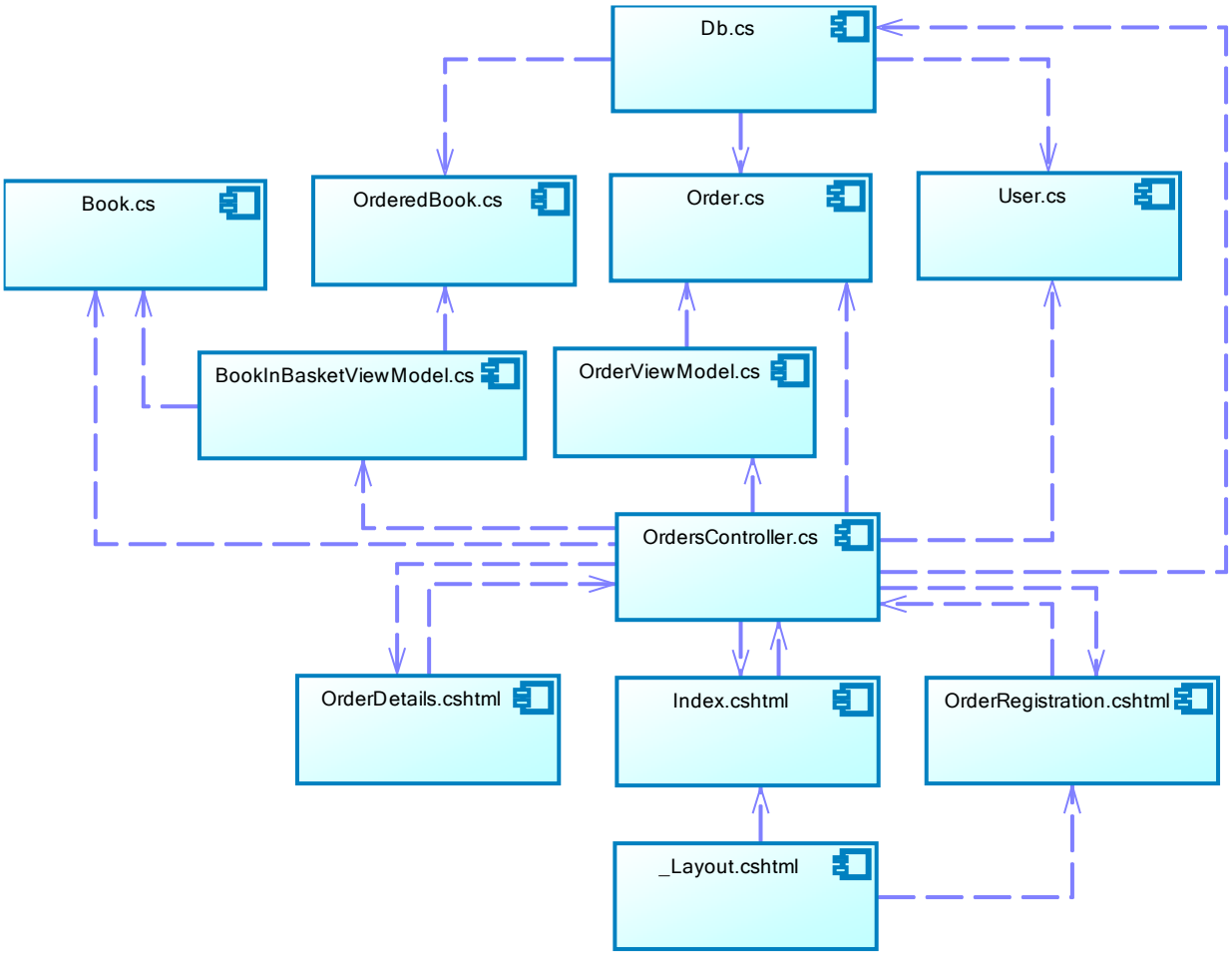

Рисунок 1.9 — Діаграма компонентів, в основі якої лежить контролер таблиці замовлень (в області користувача)

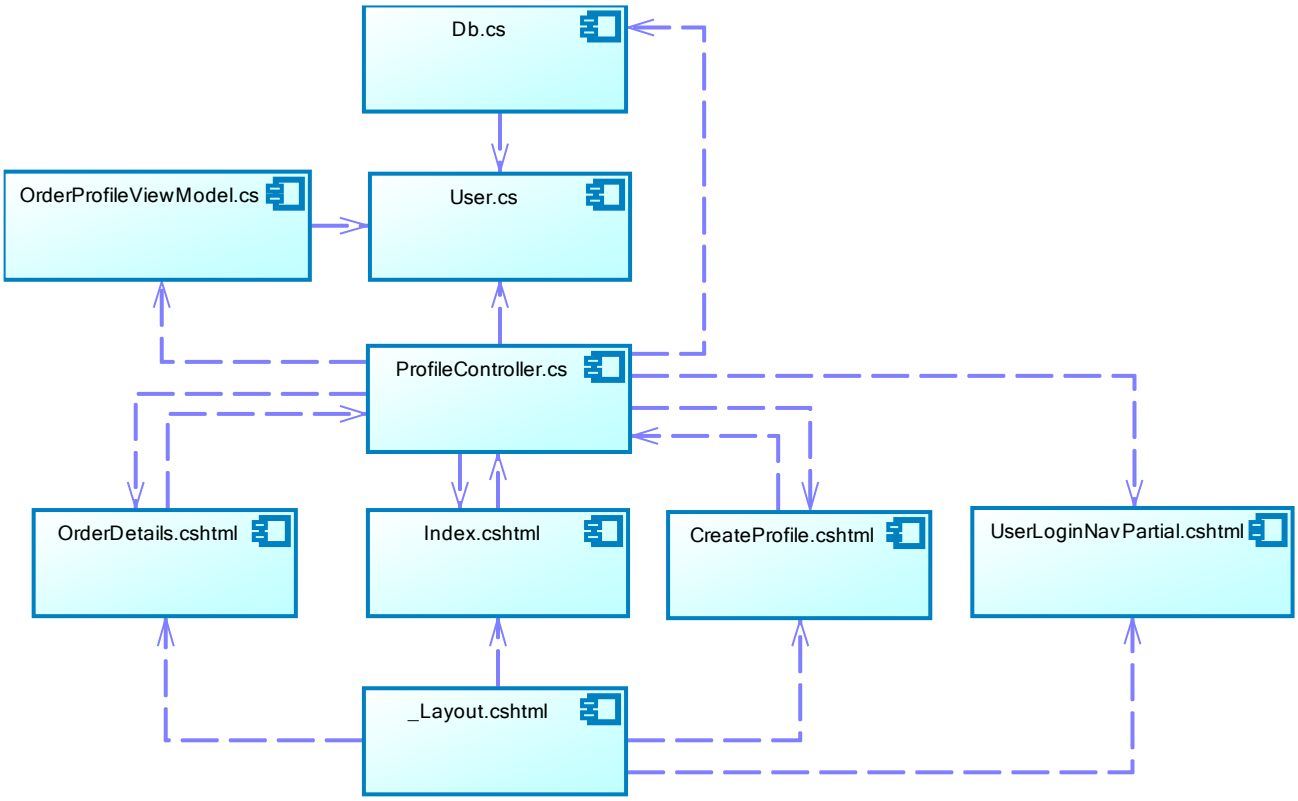

Рисунок 1.10 — Діаграма компонентів, в основі якої лежить контролер профілю користувача

2 ТЕКСТ ПРОГРАМИ

2.1 Модуль AuthorsController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Areas.Admin.Controllers
{
    [Authorize(Roles = Constants.Roles.ADMIN)]
    public class AuthorsController : Controller
    {
        // GET: Admin/Authors
        public ActionResult Index()
        {
            var authors = Enumerable.Empty<AuthorViewModel>();

            using (Db db = new())
            {
                authors = db.Authors
                    .ToArray()
                    .OrderBy(x => x.Name)
                    .Select(x => new AuthorViewModel(x))
                    .ToList();
            }

            return View(authors);
        }

        //POST: Admin/Authors/Create
        [HttpPost]
        public string Create(string authorName)
        {
            string id = string.Empty;

            using (Db db = new())
            {
                if (db.Authors.Any(x => x.Name == authorName))
                {
                    return "authorAlreadyExist";
                }

                Author author = new() { Name = authorName };

                db.Authors.Add(author);
                db.SaveChanges();

                id = author.Id.ToString();
            }

            return id;
        }

        // GET: Admin/Authors/Remove/Id
        [HttpGet]
        public ActionResult Remove(int id)
        {
            using (Db db = new())
            {
                Author author = db.Authors.Find(id);

                if (author is null)
                {
                    return Content("The author is not exist.");
                }

                db.Authors.Remove(author);
                db.SaveChanges();
            }

            TempData[Constants.Keys.SUCCESS] = "You have deleted a author";

            return RedirectToAction(nameof(Index));
        }
```

```
        // Admin/Authors/Rename
        [HttpPost]
        public string Rename(string newAuthorName, int id)
        {
            using (Db db = new())
            {
                if (db.Authors.Any(x => x.Name == newAuthorName))
                {
                    return "authorAlreadyExist";
                }

                Author author = db.Authors.Find(id);
                author.Name = newAuthorName;

                db.SaveChanges();
            }

            return Constants.Keys.SUCCESS;
        }
    }
}
```

2.2 Модуль BooksController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using BookCatalog.Extensions;
using BookCatalog.Services;
using PagedList;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookCatalog.Areas.Admin.Controllers
{
    [Authorize(Roles = Constants.Roles.ADMIN)]
    public class BooksController : Controller
    {
        private readonly ImagesService _imagesService = new ();

        #region -- Public helpers --

        // GET: Admin/Books/BookList
        [HttpGet]
        public ActionResult Index(int? page, int? genreId)
        {
            List<BookViewModel> booksViewModel;

            using (Db db = new())
            {
                booksViewModel = db.Books
                    .ToArray()
                    .Where(x => genreId is null || genreId == 0 || x.GenreId == genreId)
                    .Select(x => new BookViewModel(x))
                    .ToList();

                ViewBag.Genres = new SelectList(db.Genres.ToList(), "Id", "Name");
                ViewBag.SelectedGenre = genreId.ToString();
                ViewBag.DisplayedBooks = booksViewModel.ToPagedList(page ?? 1,
Constants.Limits.BOOKS_PER_PAGE);
            }

            return View(booksViewModel);
        }

        // GET: Admin/Books/AddBook
        [HttpGet]
        public ActionResult AddBook()
        {
            BookViewModel book = new();

            InitSelectonLists(book);

            return View(book);
        }
```

```
        // POST: Admin/Books/AddBook
        [HttpPost]
        public ActionResult AddBook(BookViewModel bookViewModel, HttpPostedFileBase file)
        {
            InitSelectonLists(bookViewModel);

            if (!ModelState.IsValid)
            {
                return View(bookViewModel);
            }

            if (file is not null && file.ContentLength > 0)
            {
                using (Db db = new())
                {
                    if (db.Books.Any(x => x.Title == bookViewModel.Title))
                    {
                        ModelState.AddModelError(string.Empty, "A book with this title already exists!");

                        return View(bookViewModel);
                    }

                    if (db.Books.Any(x => x.PosterName == file.FileName))
                    {
                        ModelState.AddModelError(string.Empty, "This poster is already used in another
book!");

                        return View(bookViewModel);
                    }
                }

                bool isExtensionAccepted = _imagesService.CheckImageExtension(file.ContentType);

                if (!isExtensionAccepted)
                {
                    InitSelectonLists(bookViewModel);

                    ModelState.AddModelError(string.Empty, "The poster was not uploaded - incorrect image
extension!");

                    return View(bookViewModel);
                }

                ImagesService.OriginalDirectory ??= new
DirectoryInfo(string.Format($"{Server.MapPath(@"\")}Images\\Uploads")).ToString();

                _imagesService.Save(file);

                using (Db db = new())
                {
                    bookViewModel.PosterName = file.FileName;
                    Book newBook = bookViewModel.ToModel();
                    db.Books.Add(newBook);
                    db.SaveChanges();
                }

                TempData[Constants.Keys.SUCCESS] = "You have added a book!";
            }
            else
            {
                ModelState.AddModelError(string.Empty, "The poster is a required field!");
                return View(bookViewModel);
            }

            return RedirectToAction(nameof(AddBook));
        }

        // GET: Admin/Books/EditBook/id
        [HttpGet]
        public ActionResult EditBook(int bookId)
        {
            BookViewModel bookViewModel;

            using (Db db = new())
            {
                Book book = db.Books.Find(bookId);

                if (book is null)
```

```
                {
                    return Content("This book does not exist.");
                }

                bookViewModel = new (book);

                InitSelectonLists(bookViewModel);
            }

            return View(bookViewModel);
        }

        // POST: Admin/Books/EditBook
        [HttpPost]
        public ActionResult EditBook(BookViewModel bookViewModel, HttpPostedFileBase file)
        {
            InitSelectonLists(bookViewModel);

            if (!ModelState.IsValid)
            {
                return View(bookViewModel);
            }

            using (Db db = new())
            {
                if (db.Books.Any(x => x.Id != bookViewModel.Id && x.Title == bookViewModel.Title))
                {
                    ModelState.AddModelError(string.Empty, "A book with this title already exists!");

                    return View(bookViewModel);
                }

                if (file is not null && db.Books.Any(x => x.Id != bookViewModel.Id && x.PosterName ==
file.FileName))
                {
                    ModelState.AddModelError(string.Empty, "This poster is already used in another
book!");

                    return View(bookViewModel);
                }

                if (file is not null && file.ContentLength > 0)
                {
                    bool isExtensionAccepted = _imagesService.CheckImageExtension(file.ContentType);

                    if (!isExtensionAccepted)
                    {
                        ModelState.AddModelError(string.Empty, "The poster was not uploaded - incorrect
image extension!");

                        return View(bookViewModel);
                    }

                    ImagesService.OriginalDirectory ??= new
DirectoryInfo(string.Format($"{Server.MapPath(@"\")}Images\\Uploads")).ToString();

                    _imagesService.RemovePoster(bookViewModel.PosterName);
                    _imagesService.Save(file);

                    bookViewModel.PosterName = file.FileName;
                }

                Book bookFromDb = db.Books.Find(bookViewModel.Id);
                bookFromDb.CopyFields(bookViewModel);

                db.SaveChanges();

                TempData[Constants.Keys.SUCCESS] = "You edited the book!";
            }

            return RedirectToAction(nameof(EditBook), new { bookId = bookViewModel.Id });
        }

        // GET: Admin/Books/DeleteBook/bookId
        [HttpGet]
        public ActionResult DeleteBook(int bookId)
        {
            using (Db db = new())
            {
```

```
                Book book = db.Books.Find(bookId);

                if (book is null)
                {
                    return HttpNotFound("Book not found.");
                }

                db.Books.Remove(book);
                db.SaveChanges();

                ImagesService.OriginalDirectory ??= new
DirectoryInfo(string.Format($"{Server.MapPath(@"\")}Images\\Uploads")).ToString();
                _imagesService.RemovePoster(book.PosterName);
            }

            return RedirectToAction(nameof(Index));
        }

        #endregion

        #region -- Private helpers --

        private void InitSelectonLists(BookViewModel book)
        {
            using (Db db = new())
            {
                book.Genres = new SelectList(db.Genres.ToList(), "Id", "Name");
                book.Authors = new SelectList(db.Authors.ToList(), "Id", "Name");
                book.Publishers = new SelectList(db.Publishers.ToList(), "Id", "Name");
                book.Languages = new SelectList(db.Languages.ToList(), "Id", "Name");
            }
        }

        #endregion
    }
}
```

2.3 Модуль GenresController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Areas.Admin.Controllers
{
    [Authorize(Roles = Constants.Roles.ADMIN)]
    public class GenresController : Controller
    {
        // GET: Admin/Genres
        public ActionResult Index()
        {
            var genres = Enumerable.Empty<GenreViewModel>();

            using (Db db = new())
            {
                genres = db.Genres
                    .ToArray()
                    .OrderBy(x => x.Name)
                    .Select(x => new GenreViewModel(x))
                    .ToList();
            }

            return View(genres);
        }

        //POST: Admin/Genres/Create
        [HttpPost]
        public string Create(string genreName)
        {
            string id = string.Empty;

            using (Db db = new ())
            {
                if (db.Genres.Any(x => x.Name == genreName))
                {
                    return "genreAlreadyExist";
```

```
                }

                Genre genre = new() { Name = genreName };

                db.Genres.Add(genre);
                db.SaveChanges();

                id = genre.Id.ToString();
            }

            return id;
        }

        // GET: Admin/Genres/Remove/Id
        [HttpGet]
        public ActionResult Remove(int id)
        {
            using (Db db = new ())
            {
                Genre genre = db.Genres.Find(id);

                if (genre == null)
                {
                    return Content("The genre is not exist.");
                }

                db.Genres.Remove(genre);
                db.SaveChanges();
            }

            TempData[Constants.Keys.SUCCESS] = "You have deleted a genre.";

            return RedirectToAction(nameof(Index));
        }

        // Admin/Genres/Rename
        [HttpPost]
        public string Rename(string newGenreName, int id)
        {
            using (Db db = new ())
            {
                if (db.Genres.Any(x => x.Name == newGenreName))
                {
                    return "genreAlreadyExist";
                }

                Genre genre = db.Genres.Find(id);
                genre.Name = newGenreName;

                db.SaveChanges();
            }

            return Constants.Keys.SUCCESS;
        }
    }
}
```

2.4 Модуль LanguagesController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Areas.Admin.Controllers
{
    [Authorize(Roles = Constants.Roles.ADMIN)]
    public class LanguagesController : Controller
    {
        // GET: Admin/Languages
        public ActionResult Index()
        {
            var categories = Enumerable.Empty<LanguageViewModel>();

            using (Db db = new())
            {
                categories = db.Languages
```

```
                .ToArray()
                .OrderBy(x => x.Name)
                .Select(x => new LanguageViewModel(x))
                .ToList();
        }

        return View(categories);
    }

    //POST: Admin/Languages/Create
    [HttpPost]
    public string Create(string languageName)
    {
        string id = string.Empty;

        using (Db db = new ())
        {
            if (db.Languages.Any(x => x.Name == languageName))
            {
                return "languageAlreadyExist";
            }

            Language language = new() { Name = languageName };

            db.Languages.Add(language);
            db.SaveChanges();

            id = language.Id.ToString();
        }

        return id;
    }

    // GET: Admin/Languages/Remove/Id
    [HttpGet]
    public ActionResult Remove(int id)
    {
        using (Db db = new ())
        {
            Language language = db.Languages.Find(id);

            if (language == null)
            {
                return Content("The language is not exist.");
            }

            db.Languages.Remove(language);
            db.SaveChanges();
        }

        TempData[Constants.Keys.SUCCESS] = "You have deleted a language.";

        return RedirectToAction(nameof(Index));
    }

    // Admin/Languages/Rename
    [HttpPost]
    public string Rename(string newLanguageName, int id)
    {
        using (Db db = new ())
        {
            if (db.Languages.Any(x => x.Name == newLanguageName))
            {
                return "languageAlreadyExist";
            }

            Language language = db.Languages.Find(id);
            language.Name = newLanguageName;

            db.SaveChanges();
        }

        return Constants.Keys.SUCCESS;
    }
}
}
```

2.5 Модуль OrdersController.cs

```
using BookCatalog.Areas.Admin.Data.ViewModels;
using BookCatalog.Data.Models;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Areas.Admin.Controllers
{
    [Authorize(Roles = Constants.Roles.ADMIN)]
    public class OrdersController : Controller
    {
        // GET: Admin/Orders
        [HttpGet]
        public ActionResult Index()
        {
            List<OrderViewModel> orders = new();

            using (Db db = new())
            {
                orders = db.Orders
                    .ToArray()
                    .OrderByDescending(x => x.CreatingDate)
                    .Select(x => new OrderViewModel(x))
                    .ToList();

                foreach (var order in orders)
                {
                    var user = db.Users.FirstOrDefault(x => x.Id == order.UserId);

                    order.UserLogin = user.Login;
                    order.Email = user.Email;
                }
            }

            return View(orders);
        }

        // GET: Admin/OrderDetails/id
        [HttpGet]
        public ActionResult OrderDetails(int id)
        {
            OrderViewModel orderViewModel = new();
            using (Db db = new())
            {
                if (!db.Orders.Any(x => x.Id == id))
                {
                    return Content("Order not found");
                }

                Order order = db.Orders.Find(id);
                orderViewModel = new(order);

                List<BookCatalog.Data.ViewModels.BookInBasketViewModel> orderedBooks =
db.OrderedBooks.Where(x => x.OrderId == id)
                    .ToArray()
                    .Select(x => new BookCatalog.Data.ViewModels.BookInBasketViewModel(x))
                    .ToList();

                TempData[Constants.Keys.ORDERED_BOOKS] = orderedBooks;
            }

            return View(orderViewModel);
        }

        // POST: Admin/OrderDetails/id
        [HttpPost]
        public ActionResult OrderDetails(OrderViewModel orderViewModel)
        {
            using (Db db = new())
            {
                Order order = db.Orders.Find(orderViewModel.Id);

                order.Status = orderViewModel.Status;
                order.DeliveryDate = orderViewModel.DeliveryDate;
                orderViewModel = new(order);
```

```
                db.SaveChanges();

                List<BookCatalog.Data.ViewModels.BookInBasketViewModel> orderedBooks =
db.OrderedBooks.Where(x => x.OrderId == order.Id)
                    .ToArray()
                    .Select(x => new BookCatalog.Data.ViewModels.BookInBasketViewModel(x))
                    .ToList();

                TempData[Constants.Keys.ORDERED_BOOKS] = orderedBooks;
                TempData[Constants.Keys.SUCCESS] = "You have edited the order.";
            }

            return View(orderViewModel);
        }
    }
}
```

2.6 Модуль OrdersController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Areas.Admin.Controllers
{
    [Authorize(Roles = Constants.Roles.ADMIN)]
    public class PublishersController : Controller
    {
        // GET: Admin/Publishers/Index
        public ActionResult Index()
        {
            var categories = Enumerable.Empty<PublisherViewModel>();

            using (Db db = new())
            {
                categories = db.Publishers
                    .ToArray()
                    .OrderBy(x => x.Name)
                    .Select(x => new PublisherViewModel(x))
                    .ToList();
            }

            return View(categories);
        }

        //POST: Admin/Publishers/Create
        [HttpPost]
        public string Create(string publisherName)
        {
            string id = string.Empty;

            using (Db db = new ())
            {
                if (db.Publishers.Any(x => x.Name == publisherName))
                {
                    return "publisherAlreadyExist";
                }

                Publisher publisher = new() { Name = publisherName };

                db.Publishers.Add(publisher);
                db.SaveChanges();

                id = publisher.Id.ToString();
            }

            return id;
        }

        // GET: Admin/Publishers/Remove/Id
        [HttpGet]
        public ActionResult Remove(int id)
        {
            using (Db db = new ())
            {
                Publisher publisher = db.Publishers.Find(id);
```

```
                if (publisher == null)
                {
                    return Content("The publisher is not exist.");
                }

                db.Publishers.Remove(publisher);
                db.SaveChanges();
            }

            TempData[Constants.Keys.SUCCESS] = "You have deleted a publisher.";

            return RedirectToAction(nameof(Index));
        }

        // Admin/Publishers/Rename
        [HttpPost]
        public string Rename(string newPublisherName, int id)
        {
            using (Db db = new ())
            {
                if (db.Publishers.Any(x => x.Name == newPublisherName))
                {
                    return "publisherAlreadyExist";
                }
                Publisher publisher = db.Publishers.Find(id);
                publisher.Name = newPublisherName;

                db.SaveChanges();
            }

            return Constants.Keys.SUCCESS;
        }
    }
}
```

2.7 Модуль BasketController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Controllers
{
    [Authorize(Roles = Constants.Roles.READER)]
    public class BasketController : Controller
    {
        // GET: Basket/Index
        [HttpGet]
        [AllowAnonymous]
        public ActionResult Index()
        {
            var basket = Session[Constants.Keys.BASKET] as List<BookInBasketViewModel> ?? new
List<BookInBasketViewModel>();

            if (!basket.Any() || Session[Constants.Keys.BASKET] is null)
            {
                ViewBag.Message = "Your basket is empty.";
                return View();
            }

            ViewBag.Summary = basket.Sum(x => x.Total);

            return View(basket);
        }

        [AllowAnonymous]
        public ActionResult BasketPartial()
        {
            BookInBasketViewModel booksInBasket = new ();

            if (Session[Constants.Keys.BASKET] is not null)
            {
                List<BookInBasketViewModel> books = Session[Constants.Keys.BASKET] as
List<BookInBasketViewModel>;
```

```
                booksInBasket.Quantity = books.Sum(x => x.Quantity);
                booksInBasket.Price = books.Sum(x => x.Quantity * x.Price);
            }
            else
            {
                booksInBasket.Quantity = 0;
                booksInBasket.Price = 0m;
            }

            return PartialView($"_{nameof(BasketPartial)}", booksInBasket);
        }

        [AllowAnonymous]
        public ActionResult AddToBasketPartial(int id)
        {
            List<BookInBasketViewModel> updatedBasket = Session[Constants.Keys.BASKET] as
List<BookInBasketViewModel> ?? new List<BookInBasketViewModel>();

            using (Db db = new())
            {
                BookInBasketViewModel bookInBasket = updatedBasket.FirstOrDefault(x => x.BookId == id);

                if (bookInBasket is null)
                {
                    Book book = db.Books.Find(id);
                    updatedBasket.Add(new BookInBasketViewModel(book));
                }
                else
                {
                    bookInBasket.Quantity++;
                }
            }

            BookInBasketViewModel basket = new()
            {
                Quantity = updatedBasket.Sum(x => x.Quantity),
                Price = updatedBasket.Sum(x => x.Quantity * x.Price),
            };

            Session[Constants.Keys.BASKET] = updatedBasket;

            return PartialView("_BookToBasketPartial", basket);
        }

        // GET: /Basket/IncreaseBooksNumber
        [HttpGet]
        [AllowAnonymous]
        public JsonResult IncreaseBooksNumber(int bookId)
        {
            List<BookInBasketViewModel> booksInBasket = Session[Constants.Keys.BASKET] as
List<BookInBasketViewModel>;

            using (Db db = new ())
            {
                BookInBasketViewModel book = booksInBasket.FirstOrDefault(x => x.BookId == bookId);
                book.Quantity++;

                var quantityPricePair = new
                {
                    quantity = book.Quantity,
                    price = book.Price
                };

                return Json(quantityPricePair, JsonRequestBehavior.AllowGet);
            }
        }

        // GET: /Basket/DecreaseBooksNumber/id
        [AllowAnonymous]
        public ActionResult DecreaseBooksNumber(int bookId)
        {
            List<BookInBasketViewModel> basket = Session[Constants.Keys.BASKET] as
List<BookInBasketViewModel>;

            using (Db db = new())
            {
                BookInBasketViewModel model = basket.FirstOrDefault(x => x.BookId == bookId);
```

```
                if (model.Quantity > 1)
                {
                    model.Quantity--;
                }
                else
                {
                    model.Quantity = 0;
                    basket.Remove(model);
                }

                var quantityPricePair = new
                {
                    quantity = model.Quantity,
                    price = model.Price
                };

                return Json(quantityPricePair, JsonRequestBehavior.AllowGet);
            }
        }

        // GET: /Basket/RemoveBook/bookId
        [AllowAnonymous]
        public void RemoveBook(int bookId)
        {
            using (Db db = new ())
            {
                List<BookInBasketViewModel> booksInBasket = Session[Constants.Keys.BASKET] as
List<BookInBasketViewModel>;
                BookInBasketViewModel book = booksInBasket.FirstOrDefault(x => x.BookId == bookId);

                if (book is not null)
                {
                    booksInBasket.Remove(book);
                }
            }
        }
    }
}
```

2.8 Модуль CatalogController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using BookCatalog.Enums;
using PagedList;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Controllers
{
    public class CatalogController : Controller
    {
        // GET: Admin/Catalog
        [HttpGet]
        public ActionResult Index()
        {
            List<CompilationViewModel> compilations = new();

            using (Db db = new())
            {
                List<BookViewModel> books = db.Books
                    .ToArray()
                    .OrderBy(x => x.Title)
                    .Select(x => new BookViewModel(x))
                    .ToList();

                compilations = new List<CompilationViewModel>
                {
                    new CompilationViewModel
                    {
                        Name = "Top",
                        Books = books
                            .OrderByDescending(x => x.Ordered)
                            .Take(Constants.Limits.BOOKS_IN_COMPILATION).ToList()
                    },
```

```
                    new CompilationViewModel
                    {
                        Name = "Best Ratings",
                        Books = books
                            .OrderByDescending(x => x.Rating)
                            .Take(Constants.Limits.BOOKS_IN_COMPILATION).ToList()
                    },
                    new CompilationViewModel
                    {
                        Name = "New",
                        Books = books
                            .OrderByDescending(x => x.PublicationYear)
                            .Take(Constants.Limits.BOOKS_IN_COMPILATION).ToList()
                    },
                    new CompilationViewModel
                    {
                        Name = "Kids",
                        Books = books.Where(x => x.AgeRating is Enums.EAgeRating.Any or
Enums.EAgeRating.Six)
                            .OrderByDescending(x => x.AgeRating)
                            .Take(Constants.Limits.BOOKS_IN_COMPILATION).ToList()
                    },
                };
            }

            return View(compilations);
        }

        // GET: Catalog/BookDetails/id
        [HttpGet]
        public ActionResult BookDetails(int id)
        {
            BookViewModel bookViewModel;

            using (Db db = new())
            {
                Book book = db.Books.FirstOrDefault(x => x.Id == id);

                if (book is null)
                {
                    return Content("Book not found.");
                }

                bookViewModel = new BookViewModel(book);
            }

            return View(nameof(BookDetails), bookViewModel);
        }

        // GET: Catalog/BookDetails/Search/age & searchQuery & sorting & selectedGenres
        [HttpGet]
        public ActionResult Search(int? page, string searchQuery, EBookSorting? sorting, int[]
selectedGenres, int[] priceRange)
        {
            using (Db db = new())
            {
                ViewBag.Genres = db.Genres.ToList();
            }

            ViewBag.SearchQuery = searchQuery;
            ViewBag.Sortings ??= new
List<EBookSorting>(Enum.GetValues(typeof(EBookSorting)).Cast<EBookSorting>());

            // если сортировка не передается извне, берем из сессии, если в сессии тоже нет - по умолчанию
            sorting = GetSelectedSorting(sorting);

            selectedGenres = GetSelectedGenres(selectedGenres);

            // TO DO: save and load range from session
            if (priceRange is null)
            {

            }
            else
            {

            }

            List<BookViewModel> books = new();
```

```
            Session[Constants.Keys.SORTING] ??= EBookSorting.Rating;

            var nextPage = page ?? 1;

            using (Db db = new())
            {
                books = db.Books
                    .ToArray()
                    .OrderBy(x => x.Title)
                    .Select(x => new BookViewModel(x))
                    .ToList();
            }

            if (sorting != null)
            {
                Func<BookViewModel, object> sortingSelector = sorting switch
                {
                    EBookSorting.Rating => x => x.Rating,
                    EBookSorting.Popularity => x => x.Title,
                    EBookSorting.New => x => x.PublicationYear,
                    EBookSorting.Cheap or EBookSorting.Expensive => x => x.Price,
                    _ => x => x.Title,
                };

                var sortedBooks = (sorting == EBookSorting.Cheap
                    ? books.OrderBy(sortingSelector)
                    : books.OrderByDescending(sortingSelector));

                books = new(sortedBooks);
            }

            string[] keyWords = searchQuery?.Split(' ', '\t');

            var foundedBooks = keyWords?.Count() > 0
                ? books.Where(x => keyWords.Any(k =>
                   x.Title.IndexOf(k, StringComparison.OrdinalIgnoreCase) > -1 ||
                   x.Description.IndexOf(k, StringComparison.OrdinalIgnoreCase) > -1 ||
                   x.AuthorName.IndexOf(k, StringComparison.OrdinalIgnoreCase) > -1))
                : books;

            if (selectedGenres is not null && selectedGenres.Length != 1)
            {
                foundedBooks = foundedBooks.Where(x => selectedGenres.Contains(x.GenreId));
            }

            if (priceRange is not null && priceRange[0] < priceRange[1])
            {
                foundedBooks = foundedBooks.Where(x => x.Price >= priceRange[0] && x.Price <=
priceRange[1]);
            }

            ViewBag.DisplayedBooks = foundedBooks.ToPagedList(nextPage,
Constants.Limits.BOOKS_PER_SEARCH_PAGE);

            if (!foundedBooks.Any())
            {
                TempData[Constants.Keys.INFO] = "Nothing found.";
            }

            return View(foundedBooks);
        }

        #region -- Private helpers --

        private int[] GetSelectedGenres(int[] selectedGenres)
        {
            if ((Session[Constants.Keys.SELECTED_GENRES] as int[])?.Length > 0)
            {
                if (selectedGenres is null)
                {
                    selectedGenres = (Session[Constants.Keys.SELECTED_GENRES] as int[]);
                }
                else
                {
                    Session[Constants.Keys.SELECTED_GENRES] = selectedGenres;
                }
            }
            else
```

```
            {
                Session[Constants.Keys.SELECTED_GENRES] = selectedGenres ?? new int[0];
            }

            return selectedGenres;
        }

        private EBookSorting? GetSelectedSorting(EBookSorting? sorting)
        {
            if (sorting is null)
            {
                if (Session[Constants.Keys.SELECTED_SORT] is null)
                {
                    Session[Constants.Keys.SELECTED_SORT] = sorting = EBookSorting.Popularity;
                }
                else
                {
                    sorting = Session[Constants.Keys.SELECTED_SORT] as EBookSorting?;
                }
            }
            else
            {
                Session[Constants.Keys.SELECTED_SORT] = sorting;
            }

            return sorting;
        }

        #endregion
    }
}
```

2.9 Модуль UserOrdersController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace BookCatalog.Controllers
{
    [Authorize(Roles = Constants.Roles.READER)]
    public class OrdersController : Controller
    {
        // GET: /Orders/
        [HttpGet]
        public ActionResult Index()
        {
            List<OrderViewModel> orders = new();

            using (Db db = new())
            {
                var currentUser = db.Users.FirstOrDefault(x => x.Login == User.Identity.Name);

                orders = db.Orders.Where(x => x.UserId == currentUser.Id)
                    .ToArray()
                    .OrderByDescending(x => x.CreatingDate)
                    .Select(x => new OrderViewModel(x))
                    .ToList();
            }

            return View(orders);
        }

        // GET: /Orders/OrderDetails/id
        public ActionResult OrderDetails(int id)
        {
            List<BookInBasketViewModel> orderedBooks = new();

            using (Db db = new())
            {
                if (!db.Orders.Any(x => x.Id == id))
                {
                    return Content("Order not found.");
                }
```

```
            orderedBooks = db.OrderedBooks.Where(x => x.OrderId == id)
                .ToArray()
                .Select(x => new BookInBasketViewModel(x))
                .ToList();

            TempData[Constants.Keys.ORDER] = db.Orders.Find(id);
        }

        return View(orderedBooks);
    }

    // GET: /Basket/Order
    [HttpGet]
    public ActionResult OrderRegistration()
    {
        var basket = Session[Constants.Keys.BASKET] as List<BookInBasketViewModel> ?? new
List<BookInBasketViewModel>();

        if (!basket.Any() || Session[Constants.Keys.BASKET] is null)
        {
            ViewBag.Message = "Your basket is empty.";
            return RedirectToAction(nameof(Index));
        }

        bool isEnoughBooksAvailable = true;

        using (Db db = new())
        {
            foreach (BookInBasketViewModel book in basket)
            {
                Book bookInDb = db.Books.Find(book.BookId);

                if (bookInDb.QuantityInStock == 0 || bookInDb.QuantityInStock < book?.Quantity)
                {
                    isEnoughBooksAvailable = false;
                    TempData[$"{Constants.Keys.DANGER}{bookInDb.Id}"] = $"Available in stock:
{bookInDb.QuantityInStock}";
                }
            }
        }

        if (!isEnoughBooksAvailable)
        {
            return RedirectToAction(nameof(Index), "Basket");
        }

        OrderViewModel orderViewModel = new();

        InitNewOrder(basket, orderViewModel);

        return View(orderViewModel);
    }

    // GET: /Basket/Order
    [HttpPost]
    public ActionResult OrderRegistration(OrderViewModel orderViewModel)
    {
        var basket = Session[Constants.Keys.BASKET] as List<BookInBasketViewModel> ?? new
List<BookInBasketViewModel>();

        InitNewOrder(basket, orderViewModel);

        if (!ModelState.IsValid)
        {
            return View(orderViewModel);
        }

        using (Db db = new())
        {
            string userLogin = User.Identity.Name;
            User user = db.Users.FirstOrDefault(x => x.Login.Equals(userLogin));

            Order newOrder = new()
            {
                UserId = user.Id,
                DeliveryAddress = orderViewModel.DeliveryAddress,
                Total = basket.Sum(x => x.Total)
            };
```

```
                db.Orders.Add(newOrder);
                db.SaveChanges();

                foreach (BookInBasketViewModel book in basket)
                {
                    OrderedBook orderedBook = new()
                    {
                        OrderId = newOrder.Id,
                        BookId = book.BookId,
                        Quantity = book.Quantity,
                    };

                    db.OrderedBooks.Add(orderedBook);

                    Book bookInDb = db.Books.Find(book.BookId);
                    bookInDb.Ordered += book.Quantity;
                    bookInDb.QuantityInStock -= book.Quantity;

                    db.SaveChanges();
                }
            }

            Session.Remove(Constants.Keys.BASKET);

            TempData[Constants.Keys.INFO] = "Your order is being processed!";

            return RedirectToAction(nameof(Index), "Basket");
        }

        #region -- Private helpers --

        private void InitNewOrder(List<BookInBasketViewModel> basket, OrderViewModel orderViewModel)
        {
            using (Db db = new())
            {
                var user = db.Users.FirstOrDefault(x => x.Login == User.Identity.Name);
                orderViewModel.UserId = user.Id;
                orderViewModel.Books = new(basket);
                orderViewModel.Total = basket.Sum(x => x.Total);
            }
        }

        #endregion
    }
}
```

2.10 Модуль ProfileController.cs

```
using BookCatalog.Data.Models;
using BookCatalog.Data.ViewModels;
using BookCatalog.Enums;
using BookCatalog.Extensions;
using System.Linq;
using System.Web.Mvc;
using System.Web.Security;

namespace BookCatalog.Controllers
{
    public class ProfileController : Controller
    {
        // GET: Profile/
        public ActionResult Index() => RedirectToAction(nameof(Login));


        // GET: Profile/Login
        [HttpGet]
        public ActionResult Login()
        {
            string userName = User.Identity.Name;

            if (!string.IsNullOrEmpty(userName))
            {
                return User.IsInRole(Constants.Roles.ADMIN)
                    ? RedirectToAction("Index", "Catalog")
                    : RedirectToAction(nameof(EditUser));
            }
```

```
            return View();
        }

        // POST: Profile/Login
        [HttpPost]
        public ActionResult Login(LoginViewModel loginViewModel)
        {
            if (!ModelState.IsValid)
            {
                return View(loginViewModel);
            }

            using (Db db = new())
            {
                if (!db.Users.Any(x =>
                    x.Login.Equals(loginViewModel.Login) &&
                    x.Password.Equals(loginViewModel.Password)))
                {
                    ModelState.AddModelError(string.Empty, "Invalid login or password.");

                    return View(loginViewModel);
                }
            }

            FormsAuthentication.SetAuthCookie(loginViewModel.Login, loginViewModel.RememberMe);

            return Redirect(FormsAuthentication.GetRedirectUrl(loginViewModel.Login,
loginViewModel.RememberMe));
        }

        // GET: Profile/Logout
        [Authorize]
        [HttpGet]
        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            ViewData.Clear();
            Session.Clear();

            return RedirectToAction(nameof(Login));
        }

        // GET: Profile/CreateUser
        [HttpGet]
        public ActionResult CreateUser() => View(nameof(CreateUser));

        // POST: Profile/CreateUser
        [HttpPost]
        public ActionResult CreateUser(UserViewModel userViewModel)
        {
            if (!ModelState.IsValid)
            {
                return View(nameof(CreateUser), userViewModel);
            }

            using (Db db = new())
            {
                if (db.Users.Any(x => x.Login.Equals(userViewModel.Login)))
                {
                    ModelState.AddModelError(string.Empty, $"Login {userViewModel.Login} is taken.");
                    userViewModel.Login = string.Empty;

                    return View(nameof(CreateUser), userViewModel);
                }

                if (string.IsNullOrWhiteSpace(userViewModel.Password))
                {
                    ModelState.AddModelError(string.Empty, $"Password is required field.");
                    return View(nameof(CreateUser), userViewModel);
                }

                if (string.IsNullOrWhiteSpace(userViewModel.ConfirmPassword))
                {
                    ModelState.AddModelError(string.Empty, $"Confirm password is required field.");
                    return View(nameof(CreateUser), userViewModel);
                }

                if (db.Users.Any(x => x.Email.Equals(userViewModel.Email)))
                {
```

```
                ModelState.AddModelError(string.Empty, $"Email {userViewModel.Email} is taken.");
                userViewModel.Email = string.Empty;

                return View(nameof(CreateUser), userViewModel);
            }

            User newUser = userViewModel.ToModel();

            newUser.Role = !db.Users.Any()
                ? ERole.Admin
                : ERole.Reader;

            db.Users.Add(newUser);
            db.SaveChanges();
        }

        TempData[Constants.Keys.SUCCESS] = "Now you are registered and can log in.";

        return RedirectToAction(nameof(Login));
    }

    [Authorize(Roles = Constants.Roles.READER)]
    public ActionResult UserLoginNavPartial()
    {
        UserLoginNavPartial userViewModel;

        using (Db db = new())
        {
            User user = db.Users.FirstOrDefault(x => x.Login == User.Identity.Name);

            userViewModel = new()
            {
                Login = user.Login
            };
        }

        return PartialView(userViewModel);
    }

    // GET: /Profile/EditUser
    [Authorize(Roles = Constants.Roles.READER)]
    public ActionResult EditUser()
    {
        UserViewModel userViewModel;

        using (Db db = new())
        {
            User user = db.Users.FirstOrDefault(x => x.Login == User.Identity.Name);

            userViewModel = new UserViewModel(user);
        }

        return View(nameof(EditUser), userViewModel);
    }

    // POST: /Profile/CreateUser
    [Authorize(Roles = Constants.Roles.READER)]
    [HttpPost]
    public ActionResult EditUser(UserViewModel userViewModel)
    {
        bool isUserLoginChanged = false;

        if (!ModelState.IsValid)
        {
            return View(nameof(EditUser), userViewModel);
        }

        if (!string.IsNullOrWhiteSpace(userViewModel.Password) &&
!userViewModel.Password.Equals(userViewModel.ConfirmPassword))
        {
            ModelState.AddModelError(string.Empty, "Passwords do not match.");
            return View(nameof(EditUser), userViewModel);
        }

        using (Db db = new())
        {
            string login = User.Identity.Name;

            if (login != userViewModel.Login)
```

```
            {
                login = userViewModel.Login;
                isUserLoginChanged = true;
            }

            if (db.Users.Any(x => x.Id != userViewModel.Id && x.Login == login))
            {
                ModelState.AddModelError(string.Empty, $"Login {userViewModel.Login} already taken.");
                userViewModel.Login = string.Empty;

                return View(nameof(EditUser), userViewModel);
            }

            User user = db.Users.Find(userViewModel.Id);
            user.CopyFiedls(userViewModel);

            if (!string.IsNullOrWhiteSpace(userViewModel.Password))
            {
                user.Password = userViewModel.Password;
            }

            db.SaveChanges();
        }

        TempData[Constants.Keys.SUCCESS] = "You have edited your profile!";

        return !isUserLoginChanged
            ? View(nameof(EditUser), userViewModel)
            : RedirectToAction(nameof(Logout));
    }
  }
}
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного університету
науки і технологій
Анатолій РАДКЕВИЧ
10.06.2022

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИГ

Опис програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01255-01 13 01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
10.06.2022

Керівник розробки
Олександр ІВАНОВ
10.06.2022

Виконавець
Андрій
10.06.2022

Нормоконтролер
Олена КУРОП'ЯТНИК
10.06.2022

2022

ЗАТВЕРДЖЕНО
44165850.01255-01 13 01-ЛЗ

ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИГ

Опис програми

44165850.01255-01 13 01

Листів 38

2022

АНОТАЦІЯ

Документ 44165850.01255-01 13 01 «ВЕБ-ПОРТАЛ ASP.NET ПРОДАЖУ КНИЖОК. Опис програми» входить до складу програмної документації на програму, яка представляє собою веб-сайт для пошуку та замовлення книг

У даному документі представленні опис програми та її функціональних можливостей.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Розроблюваний програмний продукт «Веб-портал на ASP.NET для продажу книг» призначений для полегшення пошуку книг та їх замовлення.

Для функціонування програмного продукту на стороні сервера необхідно наступне програмне забезпечення:

- операційна система Windows 8/Windows 2012;
- веб-сервер IIS7/IIS8;
- підтримка ASP.NET MVC 5;
- MSSQL Server 2012.

Для функціонування програмного продукту на стороні клієнта необхідно встановити веб-браузер Microsoft Edge/Google Chrome/Opera.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціональне призначення розроблюваного ПЗ — розробка інформаційного ресурсу для здійснення пошуку книг в каталозі магазину та оформлення замовлення.

Функціональні обмеження — веб-сайт було розроблено для таких браузерів, як:

- Microsoft Edge v.102.0.1245.33+;
- Google Chrome v.102.0.5005.63+;
- Opera v.87.0+.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм програми

Всі сторінки веб-сайту мають загальний алгоритм роботи:

- клієнт формує HTTP-запит веб-сторінки та відправляє його на сервер;
- система маршрутизації на сервері аналізує отриманий URL-запит та обирає відповідний маршрут;
- в залежності від обраного маршруту, сервер передає керування відповідному контролеру;
- контролер викликає один зі своїх action-методів, при необхідності обробляє дані, генерує представлення та відправляє його клієнту;
- клієнт отримує веб-сторінку, сформованою сервером, та працює з нею.

3.2 Використані методи

Для розробки веб-сайту за допомогою ASP.NET MVC 5 використовується патерн MVC — Model-View-Controller. Даний патерн розділяє дані додатку та логіку їх обробки на 3 окремі шари:

- Model (модель) — представляє собою дані та методи роботи з ними, при цьому вона не залежить від представлення (не знає як вони відображаються);
- View (представлення) — відповідає за відображення даних моделі користувачу, та слідкує за змінами у ній. В ASP.NET представлення використовує обробник представлень Razor для впровадження коду .NET в розмітку HTML;
- Controller (контролер) — забезпечує зв’язок між представленням та моделлю, здійснюючи обробку введених даних, формування відповіді і взаємодії з користувачем.

3.3 Структура програми

Діаграми класів модулів проєкту з урахуванням архітектури MVC зображено на рисунках 3.1 – 3.10.

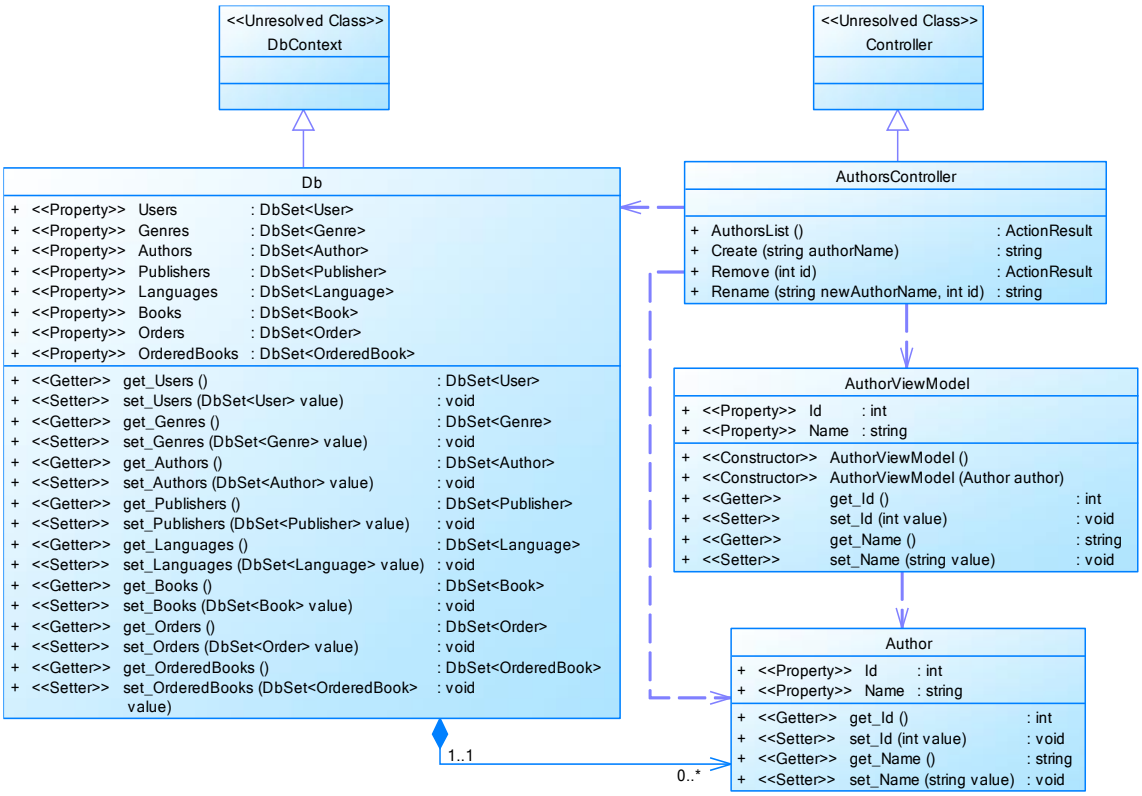

Рисунок 3.1 – Діаграма класів, в основі якої контролер таблиці авторів

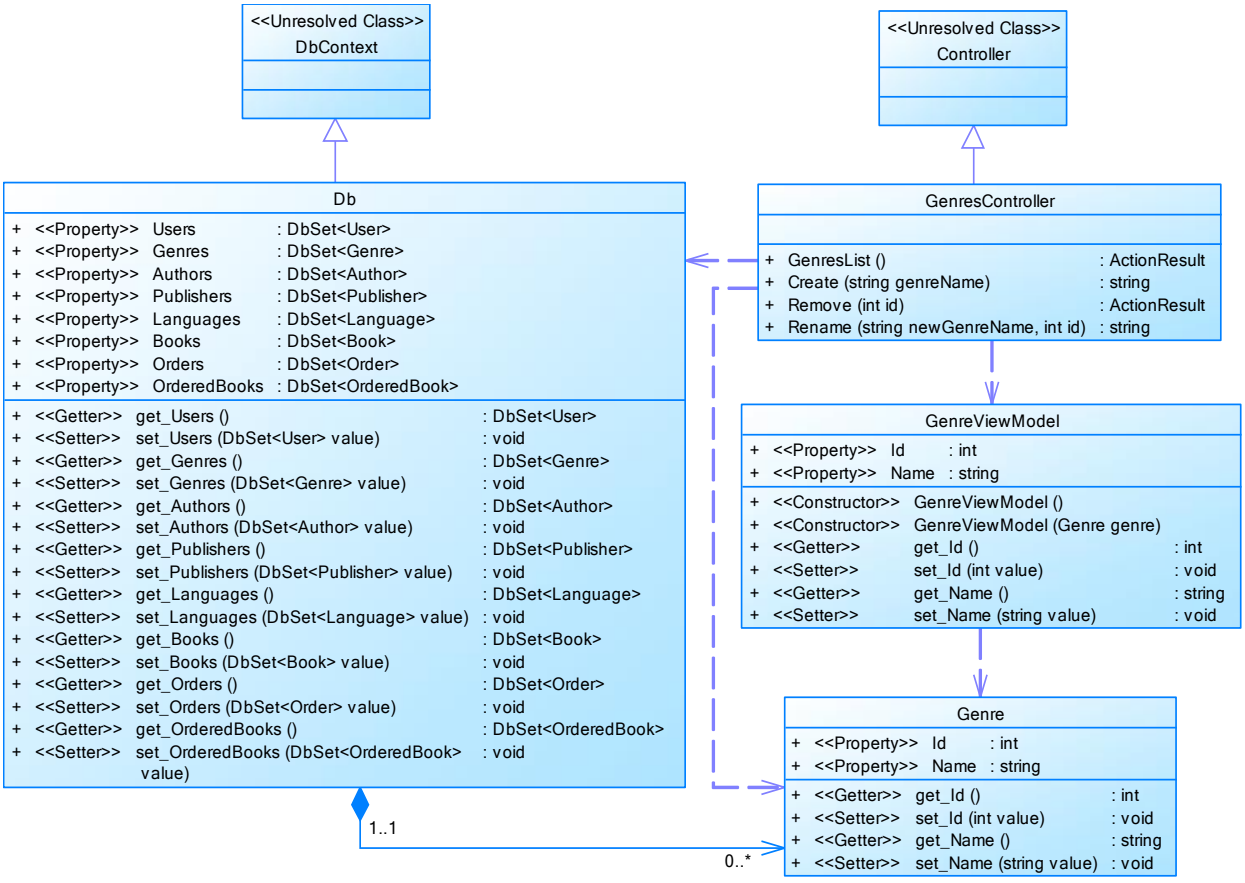

Рисунок 3.2 – Діаграма класів, в основі якої контролер таблиці жанрів

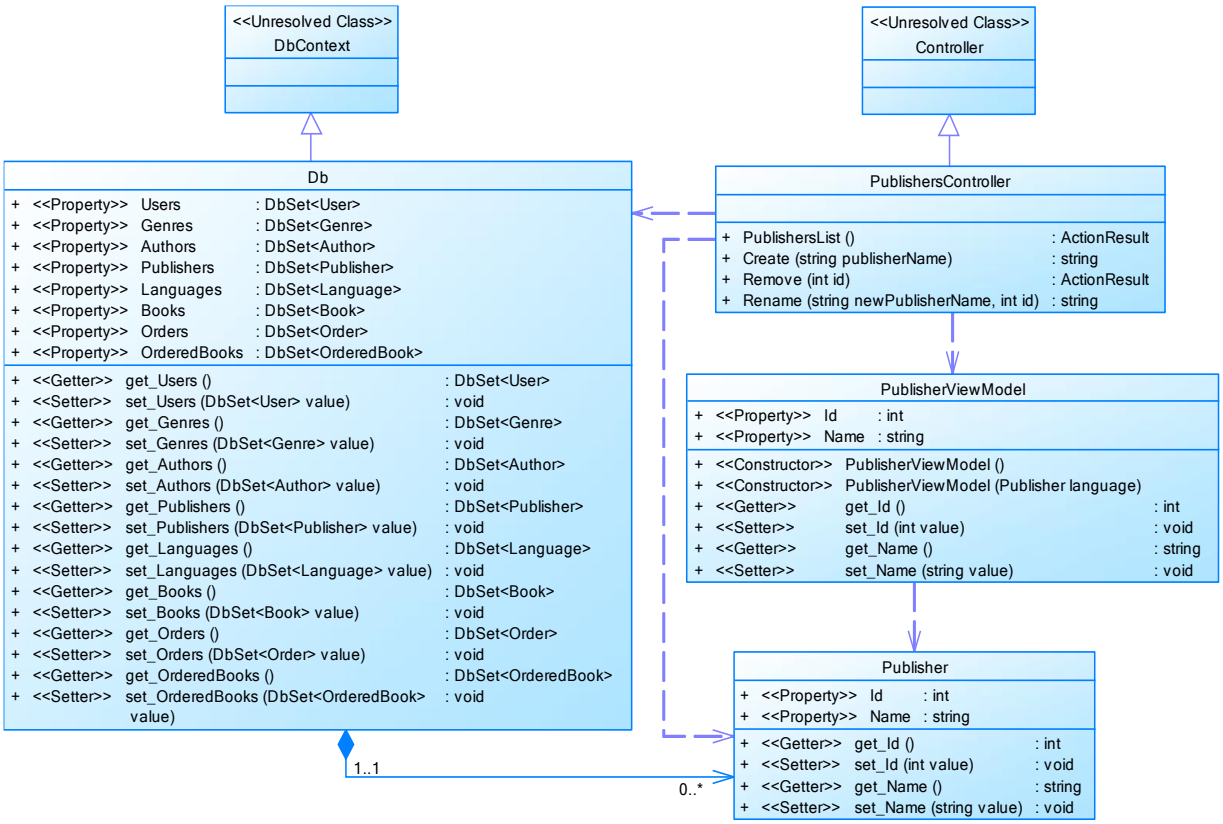

Рисунок 3.3 – Діаграма класів, в основі якої контролер таблиці видавництв

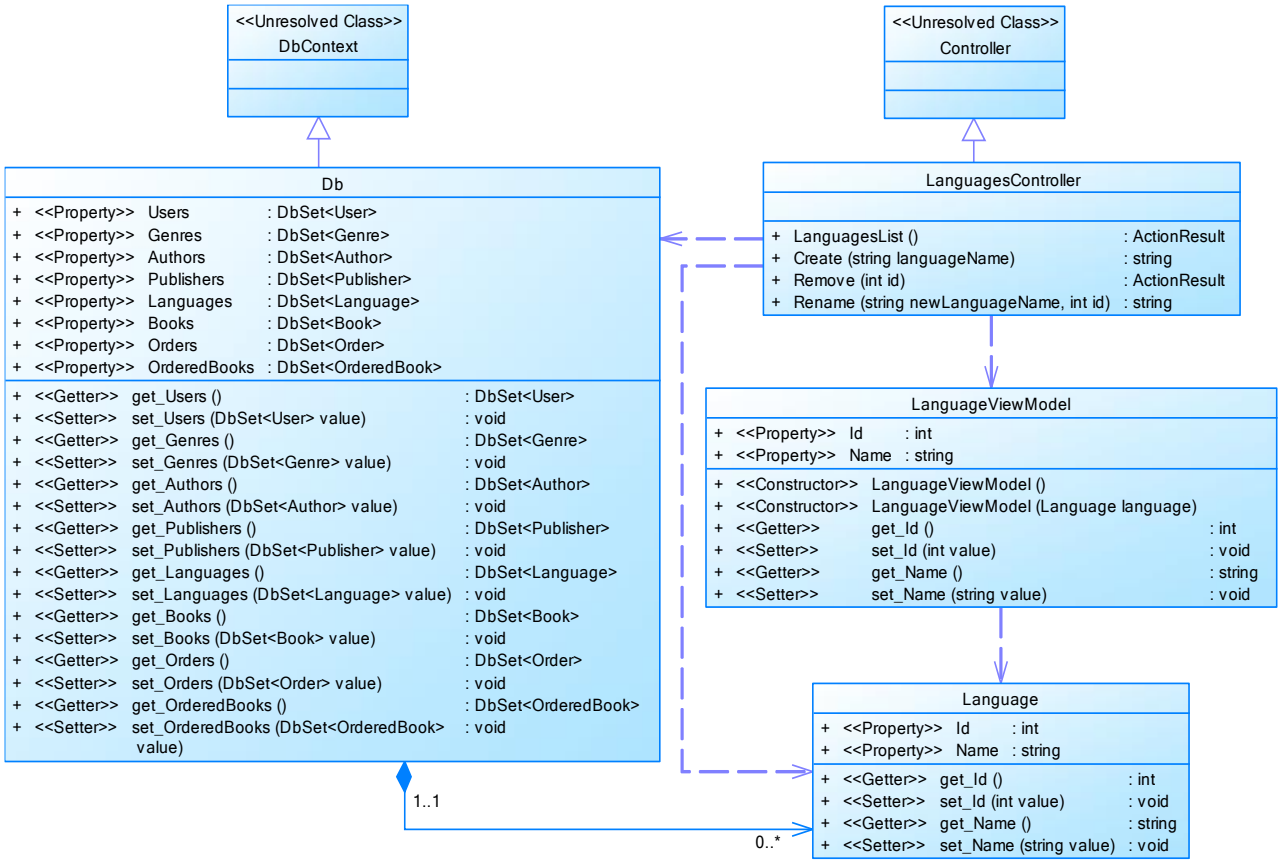

Рисунок 3.4 – Діаграма класів, в основі якої контролер таблиці мов

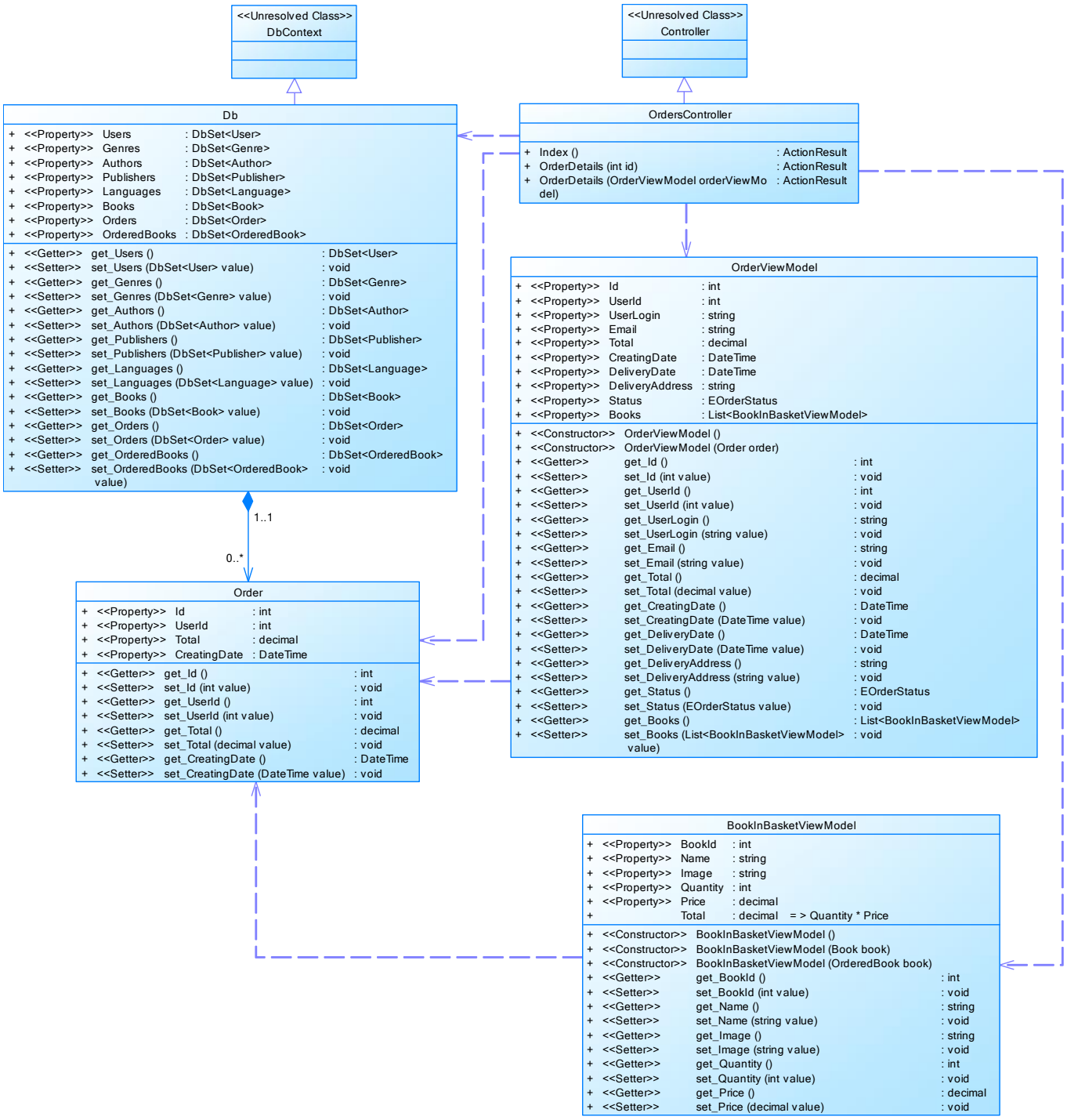

Рисунок 3.5 – Діаграма класів, в основі якої контролер таблиці замовлень (в області адміністратора)

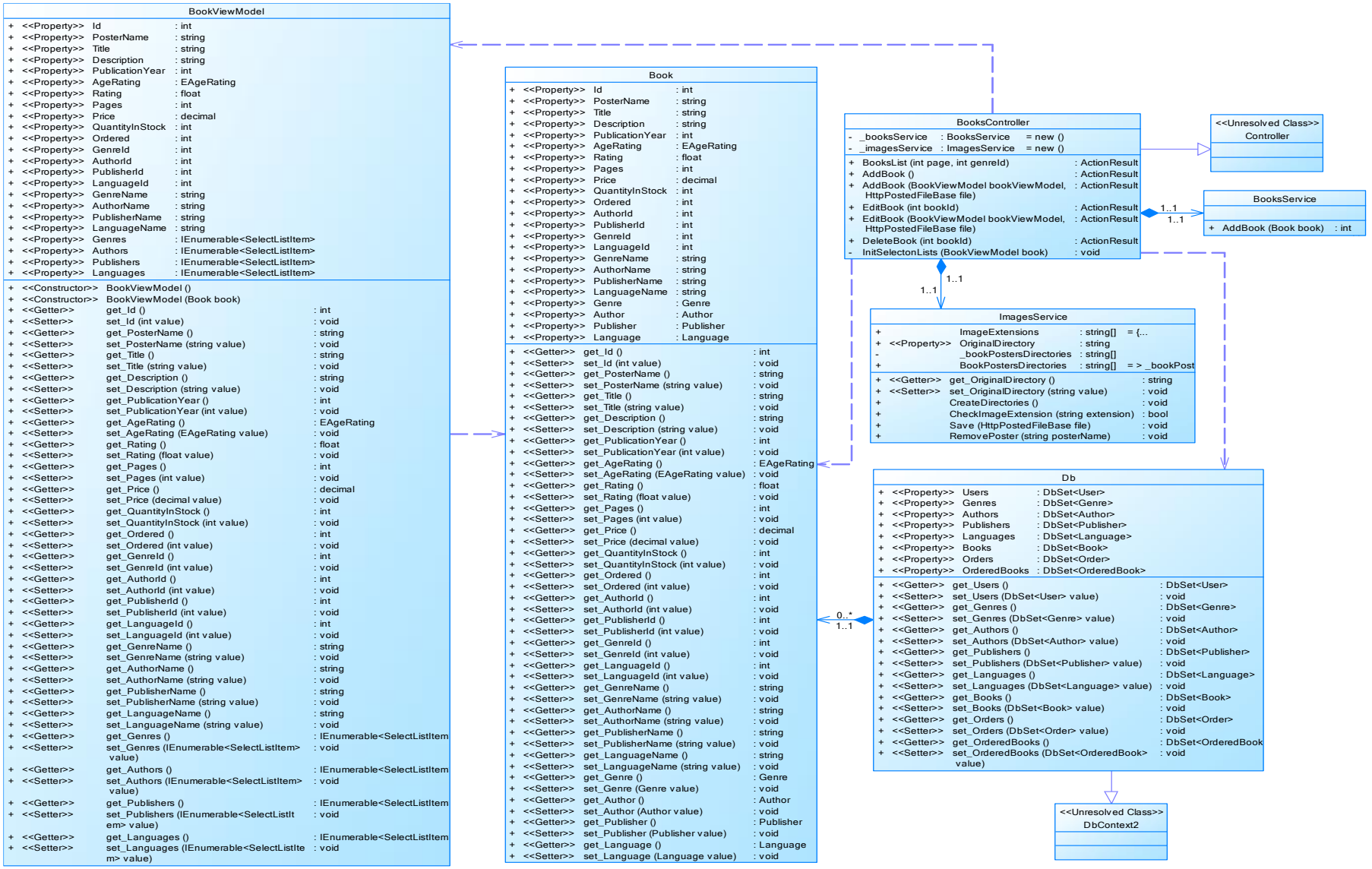

Рисунок 3.6 – Діаграма класів, в основі якої контролер таблиці книг

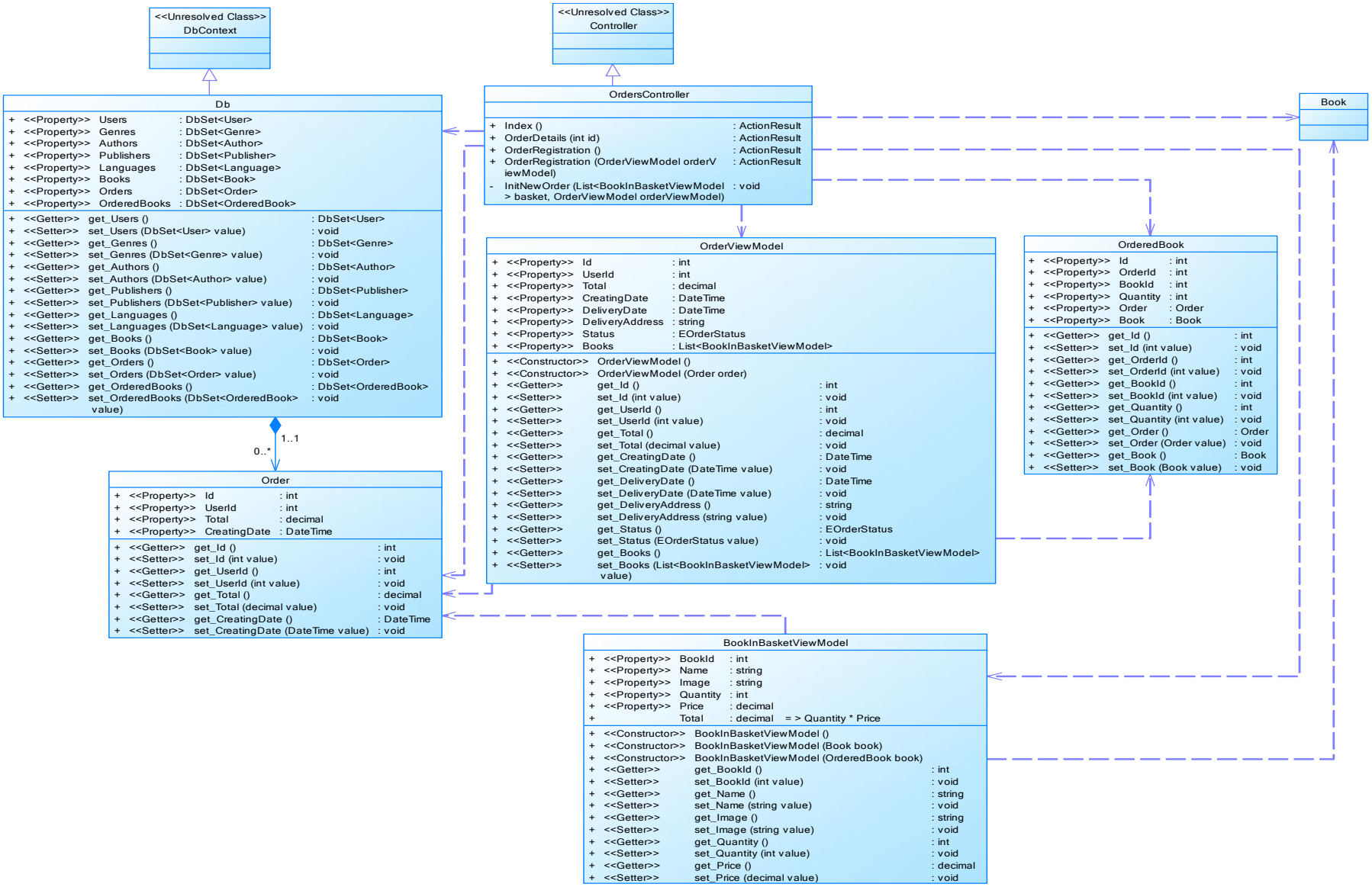

Рисунок 3.7 – Контролер таблиці замовлень (область користувача)

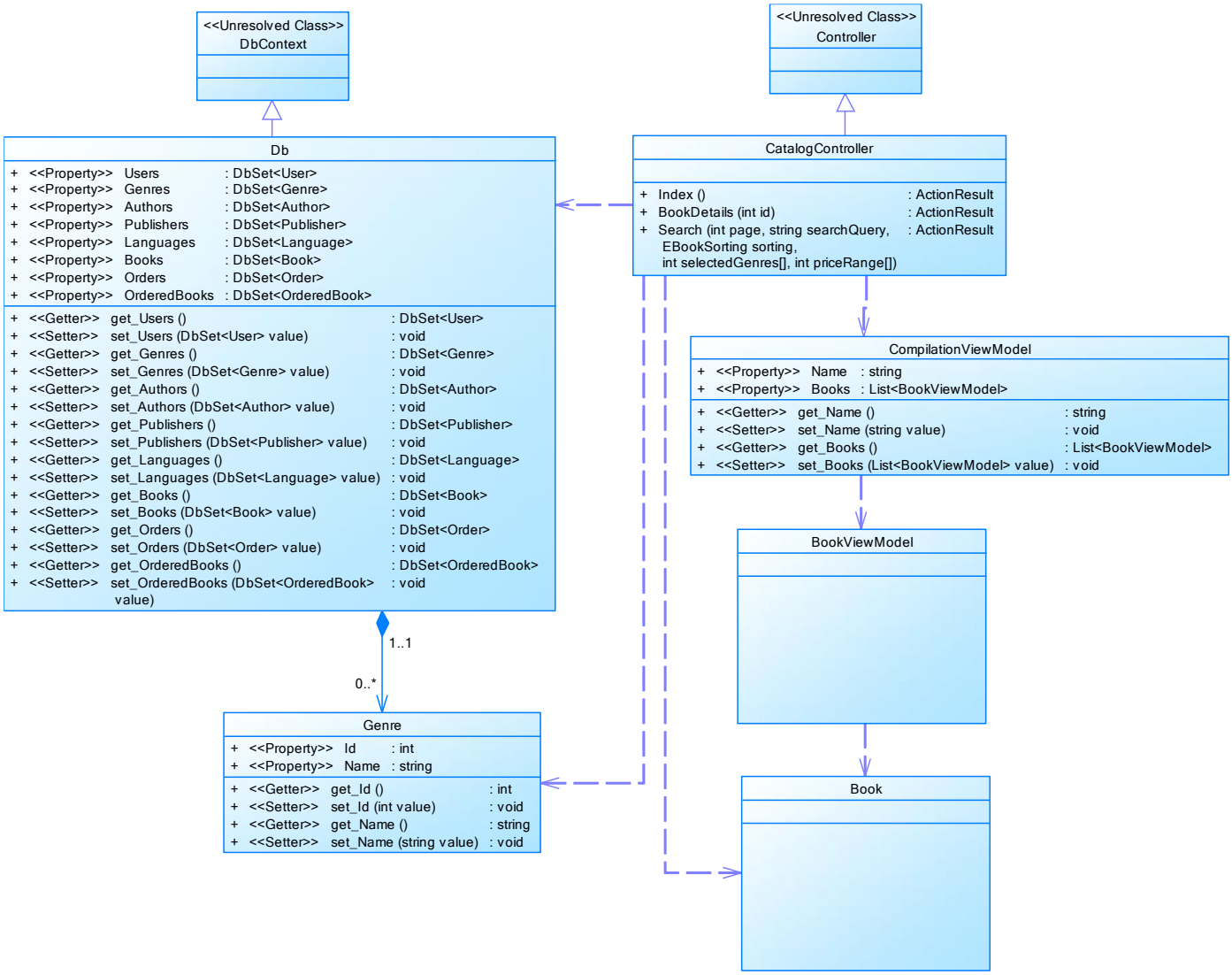

Рисунок 3.8 – Діаграма класів, в основі якої контролер книг в каталозі

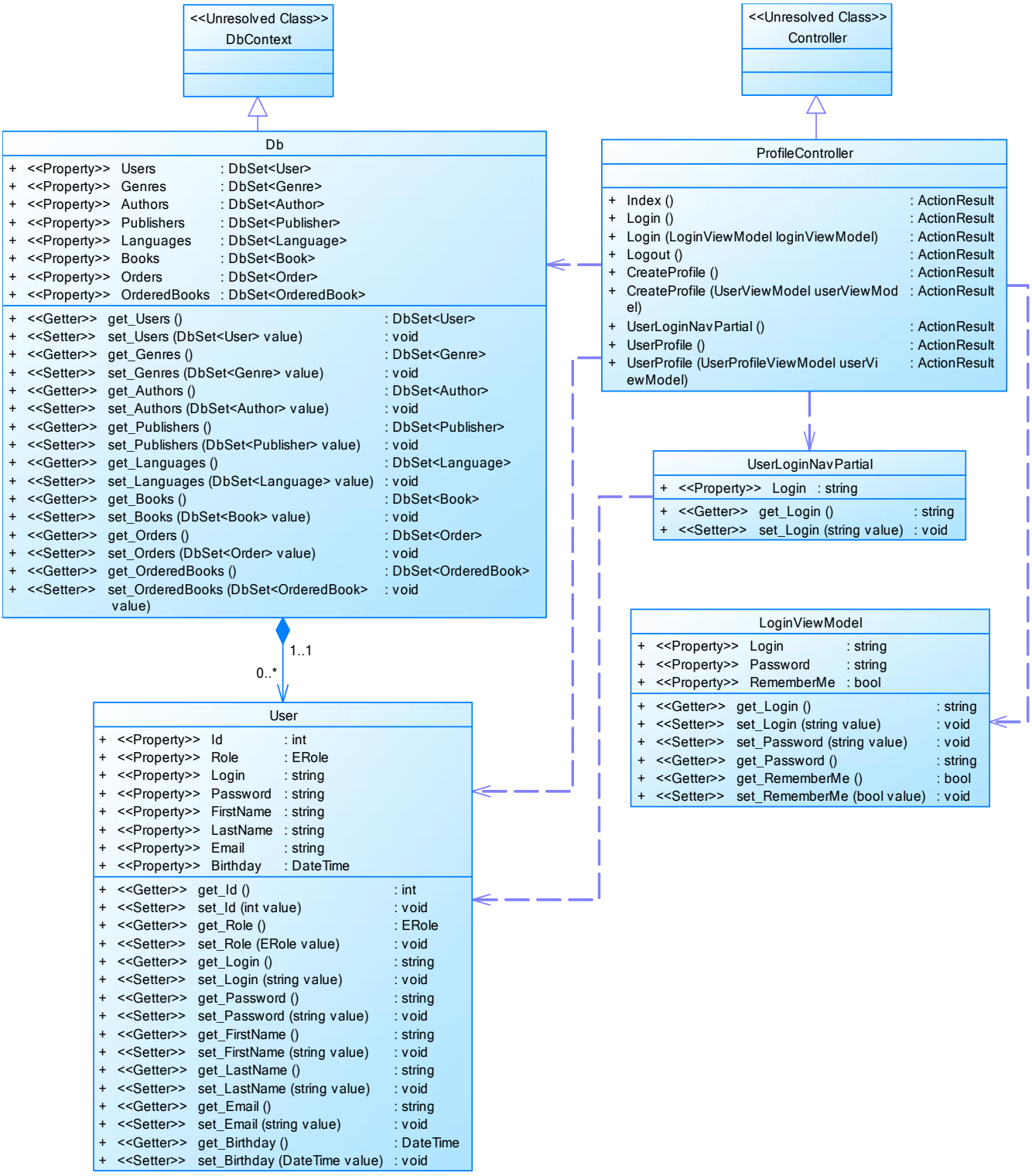

Рисунок 3.9 – Діаграма класів, в основі якої контролер облікового запису користувача

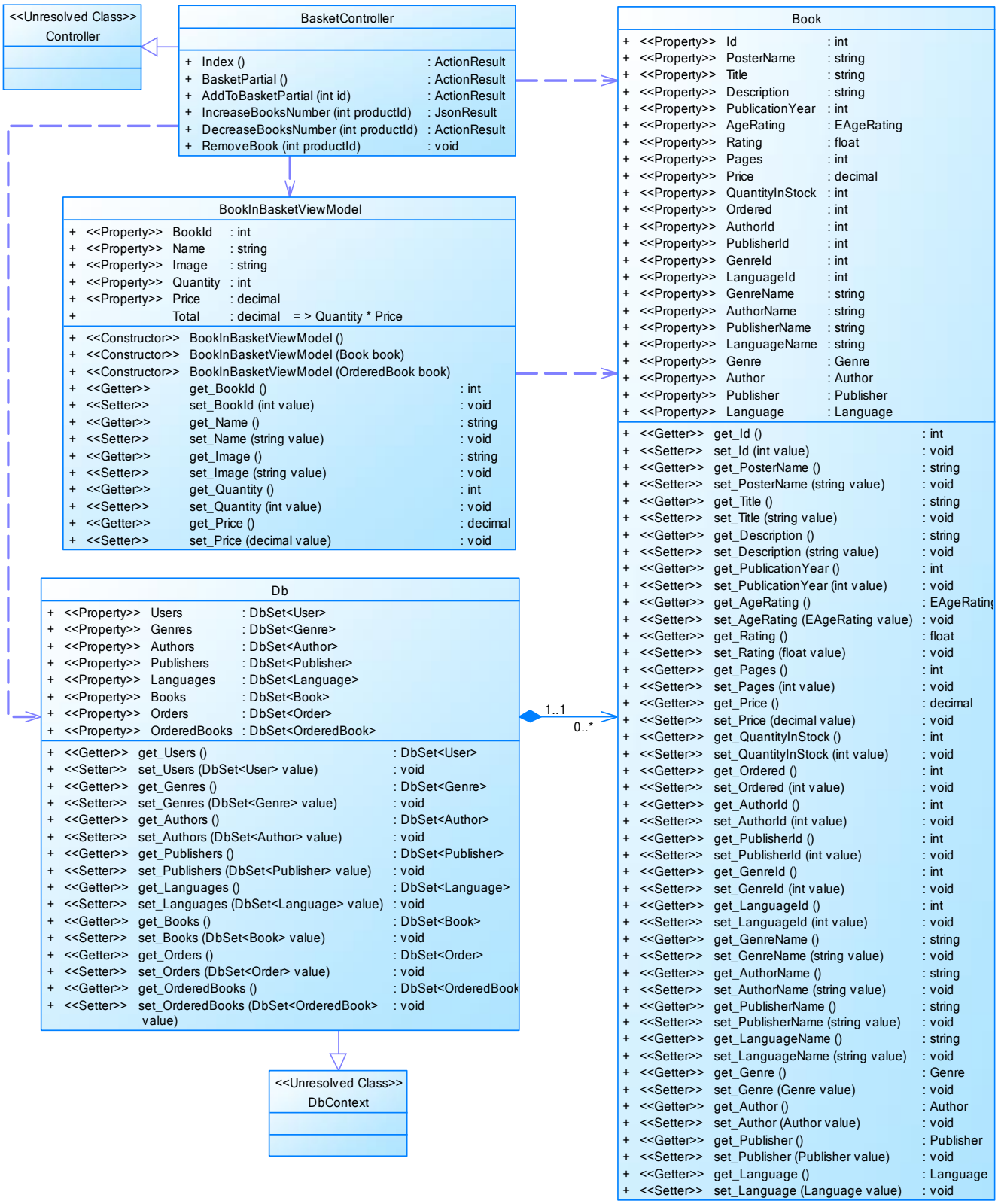

Рисунок 3.10 Діаграма класів, в основі якої контролер книг у кошику покупця

Опис обов’язків класів:

- DbContext — є базовим класом Entity Framework та надає широкі можливості для роботи з базою даних (створення запитів, відстеження змін та збереження даних в БД;
- Db — зв’язок з БД та зберігання контексту даних;
- User — зберігання даних про користувача;
- Author — зберігання даних про автора;
- Genre — зберігання даних про жанр;
- Publisher — зберігання даних про видавництво;
- Language — зберігання даних про мову;
- Order — зберігання даних про замовлення;
- Book — зберігання даних про книгу;
- OrderedBook — зберігання даних про замовлену книгу;
- AuthorViewModel — зберігання даних про автора, необхідних для відображення в представленнях;
- GenreViewModel — зберігання даних про жанр, необхідних для відображення в представленнях;
- PublisherViewModel — зберігання даних про видавництво, необхідних його відображення в представленнях;
- LanguageViewModel — зберігання даних про мову, необхідних її відображення в представленнях;
- BookViewModel — зберігання даних про книгу, необхідних її відображення в представленнях;
- BookInBasketViewModel — зберігання даних про книгу, необхідних для відображення в кошику;
- CompilationViewModel — зберігання даних про підбірку, необхідних для відображення в каталозі;
- UserLoginViewModel — зберігання логіну користувача, необхідного для відображення на панелі навігації;

- OrderViewModel — зберігання даних про замовлення, необхідних для відображення в представленнях;
- UserViewModel — зберігання даних про користувача, необхідних для відображення в представлення;
- Controller — абстрактний базовий клас для контролерів MVC з підтримкою представлень. Є головною ланкою в архітектурі MVC, реагує на запити отримані від системи маршрутизації, формує та надсилає відповідь на запит браузеру;
- AuthorsController — робота з таблицею tableAuthors;
- GenresController — робота з таблицею tableGenres;
- PublishersController — робота з таблицею tablePublishers;
- LanguagesController — робота з таблицею tableLanguages;
- AuthorController — робота з таблицею tableAuthors;
- BooksController — робота з книгами в каталозі (таблиця tableBooks)
- OrdersController (зона адміністратора) — перегляд замовлень, редагування статусу та дати доставки замовлень;
- BasketController — робота з кошиком гостя або читача, відповідає за додавання та видалення книг до кошика, змінення кількості їх примірників;
- CatalogController — відповідає за формування підбірок книг на головній сторінці сайту, перегля детальної інфомації про книгу та пошук книг по ключовим словам, сортуванням результатів пошуку та їх фільтрацією по жанрам, ціновим діапазонам;
- OrdersController — відповідає за створення нового замовлення, перегляд історії замовлень читача та детальної інфомації про певне замовлення;
- ProfileController — відповідає за авторизацію та реєстрацію коритувача, а також за можливість редагування читачем інформації про власний обліковий запис.

3.4 Зв’язки програми з іншими програмами

По завершенню всіх етапів розробки програмний продукт буде розгорнуто на веб-сервері IIS.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Розроблюваний програмний продукт розрахований для використання на ЕОМ з наступними характеристики:

– наявність веб-браузеру;
– діагональ екрану — 15.5 дюймів;
– частота процесору — від 1 ГГц та вище;
– оперативна пам’ять — 512 Мб;
– пам'ять дискового накопичувача — 1 Гб.

Сервер, на якому буде розгорнуто програмний продукт, повинен мати наступні характеристики:

– частота процесору — від 2 ГГц та вище;
– оперативна пам’ять — 8 Гб;
– пам'ять дискового накопичувача — 2 Гб.

5 ВИКЛИК ЗАВАНТАЖЕННЯ

Оскільки програмний продукт було розгорнуто на хостингу, то щоб отримати до нього доступ потрібно здійснити перехід за посиланням: http://andriitantsiura-001-site1.ftempurl.com/.

6 ВХІДНІ ДАНІ

Вхідні дані наведено в таблицях відношень БД 4.1 – 4.8.

Таблиця 4.1 – Таблиця «tableAuthors»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор автору	int	4	+
2	Name	Ім’я автора	NVARCHAR	70	-

Таблиця 4.2 – Таблиця «tableGenres»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор жанру	int	4	+
2	Name	Ім’я жанру	NVARCHAR	60	-

Таблиця 4.3 – Таблиця «tableLanguages»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор мови	INT	4	+
2	Name	Ім’я мови	NVARCHAR	60	-

Таблиця 4.4 – Таблиця «tablePublishers»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор видавництва	INT	4	+
2	Name	Назва видавництва	NVARCHAR	60	-

Таблиця 4.5 – Таблиця «tableUsers»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор користувача	INT	4	+
2	Role	Ідентифікатор ролі	INT	4	-
3	Login	Ім’я для входу в обліковий запис	NVARCHAR	16	-
4	Password	Пароль	NVARCHAR	16	-
5	FirstName	Ім'я	NVARCHAR	30	-
6	LastName	Прізвище	NVARCHAR	30	-
7	Email	Пошта	NVARCHAR	254	-

Таблиця 4.6 – Таблиця «tableBooks»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор користувача	INT	4	+
2	AuthorId	Ідентифікатор автора	INT	4	-
3	PublisherId	Ідентифікатор видавництва	INT	4	-
4	GenreId	Ідентифікатор жанру	INT	4	-
5	LanguageId	Ідентифікатор мови	INT	4	-
6	PosterName	Ідентифікатор ролі	INT	4	-
7	Title	Назва книги	NVARCHAR	150	-
8	Description	Опис книги	NVARCHAR	100	-
9	PublicationYear	Рік видавництва	INT	4	-
10	AgeRating	Вікове обмеження	INT	4	-
11	Rating	Рейтинг	FLOAT	4	-
12	Pages	Кількість сторінок	INT	4	-
13	Price	Вартість	DECIMAL	16	-
14	QuantityInStock	Кількість на складі	INT	4	-
15	Ordered	Кількість замовлень	INT	4	-

Таблиця 4.7 – Таблиця «tableOrders»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор видавництва	INT	4	+
2	UserId	Ідентифікатор читача	INT	4	-
3	CreatingDate	Дата створення	DATETIME	16	-
4	DeliveryDate	Дата доставки	DATETIME	16	-
5	DeliveryAddress	Адреса доставки	NVARCHAR	2147483646	-
6	Total	Вартість замовлення	DECIMAL	16	-
7	Status	Статус замовлення	INT	4	-

Таблиця 4.8 – Таблиця «tableOrderedBooks»

№	Стовпець	Опис	Тип	Розмір	Ключ
1	Id	Ідентифікатор видавництва	INT	4	+
2	OrderId	Ідентифікатор замовлення	INT	4	-
	BookId	Ідентифікатор книги	INT	4	-
3	Quantity	Кількість екземплярів книги	INT	4	-

7 ВИХІДНІ ДАНІ

Вихідні дані програми:

- інтерфейс, що відображає дані про: книги в каталозі, книги в підбірках, книги в кошику, авторів, жанри, видавництва, мови, користувача, замовлення, замовлені книги;
- рядки в БД;
- повідомлення про стан системи.

8 ОПИС ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ

8.1 Опис станів програми

Стани програми, в яких вона може знаходитись, приведено у таблиці 8.1.

Таблиця 8.1 – Стани програми

№	Стан	Опис стану	Рекомендовані дії
1	Відправка HTTP-запиту	Більшість дій користувача надсилають серверу запити, що призводить до оновлення сторінки	Дочекатись завершення оновлення сторінки
2	Отримання відповіді на HTTP-запит	Отримавши запит, система маршрутизації аналізує його та передає керування відповідному контролеру, який формує відповідь та надсилає її клієнту	Дочекатись завершення отримання відповіді
3	Помилка 404	Сервер не може знайти дані за запитом	Повторити запит. У випадку, якщо помилка є постійною — звернутися до технічної підтримки
4	Пошук книг не дав результатів	Користувач увів ключові слова та/або обрав фільтри, за якими не вдалося знайти жодної книги	Повторити пошук з іншими ключовими словами, обрати інші фільтри
5	Книгу не знайдено	В адресному рядку введено запит на дію над книгою, якої немає в БД	Повторити запит, увівши інший ідентифікатор книги
6	Замовлення не знайдено	В адресному рядку введено запит на дію над замовленням, якого немає в БД	Повторити запит, увівши інший ідентифікатор замовлення

8.2 Опис керування діалогом програми

На кожній сторінці сайту є навігаційна панель, яка змінює свій зовнішній вигляд в залежності від ролі користувача.

В ролі гостя, користувачу на навігаційній панелі доступні наступні посилання:

- перехід на головну сторінку сайту;
- перехід на сторінку пошуку книг;

- перехід на сторінку кошика. Якщо в кошик було додано книги, то поруч з посиланням буде відображено їх кількість та сумарну вартість.
- перейти на сторінку входу в систему.

В ролі читача, з панелі зникає посилання на вхід та з'являються наступні посилання:

- перехід на сторінку історії замовлень;
- перехід на сторінку редагування облікового запису.
- вихід.

В ролі адміністратора, з навігаційної панелі зникає посилання на кошик, та з’являються наступні посилання

- перехід на адміністративну панель;
- вихід.

На головній сторінці сайту користувач може переглянути сформовані системою підбірки книг, додати книгу до кошика або перейти на сторінку детальної інформації про книгу (для ролі адміністратора робота з кошиком не доступна).

Здійснивши пошук по ключовим словам, користувач переходить на сторінку пошуку, де може здійснювати фільтрацію знайдених книг по жанрам, ціновому діапазоні та встановити сортування результатів пошуку, додати книгу до кошика або перейти на сторінку детальної інформації про книгу (для ролі адміністратора робота з кошиком не доступна). Якщо в пошуковому запиті не було введено ключові слова, то користувач матиме змогу переглядати всі книги в каталозі.

На сторінці кошика, користувач може переглядати додані книги, змінювати кількість їх примірників та видаляти їх. У читача з’являється можливість здійснити перехід на сторінку створення замовлення, якщо обрана кількість книг не є більшою за кількість є на складі; якщо користувач є гостем, він буде перенаправлений на сторінку входу.

На сторінці входу в систему, користувач може увійти в систему або створити новий обліковий запис.

На сторінці історії замовлень, читач може переглянути детальну інформацію про кожне замовлення.

На адміністративній панелі, праворуч з'являється додаткове меню з наступними посиланнями:

– робота з замовленнями;
– робота з книгами;
– робота з авторами;
– робота з жанрами;
– робота з видавництвами;
– робота з мовами.

На сторінці з замовленнями, адміністратор може переглянути деталі кожного замовлення, змінити їх статус і дату доставки.

На сторінці з книгами, адміністратор може перемикатися між книгами по жанрам, додавати нові книги, здійснювати їх редагування та видалення.

На сторінках роботи з авторами, жанрами, видавництвами та мовами адміністратор може додавати нові записи, видаляти та редагувати їх.

8.3 Формування екранів

Сторінки сайту представлено на рисунках 8.1 – 8.22.

Рисунок 8.1 – Навігаційна панель (гість)

Рисунок 8.2 – Навігаційна панель (читач)

Рисунок 8.3 – Навігаційна панель (адміністратор)

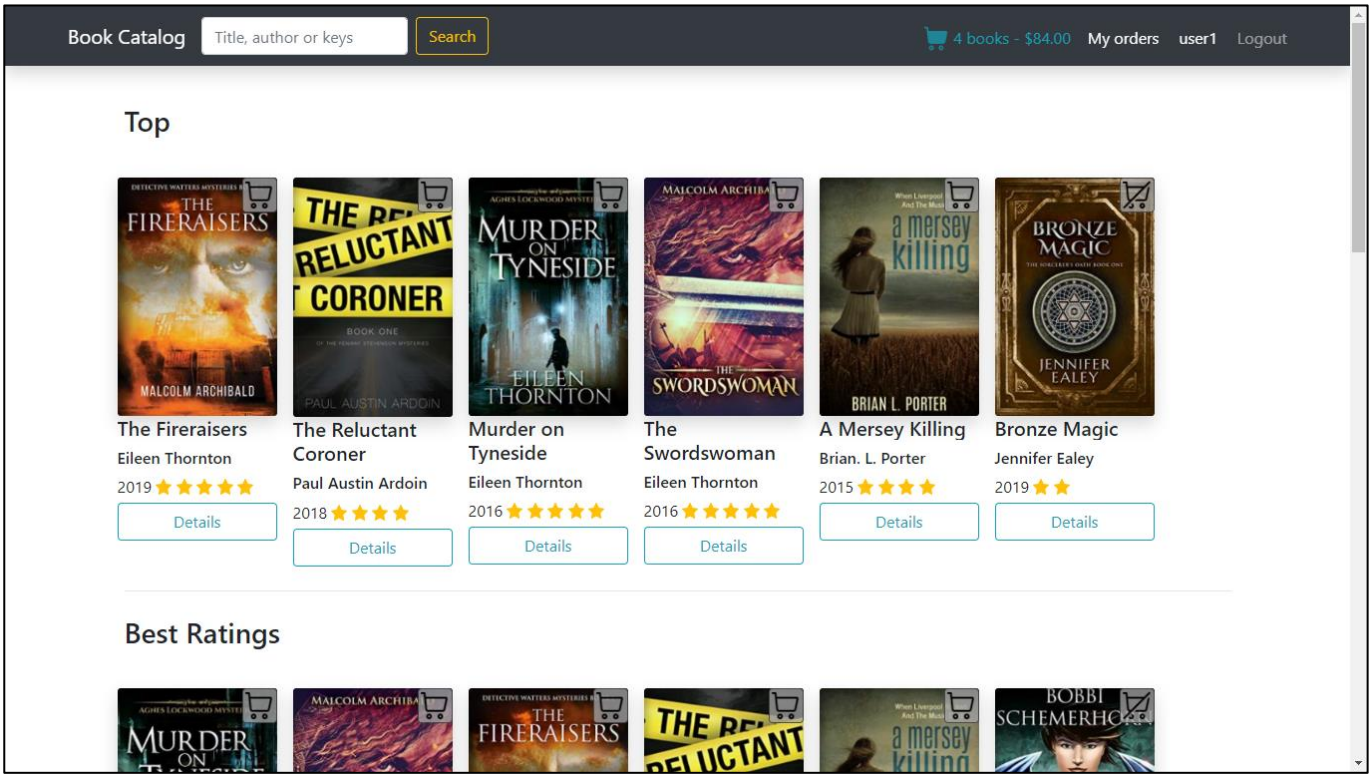

Рисунок 8.4 – Головна сторінка сайту

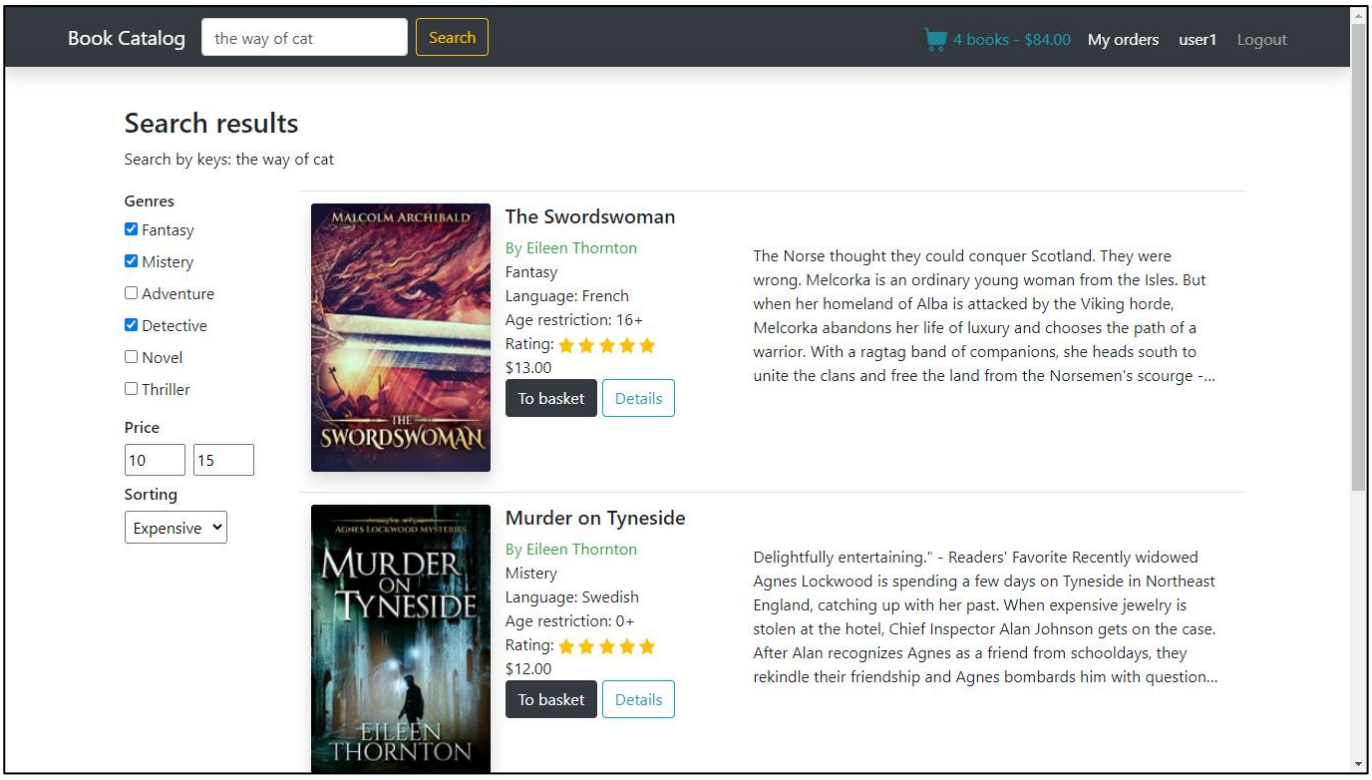

Рисунок 8.5 – Сторінка пошуку книг

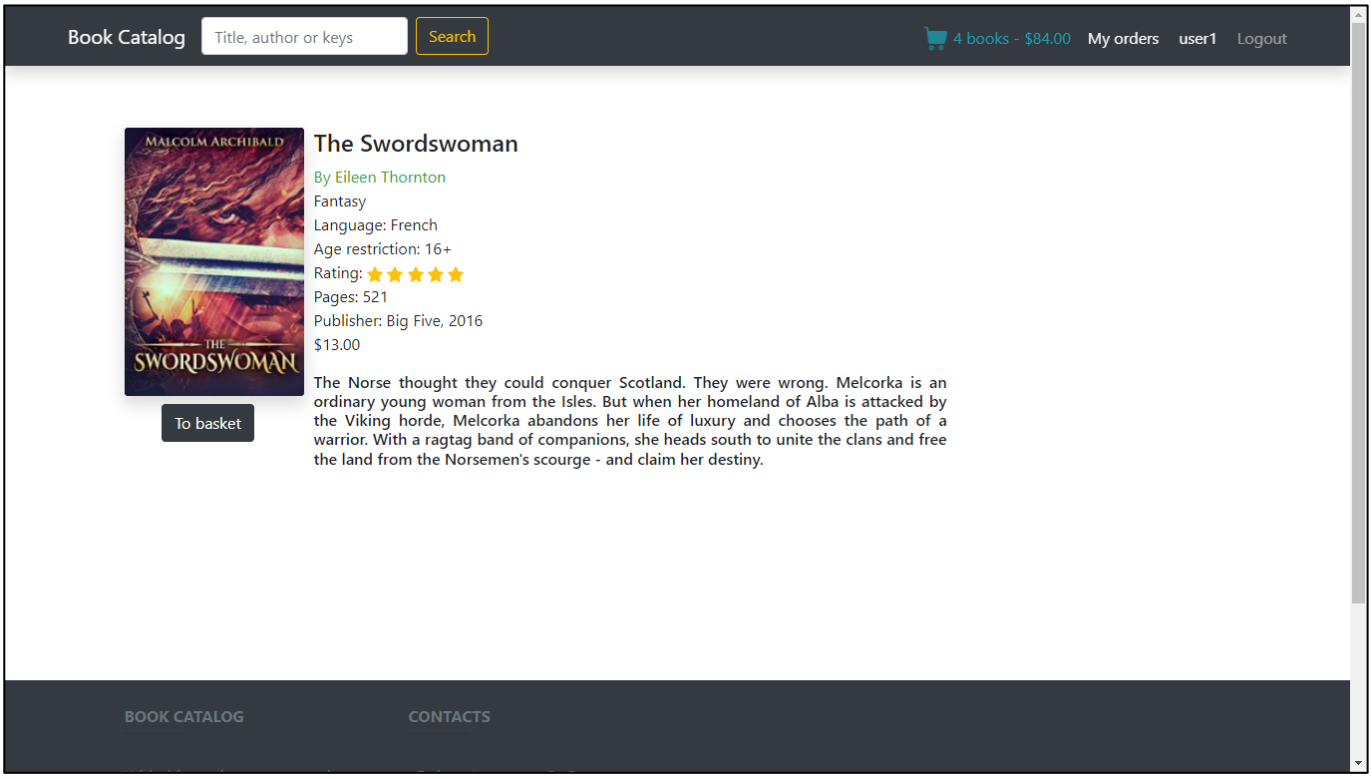

Рисунок 8.6 – Сторінка перегляду детальної інформації про книгу

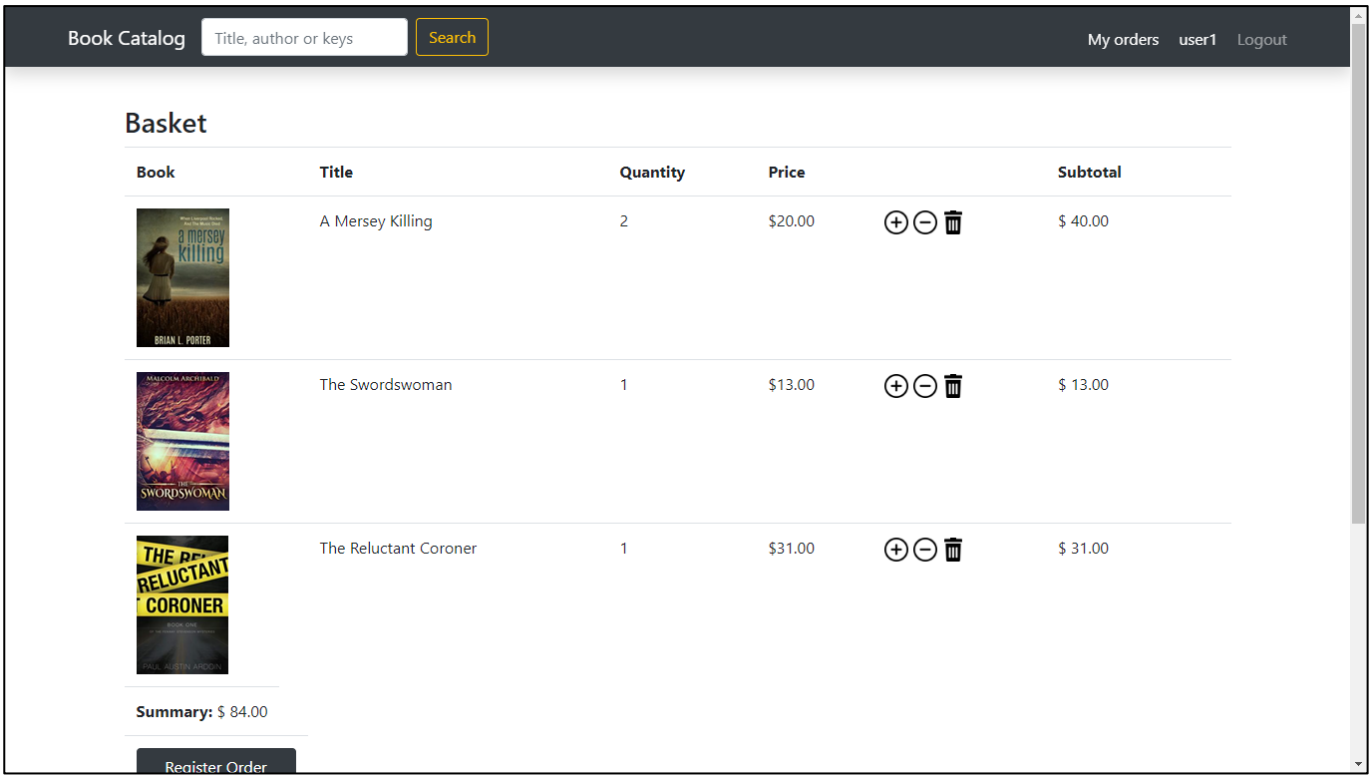

Рисунок 8.7 – Сторінка кошика

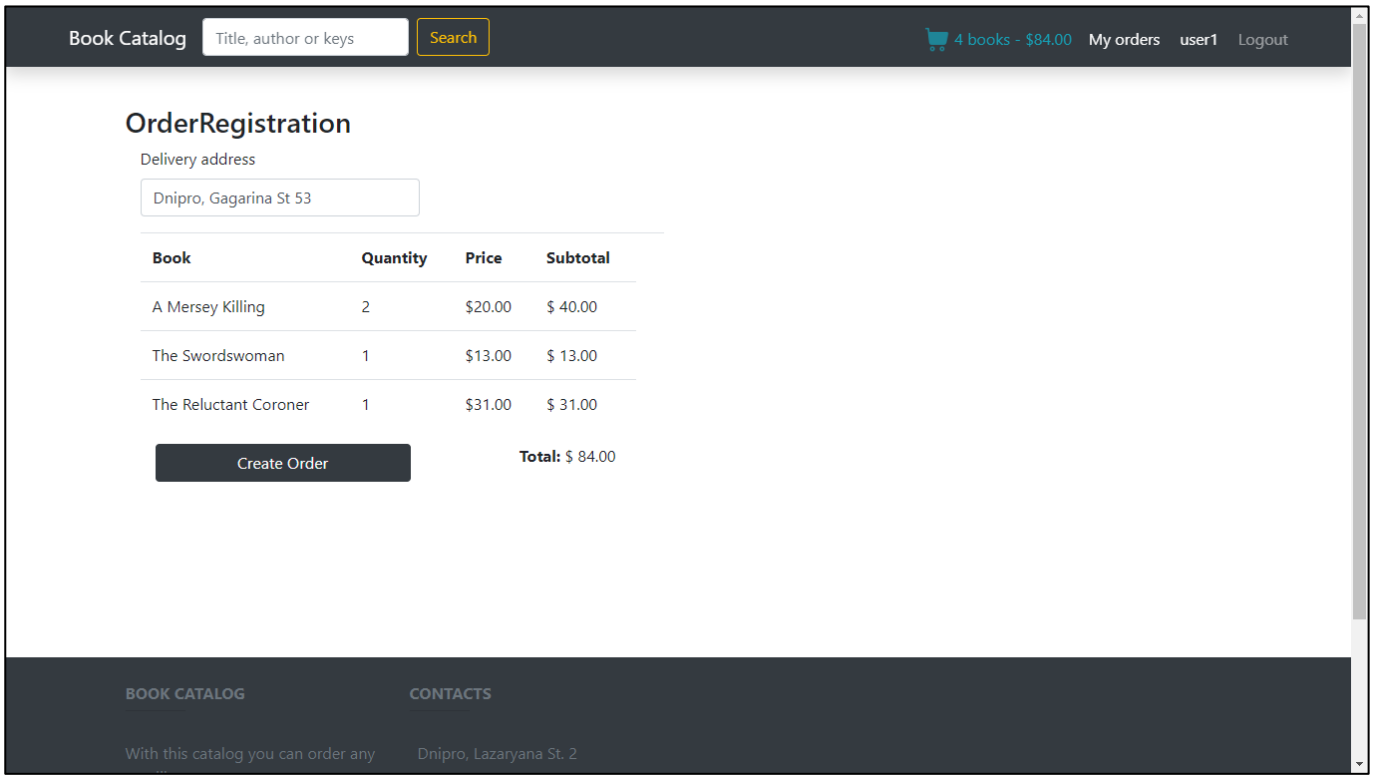

Рисунок 8.8 – Сторінка оформлення замовлення

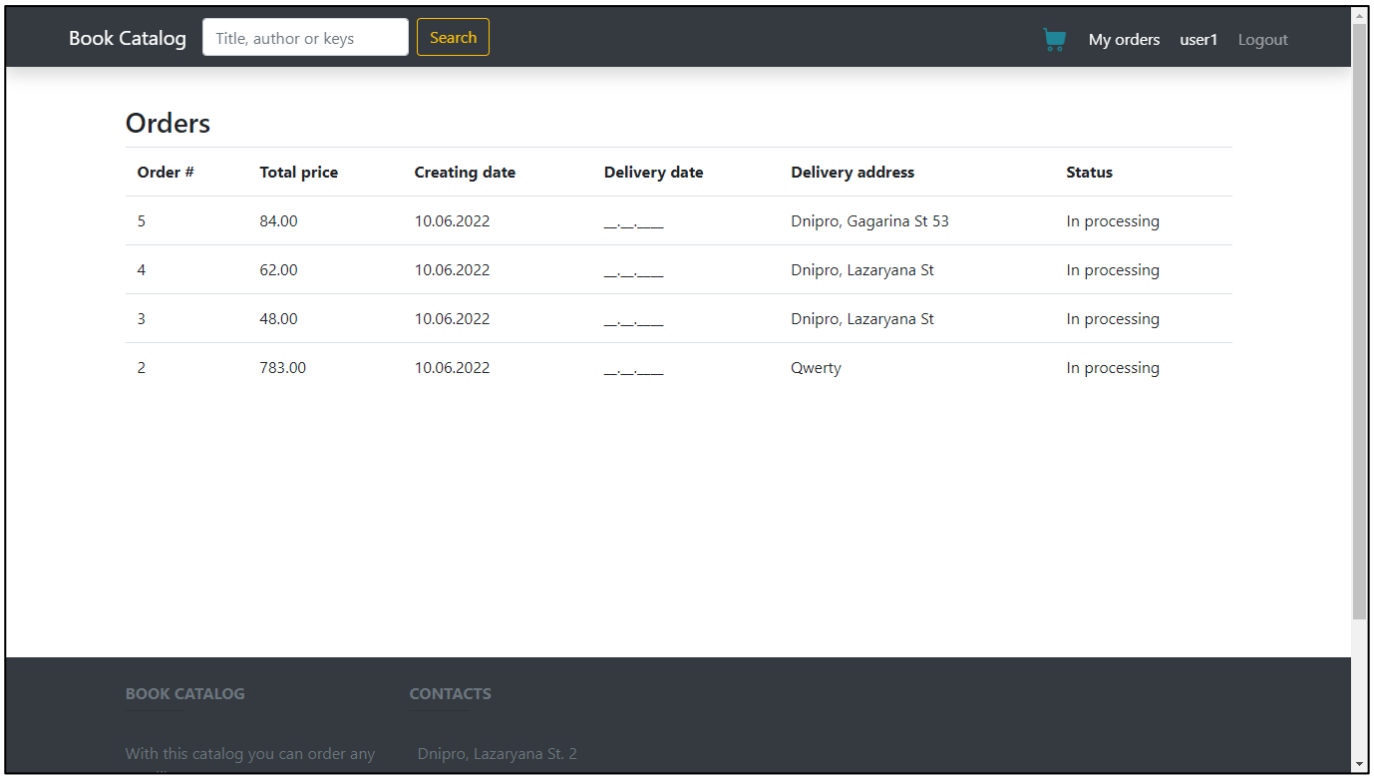

Рисунок 8.9 – Сторінка історії замовлень

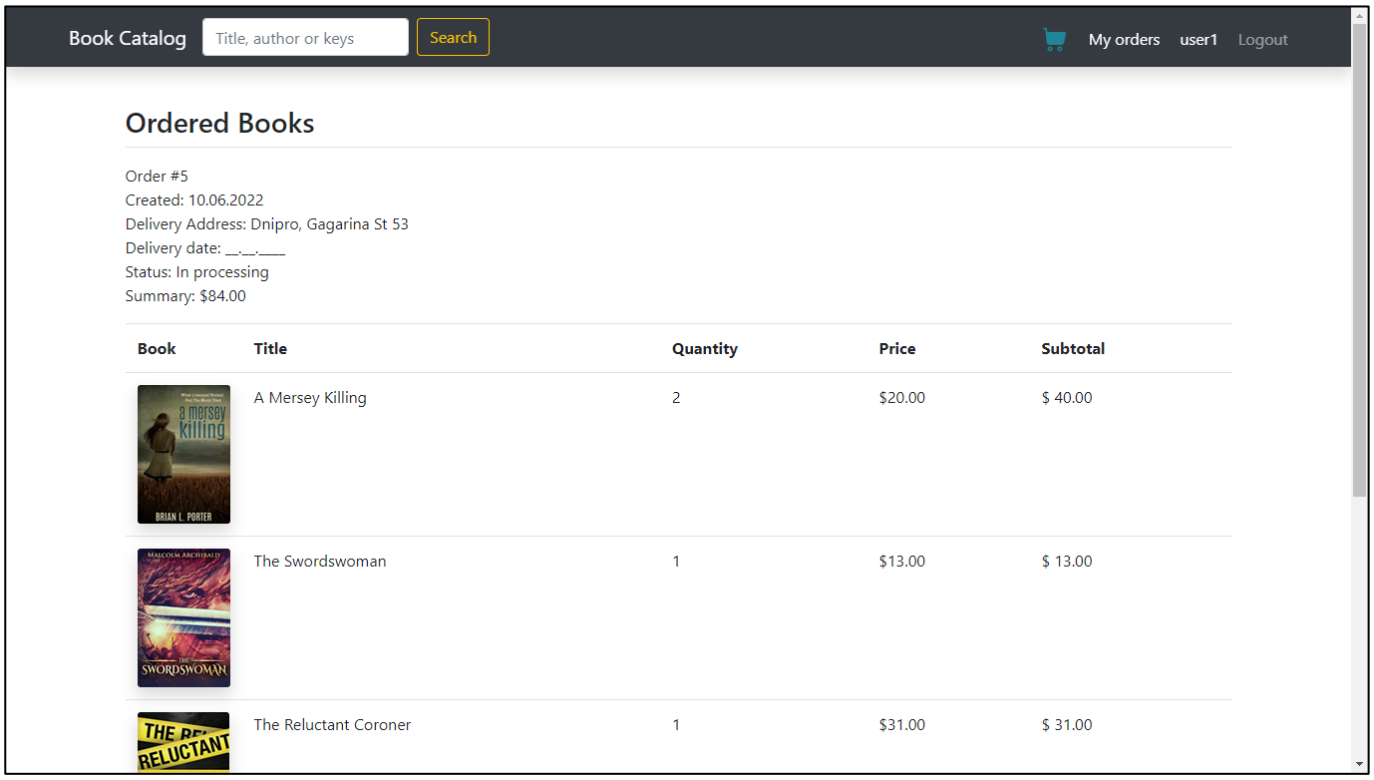

Рисунок 8.10 – Сторінка перегляду детальної інформації про замовлення

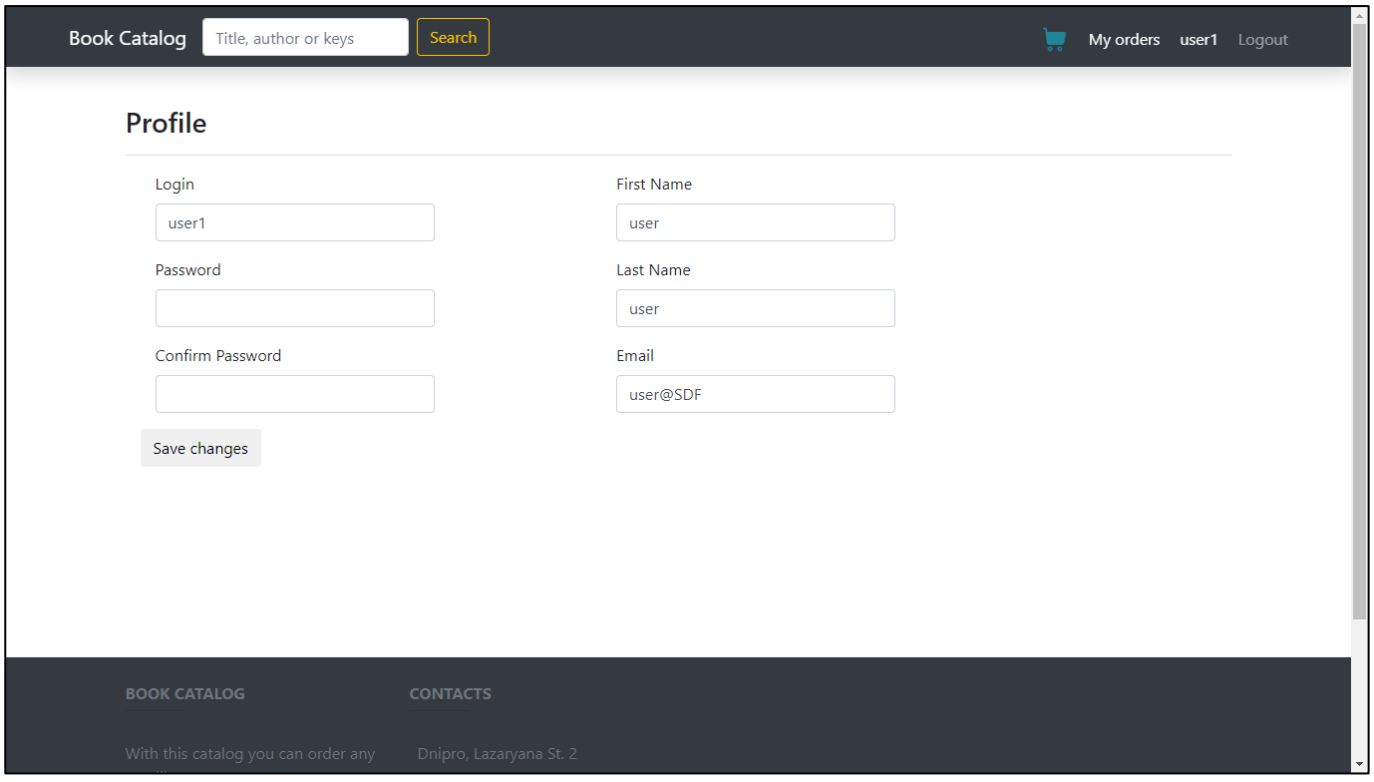

Рисунок 8.11 – Сторінка редагування облікового запису

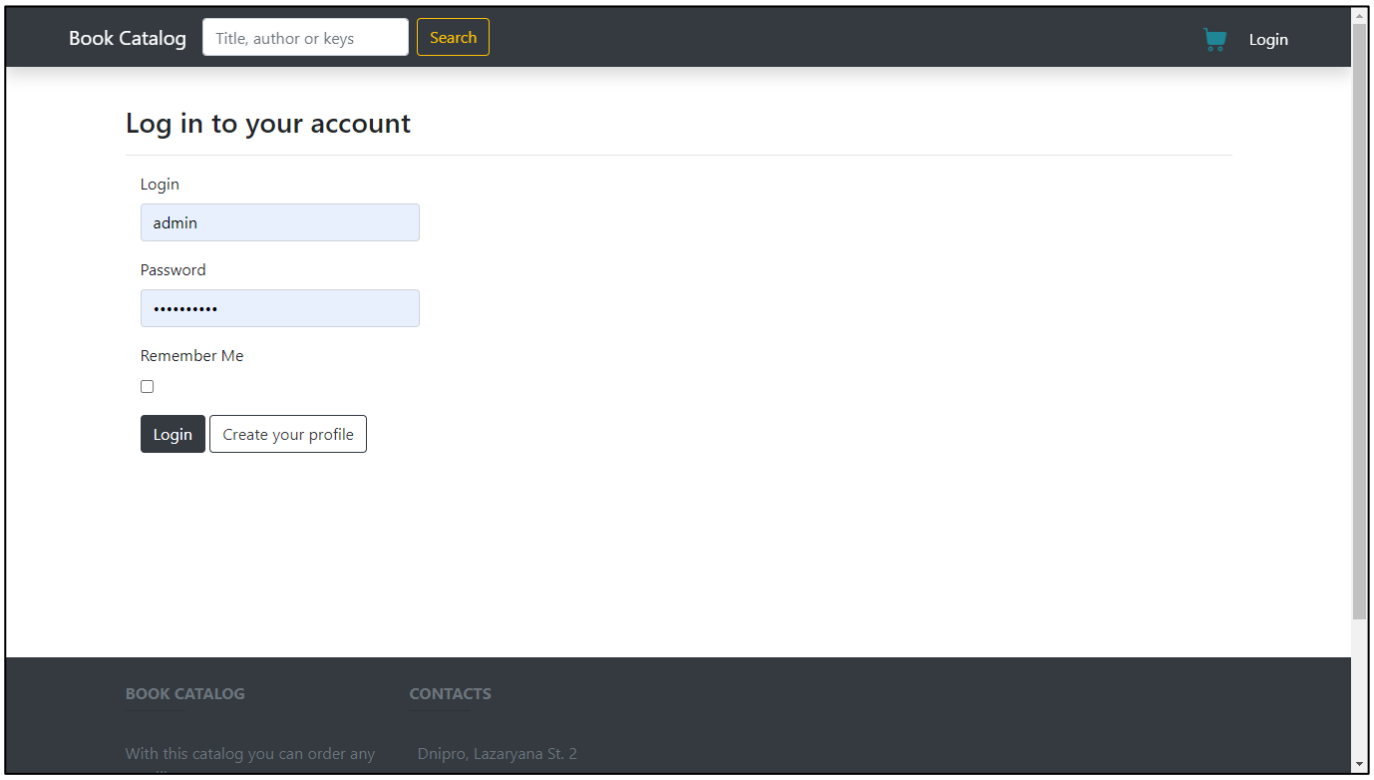

Рисунок 8.12 – Сторінка входу

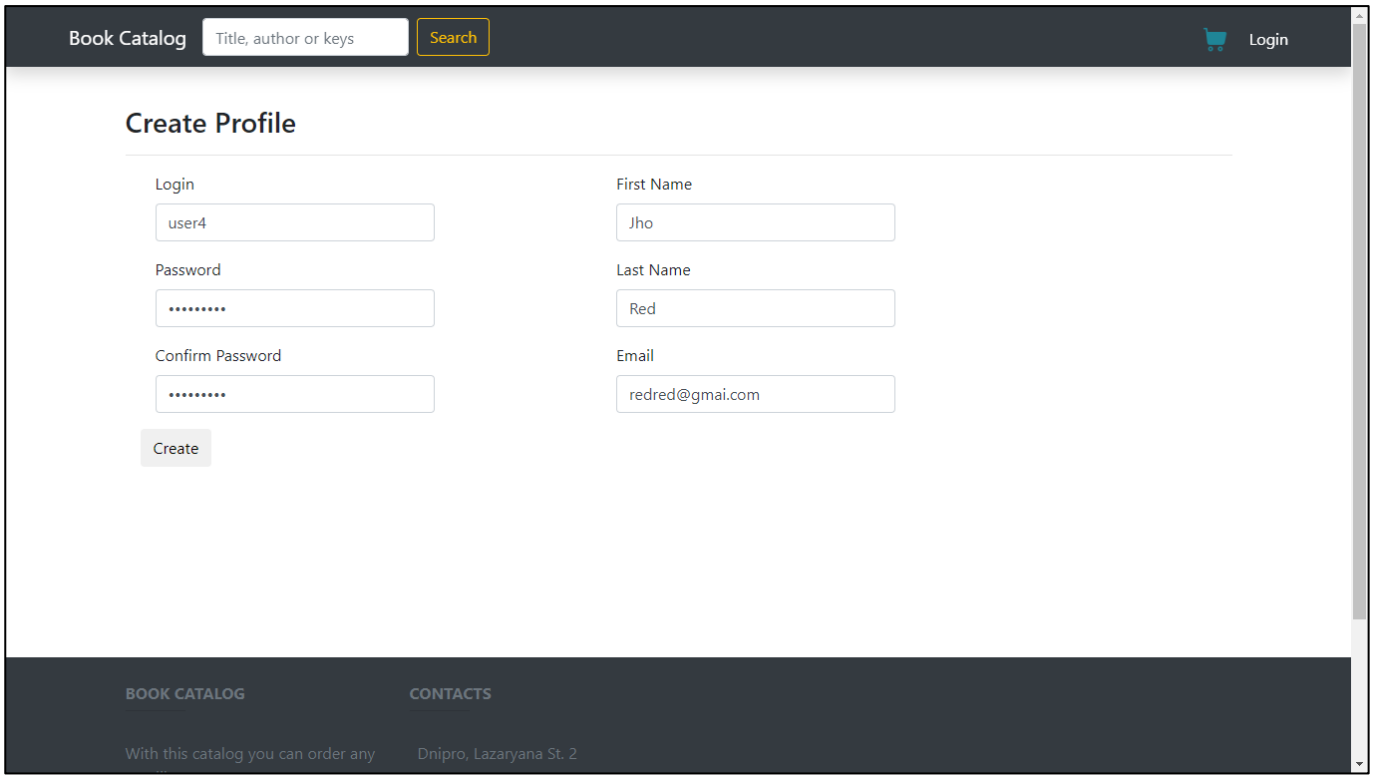

Рисунок 8.13 – Сторінка створення облікового запису

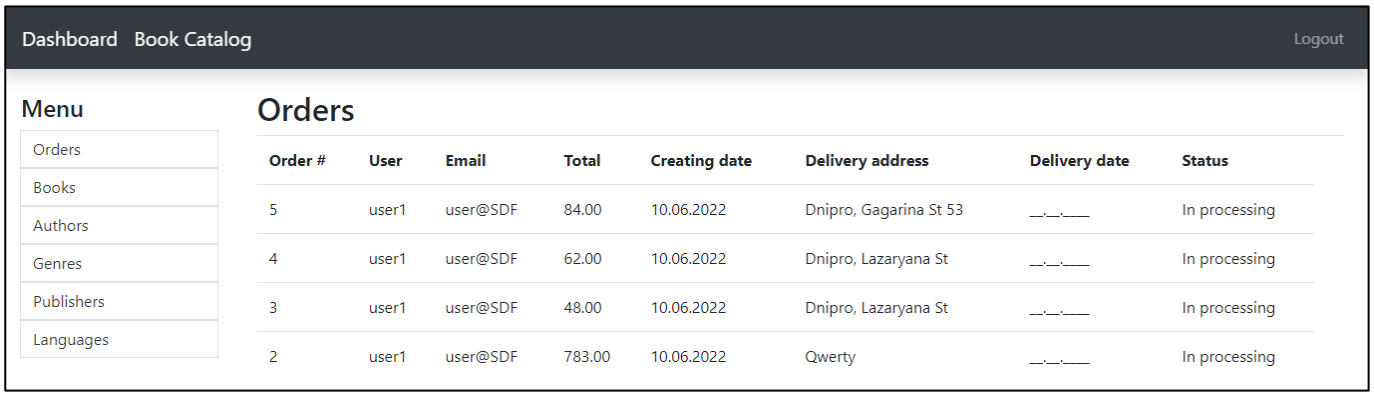

Рисунок 8.14 – Адміністративна панель, сторінка замовлешь читачів

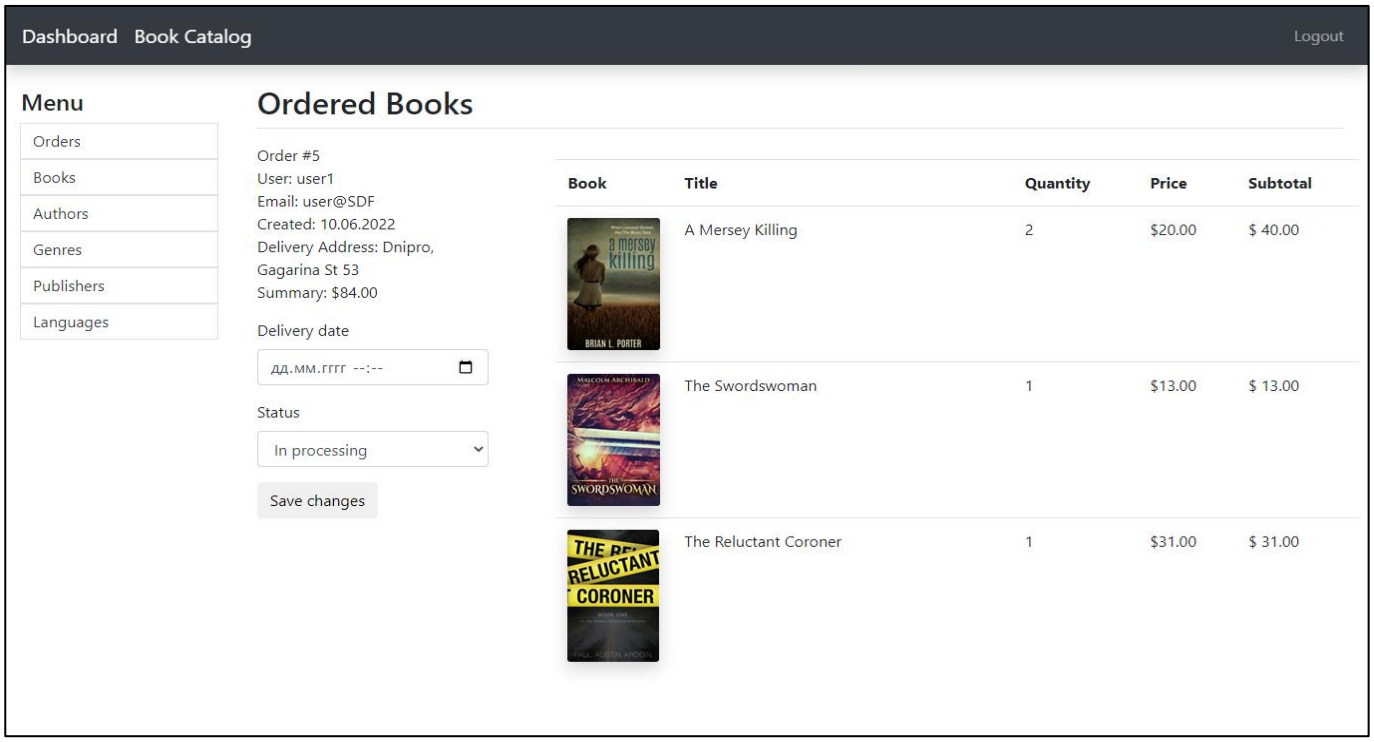

Рисунок 8.15 – Адміністративна панель, перегляд детальної інформації про замовлення читача

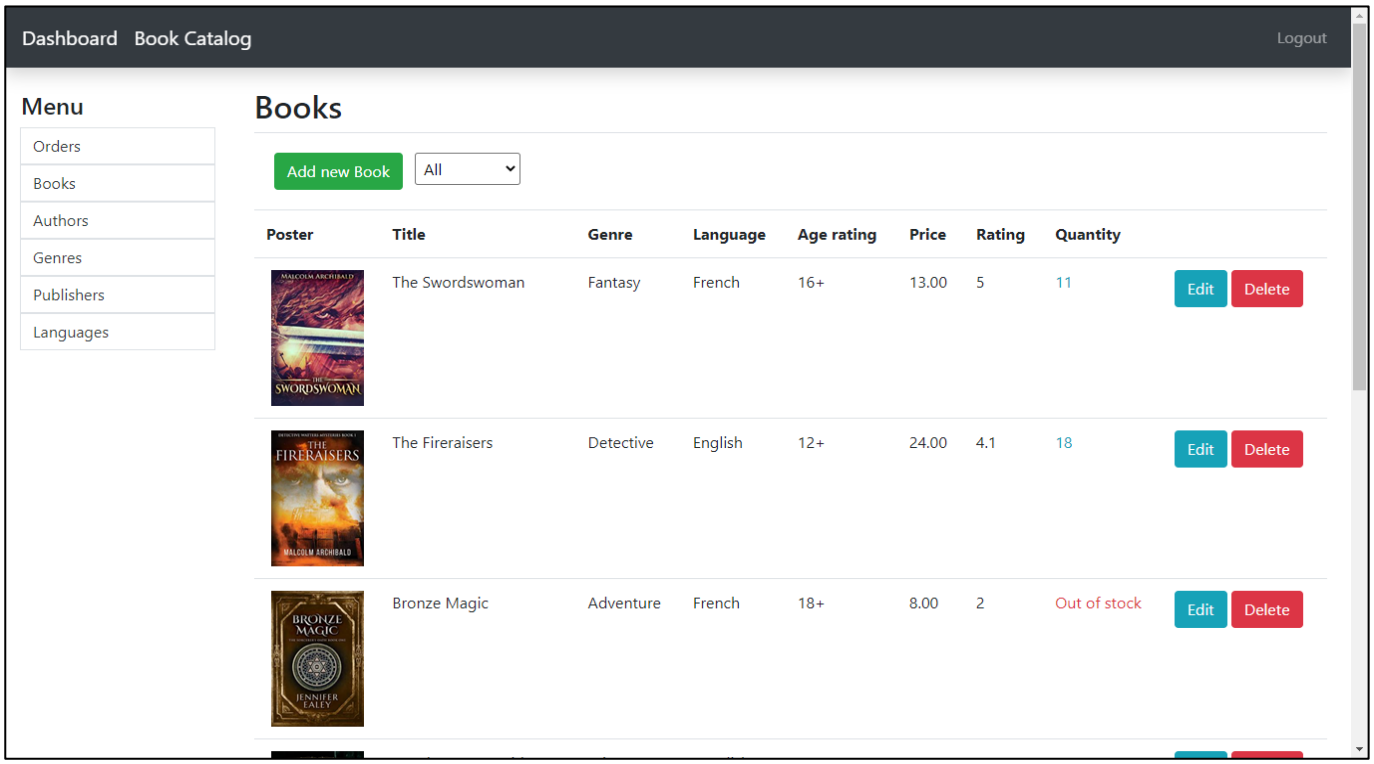

Рисунок 8.16 – Адміністративна панель, сторінка книг

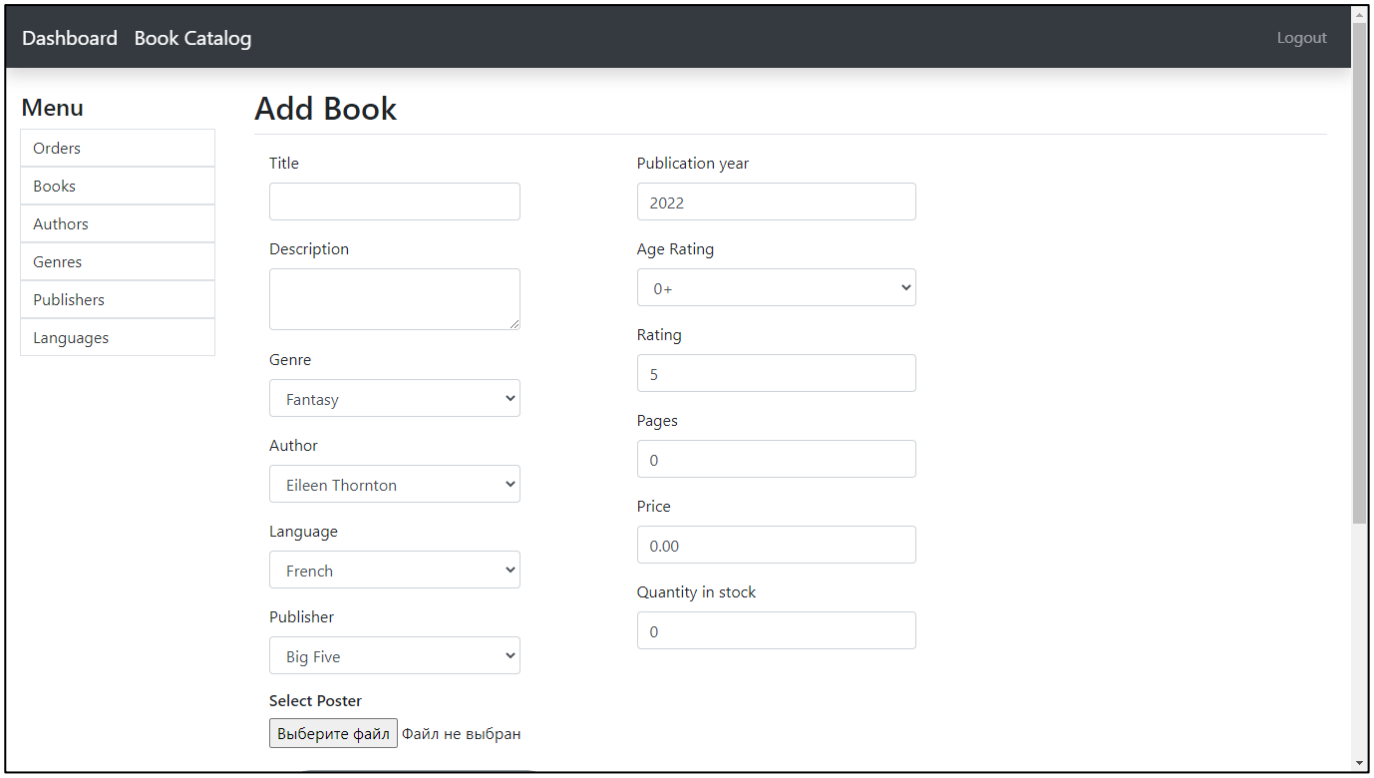

Рисунок 8.17 – Адміністративна панель, додавання нової книги

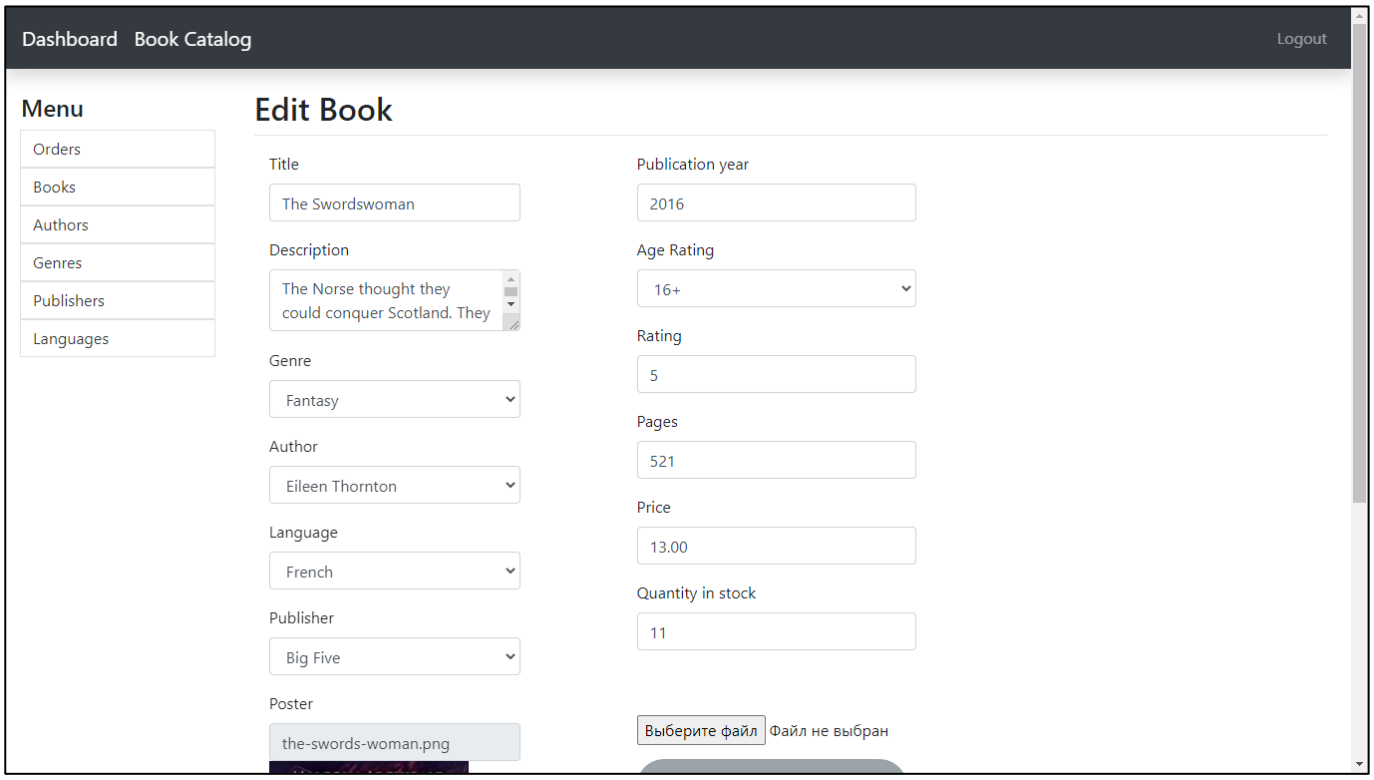

Рисунок 8.18 – Адміністративна панель, редагування книги

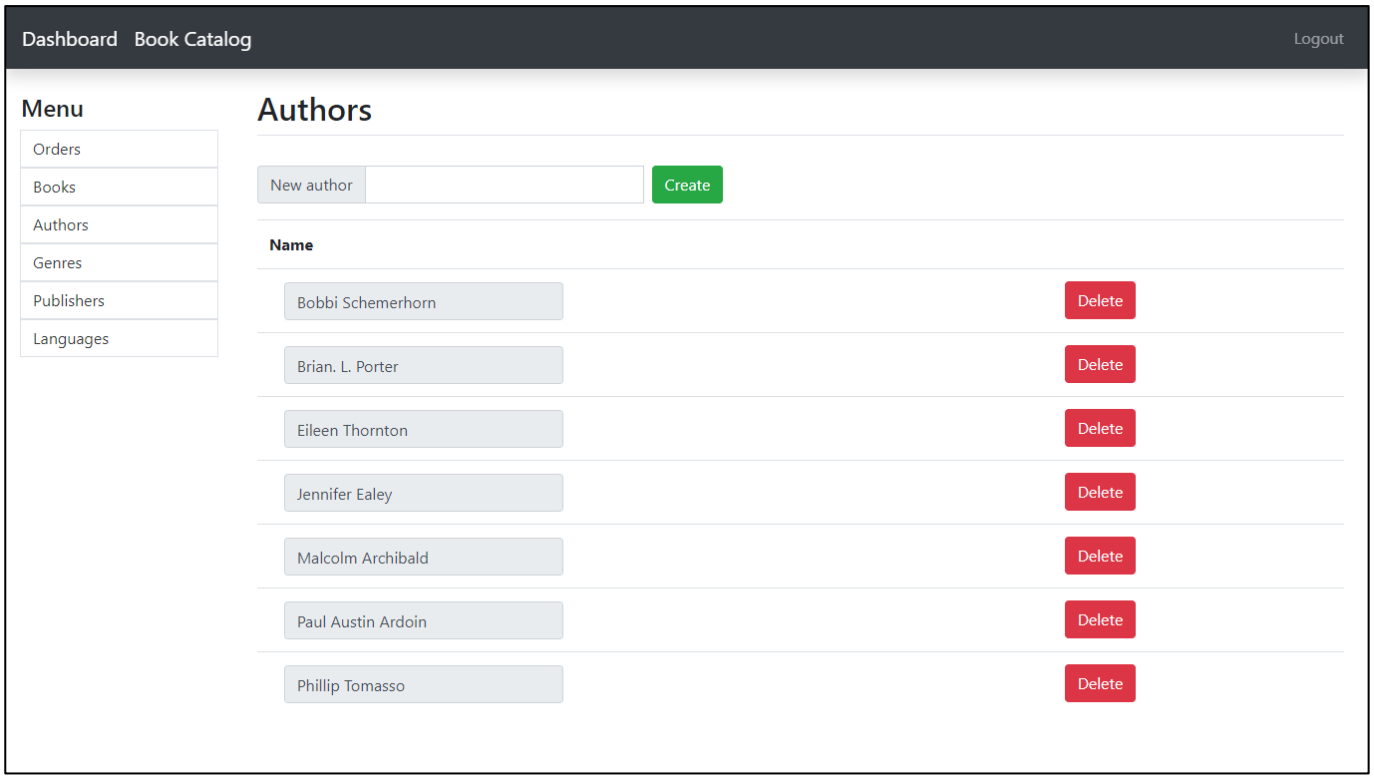

Рисунок 8.19 – Адміністративна панель, робота з авторами

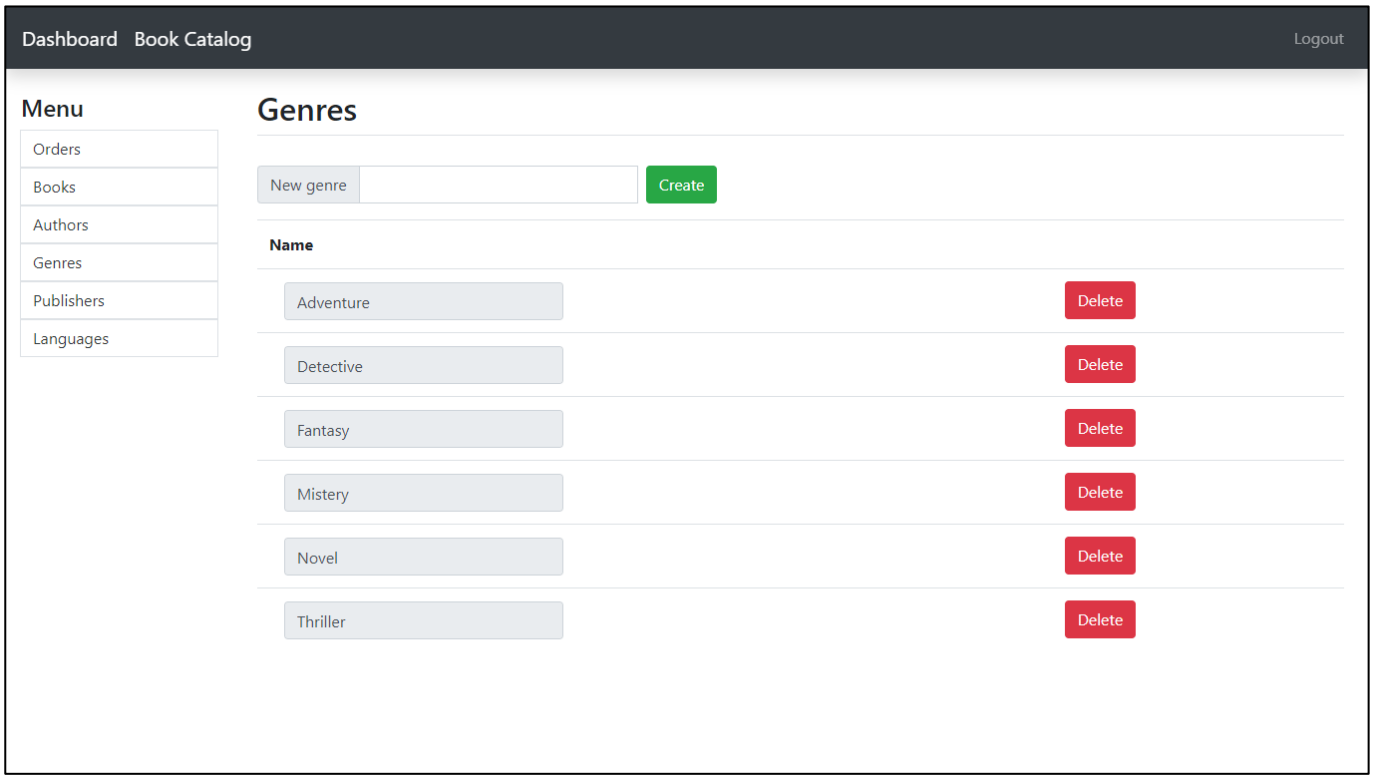

Рисунок 8.20 – Адміністративна панель, робота з жанрами

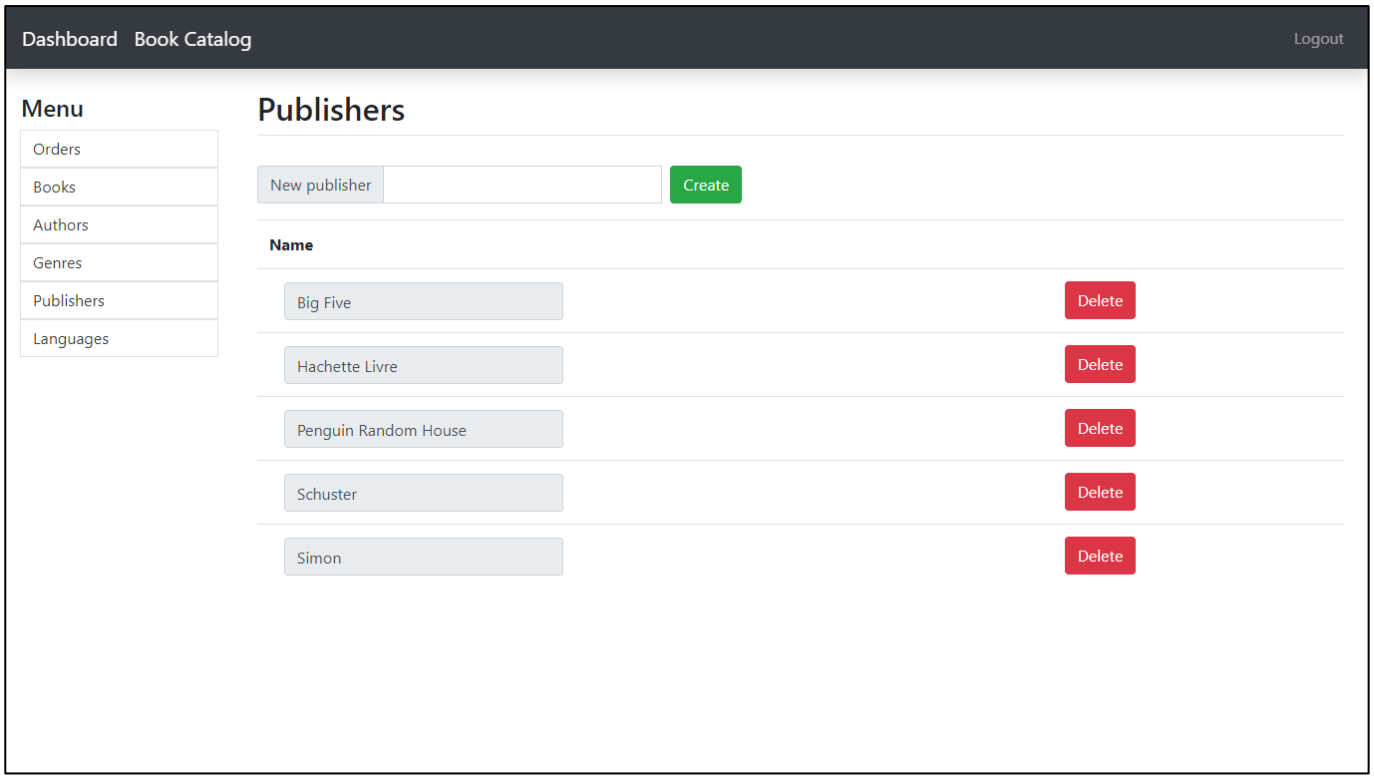

Рисунок 8.21 – Адміністративна панель, робота з видавництвами

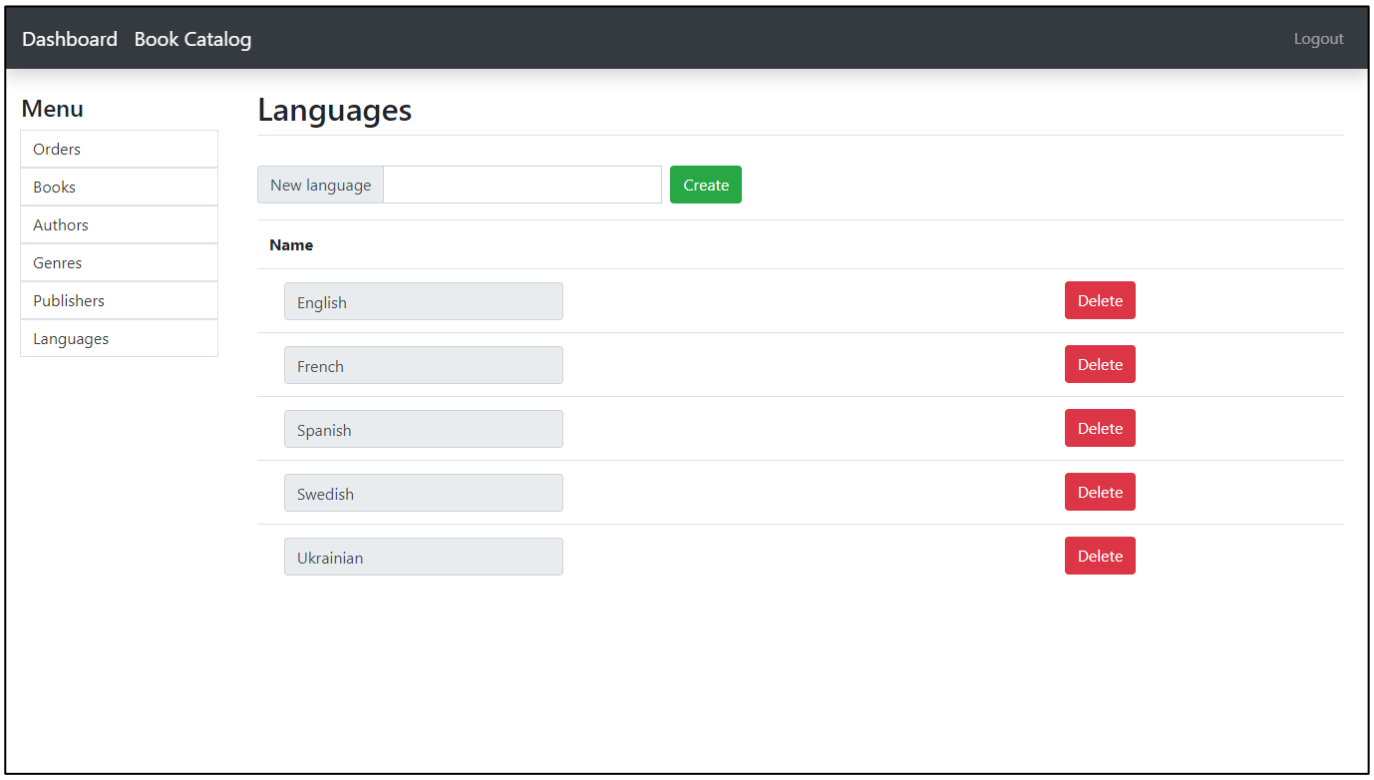

Рисунок 8.22 – Адміністративна панель, робота з мовами

9 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

У разі виникнення питань, пов’язаних з продуктом «ВЕБ-ПОРТАЛ НА ASP.NET ДЛЯ ПРОДАЖУ КНИГ», можна подати зв’язати запит за електронною адресою: andreytancyura@gmail.com. Робочий час надання підтримки: з 9:00 до 18:00 по буднях.

10 ПОВІДОМЛЕННЯ

Для оповіщення користувача про різні ситуації, що можуть виникнути під час його взаємодії з веб-сайтом, використовуються текстові повідомлення, наведені в таблиці 10.1.

Таблиця 10.1 — Повідомлення системи

Текст	Адресат	Ситуація	Рекомендовані дії
1	2	3	4
The author already exist!	Адміністратор	Автор вже існує	Ввести іншу назву жанру
The author has been added	Адміністратор	Додано нового автора	Переглянути
The author name has been changed	Адміністратор	Автора відредаговано	Переглянути
You have deleted a author	Адміністратор	Автора видалено	Переглянути
The genre already exist!	Адміністратор	Автор вже існує	Ввести інше ім’я автора
The genre has been added	Адміністратор	Додано новий жанр	Переглянути
The genre name has been changed	Адміністратор	Жанр відредаговано	Переглянути
You have deleted a genre	Адміністратор	Жанр видалено	Переглянути
The publisher already exist!	Адміністратор	Видавництво вже існує	Ввести іншу назву видавництва
The publisher has been added	Адміністратор	Додано нове видавництво	Переглянути
The publisher name has been changed	Адміністратор	Видавництво відредаговано	Переглянути
You have deleted a publisher	Адміністратор	Видавництво видалено	Переглянути
The language already exist!	Адміністратор	Мова вже існує	Ввести іншу назву мови
The language has been added	Адміністратор	Додано нову мову	Переглянути
The language name has been changed	Адміністратор	Мову відредаговано	Переглянути
You have deleted a language	Адміністратор	Мову видалено	Переглянути
A book with this title already exists!	Адміністратор	Введено вже існуючу назву книги	Ввести іншу назву книгу

Продовження таблиці 10.1

1	2	3	4
This poster is already used in another book!	Адміністратор	Обрано постер існуючої книги	Обрати інший постер
The poster was not uploaded - incorrect image extension!	Адміністратор	Обрано зображення з некоректним розширенням	Обрати зображення з розширенням jpg, jpeg, pjpeg, gif, x-png, png
You have added a book!	Адміністратор	Додано нову книгу	Переглянути
The poster is a required field!	Адміністратор	Постер не обрано	Обрати постер
You edited the book!	Адміністратор	Книгу відредаговано	Переглянути
You have edited the order	Адміністратор	Замовлення відредаговано	Переглянути
Your basket is empty.	Читач, гість	Кошик порожній	Додати книгу до кошика
Available in stock:	Читач, гість	Недостатньо книг на складі	Зменшити кількість книг у кошику або обрати іншу книгу
Nothing found.	Адміністратор, читач, гість	Пошук книг не дав результатів	Виконати пошук з іншими критеріями
Your order is being processed	Читач	Оформлено нове замовлення	Переглянути
Invalid login or password.	Адміністратор, читач	Логін або пароль введене не правильно	Ввести коректний логін та пароль
Login ### is taken	Гість	Логін вже зареєстровано	Ввести інший логін
Now you are registered and can log in.	Гість	Реєстрацію завершено	Переглянути
Passwords do not match.	Читач	Паролі не співпадають	Ввести співпадаючі паролі
You have edited your profile	Читач	Обліковий запис відредаговано	Переглянути
The field is required	Гість читач, адміністратор,	Користувач не заповнив необхідне поле	Заповнити поле