Міністерство освіти і науки України

Національний авіаційний університет

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра інженерії програмного забезпечення

**Курсова робота**

з дисципліни «**Архітектура програмного забезпечення**»

на тему «**Інтернет-магазин**»

Виконали студенти гр. ПІ-220 ст:

Манжула К.О.

Давидович В.І.

Єрофєєв Т.О.

Прийняв:

Дишлевий О.П.

Київ 2020

**Зміст**

Вступ………………………………………………………………………….3

1. Загальне завдання та варіант……………………………………………..4
2. Загальна архітектура проекту…………………………………………….5
3. Описання процесу розробки з використанням системи розподілу версій та відслідковування завдань………………………………………………..16
4. Діаграми рівнів…………………………………………………………...18
5. Деталі реалізації………………………………………………………….20
6. Інструкція з використання застосування……………………………….21
7. Висновки………………………………………………………………….27
8. Використані джерела…………………………………………………….28

**Вступ**

Якщо людині необхідно отримати певний об’єкт чи послугу, то найпростіший спосіб це зробити – це довідатися до відповідного магазину. Магазин дозволяє будь-якому користувачу переглянути каталог з доступними для купівлі товарами та, в обмін на грошові ресурси, отримати обраний товар (товари). Однак, у магазинів є певні недоліки, а саме: якщо у магазині вихідний – то він не працюватиме, теж саме стосується і перерв у магазині. Також, частіш всього, у магазині немає можливості консультацій великої кількості покупців одночасно. До того ж, велику роль у звичайному магазині відіграє людський фактор, тобто задоволеність роботою, характер робітників, ступінь втоми робітників та багато іншого. Усе вищезазначене може призвести до зменшення популярності магазину і, як наслідок, зменшення прибутків та закриття магазину.

Як аналог, набув популярності вид сайту, який надає користувачам повний асортимент товарів різних типів, їх опис та ціни, тобто інтернет-магазини. Будь-який інтернет магазин має ряд переваг перед реальним магазином, а саме: інтернет-магазин працює постійно(24 години в день) без перерв, надає доступ до вітрин з товарами будь-якому користувачеві, коштує менше (підтримка), ніж реальний магазин, людський фактор майже не відіграє ролі, адже магазин є автоматизованим, та багато іншого.

В межах курсової роботи необхідно розробити ASP.NET MVC WEB API - зостосування на тему «Інтернет - магазин», в якому необхідно спроектувати та реалізувати систему класів та скриптів, в основу якої покладено логічну структуру даних, наведену у поставленому завданні, для накопичення та обробки даних домену відповідно варіанту курсової роботи. Код програмної системи має складатись не менш як з трьох частин (окремих проектів) відповідно до багатошарової архітектури системи, де шарами архітектури є: шар (рівень) доступу до даних (DAL), шар бізнес-логіки (BLL), шар представлення (інтерфейс програмної системи) (PL). Також обов’язковим є проектування відповідно до завдання UNIT-тестів, використовування маппінгу, шаблонів проектування Repository та UnitOfWork, систему контролю версій Git та систему відслідковування завдань (Azure).

1. **Загальне завдання та варіант**

В межах курсової роботи необхідно розробити ASP.NET MVC WEB API - зостосування на тему «Інтернет - магазин», а саме інтернет-магазин книг, в якому необхідно спроектувати та реалізувати систему класів, в основу якої покладено логічну структуру даних, наведену у поставленому завданні, для накопичення та обробки даних домену відповідно варіанту курсової роботи. Код програмної системи має складатись не менш як з трьох частин (окремих проектів) відповідно до багатошарової архітектури системи, де шарами архітектури є: шар (рівень) доступу до даних (DAL), шар бізнес-логіки (BLL), шар представлення (інтерфейс програмної системи) (PL). Також обов’язковим є проектування відповідно до завдання UNIT-тестів. Розробку необхідно вести у команді, яка налічує не менше ніж 2 чоловіки (максимум, 3). Для передачі даних між рівнями необхідно використати маппінг. Також, необхідно використовувати систему контролю версій Git та систему відслідковування завдань (Azure).

Програма повинна включати в себе такі функції та можливості:

* Перегляд списку товарів.
* Перегляд відомостей про товари.
* Сортування списку товарів по категоріям.
* Додавання товару до кошика.
* Перегляд вмісту кошика.
* Видалення книг із кошика.
* Відображення стану кошика.
* Оформлення замовлення.
* Додавання книги до каталогу.
* Редагування даних книги.
* Видалення книги.
* Перегляд списку замовлень.
* Видалення замовлення

Доступ до даних повинен бути організований таким чином, щоб збереження даних відбувалося у відповідній БД із застосуванням Entity Framework.

Принцип багатошаровості при будуванні архітектури є обов’язковим до застосування.

Зовнішній вигляд сайту (для користувачів) повинен складатися із списку книг, графічних компонентів для сортування списку книг, додавання книги у кошик, перегляду вмісту кошику, видалення книг з кошику, повернення до перегляду товарів та здійснення замовлення (кнопки).

Зовнішній вигляд сайту (для менеджера/адміністратора) повинен складатися із списку книг та замовлень, графічних компонентів для додавання книги у список, редагування книги, видалення книги та видалення (кнопки).

На вхід до системи потрапляє така інформація, як ім’я користувача, логін/пароль для адміністратора/менеджера, номер книги, назва книги, опис книги, автор книги, жанр книги, ціна книги, номер замовлення.

Дані про товари (каталог) та замовлення повинні зберігатися у БД BookDbContext.

Цільова аудиторія: програма розрахована на усі вікові категорії користувачів персональних комп’ютерів.

1. **Загальна архітектура проекту**

ASP.NET MVC Web API - додаток «BOOK STORE» включає в себе проекти, представлені у таблиці 1.

Таблиця 1 – Проекти програми «BOOK STORE».

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
| 1 | BookStorePL | Проект, який містить візуальну частину сайту. |
| 2 | BookStoreBLL | Проект бізнес-операцій над даними, отриманими з BookStoreDAL. |
| 3 | BookStoreDAL | Проект доступу к даним, які містяться у відповідній БД. |
| 4 | BookStoreTests | Проект звичайних та NUnit модульних тестів |

Проект представлення (BookStorePL): призначений для отримання даних від користувача (користувач/менеджер/адміністратор), передачі цих даних до проекту бізнес-логіки та отримання оброблених даних. Налічує у собі директорії, представлені у таблиці 2.

Таблиця 2 – Директорії проекту BookStorePL

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
|  |  |  |
| 1 | App\_Data | Містить файли, ресурси і бази даних, що використовуються додатком. |
| 2 | App\_Start | Зберігає ряд статичних файлів, які містять логіку ініціалізації програми при запуску. Містить класи IdentityConfig.cs, NinjectWebCommon.cs, RouteConfig.cs, WebApiConfig.cs. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 3 | Controllers | Містить файли класів контролерів. Містить класи-контролери AdminController.cs, BooksController.cs, HomeController.cs, OrdersController.cs |
| 4 | fonts | Зберігає додаткові файли шрифтів, використовуваних додатком. |
| 5 | Models | Містить моделі для відображення даних. Включає в себе класи BookVM.cs, та OrderVM.cs |
| 6 | Scripts | Каталог зі скриптами і бібліотеками на мові javascript. |
| 7 | Views | Каталог із представленнями (Views). Включає в себе каталоги Admin, Book, Home, Prep, Shared. |
| 8 | Admin | Каталог для представлень адміністратора. Включає в себе представлення AdminBookAdd.cshtml, AdminBookList.cshtml, AdminOrderList.cshtml, Index.cshtml. |
| 9 | Book | Директорія для тестових представлень. Пуста, та у курсовій роботі не використовується. |
| 10 | Home | Директорія для представлень сайту для користувача. Включає в себе представлення BookCart.cshtml, BookList.cshtml, CartWiget.cshtml, Checkout.cshtml, Index.cshtml, ThankYou.cshtml. |
| 11 | Prep | Директорія для тестових представлень. Включає в себе представлення Index.cshtml та Orders.cshtml. У курсовій роботі не використовується. |
| 12 | Shared | Папка для загального представлення. Включає в себе представлення-шаблон \_Layout.cshtml. |

Опис створених класів, скриптів та представлень проекту BookStorePL представлений у таблиці 3.

Таблиця 3 – Класи, скрипти та представлення проекту BookStorePL

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
|  |  |  |
| 1 | AdminController.cs | MVC контролер для відображення меню роботи для менеджера/адміністратора. |
| 2 | BooksController.cs | MVC Web Api контролер для здійснення основного функціоналу для роботи із списком книг. |
| 3 | HomeController.cs | MVC контролер для відображення меню роботи для користувача. |
| 4 | OrdersController.cs | MVC Web Api контролер для здійснення основного функціоналу для роботи із списком замовлень. |
| 5 | NinjectResolver.cs | Клас для створення ядра Ninject . Використовує клас NinjectScope. |
| 6 | NinjectScope.cs | Допоміжний клас для класу NinjectResolver.cs. |
| 7 | OrderVM.cs | Модель для відображення інформації про замовлення. |
| 8 | BookVM.cs | Модель для відображення інформації про книгу. |
| 9 | \_references.js | Скрипт для регістрації інших скриптів. |
| 10 | storeAdminController.js | Скрипт для аутендифікації користувача та здійснення функціоналу адміністратора/менеджера. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 11 | storeAdminModel.js | Скрипт для позначення моделі адміністратора/менеджера. |
| 12 | storeAjax.js | Скрипт для здійснення Ajax-запитів. |
| 13 | storeBooksController.js | Скрипт для оформлення Ajax-запитів, пов’язаних з книгами. |
| 14 | storeCommonController.js | Скрипт для заповнення Ajax-запитів, пов’язаних з аутендифікацією. |
| 15 | storeCustomerController.js | Скрипт для здійснення функціоналу для користувача, тобто, наприклад, сортування книг по категорії. |
| 16 | storeCustomerModel.js | Скрипт для позначення моделі користувача. |
| 17 | storeModel.js | Скрипт для позначення моделі сайту. |
| 18 | storeOrdersController.js | Скрипт для заповнення Ajax-запитів пов’язаних з замовленнями. |
| 19 | AdminBookAdd.cshtml | Представлення для додавання/оновлення даних книги. |
| 20 | AdminBookList.cshtml | Представлення для відображення каталогу книг. |
| 21 | AdminOrderList.cshtml | Представлення для відображення списку замовлень. |
| 22 | Index.cshtml | Представлення для авторизації та менеджменту книг та замовлень. |
| 23 | BookCart.cshtml | Представлення для кошику для купівель. |
| 24 | BookList.cshtml | Представлення для відображення однієї книги. |
| 25 | CartWiget.cshtml | Представлення для відображення стану кошику. |
|  |  |  |
| 26 | Checkout.cshtml | Представлення для оформлення замовлення. |
| 27 | Index.cshtml | Загальне представлення для відображення списку книг, категорій сортування, стану кошика. |
| 28 | ThankYou.cshtml | Представлення для повідомлення про здійснення покупки. |
| 29 | \_Layout.cshtml | Шаблон для представлень, який використовує різні бібліотеки (jquery, knockout,bootstrap), а також скрипти, зазначені у цій таблиці вище. |

Проект бізнес-логіки (BookStoreBLL): призначений для отримання даних від представлення (BookStorePL), здійснення бізнес-операцій над даними та повертання оброблених даних. Налічує у собі директорії, представлені у таблиці 4.

Таблиця 4 – Директорії проекту BookStoreBLL.

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
| 1 | DTO | Директорія для зберігання проміжних сутностей. Включає в себе класи BookDTO.cs та OrderDTO. |
| 2 | Infrastructure | Директорія для інфраструктури проекту. Включає в себе класи ServiceModule.cs та ValidationException.cs. |
| 3 | Interfaces | Директорія для інтерфейсів сервісів. Включає в себе IBookService.cs та IOrderService.cs. |
| 4 | Services | Директорія для сервісів проекту. Включає в себе класи BookSercice.cs та OrderService.cs. |

Опис створених класів та інтерфейсів проекту BookStoreBLL представлений у таблиці 5.

Таблиця 5 – Класи та інтерфейси проекту BookStoreBLL.

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
| 1 | BookDTO.cs | Проміжна сутність, яка позначає книгу. |
| 2 | OrderDTO.cs | Проміжна сутність, яка позначає замовлення. |
| 3 | ServiceModule.cs | Спеціальний клас, який слугує для організації зіставлення залежностей. (Ninject) |
| 4 | ValidationException.cs | Користувацьке виключення. Слугує для валідації даних на рівні. |
| 5 | IBookService.cs | Інтерфейс для BookService. |
| 6 | IOrderService.cs | Інтерфейс для OrderService. |
| 7 | BookService.cs | Клас для здійснення бізнес-операцій з книгами. |
| 8 | OrderService.cs | Клас для здійснення бізнес-операцій з замовленнями. |
| 9 | BookStoreTestHelpers.cs | Допоміжний клас для здійснення звичайних unit-тестів. Цей клас використовує принцип симуляції. |

Проект доступу до даних (BookStoreDAL): містить всі моделі даних магазину, що зберігається у БД, а також класи, через які йде взаємодія з БД. Налічує у собі директорії, представлені у таблиці 6.

Таблиця 6 – Директорії проекту BookStoreDAL.

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
| 1 | Abstract | Директорія для зберігання депозитаріїв. Включає в себе інтерфейси IRepository.cs, та IUnitOfWork.cs. |
| 2 | Concrete | Директорія для контексту та ініціалізатора БД. Включає в себе класи BookDBContext.cs та BookDbInitializer.cs. |
| 3 | Entities | Директорія для зберігання сутностей БД. Включає в себе класи Book.cs, Order.cs та OrderLine.cs. |
| 4 | Repositories | Директорія для зберігання реалізації інтерфейсів (директорія Abstract). Включає в себе класи BookRepository.cs, EFUnitOfWork.cs та OrderRepository.cs, |
| 5 | Identity | Директорія для зберігання інфраструктури проекту. Містить класи StoreAuthProvider.cs, StoreIdentityDBContext.cs, StoreIdentityDbInitializer.cs, StoreRole.cs, StoreRoleManager.cs, StoreUser.cs, StoreUserManager.cs. |

Опис створених класів та інтерфейсів проекту BookStoreDAL представлений у таблиці 7.

Таблиця 7 – Класи та інтерфейси проекту BookStoreDAL.

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
|  |  |  |
| 1 | IRepository.cs | Інтерфейс загального репозиторію. |
| 2 | IUnitOfWork.cs | Інтерфейс для реалізації шаблону проектування UnitOfWork. |
| 3 | BookDBContext.cs | Контекст БД інтернет-магазину. |
| 4 | BookDbInitializer.cs | Ініціалізатор для БД інтернет-магазину. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 5 | Book.cs | Модель книги. |
| 6 | Order.cs | Модель замовлення. |
| 7 | OrderLine.cs | Модель рядка замовлення. (Допоміжна модель для Order.cs) |
| 8 | BookRepository.cs | Реалізація інтерфейсу IRepository для книг. |
| 9 | EFUnitOfWork.cs | Реалізація інтерфейсу IUnitOfWork. |
| 10 | OrderRepository.cs | Реалізація інтерфейсу IRepository та IRepositoryOrders для замовлень. |
| 11 | StoreAuthProvider.cs | Провайдер ролей для сайта. |
| 12 | StoreIdentityDBContext.cs | Контекст БД ролей. |
| 13 | StoreIdentityDbInitializer.cs | Ініціалізатор БД ролей. |
| 14 | StoreRole.cs | Модель ролі користувача. |
| 15 | StoreRoleManager.cs | Менеджер ролей. Відповідає за створення ролі. |
| 16 | StoreUser.cs | Модель користувача. |
| 17 | StoreUserManager.cs | Менеджер користувачів. Відповідає за створення користувача. |

Проект unit-тестів (BookStoreTests): містить unit(модульні)-тести для проекту бізнес-логіки (BookStoreBLL). Налічує у собі директорію NUnit , яка включає в себе NUnit – тести. Модульні тести проекту BookStoreTests представлені у таблиці 8.

Таблиця 8 – Класи модульних тестів проекту BookStoreTests.

|  |  |  |
| --- | --- | --- |
| № | Назва | Опис |
|  |  |  |
| 1 | Can\_Add\_New\_Book | Звичайний тест для перевірки функції додавання нової книги у каталог. Використовує клас BookStoreTestHelpers. |
| 2 | Can\_Change\_Book\_With\_Id\_3 | Звичайний тест для перевірки функції редагування даних книги з вказаним номером. Використовує клас BookStoreTestHelpers. |
| 3 | Can\_Delete\_Book\_With\_Id\_3 | Звичайний тест для перевірки функції видалення книги з вказаним номером з каталогу. Використовує клас BookStoreTestHelpers. |
| 4 | Can\_Get\_All\_Books | Звичайний тест для перевірки функції отримання повного списку книг. Використовує клас BookStoreTestHelpers. |
| 5 | Can\_Get\_Book\_With\_Id\_2 | Звичайний тест для перевірки функції отримання книги із вказаним номером. Використовує клас BookStoreTestHelpers. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 6 | Can\_Add\_New\_Order | Звичайний тест для перевірки функції додавання нового замовлення у каталог. Використовує клас BookStoreTestHelpers. |
| 7 | Can\_Delete\_Order\_With\_Id\_3 | Звичайний тест для перевірки функції видалення замовлення із вказаним номером. Використовує клас BookStoreTestHelpers. |
| 8 | Can\_Get\_All\_OrderLines\_With\_Order\_With\_Id\_2 | Звичайний тест для перевірки функції отримання всіх рядків замовлення з вказаним номером. Використовує клас BookStoreTestHelpers. |
| 9 | Can\_Get\_All\_Orders | Звичайний тест для перевірки функції отримання списку всіх замовлень. Використовує клас BookStoreTestHelpers. |
| 10 | Can\_Get\_Order\_With\_Id\_2 | Звичайний тест для перевірки функції отримання замовлення із вказаним номером. Використовує клас BookStoreTestHelpers. |
| 11 | Can\_BookService\_Add\_New\_Book | NUnit тест для перевірки функції додавання нової книги у каталог. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 12 | Can\_ BookService\_Delete\_Book\_With\_Id\_2 | NUnit тест для перевірки функції видалення книги з вказаним номером. |
| 13 | Can\_ BookService\_Edit\_Book\_with\_Id\_2 | NUnit тест для перевірки функції редагування даних книги з вказаним номером. |
| 14 | Can\_ BookService\_Get\_All\_Books | NUnit тест для перевірки функції отримання всіх книг з каталогу. |
| 15 | Can\_BookService\_Get\_Book\_With\_Id\_2 | NUnit тест для перевірки функції отримання книги з вказаним номером. |
| 16 | Can\_OrderService\_Add\_New\_Order | NUnit тест для перевірки функції додавання нового замовлення у каталог. |
| 17 | Can\_ OrderService\_Delete\_Order\_With\_Id\_1 | NUnit тест для перевірки функції видалення замовлення із вказаним номером. |
| 18 | Can\_ OrderService\_ Get\_All\_OrderLines\_With\_Order\_With\_Id\_1 | NUnit тест для перевірки функції отримання всіх рядків замовлення із вказаним номером. |
| 19 | Can\_ OrderService\_Get\_All\_Orders | NUnit тест для перевірки функції отримання всіх замовлень. |
|  |  |  |
| 20 | Can\_OrderService\_Get\_Order\_With\_Id\_1 | NUnit тест для перевірки функції отримання замовлення із вказаним номером. |

1. **Описання процесу розробки з використанням системи розподілу версій та відслідковування завдань.**

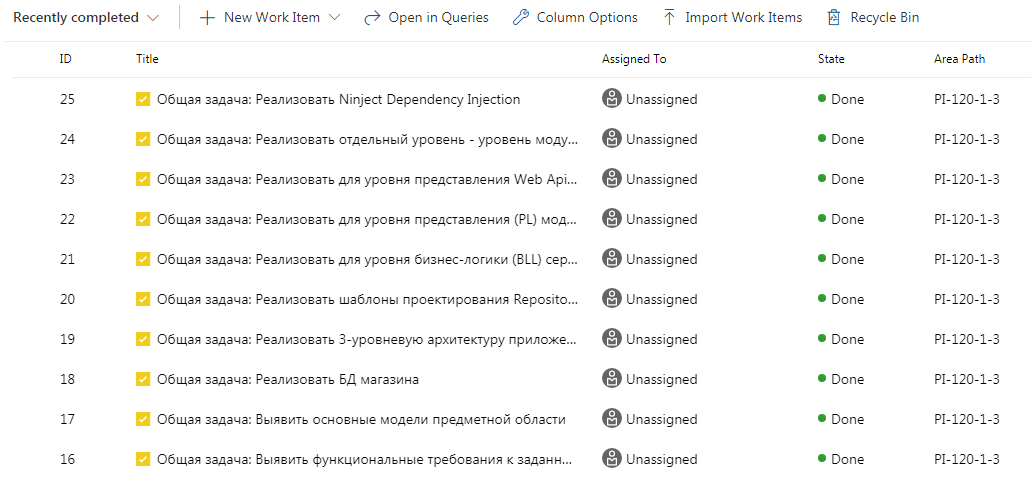
Для розробки курсової роботи було обрано ітеративну модель розробки, яка передбачає можливість паралельно виконувати ряд поставлених перед командою задач з неперервним аналізом результатів та корегуванням попередніх етапів роботи. Відповідно до обраної моделі була поставлена задача - розробити курсову роботу. Відповідно до принципу декомпозиції, поставлене глобальне завдання було розділено на окремі підзавдання. Список цих підзавдань можна побачити у таблиці 9.

Таблиця 9 – Підзавдання глобального (основного) завдання.

|  |  |
| --- | --- |
| № | Підзавдання |
|  |  |
| 1 | Виявити функціональні вимоги до заданого ПО (курсовий проект). |
| 2 | Виявити основні моделі предметної області |
| 3 | Реалізувати БД інтернет-магазину |
| 4 | Реалізувати 3-рівневу архітектуру додатку (PL, BLL, DAL) |
| 5 | Реалізувати шаблони проектування Repository та UnitOfWork. (DAL) |
| 6 | Реалізувати для рівня бізнес-логіки (BLL) сервіси та моделі для взаємодії з рівнем доступу до даних (DAL). |
| 7 | Реалізувати для рівня представлення моделі для взаємодії з рівнем бізнес-логіки та загальну інфраструктуру. |

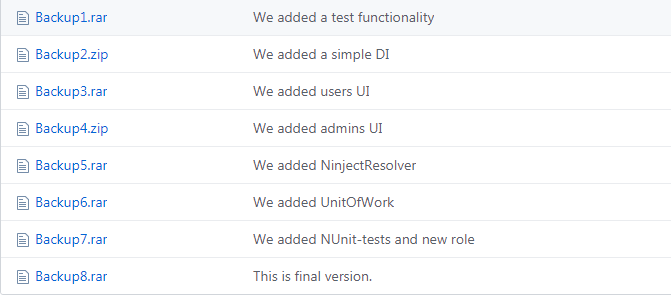
|  |  |
| --- | --- |
|  |  |
| 8 | Реалізувати для рівня представлення (PL) Web Api функціонал. Web api контролери не повинні повертати View. |
| 9 | Реалізувати окремий рівень – рівень модульних тестів (NUnit). Тестуватися повинна бізнес логіка. |
| 10 | Реалізувати Ninjеct Dependency Injection. |

Виконання цих підзавдань відслідковувалося у системі відслідковування завдань Azure. Тип поставлених завдань – Task. Це можна побачити на рисунку 1.

Рисунок 1 – Структура віддаленого репозиторію TermPaper.

Відповідно до вищезазначених підзавдань, була використана система контролю версій Git. Був створений локальний репозиторій, в який під час розробки курсової роботи додавалися робочі версії проекту, відповідно до виконаного підзавдання чи групи завдань. Потім, локальний репозиторій був зв’язаний із віддаленим репозиторієм, створеним за допомогою інтернет-ресурсу Github. Як наслідок, віддалений репозиторій TermPaper має структуру, зображену на рисунку 2.

Рисунок 2 – Структура віддаленого репозиторію TermPaper.

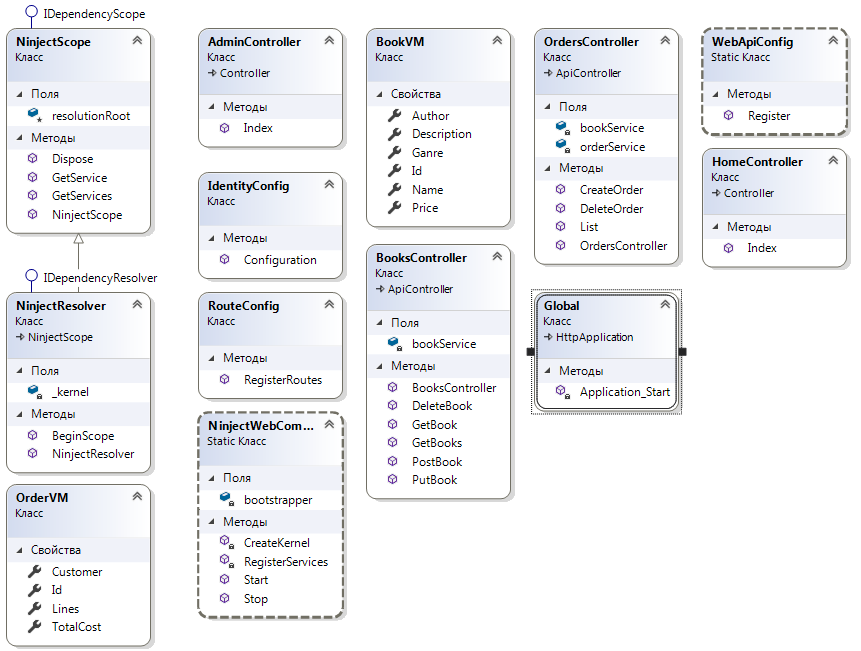


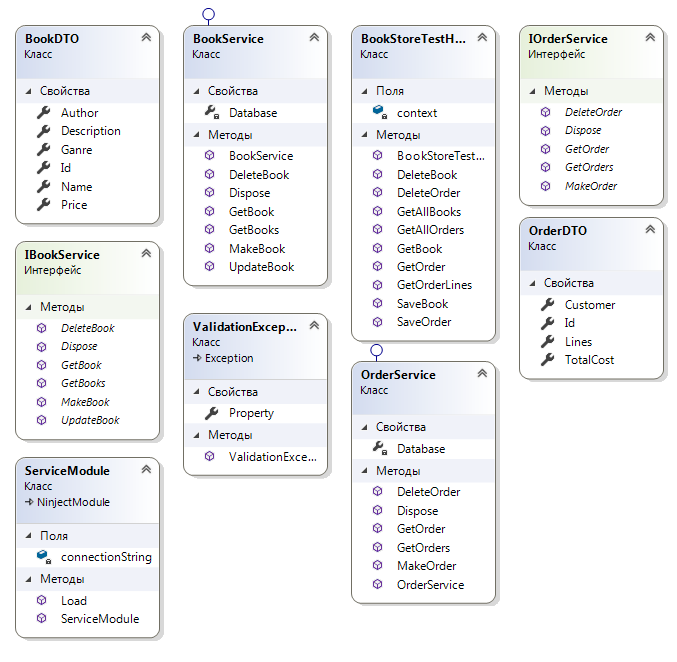
Всі робочі версії продукту після виконання поставлених підзавдань були додані до репозиторію у вигляді \*.rar або \*.zip архівів, з метою забезпечення зручності у завантаженні при необхідності. Зважаючи на те, що курсовий проект розроблявся по заданим завданням (крокам), не було необхідності робити розгалуження на головній гілці. Відповідно до цього, репозиторій має лише одну пряму головну гілку, яка складається з 8-ми елементів.

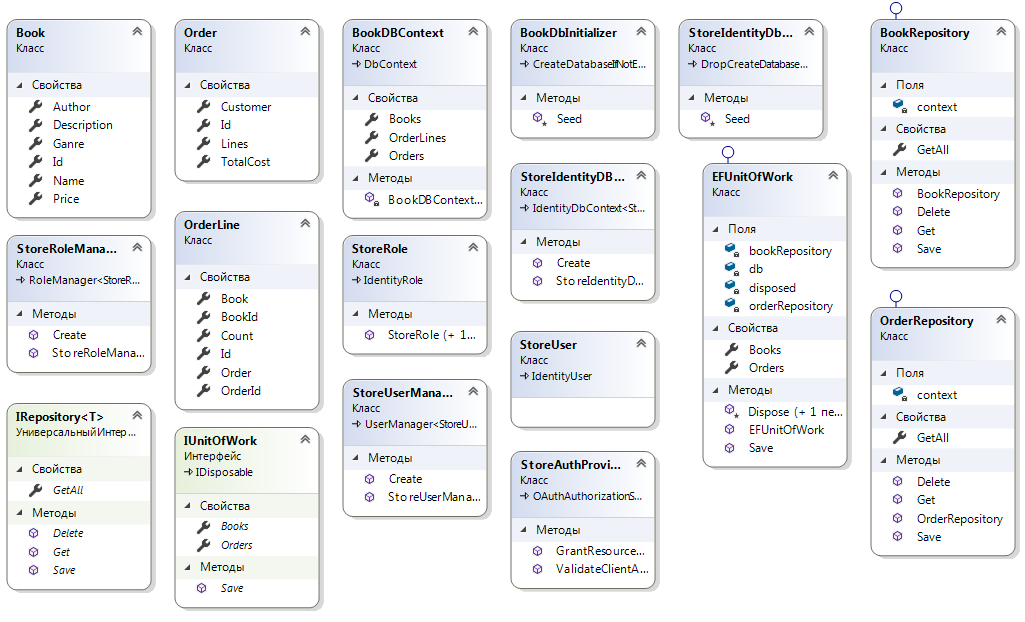
1. **Діаграми рівнів**

Діаграми рівнів послідовно представлені на рисунках 3, 4, 5.

Рисунок 3 – Діаграма класів рівня представлення.

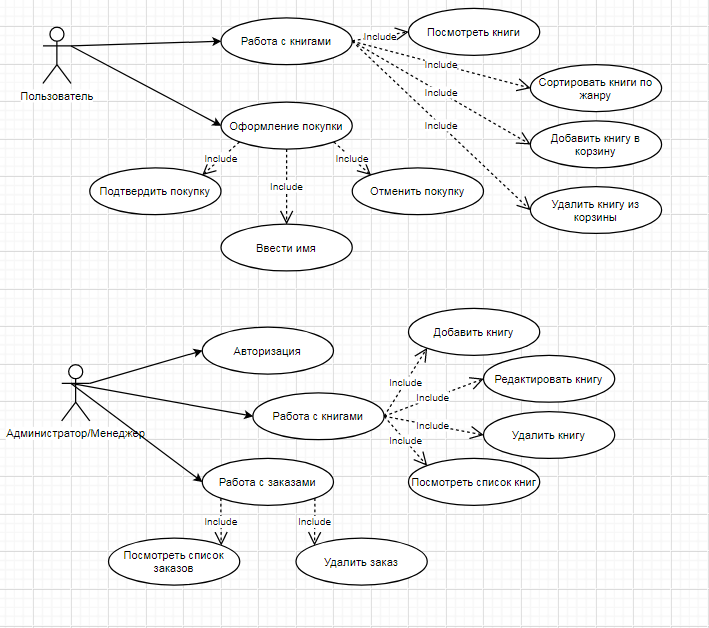


Рисунок 4 – Діаграма класів рівня бізнес-логіки.

Рисунок 5 – Діаграма класів рівня доступу до даних

Uml-діаграму прецедентів (Use-Case) можна побачити на рисунку 6.

Рисунок 6 – Use-Case діаграма інтернет-магазину «BOOK STORE»



1. **Деталі реалізації**

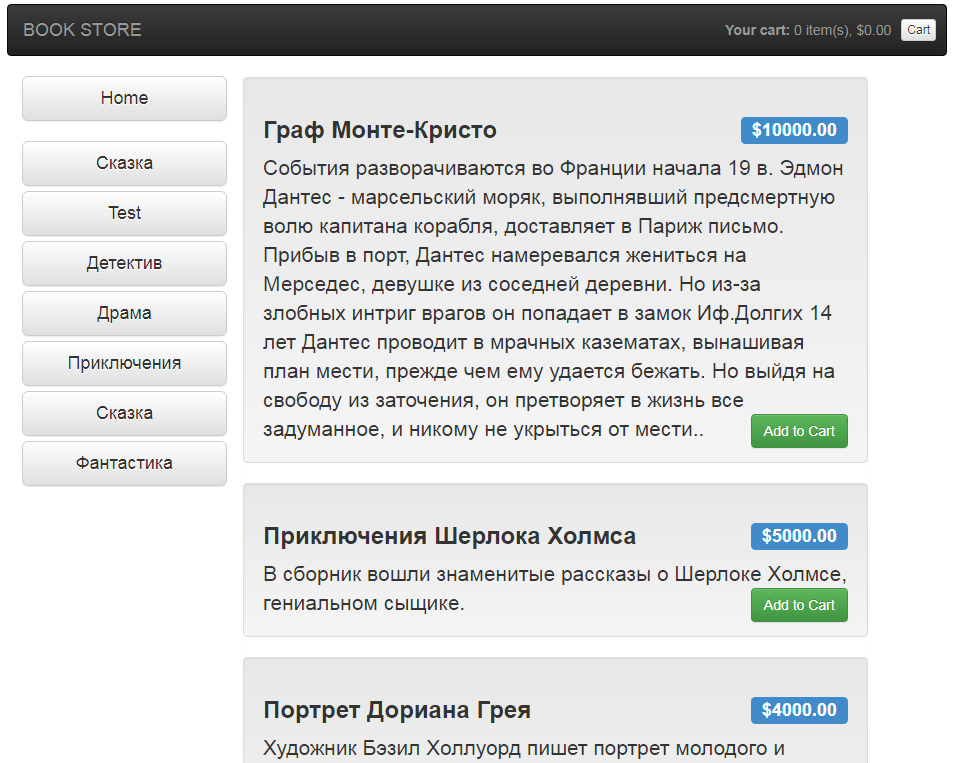
До деталей реалізації інтернет – магазину «BOOK STORE» відносяться:

* Новий жанр для сортування і, відповідно, кнопка для нього створюються автоматично при створенні книги із новим жанром.
* Є можливість переглянути стан кошика у будь який момент.
* Після додавання книги у кошик є можливість оформити замовлення, або продовжити процес перегляду книг з метою додавання необхідної книги до кошику.
* Для оформлення замовлення необхідно ввести ім’я того, хто здійснює купівлю. Без цього замовлення не буде виконане. (Це полегшена версія авторизації).
* Для закінчення оформлення замовлення необхідно оновити сторінку. Якщо оновити сторінку раніше, то замовлення буде скасовано.
* Для доступу до функціонала адміністратора/менеджера необхідно ввести у адресний рядок замість слова Home слово Admin. Для повернення до режима звичайного користувача необхідно замість слова Admin у адресного рядка ввести слова Home.
* Також, для доступу до функціонала адміністратора/менеджера необхідно ввести логін та пароль (Повна форма авторизації). За замовчуванням, логін для адміністратора – admin, для менеджера – manager, а пароль однаковий для обох ролей – secret.
* Доступ до меню книг чи замовлень можна отримати, натиснувши на відповідні кнопки. Додавання та редагування книги можна здійснити, натиснувши на одну й ту саму кнопку.
* Для редагування даних книги необхідно, крім заповнення інших полей, у поле Id ввести номер книги, що редагується.
* Для додавання книги у каталог (БД) необхідно заповнити всі поля, окрім поля Id.
* Адміністратор/менеджер не може додавати чи редагувати замовлення. Тобто можливе лише видалення замовлення (ніби його здійснення).
* Оновлення сторінки у режимі адміністрування призведе до необхідності повторної авторизації.
* Сайт не використовує cookies.
* Якщо база даних сайту не існувала раніше, то вона автоматично створюється та заповнюється даними.

1. **Інструкція з використання застосування**

При запуску додатку користувач автоматично переходить до режиму звичайного користувача, тобто до режиму перегляду товарів, як показано на рисунку 7.

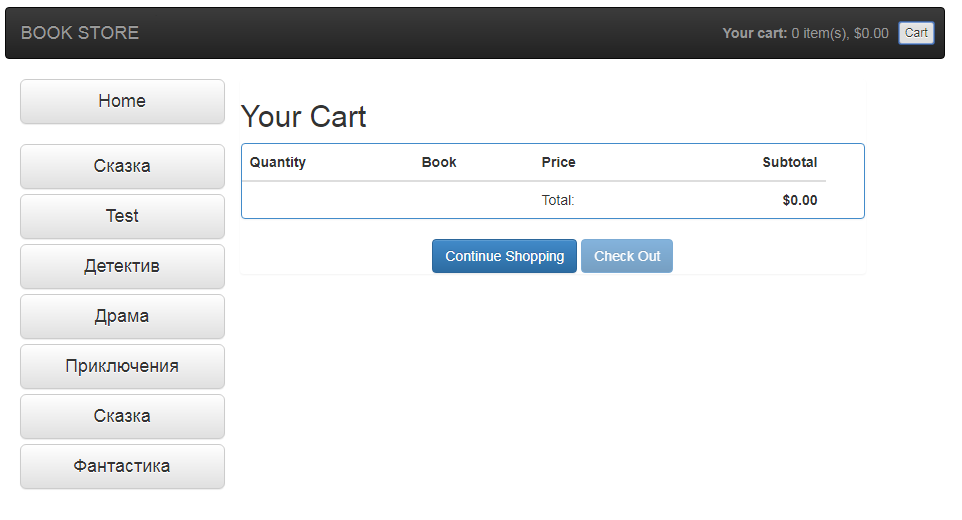
Рисунок 7 – Перегляд товарів.

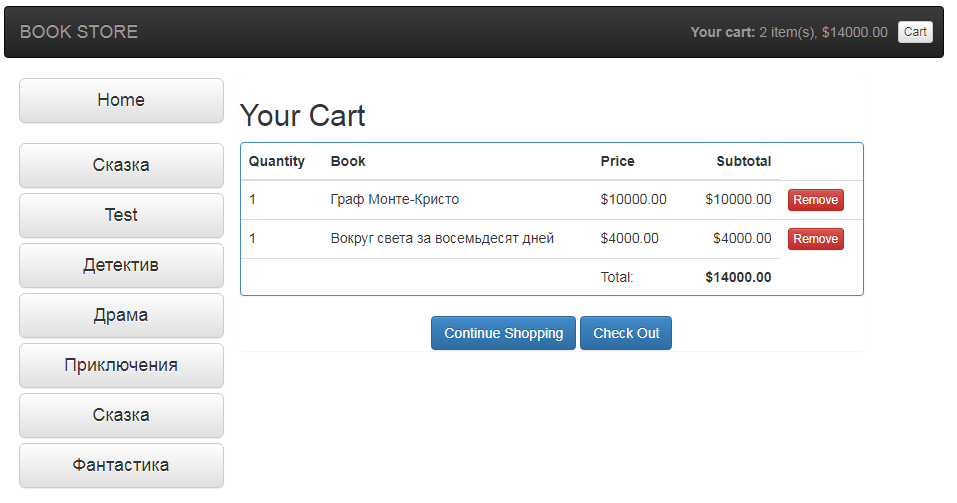


Кнопки для сортування книг за певною категорією знаходяться зліва. При натисканні на відповідну кнопку користувачеві будуть показані книги лише із тим жанром, який був вказаний на кнопці. Кнопка обраної категорії змінює свій колір на синій. Для перегляду всіх книг необхідно натиснути на кнопку Home. Це продемонстровано на рисунку 8.

Рисунок 8 – Сортування книг по жанру.

Стан корзини вказаний зверху справа. Більш детальну інформацію можна побачити, якщо натиснути на кнопку Cart. Перейти до перегляду каталогу книг можна при натисканні на кнопку Continue Shopping. За здійснення замовлення відповідає кнопка Check Out. Однак, неможливо здійснити замовлення, якщо у корзині нема книг. Все це показано на рисунках 9 та 10.

Рисунок 9 – Робота із кошиком (Частина 1).

Рисунок 10 – Робота із кошиком (Частина 2).

При натисканні на кнопку Check Out відбувається перехід до меню оформлення замовлення. Для здійснення замовлення необхідно ввести ім’я (полегшена форма авторизації). Якщо не вводити ім’я, то користувач побачить помилку, що показана на рисунку 11. Якщо ввести ім’я, то замовлення буде здійснено, як показано на рисунку 12.

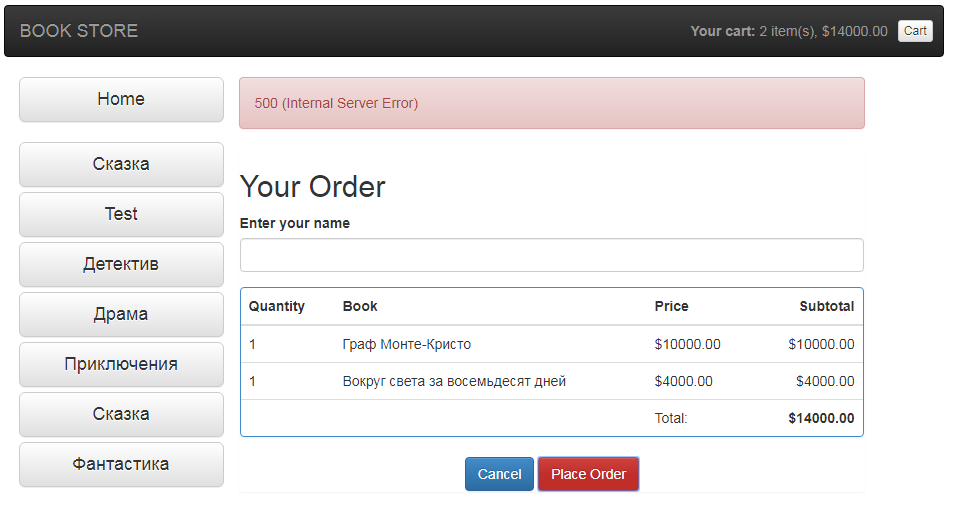
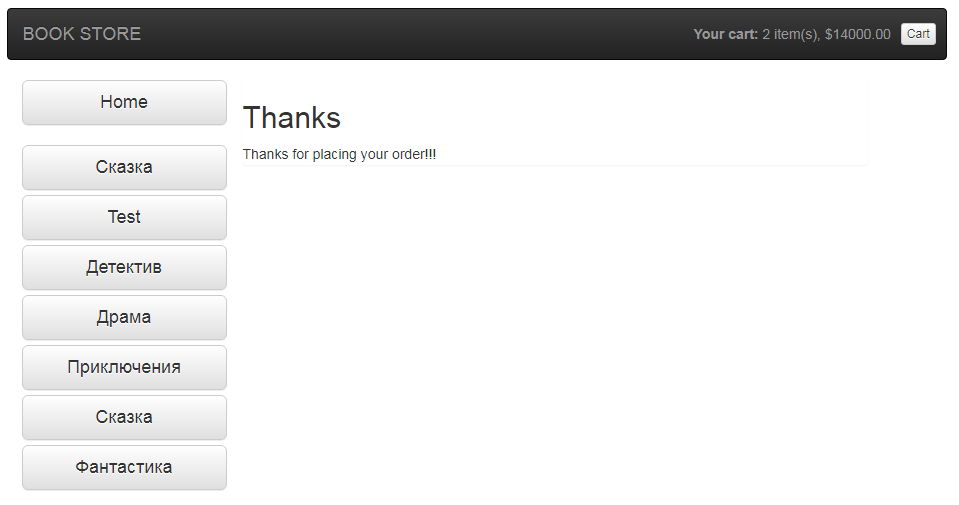
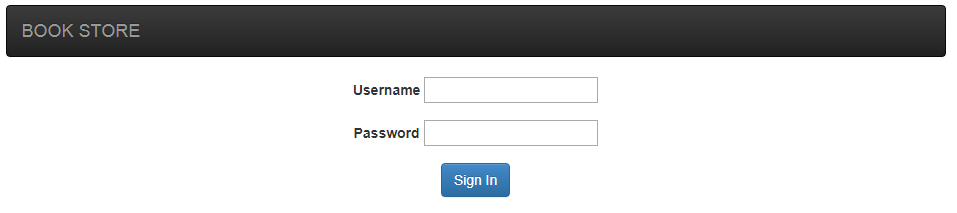
Рисунок 11 – Помилка при не введеному імені користувача.

Рисунок 12 – Вдале оформлення замовлення.

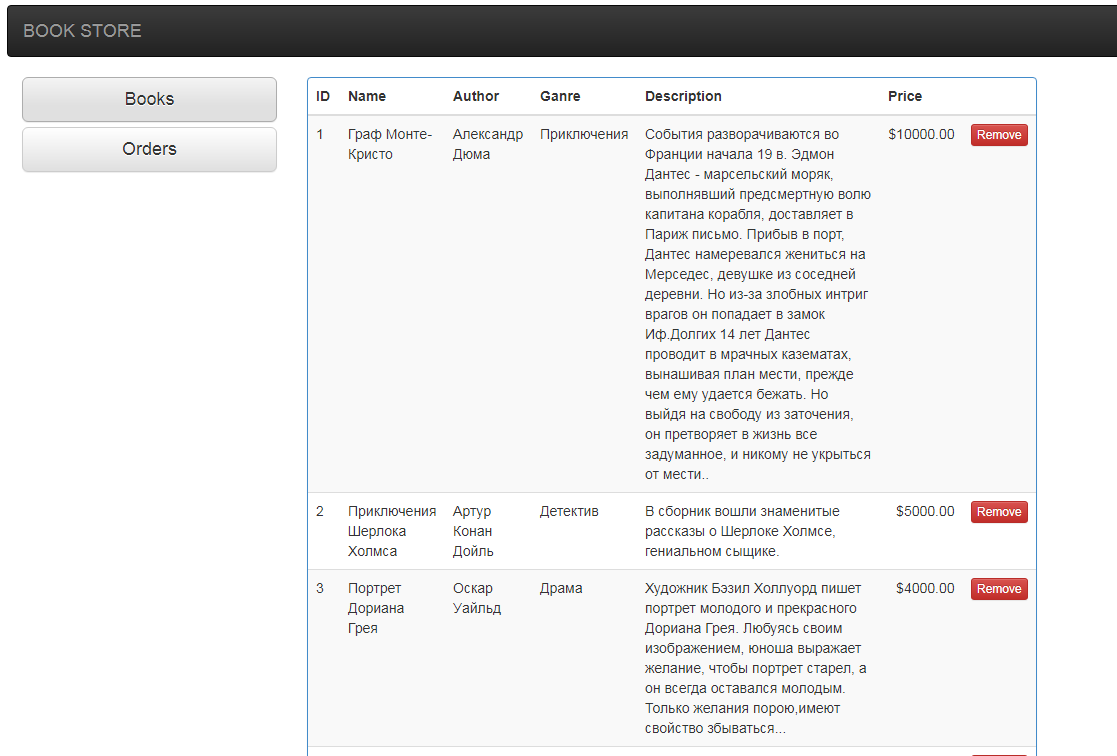


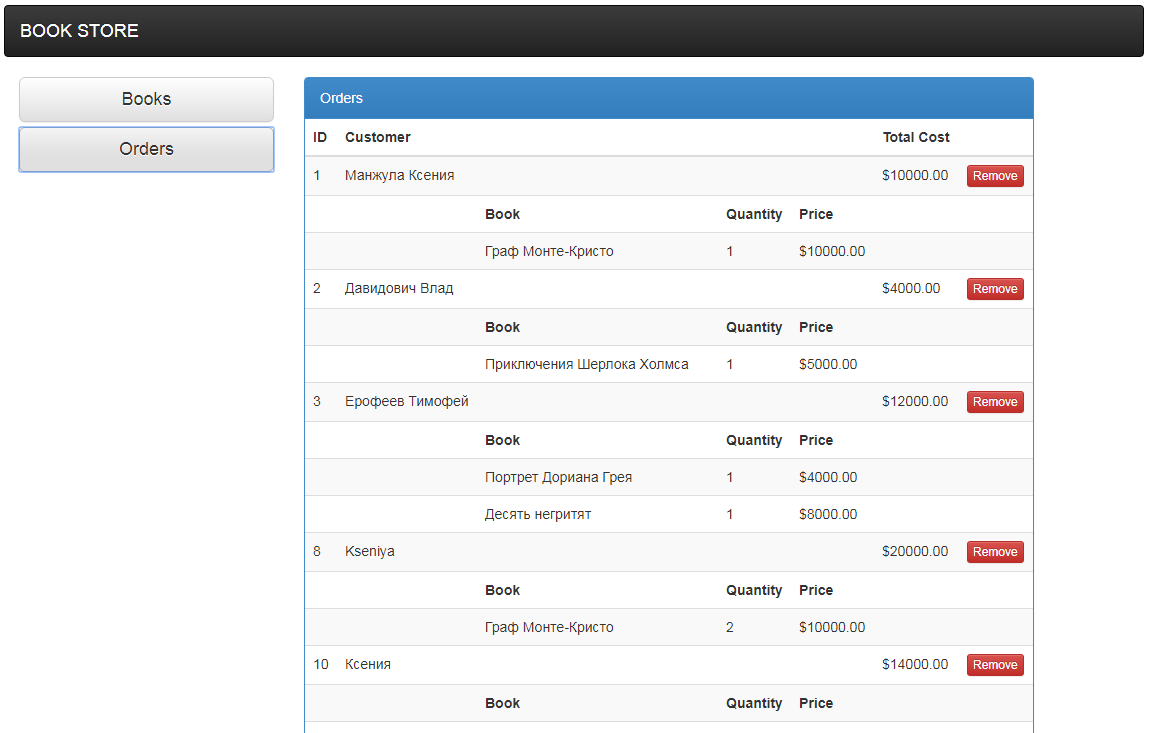
При введенні у адресний рядок замість слова Home слова Admin відбувається перехід до меню авторизації, зображене на рисунку 13.

Рисунок 13 – Меню авторизації для адміністраторів/менеджерів.

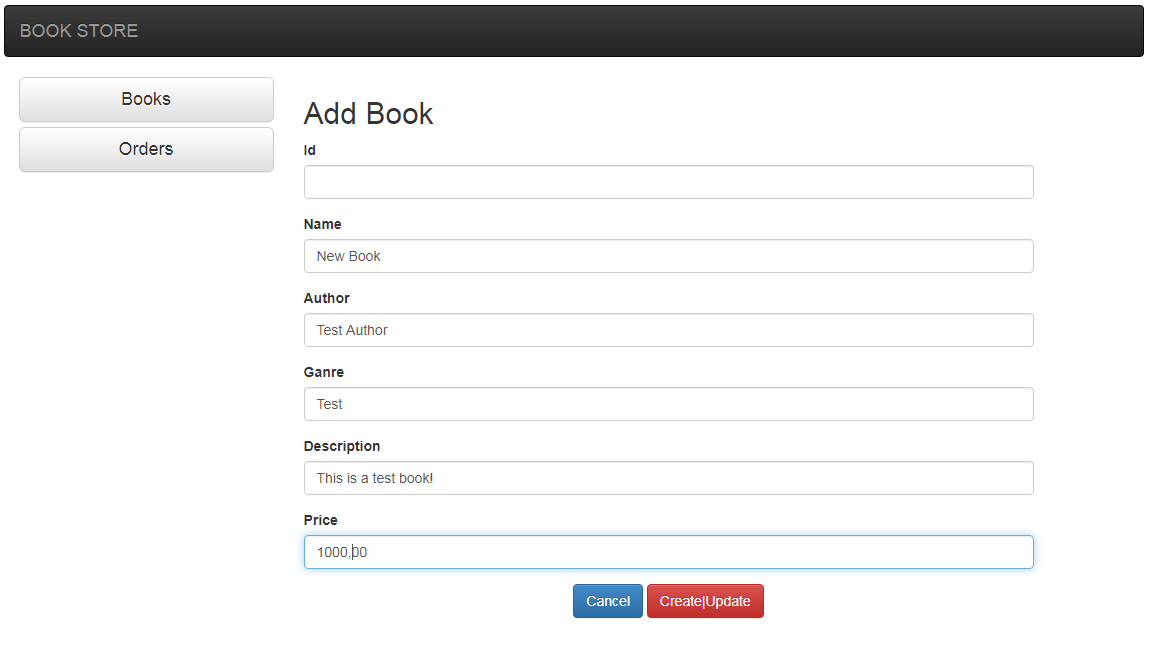
При введенні логіна та пароля адміністратора чи менеджера відбувається перехід до меню керування каталогом, зображене на рисунку 14. Для переходу до меню керування замовленнями необхідно натиснути на кнопку Orders, для повернення – натиснути на кнопку Books. Меню керування замовленнями показано на рисунку 15.

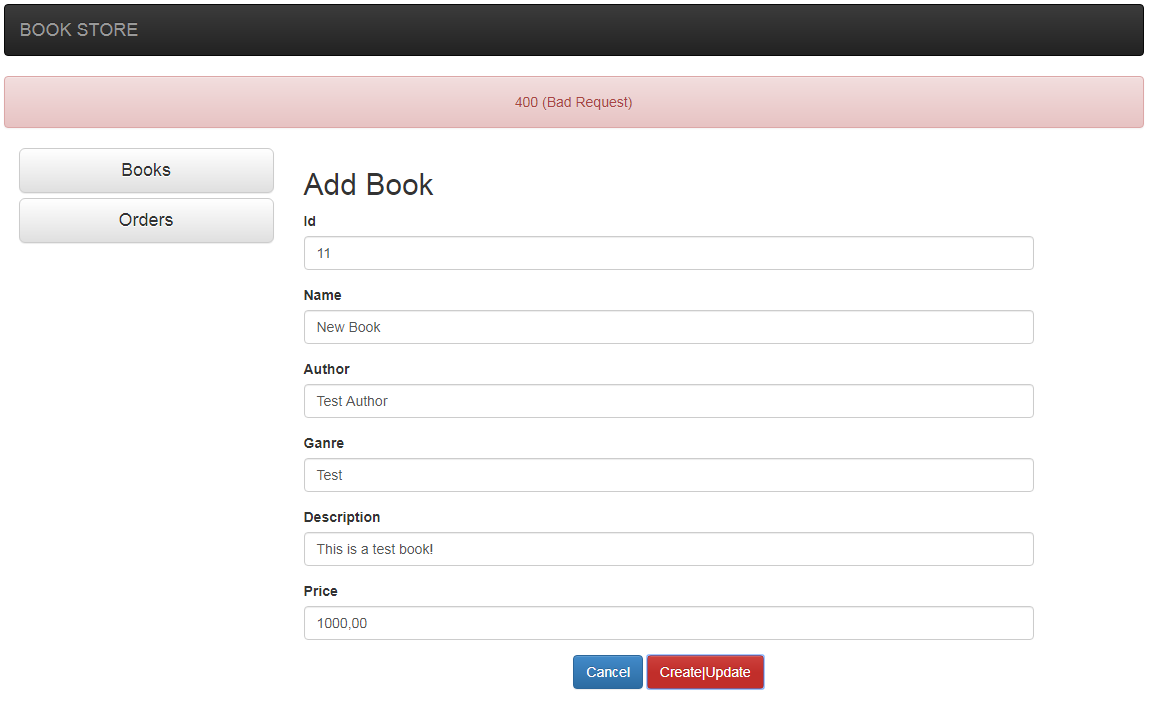
Рисунок 14 – Меню керування каталогом книг.



Рисунок 15 – Меню керування замовленнями.

Для створення чи оновлення книги необхідно натиснути на кнопку Add/Update Book, яка знаходиться в кінці списку книг. При натисканні на цю кнопку відбувається перехід до меню додавання/редагування книги, яке зображено на рисунку 16. Якщо заповнити всі поля, крім Id, відбудеться додавання книги. Якщо із заповненим полем Id – оновляться дані книги із номером, вказаним у полі Id. При введенні у поле Id номеру неіснуючої книги користувач побачить помилку, зображену на рисунку 17.

 Рисунок 16 – Меню додавання/оновлення книги із полями, заповненими для додавання книги.

 Рисунок 17 – Помилка при оновленні неіснуючої книги.

**Висновки**

В ході написання курсової роботи було розроблено ASP.NET MVC Web API інтернет - магазин «BOOK STORE». Розроблена програма основана на об’єктно-орієнтованому підході до програмування мовою С# та з використанням механізму зберігання даних у БД за допомогою Entity Framework, шаблонів проектування Repository та UnitOfWork, маппінгу, системи контролю версій Git, системи відслідковування завдань (Azure), і NUnit-тестування, багатощарової структури даних.

Практичне значення дослідження полягає в тому, що розроблена програма допомагає організувати та підтримувати постійне ознайомлення клієнтів із каталогом товарів без перерв.

Для виконання курсової роботи було проведено ознайомлення з існуючими особливостями мови С# та проведено аналіз технологій Entity Framework, шаблонів проектування Repository та UnitOfWork, маппінгу, системи контролю версій Git, системи відслідковування завдань (Azure), NUnit-тестування, і багатощарової структури даних. В результаті було сформовано технічне завдання на розробку інтернет – магазину, який реалізує всі вищезазначені технології.

Для написання даної програми було обрано середовище Microsoft Visual Studio 2017. Також, необхідні компоненти були завантажені та встановлені за допомогою системи керування макетами Nuget.

Під час виконання дипломної роботи було закріплено навички та уміння з програмування в Visual Studio C#, використовуючи мову програмування C#, а також покращено навички використання системи контролю версій Git та системи відслідковування завдань (Azure).

**Використані джерела**

1. Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Язык программирования C#. Классика Computers Science. 4-е изд. – СПб.: Питер 2012 – 784 с.
2. MVC 5 Интернет-Магазин [Електронний ресурс] / Master Progger. – 2018. – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=ZSCrfeEnVXc&list=PLrTvxXQSRKAMwJyujBX13GbTYpvbaLyYP&index=1>.
3. ASP NET MVC. Создание сайта с нуля. Интернет магазин. [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=Yru7B60GSjk&list=PLH3y3SWteZd1O84-ovNIG4kkBI2F0WCjn&index=2>.
4. ASP.NET MVC 5. Интернет-магазин. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.youtube.com/watch?v=ZktldWnQHEU&list=PLzpdsX4gjbEdi5gCen_DxGF3YLfltGJMs&index=2>.
5. Введение в Web API [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: <https://metanit.com/sharp/mvc/12.1.php>.
6. Начало работы с ASP.NET MVC 4 [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: <https://metanit.com/sharp/mvc/1.2.php>.
7. Интернет магазин на ASP.NET [Електронний ресурс] – Режим доступу до ресурсу: <https://professorweb.ru/my/ASP_NET/gamestore/level2/2_1.php>.
8. Freeman A. Expert ASP.NET Web API 2 for MVC Developers. / Adam Freeman., 2014. – 651 с. – (1).