Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Інтернет-магазин книжок з чат-ботом у "Телеграм"*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студентка 4 курсу, група КН-17-1* — Курс, група виконавця — Підпис — *О.О. Яницький* — Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КНІТ* — Науковий ступінь, посада — Підпис — *Р.О. Багрій* — Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* — Науковий ступінь, посада — Підпис — *Р.О. Багрій* — Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор — Підпис — *О.В. Бармак* — Ініціали, прізвище

08 червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*

Кафедра *комп'ютерних наук та інформаційних технологій*

Освітній ступінь *бакалавр*

Галузь знань *12 – Інформаційні технології*

Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

(підпис)

д.т.н., професор О.В. Бармак

«*08*» *лютого* *2021* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Інтернет-магазин книжок з чат-ботом у "Телеграм"»*

2. Завдання видано студенту *Яницькому Олегу Олеговичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КНІТ Багрій Руслан Олександрович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*05*» *лютого* *2021* р. № *11*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи - розробка веб-застосунку магазин книг, який буде працювати по бізнес-моделі B2C - бізнес для споживача, метою якої є прямі продажі для споживача. Слід забезпечити виконання функцій внесення, редагування та перегляду всіх необхідних для роботи системи даних й реалізувати основний функціонал електронної комерції: автентифікація користувачів, управління продуктами, система покупок, система управління. А також розробка бота, який буде реалізовувати меншу частину функціонала сайту у форматі чата для месенджера «Телеграм».

Виконавець: *студентка 4 курсу, група КН-17-1* (Курс, група виконавця) _____ (Підпис) *О.О. Яницький* (Ініціали, прізвище)

Керівник: *к.т.н., доцент кафедри КНІТ* (Науковий ступінь, посада) _____ (Підпис) *Р.О. Багрій* (Ініціали, прізвище)

Анотація

Тема кваліфікаційної роботи бакалавра: *«Інтернет-магазин книжок з чат-ботом у "Телеграм"»*

Виконавець кваліфікаційної роботи бакалавра: *студентка групи КН-17-1 Яницький Олег Олегович*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КНІТ Багрій Руслан Олександрович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
51	38	14	15	13

Метою бакалаврської роботи є створення веб-застосунку «Інтернет-магазин книг» та чат-боту в "Телеграм" на мові програмування C#.

Результатом виконання бакалаврської роботи є веб-застосунок та телеграм бот розроблений на платформі ASP.NET MVC 5, а також база даних на MS SQL Server. Створений програмний застосунок дозволяє спростити та автоматизувати продаж книг у магазині. Розроблений програмний продукт призначений для працівника магазину та покупця, а також для покупців, що користуються месенджером "Телеграм".

Виконавець: *студентка 4 курсу, група КН-17-1* (Курс, група виконавця) _____ (Підпис) *О.О. Яницький* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	**Пояснення**
БД	База даних
ІТ	Інформаційні технології
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
ПП	Програмний продукт
СКБД	Система керування базами даних
ЕК	Електронна комерція
МК	Магазин книг

Вступ

Протягом останніх років електронна комерція швидко розвивається у всьому світі, забезпечуючи при цьому безпрецедентні засоби і можливості для розвитку світової торгівлі. Все більше людей вважають за краще вести діяльність через Інтернет і не тільки професійну. Для особистих цілей завжди зручно, не витрачаючи час на дорогу і розмови по телефону, знайти необхідну інформацію. Комерційна діяльність в Інтернеті включає і має на увазі роботу в рекламній сфері, поширення товарів і послуг в мережі. Електронна комерція та інтернет магазин невіддільні одне від одного. Також одним із важливих трендів в сфері електронної комерції є активне застосування чат-ботів, цей інструмент може дати певні переваги інтернет-магазину. Він здатний до певної міри ефективно замінити операторів кол-центру, він працює 24 години на добу без перерв на обід.

Книгарня виставляє на продаж велику кількість книг, в чому дуже складно орієнтуватися. В ньому є примірники [1], видання[2], покупки, прокат, т. д.. Для кращого обліку цих речей, пропонується створити облік в електронному вигляді. В даному випадку, використовуючи засоби MS SQL Server [3] для створення бази даних і зручного інтерфейсу до неї.

Використання електронного обліку дозволить автоматизувати багато процесів, що дозволить пришвидшити та спростити роботу магазину. Електронний облік дозволить з легкістю виконувати процедуру продажі книг, прокату книг, реєстрацію нових книг, т. д.. Також дозволяє без додаткових економічних витрат організувати зберігання і використання існуючої інформації, та автоматизувати дії по створенню нових даних. Дозволить легко отримувати потрібну інформації.

При створенні інтернет магазину книжок, одним із недоліків є те що, покупці звикли вибирати товар в звичайному магазині, де вони його можуть побачити, а також їм зможе допомогти консультант в виборі потрібної їм книги відносно їх побажань, чого вони не можуть онлайн. Часткове вирішення цієї проблеми буде використання чат-боту. За допомогою простих маніпуляцій чат-

бот здатний максимально полегшити процес вибору товару. Це особливо важливо для тих відвідувачів, які зайшли в інтернет-магазин вперше, вони можуть заплутатися в навігації та не знайти того що шукали, саме тому їм допоможе бот. Цей інструмент допомагає підвищити лояльність відвідувачів інтернет-магазину, при цьому мінімізувавши витрати.

Розділ 1
Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області

Електронна комерція[4] – це ведення бізнесу в онлайн режимі, яке на сьогодні присутнє в чотирьох наступних сферах: прямі продажі товарів і послуг; банківська справа та фактурування (платіжні системи); безпечне розміщення інформації; корпоративні закупівлі. Електронна комерція стала невід'ємною частиною сучасної економіки. Все більше споживачів купують товари за допомогою мережі Інтернет, а комерційні організації так чи інакше використовують можливості даної мережі при здійсненні підприємницької діяльності.

Інтернет-магазин[5] - місце в інтернеті, де відбувається прямий продаж товарів споживачеві (юридичній або фізичній особі), враховуючи доставку. При цьому розміщення споживацької інформації, замовлення товару і угода відбуваються там само, всередині мережі (на сайті інтернет-магазину). Вибравши необхідні товари або послуги, користувач зазвичай має можливість тут же на сайті вибрати метод оплати і доставки. Сукупність відібраних товарів, спосіб оплати і доставки являють собою завершене замовлення, яке оформляється на сайті шляхом повідомлення необхідної інформації про покупця. Інформація про покупця може зберігатися в базі даних магазину якщо бізнес-модель магазину розрахована на повторні покупки, або ж відправлятися одноразово. В інтернет-магазинах, розрахованих на повторні покупки, також ведеться відстеження повернень відвідувача і історія покупок. Часто при оформленні замовлення передбачається можливість повідомити деякі додаткові побажання від покупця продавцю.

Плануючи Інтернет-магазин, перш за все слід визначитися з товарами, які будуть продаватися. В електронному магазині покупець не може як слід оглянути товар, вивчити його, тому магазинам потрібен якісний інтерфейс і вітрина товару. Інформація, доступна користувачам інтернету, розташовується на комп'ютерах

(веб-серверах[6]), на яких встановлено спеціальне програмне забезпечення. Значна частина цієї інформації організована у вигляді веб-сайтів[7] або веб-застосунків[8]. Кожен з них має свою інтернет адресу. Більшість користувачів розглядають кожен відвідуваний ними адреса в інтернеті як веб-сайт. Насправді не всі URL, що вводяться в адресний рядок браузера, є такими. Деякі з них можуть бути визначені як веб-застосунки. Різниця між поняттями веб-сайту і веб-застосунки може бути неважлива для простого користувача. Однак вона буде мати значення при розробці інтернет-ресурсу для бізнесу з точки зору розуміння його вимог.

Веб-сайти зазвичай носять інформаційний характер. Складаються з веб-сторінок, об'єднаних один з одним в єдиний ресурс. Мають просту архітектуру на основі HTML-коду. Служать в якості платформи для надання контенту для відвідувачів: можуть містити текст, зображення або музику. Сайти не пропонують можливості взаємодії з програмою. Його користувачі не мають доступу до розміщення своєї інформації крім заповнення форми для отримання підписки. Найбільш яскравими прикладами типових сайтів є новинні, кулінарні, прогнозу погоди.

Веб-застосунки - це інтерактивні комп'ютерні програми, розроблені для мережі інтернет, що дозволяють користувачам вводити, отримувати і маніпулювати даними за допомогою взаємодії. Такі програми зазвичай мають дуже тісний зв'язок з сервером, відправляючи на нього безліч запитів. Можуть бути вбудовані в веб-сторінки, або самі веб-сторінки можуть бути додатками. Наприклад, Facebook, Gmail, YouTube, Ebay, Twitter, Amazon. Веб-застосунки використовують ім'я користувача і пароль для аутентифікації. Дозволяють своїм відвідувачам обмінюватися миттєвими повідомленнями (чат-платформи, соцмережі, блоги), створюють контент на основі призначених для користувача переваг, забезпечують до нього необмежений доступ, використовують міні-вбудовані програми для розваг. Оскільки використання інтернет-додатків часто пов'язано з введенням особистих даних, платіжних реквізитів, розробники несуть

відповідальність за захист конфіденційної інформації. Вони стикаються з додатковими вимогами, такими як відповідність стандарту безпеки PCI DSS[9].

У сучасному світі інтернету майже не залишилося сайтів, які не мали б ніякого елемента інтерактивності. І навпаки, багато веб-застосувань часто включають в себе пошук інформації. Проте, веб-сайти як і раніше є інформаційними джерелами, а веб-застосунки залишаються інструментами призначені для користувача.

Книгарня [10]- крамниця, що займається роздрібним та оптовим продажом книг та інших товарів, таких як журнали, географічні карти, путівники та ін. А також оперативно-виконавчий орган, що входить до складу книготорговельної галузі, точка роздрібного продажу книг та інших публікацій, приміщення з відповідним обладнанням для експозиції та продажу.

Робота книжкового магазину є досить складним і трудомістким процесом. У магазин приходить багато книг з різних видавництв, також кожна книга може бути присутня в декількох примірниках. Для обліку книг необхідно оформляти і вести досить велику кількість документації (оформлення накладних на прихід книг, оформлення знижок і ліцензій і т. д.). Основними напрямками роботи будь-якої книгарні є: комплектування і організація книжкового каталогу; обслуговування покупців.

Комплектування нових книжок складається з систематичного виявлення потрібних для даного магазину видань і придбання їх. Від своєчасності і повноти комплектування книгарні в значній мірі залежить рівень обслуговування покупців.

Організація книжкового фонду включає питання обліку, розстановки, зберігання літератури і продажу її покупцю. Облік книжкового каталогу - це прийом надійшовших видань, запис прибулих і виключення з тих чи інших причин непридатних для продажу видань в документах, які дають точні відомості про склад фонду і допомагають забезпечити його збереження.

Обслуговування покупців магазину здійснюється різними шляхом продажу літератури і допомоги окремим читачам в підборі необхідної їм літератури.

Діяльність книжкового магазину пов'язана з урахуванням великої кількості операцій, безліч книг і покупців серйозно уповільнюють роботу працівників. Складність пошуку потрібної книги в каталозі займає тривалий час, і цілком спирається на компетентність продавців. Для ведення каталогів, організації пошуку необхідних видань і статистики в базі повинні зберігатися відомості, велика частина яких розміщуються в анотованих каталожних картках.

Таким чином, автоматизація роботи книгарні у вигляді веб-застосунку допомагає ефективно використовувати ресурси, а також допомагає облегшити роботу, як персоналу магазину так і його покупців.

Отже, основними параметрами предметної області, для задачі, що розглядається, є:

Параметр	**Опис**
Назва	Назва книжки певного видавництва, яка зберігається на складі.
Видавництво	Назва видавництва, яке надрукувало та продало книгу.
Жанр	Кожна книга яка зберігається в магазині має свій жанр.
Автор	Інформація про автора, який написав книгу.
Рік видавництва	Видавництво може перевидати стару книгу, тому доцільно зберігати цю інформацію .
Кількість екземплярів	В магазині може зберігатися кілька екземплярів книги.
Ціна	На кожну книгу установлена фіксована вартість за яку вона продається.

Фото	Вигляд книги може відрізнятися в залежності від видавництва, тому для кращого ознайомлення буде зберігатися фото кожної книги.
Логін	Ідентифікатор кожного користувача для системи реєстрації.
Пароль	Для аутентифікації користувача в системі реєстрації.
ПІБ	Повні персональні дані користувача, який замовляє товар.
Адреса	Повна адреса користувача, на яку буде відправлено замовлення.
Телефон	Телефон користувача, який замовляє товар

1.2 Аналіз існуючого програмного забезпечення предметної області

На сьогоднішній день у цій предметній області без інформаційних технологій не обійтись, і якщо в книгарні немає сайту то вона не буде популярною.

Зазвичай використовують такі популярні технології як ASP.NET, Laravel, PrimeFace та інші. Такі сайти більш функціональні, мають оригінальний дизайн, а також унікальну адресу, але при цьому дорожчі та набагато важчі в реалізації . В противагу таким технологіям сайти також створюють за допомогою конструкторів сайтів, найбільш популярні, а саме: uCoz, uKit, Wix. Такий сайт досить легко та швидко створити, крім того набагато дешевше, але він буде не таким оригінальним , і тому менш популярним.

Розглянемо на прикладі популярної мережі книгарень «Є»[11], їхні магазини розташовані по всій Україні, крім того вони також мають сайт, який зображено на рисунку 1.1.

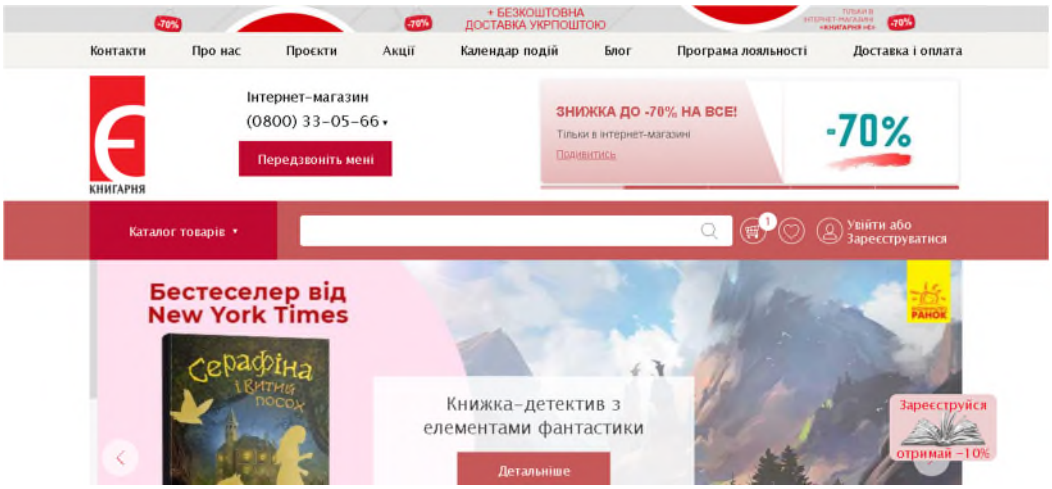


Рисунок 1.1 – Головна сторінка сайту «book-ye.com.ua»

На цьому сайті реалізовано перегляд всього асортименту книг, що знаходяться в магазині при цьому з можливістю фільтрувати їх за жанром, авторами та ціною, що продемонстровано на рисунку 1.2. Також можна переглядати кожен товар окремо(рисунок 1.3) та отримати про нього додаткову інформацію.

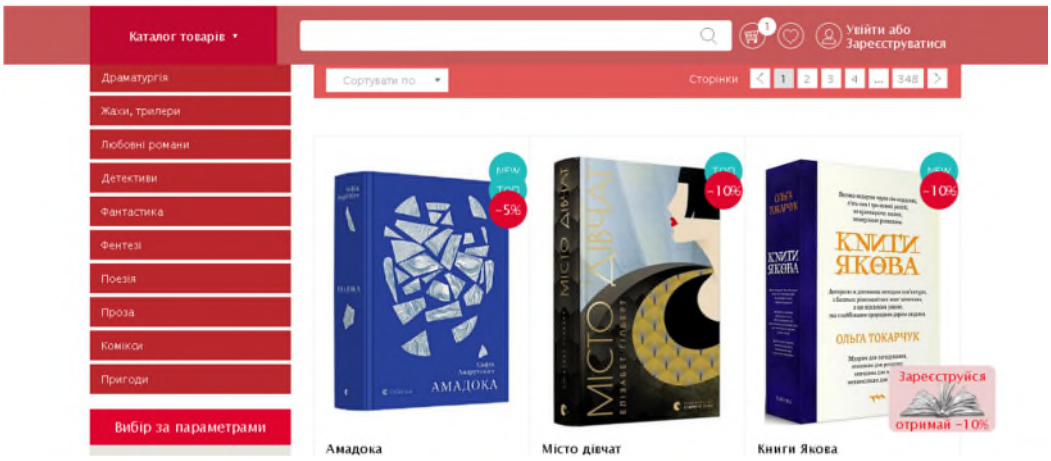


Рисунок 1.2 – Перегляд асортименту магазина «Є»

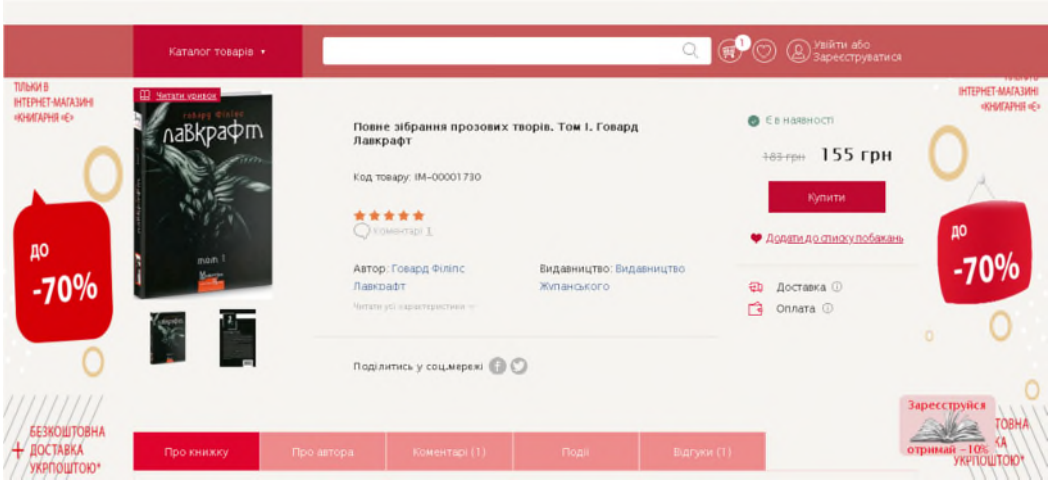


Рисунок 1.3 – Перегляд книжки на сайті книгарні «Є»

Ну і остання та найбільш важлива функція це покупка товару, вибравши потрібні книги та їхню кількість, і якщо ви новий покупець заповнити персональні дані, після чого вибрати спосіб доставки та оплати, що зображено на рисунку 1.4.

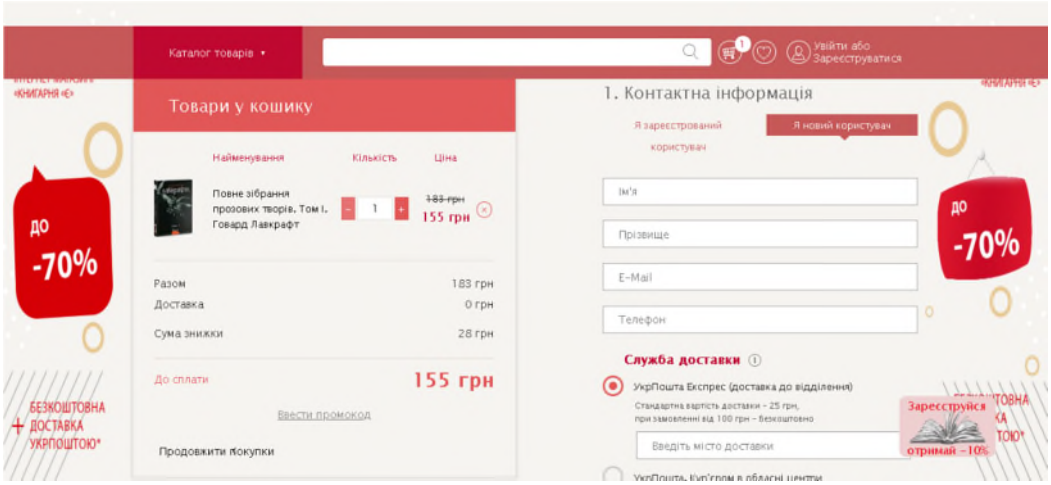


Рисунок 1.4 – Покупка товару на сайті книгарні «Є»

Хоча такий сайт має багато плюсів, є один мінус відсутність електронних книг. В наш час інформаційних технологій електронні книжки все більше витісняють паперові, і можливість їх покупки на сайті надає йому велику перевагу.

На Рисунку 1.5 зображений сайт книжкового магазину «Yakaboo» .

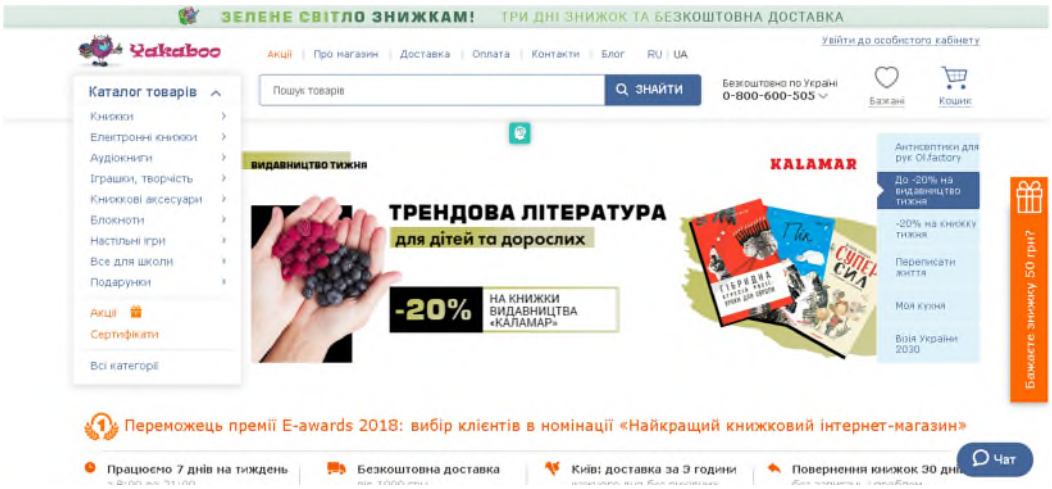


Рисунок 1.5 – Головна сторінка сайту «yakaboo.ua»

Цей сайт відрізняється від попереднього тим, що на ньому окрім паперових книг можна також купувати їхні електронні версії (рисунок 1.6).

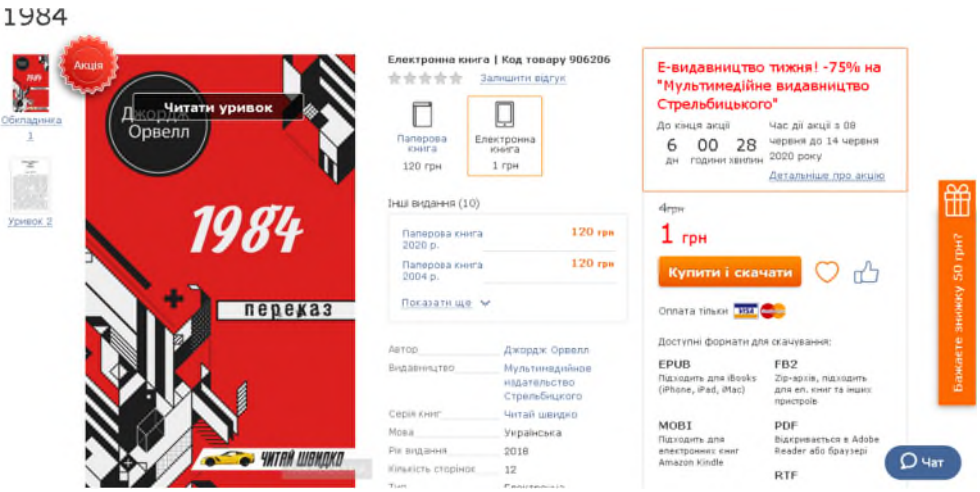


Рисунок 1.6 – Перегляд електронної версії книги на сайті «yakaboo.ua»

Здавалося невелика перевага, крім того можна побачити, що перший сайт має кращий дизайн і більш привабливий. Але провівши дослідження статистики (рисунок 1.7) сайтів можна побачити, що відвідуваність сайту «yakaboo.ua» приблизно в 4 раз більше ніж «book-ye.com.ua», що демонструє результат цієї переваги.

Рисунок 1.7 – Перегляд статистики сайтів за допомогою SimilarWeb

Отже, наразі можна знайти багато сайтів книжкових магазинів. В основному вони схожі і надають клієнтам можливість перегляду книжок та їх купівлі. Їхня відмінність полягає в дизайні та функціоналу сайту, крім того сайти які надають можливість купівля електронної версії книжок мають більшу популярність ніж їхні конкуренти які продають лише паперові книги.

1.3 Аналіз сучасних засобів створення програмного забезпечення

На даний момент для розробки веб застосунків [12] можна використати одну з двох найбільш відомих існуючих платформ: ASP.NET та PHP.

ASP.NET[13] - це серверний веб фреймворк, що базується на C #. Він був представлений Microsoft на початку 2000-х років як розширення платформи .NET. Цей фреймворк пропонує безліч бібліотек та інструментів, особливо для веб-розробки. Більше того, будь-які інші пакети, доступні на платформі .NET, такі як Visual Studio, можуть зручно використовувати веб-розробники.

Серед основних функціональних можливостей - синтаксис шаблону веб-сторінок під назвою Razor. Це дозволяє розробникам писати внутрішній код на C #. Після оцінки коду на сервері вміст HTML буде надіслано користувачам. Крім того, сторона користувача кодується на JavaScript, що ефективно інтегрується з ASP.NET. Різні розширення редактора забезпечують можливість заповнення коду та підсвічування синтаксису.

Раніше веб-програми ASP.NET працювали лише на серверах Microsoft, які зазвичай вимагають вищої плати за хостинг. З новою версією ASP.NET Core, випущеною в 2016 році, інженери-програмісти можуть подавати заявки на послуги хостингу Linux та macOS.

ASP.NET - це зрілий фреймворк, який існує на ринку вже більше кількох десятиліть. Порівняно з' іншими, він завоював гідну частку ринку і продовжує рости шляхом постійного розвитку. Будучи фреймворком з відкритим кодом, він підтримується Microsoft та спільнотою програмістів. Ця тісна співпраця призводить до різних бібліотек та чітких керівних принципів.

ASP.NET часто вибирають завдяки швидкій та ефективній роботі. Компіляцію потрібно виконати лише один раз, оскільки платформа .NET здатна виконувати скомпільований код знову і знову. На відміну від інтерпретованого коду, який потрібно виконувати щоразу машиною, скомпільований є більш послідовним і масштабованим.

Однією з відмінних переваг є міцна безпека. Цей фреймворк підтримує всі необхідні протоколи автентифікації. ASP.NET пропонує ряд вбудованих функцій для загального захисту додатків, а також міжсайтові сценарії та підробку запитів. На додаток до багатофакторної автентифікації, фреймворк забезпечує її зовні, наприклад, із Google.

PHP[14] - це серверна мова для веб-розробки, яка була створена з невеликого проєкта з відкритим кодом. Він був створений в 1995 році Расмусом Лердорфом, який написав кілька програм CGI і зумів змусити їх працювати з веб-формами та базами даних. Він не прагнув створити нову мову програмування, але вона стала однією з найбільших мов з відкритим кодом. Будь-хто може зробити свій внесок у новий випуск, включаючи розгортання завдань, тестування, документацію та джерела.

Спільнота PHP має різні безкоштовні бібліотеки, інструменти та розширення з відкритим кодом. Наприклад, розширення MySQL, воно дозволяє розробникам створювати програми PHP, які взаємодіють з базою даних, одночасно забезпечуючи вдосконалену підтримку операторів та транзакцій.

У світлі порівняння між ASP та PHP розробники PHP мають справу з іншим типом коду, який інтерпретується повністю і повністю написаний на C. Це, однак, дозволяє відмінне налаштування, що може бути причиною деяких помилок та поганого кодування.

Оскільки ця серверна мова була спочатку випущена для веб-розробки, вона ефективно обмінюється даними та отримує доступ до більшості типів баз даних. Завдяки своїй пристосованості та гнучкості розробники не обмежуються у своєму виборі. Більше того, PHP може мати підтримку для інших служб, що охоплюють різні протоколи, такі як POP3, HTTP, IMAP, LDAP тощо.

Завдяки своїй доступності PHP набув популярності серед величезної кількості розробників. Вони завоювали найбільшу частку ринку і постійно вдосконалюються та регулярно випускаються. PHP часто вибирають для більшої масштабованості. Це досягається додаванням додаткових серверів. Загальне навантаження буде розподілено між ними і може обробляти більше запитів користувачів. Цей підхід спирається на горизонтальну масштабованість, яка відрізняється PHP від ASP.NET.

Маючи ряд вбудованих функцій безпеки, PHP все ще вважається менш безпечним у порівнянні з іншими мовами програмування веб-серверів. Отримавши більшу гнучкість у написанні коду, інженери несуть більше відповідальності за його захист і не можуть покладатися лише на вбудовані інструменти.

Обидві платформи мають свої переваги та недоліки але загалом, обидві придатні для створення програмних продуктів і в нашому випадку інтернет магазину книг, але в зв’язку з тим, що .NET є більш популярним, надійнішим та має більшу кількість бібліотек та фреймворків, крім того є можливість багатомовності, що дає змогу писати код сторінок різними мовами, було обрано саме її.

1.4 Постановка задачі та вимоги до розробки інформаційної системи

Метою кваліфікаційної роботи бакалавра є розробка веб-застосунку магазин книг з чат-ботом за допомогою фреймворку ASP.NET MVC. В інтернет-магазині необхідно система реєстрації, що розділяє користувачів за ролями на адміністратора та клієнта, а тому їхні можливості будуть відрізнятися.

Для адміністратора має бути передбачений наступний функціонал:

- робота з книжками (перегляд, додавання, редагування та видалення);
- робота з клієнтом (перегляд, редагування, видалення);
- робота з допоміжними даними інтернет-магазину (працівниками, магазинами, жанрами, виданнями, авторами тощо);
- робота з замовленнями (перегляд, редагування).

Для клієнта(покупець) мають бути передбачені наступні функції

- робота з книжками (перегляд, додавання у корзину);
- робота з кошиком (перегляд, редагування);
- робота з замовленнями (перегляд статусу замовлення);
- робота з допоміжними даними (контактні дані інтернет-магазину).

Розділ 2

Проєктування інформаційної системи

2.1 Функціональна структура та бізнес-процеси системи

Інтернет-магазин книг призначений для планування ресурсів магазину, які використовуються для відстеження книжкових фондів, від їх замовлення та придбання до продажі покупцям книг. Основні сутності, що необхідні для інтернет магазину є напрямами роботи адміністратора з застосунком, наступні: замовлення, книжковий фонд, працівники, а також для користувача: товари, кошик та робота з ботом. Ці напрямки роботи адміністратора та користувача з інтернет-магазином відображено на рисунку 2.1 та 2.2 відповідно.

Рисунок 2.1 – Групи функцій при роботі адміністратора з інтернет магазином

Рисунок 2.2 – Групи функцій при роботі користувача з інтернет магазином

Відповідно, бізнес-процеси у інтернет-магазину книг , які підлягають автоматизації, наступні:

1. Бізнес-процес «Замовлення» – призначений для роботи адміністратора з даними про покупки та клієнтами які їх зробили (замовлення, клієнти);
2. Бізнес-процес «Книжковий фонд» – призначений для роботи адміністратора з даними каталогу книжок які зберігаються (примірники, видання, автори, жанри);
3. Бізнес-процес «Кадри» – призначений для роботи адміністратора з даними допоміжної інформації (працівники, посади, магазини).
4. Бізнес-процес «Книжки» - призначений для роботи користувача з даними про книгу (замовлення)
5. Бізнес-процес «Кошик» - призначений для роботи користувача з даними про кошик (кошик)
6. Бізнес-процес «Бот» - призначений для роботи користувача з телеграм ботом

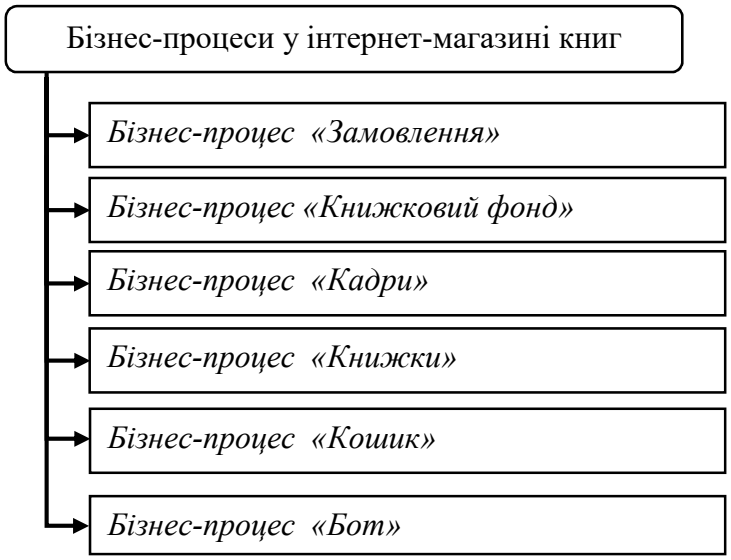


Рисунок 2.3 – Бізнес-процеси у інтернет-магазині книг

Бізнес-процес «Замовлення». Даний бізнес-процес включає функції по роботі адміністратора з замовлення які зробили клієнти, зокрема він надає можливість перегляду та редагування цих даних про замовлення, а також перегляду клієнтів які зробили замовлення. Даний бізнес-процес включає наступні функції:

- перегляд переліку клієнтів;
- перегляд переліку замовлень;
- редагування відомостей обраного клієнта;
- редагування відомостей обраного замовлення;
- перегляд відомостей про клієнта;
- перегляд відомостей про замовлення.

Бізнес-процес «Книжковий фонд». Даний бізнес-процес включає функції по роботі адміністратора з каталогом книг магазина, зокрема він надає можливість додавати нові книги, а також можливість перегляду, редагування цих даних. Даний бізнес-процес включає наступні функції:

- перегляд переліку книг;
- додавання нової книги;
- редагування обраної книги;

- перегляд відомостей обраної книги;
- перегляд даних про видання книги.

Бізнес-процес « Кадри». Даний бізнес-процес включає функції по роботі адміністратора з працівниками, зокрема він надає можливість додавати записи нового працівника магазину , а також можливість перегляду, редагування цих даних. Даний бізнес-процес включає наступні функції:

- перегляд переліку працівників;
- додавання нового працівника;
- редагування відомостей обраного працівника;
- перегляд відомостей про працівника.

Бізнес-процес «Книжки». Даний бізнес-процес включає функції по роботі користувача з книгами, зокрема він надає можливість перегляду каталогу книг , а також можливість перегляду відомостей про обрану книгу та додавання її в кошик. Даний бізнес-процес включає наступні функції:

- перегляд переліку книг;
- перегляд відомостей обраної книги;
- додавання обраної книги в кошик;

Бізнес-процес «Кошик». Даний бізнес-процес включає функції по роботі користувача з кошиком, зокрема він надає можливість перегляду кошика , а також можливість його редагування та оформлення замовлення. Даний бізнес-процес включає наступні функції:

- перегляд переліку товарів у кошику;
- видалення товарів з кошика;
- редагування кількості товарів в кошику
- перегляд загальної вартості;
- оформлення товару;

Бізнес-процес «Бот». Даний бізнес-процес включає функції по роботі користувача з телеграм ботом, зокрема він надає можливість працювати з ботом,

переглядати весь товар, дізнатися про нього інформацію і його купити. Даний бізнес-процес включає наступні функції:

- Перегляд товару;
- Покупка товару;
- Додаткова інформація

Процеси «Замовлення», «Книжковий фонд» та «*Кадри»*, відповідають за роботу адміністратора з сайтом, вони включають в себе авторизацію та підтвердження прав адміністратора, після успішного входу він має доступ до панелі адміністратора і може переглядати дані відповідно до процесів, а саме з замовленнями, які зробили клієнти, каталогами книг, та працівниками магазину. Крім перегляду, адміністратор може додавати, редагувати та видаляти дані (рисунок 2.4).

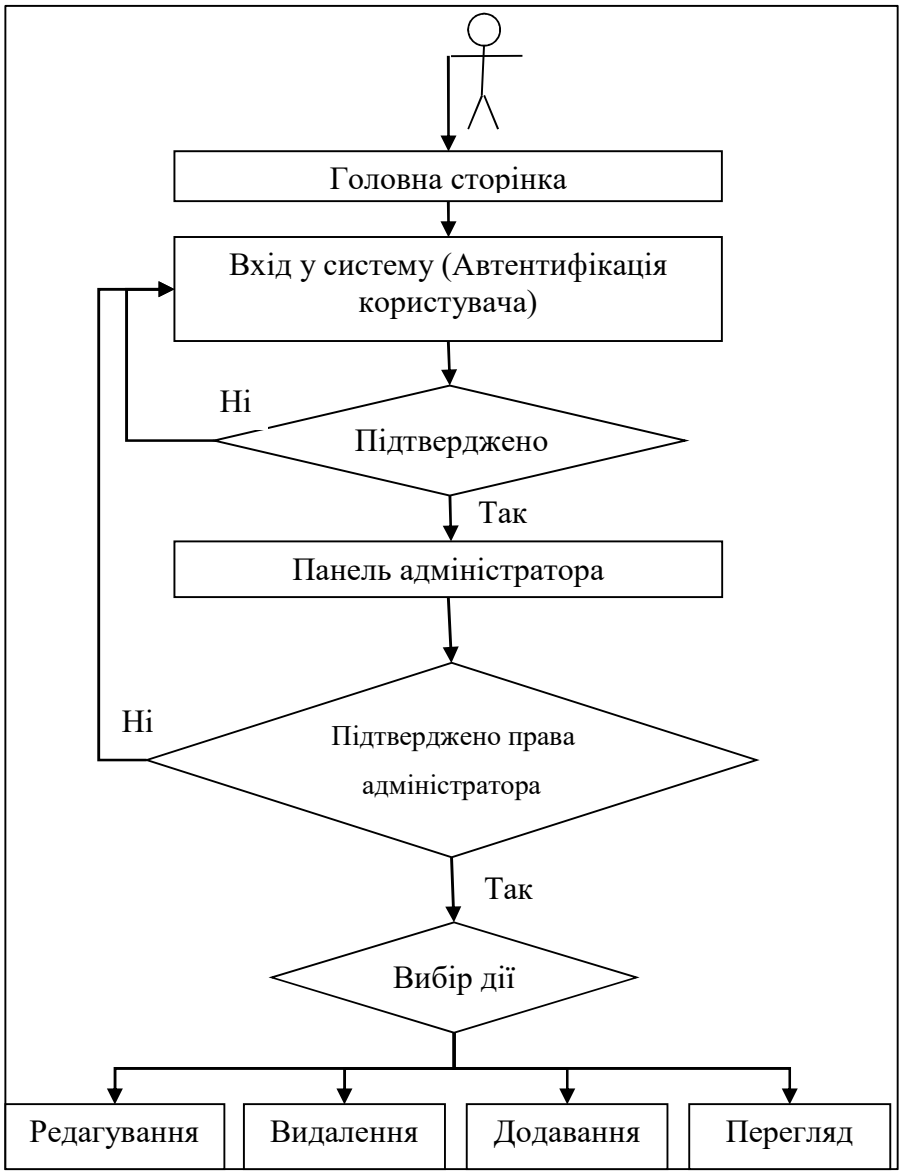


Рисунок 2.4 – Діаграма активності роботи адміністратора з сайтом

Відповідно *«Кошик», «Книжки»* та *«Бот»* це процеси роботи користувача з сайтом та телеграм ботом. Після авторизації на сайті, клієнт переходить на головну сторінку. Також він може переглядати каталог товарів і добавити в кошик товар який сподобався. В кошику користувач має можливість переглянути, редагувати(добавити, видалити) та перейти до оформлення товарів після чого їх оплатити (рисунок 2.5). При роботі з телеграм ботом користувач може переглянути весь список товарів, а також інформацію про магазин. Крім того він може переглянути всю інформацію про книжку, яка доступна на сайті та купити її (рисунок 2.6).

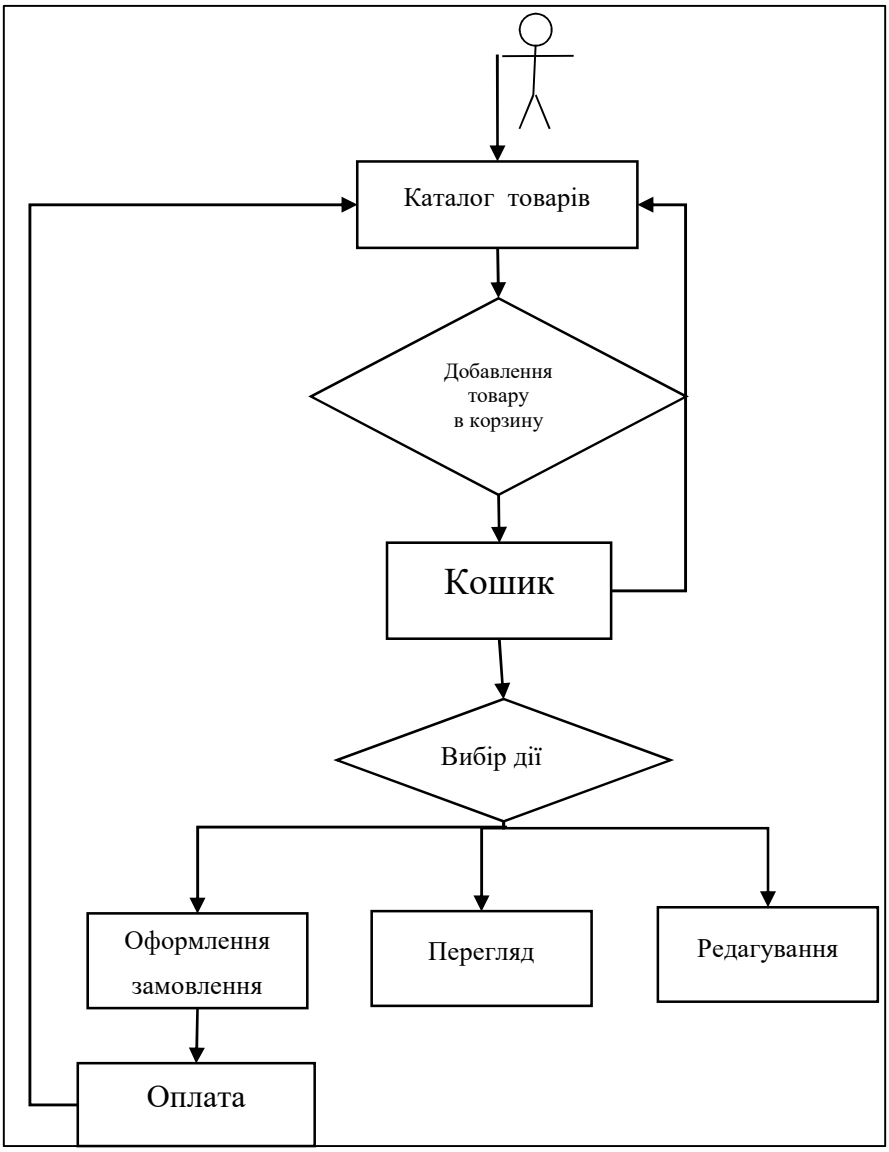


Рисунок 2.5 – Діаграма активності роботи клієнта з сайтом

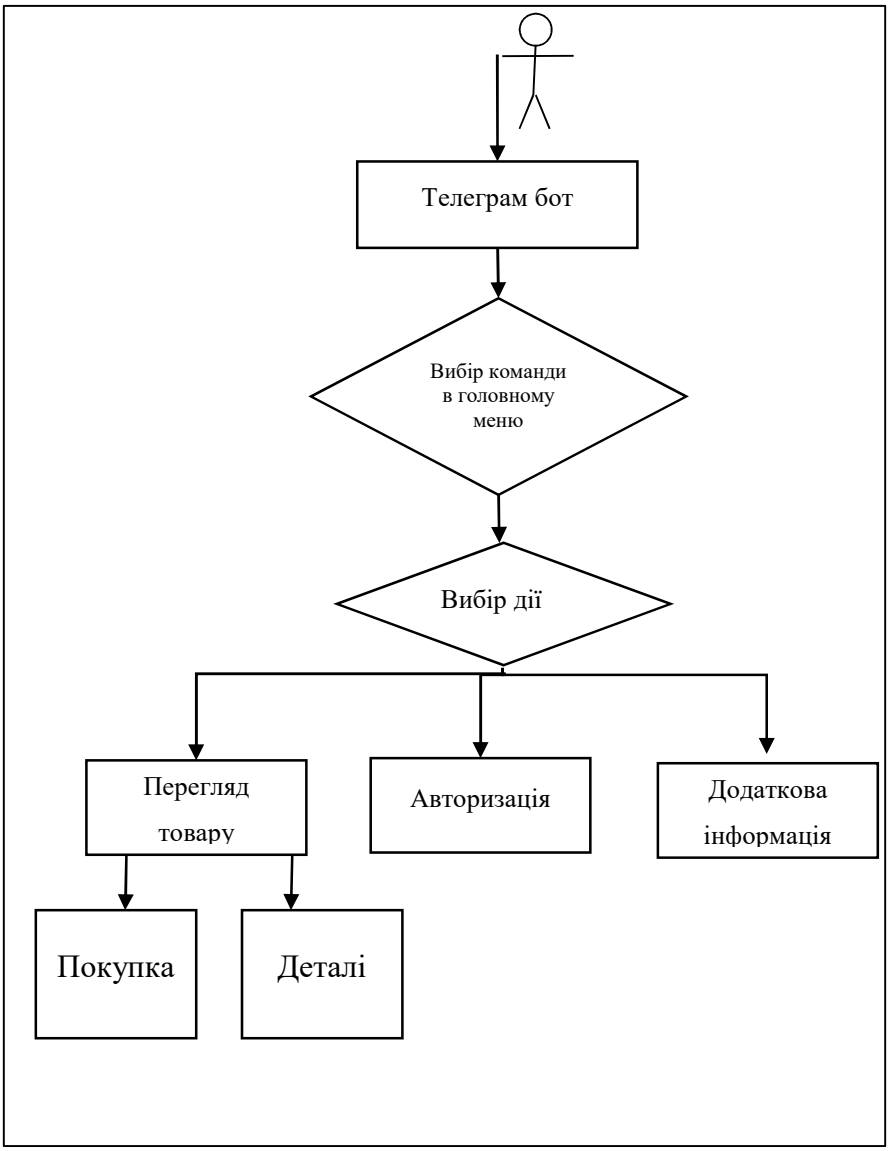


Рисунок 2.6 – Діаграма активності роботи клієнта з ботом

Дотримуючись усіх цих бізнес процесів можна створити систему яка допомагатиме адміністратору магазину вести електронну версію книгарні, аналізувати статистику (якщо буде достатньо інформації для її створення), а також допомагатиме користувачу купити товар. Така система буде здатна конкурувати з іншими оскільки вона включає в себе все що потрібно для роботи з інтернет магазином. Плюси такої системи - це швидке контролювання каталогу книг, моніторинг покупок для адміністратора, а також можливість клієнта швидкої покупки товару через інтернет.

2.2 Інформаційна структура системи

Для розробки структури бази даних інтернет-магазину книг необхідно зберігати дані про сутності предметної області, а саме: Workers, Orders, Posts, Clients, Authors, Editions, Type_edition, Genres, Books. Таблиці названі латиницею для зручності створення за допомогою SQL запиту. В результаті чого одержимо базу даних, структуру якої зображено на рисунку 2.7.

Рисунок 2.7 – Структура бази даних

Таблиця «Posts» призначена для зберігання даних про посади працівників, й містить атрибути, наведені в таблиці 2.1.

Таблиця 2.1 – Атрибути таблиці БД «Posts»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Post_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	Name	VARCHAR	Поле для назви посади

Таблиця «Type_edition» призначена для зберігання даних про тип видання, й містить атрибути, наведені в таблиці 2.2.

Таблиця 2.2 – Атрибути таблиці БД «Type_edition»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Etype_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	Name	VARCHAR	Поле для назви типу видання

Таблиця «Genres» призначена для зберігання даних про жанри видання, й містить атрибути, наведені в таблиці 2.3.

Таблиця 2.3 – Атрибути таблиці БД «Genres»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Genre_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	Name	VARCHAR	Поле для назви жанру

Таблиця «Authors» призначена для зберігання даних про авторів видання, й містить атрибути, наведені в таблиці 2.4.

Таблиця 2.4 – Атрибути таблиці БД «Authors»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Author_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	Name	VARCHAR	Поле для назви жанру

Таблиця «Workers» призначена для зберігання даних про працівників, й містить атрибути, наведені в таблиці 2.5.

Таблиця 2.5 – Атрибути таблиці БД «Workers»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Worker_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	First_name	VARCHAR	Поле для ім’я працівник
3.	Second_name	VARCHAR	Поле для прізвища працівник
4.	Address	Date	Поле для адресу
5.	Login	VARCHAR	Поле для логіну працівник
6.	Password	VARCHAR	Поле для пароля для працівника
7.	Email	VARCHAR	Поле для пошти працівника
8.	Phone	VARCHAR	Поле для номера телефону працівник
9.	Date_hiring	Date	Поле для запису дати прийняття на роботу працівника
10.	Date_birth	Date	Поле для запису дати народження працівника
11.	Date_fired	Date	Поле для запису дати звільнення на роботу працівника
12.	PostId	INT	Поле для запису зовнішніх ключів з таблиці Posts (таблиці зєднані між собою)
13.	- Library_id	INT	Поле для запису зовнішніх ключів з таблиці Libraries (таблиці зєднані між собою)

Таблиця «Clients» призначена для зберігання даних про клієнтів , й містить атрибути, наведені в таблиці 2.6.

Таблиця 2.6 – Атрибути таблиці БД «Clients»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Client_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	First_name	VARCHAR	Поле для ім’я клієнта
3.	Second_name	VARCHAR	Поле для прізвища клієнта
4.	Address	VARCHAR	Поле для адресу клієнта
5	Phone	VARCHAR	Поле для номера телефону клієнта

Таблиця «Edition» призначена для зберігання даних про видання , й містить атрибути, наведені в таблиці 2.7.

Таблиця 2.7 – Атрибути таблиці БД «Edition»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Edition_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
3.	Etype_id	INT	Поле для запису зовнішніх ключів з таблиці Type_editions (таблиці зєднані між собою)
5	Name	VARCHAR	Поле для назви видавництва
6	Price	INT	Поле для ціни видання
5	Data_Start	DATE	Поле для дати надрукування видання
6	Data_End	DATE	Поле для дати закінчення терміну придатності видання

Таблиця «Books» призначена для зберігання даних про примірники , й містить атрибути, наведені в таблиці 2.8.

Таблиця 2.8 – Атрибути таблиці БД «Books»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Book_id	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	Author_id	INT	Поле для запису зовнішніх ключів з таблиці Authors (таблиці зєднані між собою)
3.	Name	INT	Поле для назви книги
4.	Genre_id	INT	Поле для запису зовнішніх ключів з таблиці Genres (таблиці зєднані між собою)
5.	Edition_id	INT	Поле для запису зовнішніх ключів з таблиці Editions (таблиці зєднані між собою)

Таблиця «Orders» призначена для зберігання даних про замовлення , й містить атрибути, наведені в таблиці 2.9.

Таблиця 2.9 – Атрибути таблиці БД «Orders»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	OrdersId	INT	Первинний ключ для однозначності ідентифікації записів у таблиці
2.	Book_id	INT	Поле для запису зовнішніх ключів з таблиці Books (таблиці зєднані між собою)
3.	Client_id	INT	Поле для запису зовнішніх ключів з таблиці Clients (таблиці зєднані між собою)
4.	Workerk_id	INT	Поле для запису зовнішніх ключів з таблиці Workers (таблиці зєднані між собою)
5.	Data_Issue	DATE	Поле для дати видачі
6.	Data_Return	DATE	Поле для дати повернення
7.	Count	INT	Поле для кількості товару
7.	Sum	INT	Поле загальної вартості замовлення

Отже, для вирішення поставленої задачі підходить наведена база даних інтернет-магазину книг.

Для кожного бізнес-процесу(Рисунок 2.3) відповідає одна, або декілька таблиць БД, детальний опис наведений нижче:

- Бізнес-процес «Замовлення». Процес передбачає взаємодію адміністратора з такими таблицями, як Orders, Clients, Workers, тобто користувач, який має права адміністратора може переглянути список замовлень які зробили покупці, крім того він може їх змінити(редагувати, добавити, видалити).
- Бізнес-процес «Книжковий фонд». Процес передбачає взаємодію адміністратора з такими таблицями, як Books, Edition, Genre, Type_Edition, Authors, тобто користувач, який має права адміністратора може переглядати весь

товар та всю інформацію про нього(автор, жанр, видання), яка зберігається в інших таблицях, крім того він може їх змінити(редагувати, добавити, видалити).

- Бізнес-процес «Кадр». Процес передбачає взаємодію адміністратора з такими таблицями, як Workers, Posts, тобто він може переглянути інформацію по працівникам магазину та їх посаді, а також змінювати ці дані.
- Бізнес-процес «Книжки». Процес передбачає взаємодію звичайного користувача, тобто покупця з такими таблицями, як Books, Edition, Genre, Type_Edition, це дає йому змогу переглянути весь товар, який наявний в магазині та його деталі, але він не може змінювати цю інформацію.
- Бізнес-процес «Кошик». Процес передбачає взаємодію користувача з такими таблицями, як Books, Orders, Clients, це дає йому змогу додавати обраний товар в кошик, його оформити та купити.
- Бізнес-процес «Бот». Процес передбачає взаємодію користувача з аналогічними таблицями бізнес-процесу «книжки», єдина різниця користувач переглядає товар через бота в телеграмі.

2.3 Вибір засобів розробки інформаційної системи

Для розробки веб-додатку буде використовуватися декілька технологій. Спершу було обрано C#, як мову програмування та платформу .NET на якій ця мова працює. C# [15] – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Хоча ця мова відносно молода, на теперішній час вона являється однією із топ 3 мов програмувань і за допомогою можна розробляти будь які програми та будь якої складності.

Для кращої структуризації проєкту буде використовуватись ASP.NET MVC Framework [16] – фреймворк для створення веб-застосунків, який реалізує шаблон Model-view-controller. Цей шаблон передбачає поділ системи на три частини: модель, вигляд та контролер. Модель застосовується для опису даних та їх використання, вигляд – це інтерфейс користувача, зазвичай це просто веб-

сторінка і контролер, який з’єднує модель і вигляд та відповідає на запроси користувача.

І для того щоб зберігати дані в проєкті буде використано систему керування базою даних SQL Server та для доступу цих даних технологія Entity Framework

Для створення бота відбувається за допомогою спеціальної утиліти «@BotFather», який значно спрощує процес розробки. також буде використовуватись мова C# і бібліотека “Telegram.Bot”. Ця бібліотека є API, тобто інтерфейсом на основі HTTP за допомогою якого користувач зможе звертатися до сервера, ця бібліотека є обов’язковою для працездатності «Телеграм» бота.

Розділ 3

Програмна реалізація інформаційної системи

3.1 Структура та функціональне призначення складових системи

Проєкт використовує фреймворк ASP.NET MVC Framework, який реалізує шаблон проєктування MVC. Суть цього шаблону полягає в розділенні програми на моделі, контролери та представлення. У Моделі реалізується бізнес-логіка програми. У представленні знаходиться вивід даних системи. Контролер реалізує зв’язок між моделями та представленнями.

Для кращої структури проєкту вирішено створити рішення Visual Studio з чотирма проєктами(рисунок 3.1), перших два основних, які будуть відповідати за веб-сайт: BookStore.Domain та BookStore.WebUI. BookStore.Domain (рисунок 3.2) проєкт який містить сутність і логіку предметної області, налаштовується на забезпечення сталості за допомогою сховища, яке створено за допомогою інфраструктури Entity Framework. BookStore.WebUI (рисунок 3.3) - містить контролери, моделі та представлення, виступає в якості інтерфейсу додатку. Наступні два це BookStore.Bot та BookStore.UnitTests, перший відповідає за роботу бота в телеграмі а другий буде містити в собі тести для перевірки додатка. Хоча така архітектура може здаватися складної та трудомісткою, в подальшому розвитку програми вона окупаються, забезпечуючи придатний для супроводу, розширюваний, добре структурований код з чудовою підтримкою модульного тестування. Крім того телеграм бот та веб сайт будуть працювати окремо, саме тому якщо виникнуть проблеми з одним із них, інший буде працювати без проблем.

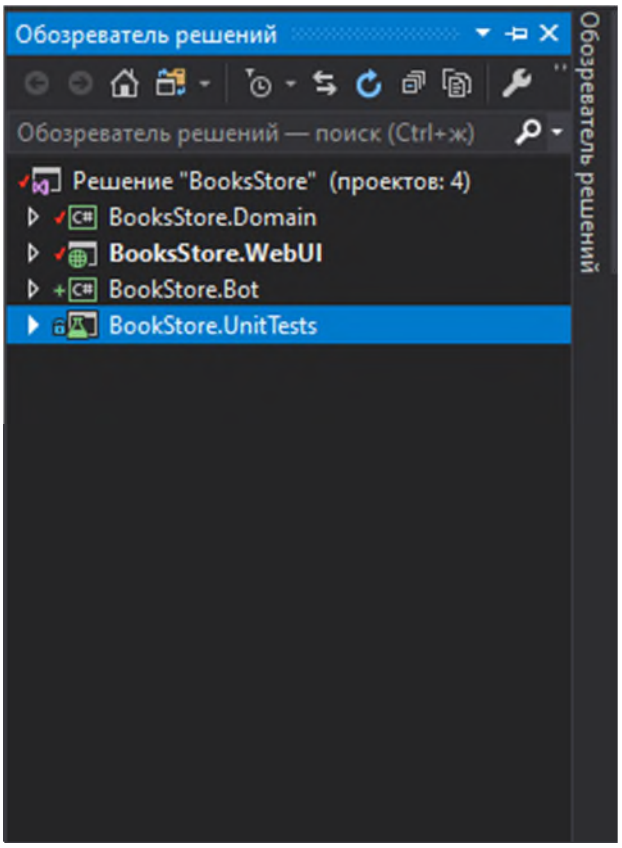


Рисунок 3.1 – Структура проєкту «BookSotore»

В MVC проєкті знаходяться 6 контролерів які відповідають за функціонування веб сайту, а саме:

- AccountController - відповідає за автентифікацію реєстрацію користувача;
- AdminController - відповідає за реалізацію можливостей адмінстратора;
- BookController - відповідає за відображення товарів;
- CartController - відповідає за реалізацію роботи користувача з корзиною;
- NavController – відповідає за фільтрацію товарів;

Кожен контролер пов’язаний зі своєю моделлю для отримання даних, та має представлення для відображення цих даних.

Рисунок 3.2 - Діаграма класів проєкту BookStore. Domain

Рисунок 3.3 - Діаграма класів проєкту BookStore.WebUI

3.2 Особливості реалізації складових системи

Головною сторінкою сайту інтернет-магазину буде каталог товарів, яка зображена на рисунку 3.4. Потрапляючи на цю сторінку покупець одразу бачить весь список книжок які наявні в магазині.

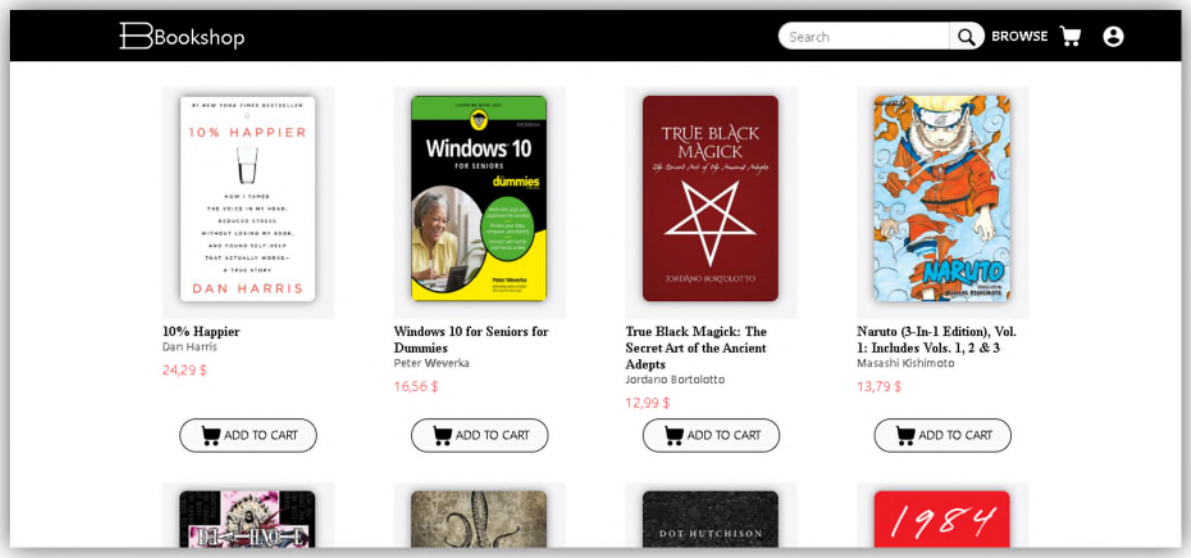


Рисунок 3.4 – Головна сторінка сайту.

Весь каталог розбитий на сторінки, на одній сторінці вміщається 8 товарів. Про кожну книгу клієнт бачить основну інформацію (рисунок 3.5), а саме: назва, автор, ціна та фото, завдяки чому легко може орієнтуватися і вибрати потрібний йому товар, який його зацікавив. Окрім того на сайті також можливо переглядати товар за вибраним жанром (рисунок 3.6).

Рисунок 3.5 – Інформація про товар.

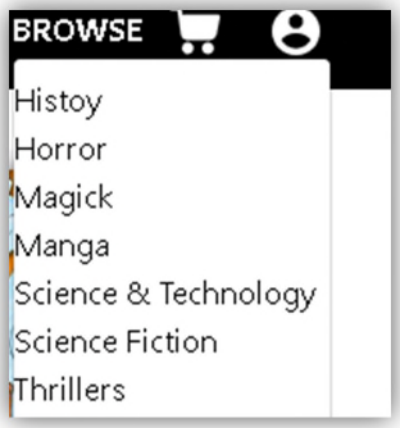


Рисунок 3.6 – Сортування товару за жанром.

Користувач, який вперше потрапив на сайт може одразу зробити замовлення обраних ним книг без будь-яких обмежень. За допомогою кнопки «ADD TO CART», вибраний товар потрапляє в корзину, і покупець потрапляє на сторінку кошика яка продемонстрована на рисунку 3.7.

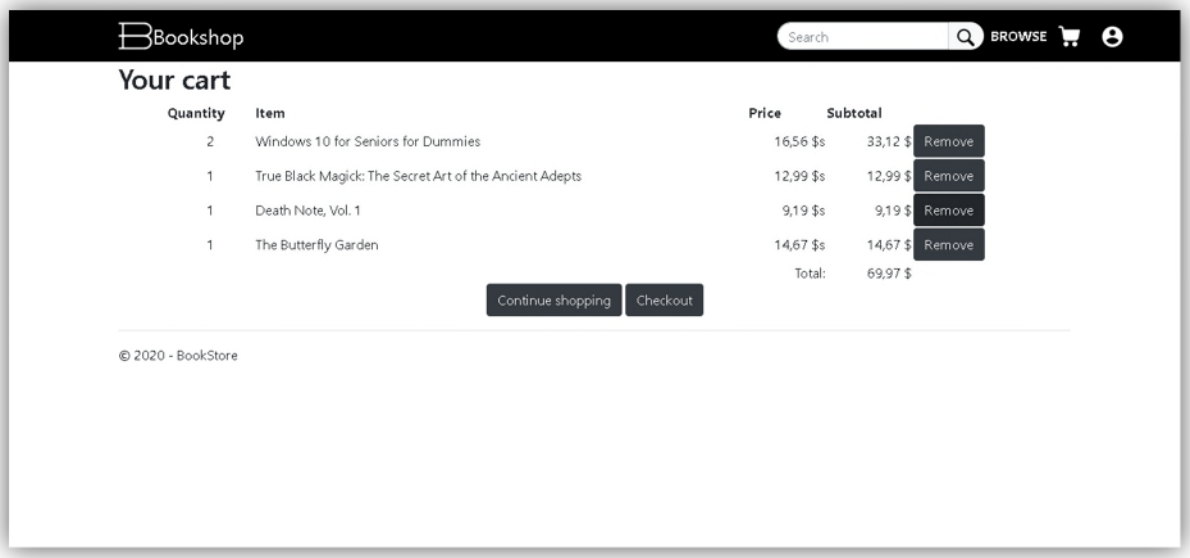


Рисунок 3.7 – Кошик покупця.

На цій сторінці відображені всі товари, які були добавлені в кошик, а також інформацію про їхню кількість, назву, ціну за одиницю книжки та повну ціну. Крім цього покупець може видалити не потрібну книгу натиснувши кнопку «Remove», продовжити покупку «Continue shopping» та перейти оформлення замовлення «Checkout». Також для того, щоб кожного разу не переходити в кошик, можна переглянути основну інформацію за допомогою часткового представлення, яке викликається за допомогою іконки (кошик) на шапці сайту (рисунок 3.8), окрім того можна перейти в кошик за допомогою посилання «Checkout».

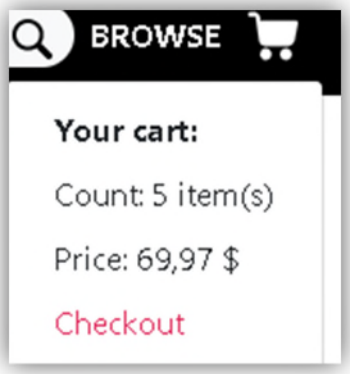


Рисунок 3.8 – Часткове представлення кошика.

Вибравши потрібний товар та перейшовши до його оформлення, покупець потрапляє на сторінку (рисунок 3.9) в якій потрібно ввести свої дані (ім’я, фамілія, адрес), кожне поле є обов’язковим, якщо воно не заповнене, або ж не коректне, то з’явиться оповіщення, яке повідомить що це за помилка.

Рисунок 3.9 – Деталі замовлення.

Якщо клієнт заповнить все вірно та на натисне кнопку «Buy» він побачить повідомлення про успішність покупки (рисунок 3.10).

Рисунок 3.10 – Повідомлення про успішне замовлення.

Хоча незареєстрований покупець ніяк необмежений в функціоналі, для того щоб при замовленні кожного разу він не вводив постійно одні і ті ж самі дані, на сайті передбачена система реєстрації (рисунок 3.11) та авторизації (рисунок 3.12)

Під час реєстрації нового користувача відбувається перевірка чи вже існує користувач з таким логіном, а при вході на сайт перевіряється коректність паролю.

Окрім того всі поділення по ролям на звичайних користувачів та адміністраторів, і в залежності від тої чи іншої ролі контент сайту буде відрізнятися.

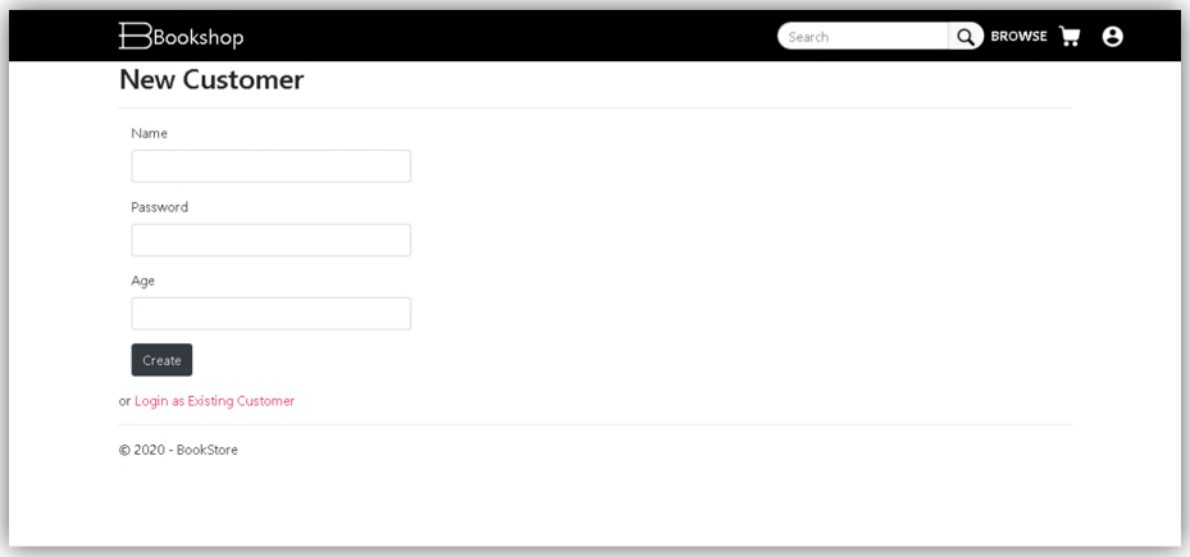


Рисунок 3.11 – Форма для реєстрації користувача.

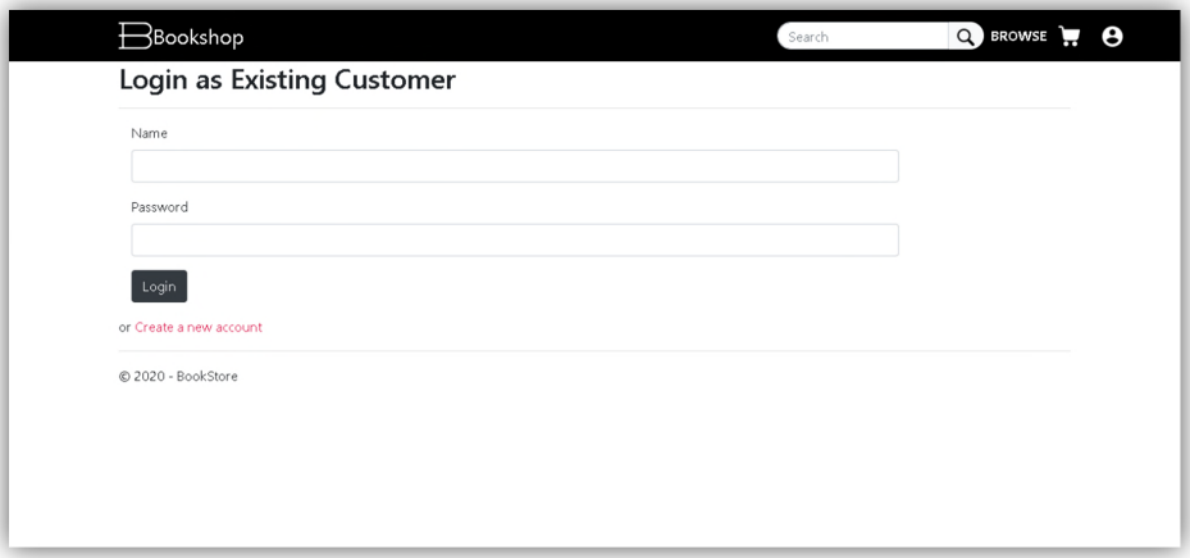


Рисунок 3.12 – Форма для авторизації користувача.

Якщо це звичайний користувач він буде бачити представлення яке зображено на рисунку 3.13.

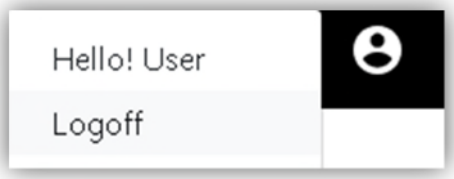


Рисунок 3.13 – Представлення інформації зареєстрованого користувача.

Адміністратор сайту крім цього буде мати доступ до панелі адміністрування (рисунок 3.14).

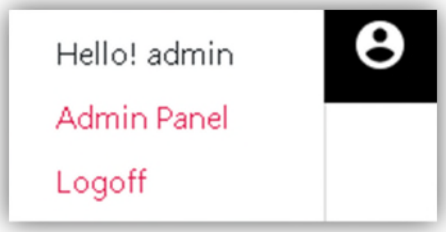


Рисунок 3.14 – Представлення інформації адміністратора сайту.

За допомогою цієї панелі адмін може переглядати всі товари (рисунок 3.15), а також редагувати (рисунок 3.16), видаляти та додавати новий товар (рисунок 3.17).

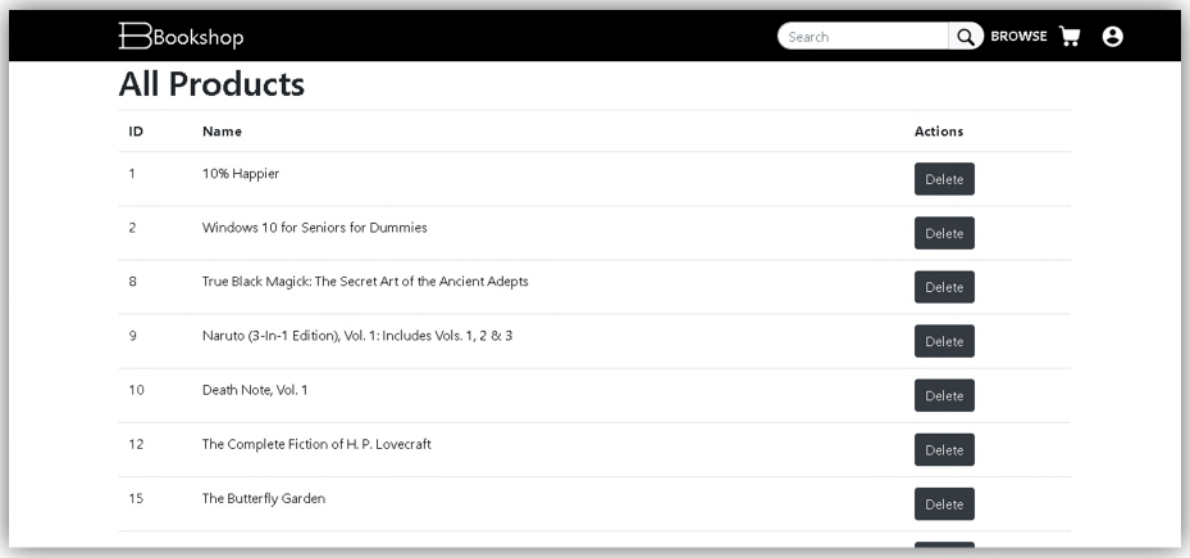


Рисунок 3.15 – Список товару на панелі адміністрування.

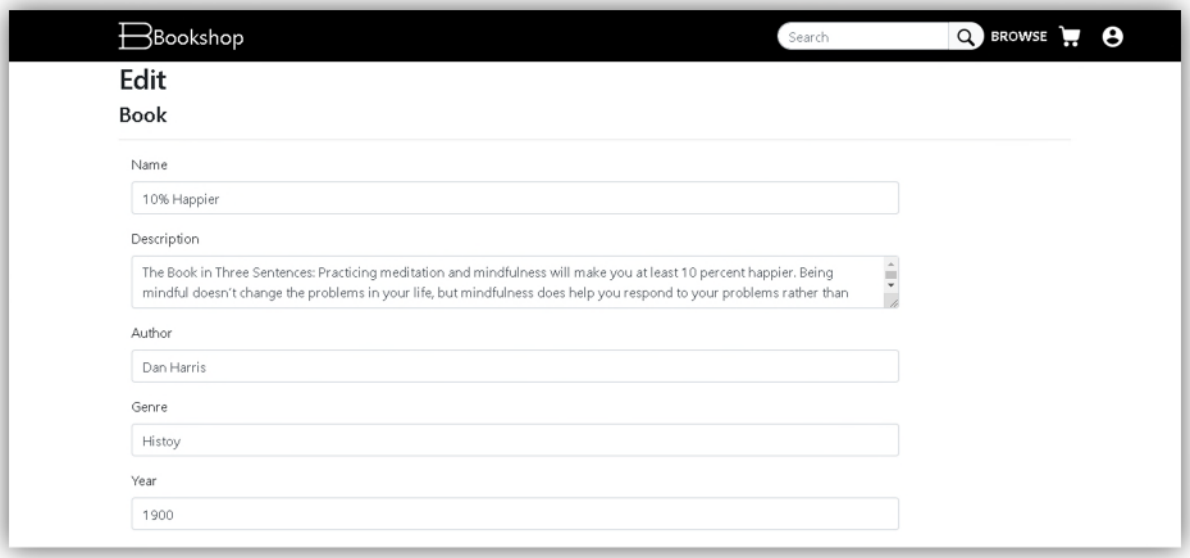


Рисунок 3.16 – Редагування товару за допомогою панелі адміністрування.

Рисунок 3.17 – Додавання нового товару за допомогою панелі адміністрування.

Робота бота здійснюється за допомогою самого «Телеграму», для цього потрібно лише його зареєструвати, після чого в проєкті потрібно налаштувати підключення до нього використовуючи потрібну бібліотеку.

Алгоритм роботи бота досить простий. Повідомлення, команди і запити, надіслані користувачами, передаються на програмне забезпечення, запущене на сервері. Посередницький анонімний сервер Telegram обробляє шифрування і

здійснює зворотний зв'язок між утилітою і користувачем. Взаємодію між користувачем та ботом можна переглянути на рисунку 3.18.

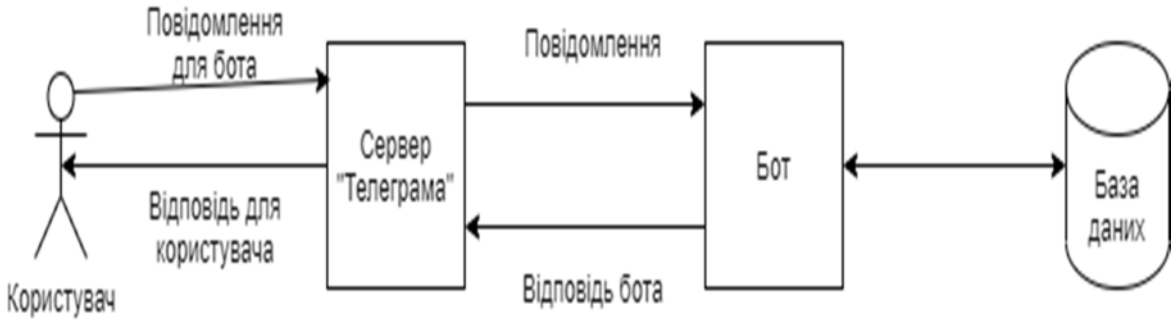


Рисунок 3.18 – Схема взаємодії користувача та бота.

Перейшовши за посланням в телеграм та розпочавши роботу бота з’явиться його основне меню (рисунок 3.19).

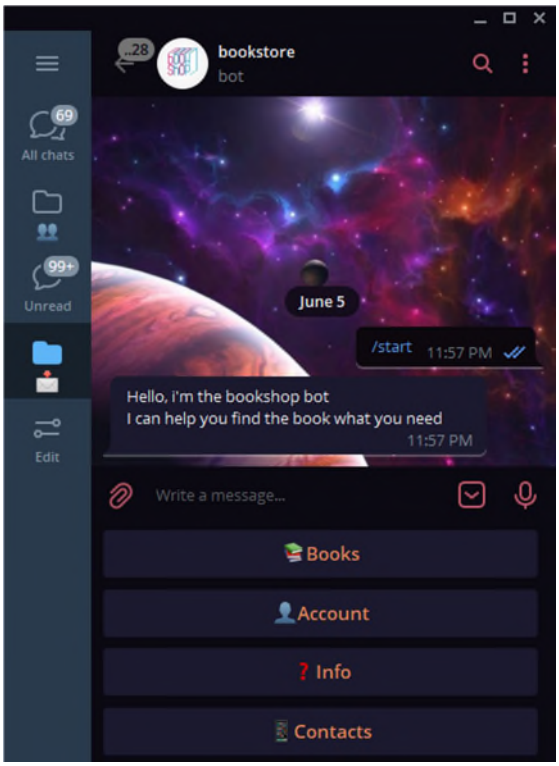


Рисунок 3.19 – Основне меню телеграм бота.

Основна функція бота це перегляд товару магазина, що здійснюється за допомогою пункту меню «Books». Бот спочатку виводить одну книжку з бази даних, після чого надає доступ до іншого меню, за допомогою якого з’являється можливість перегляду всього товару магазину (Рисунок 3.20).

Рисунок 3.20 – Вивід товару за допомогою бота.

Крім цього з’являється можливість перегляду описання книги і її купівлі, але для цього ще потрібно авторизуватися в головному меню за допомогою пункту «Account» (Рисунок 3.21). Для того, щоб зайти під своїм акаунтом потрібно попередньо зареєструватись на сайті, після аутентифікації користувача в нього з’являється можливість покупки товару. Також бот надає додаткову інформацію про магазин (Рисунок 3.22).

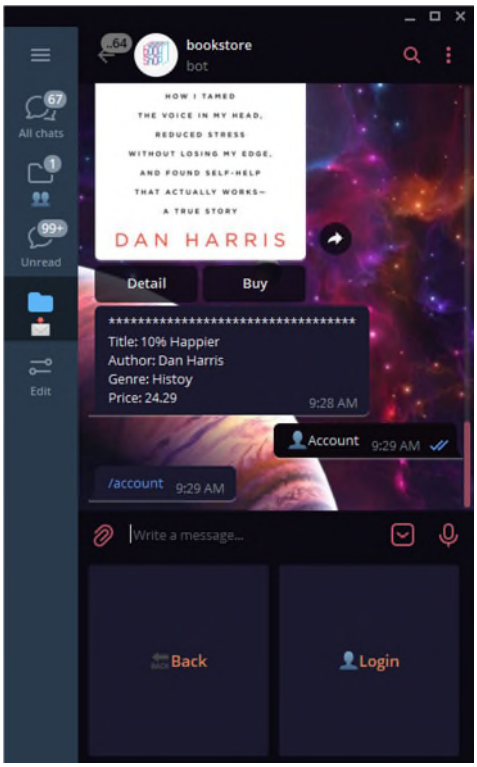


Рисунок 3.21 – Авторизація користувача за допомогою бота.

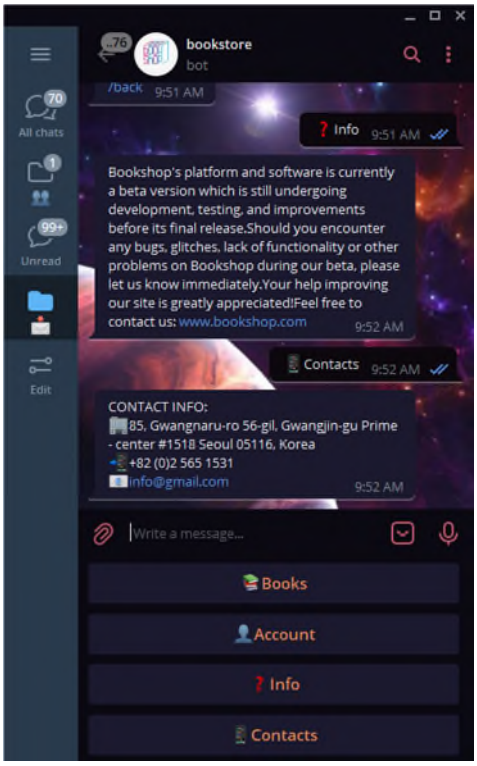


Рисунок 3.22 – Додаткова інформація про магазин в боті.

3.3 Тестування

Для тестування даного сайту було використано вбудовану підтримку модульного тестування Visual Studio, хоча доступні й інші пакети модульного тестування .NET. Найбільш популярним з них є, мабуть, NUnit, проте всі пакети тестування в основному роблять одне і те ж. Причина вибору інструментів тестування Visual Studio пов'язана з привабливістю інтеграції з іншими частинами IDE-середовища.

Було створено ряд тестів які перевіряють коректність роботи контролерів. Для прикладу на рисунку 3.23 продемонстровано результат тесту який перевіряє роботу методу дії, який реалізую відображання всіх товарів на адмін панель.

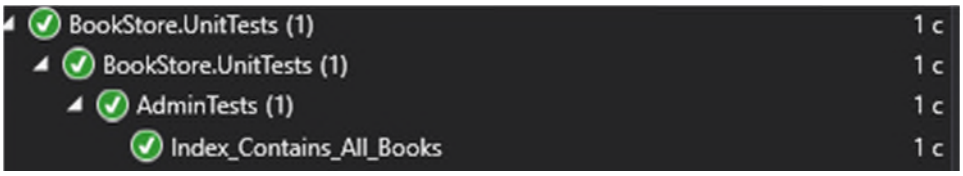


Рисунок 3.23 – Результат роботи тесту для перевірки методу Index контролера Admin.

Також було створено модульне тестування: відправки, пов'язані з редагуванням товару за допомогою панелі адміністрування (рисунок 3.24)

Рисунок 3.24 – Результат роботи тестів для перевірки методу Edit контролера Admin.

Крім цього створено декілька тестів для перевірки функціоналу кошика (рисунок 3.25).

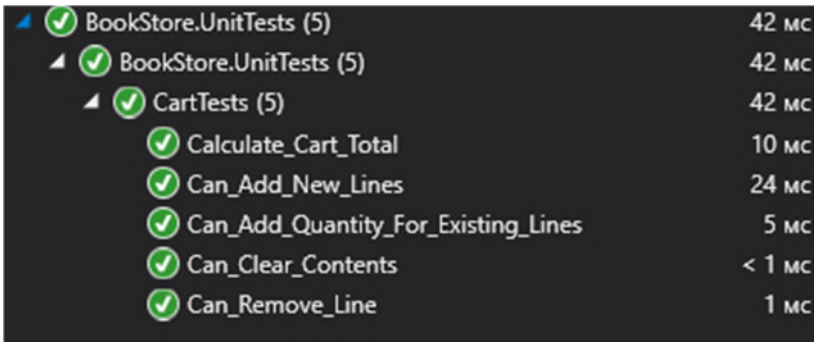


Рисунок 3.25 – Результат роботи тестів які перевіряють функціонал кошика.

Отже, з результатів тестувань можна побачити, що основний функціонал сайту працює правильно при задовільному часі, з чого можна зробити висновок, що навіть при великій кількості користувачів сайт буде працювати стабільно.

3.4 Інструкція користувача

Зайшовши на сайт, користувач одразу потрапить на головну сторінку, на якій відображено каталог книжок, зображена на рисунку 3.4. Вибравши потрібну книгу він може одразу перейти до її покупки, для цього потрібно натиснути кнопку «ADD TO CART» (рисунок 3.26) .

Рисунок 3.26 – Кнопка покупки товару.

Після чого покупець перейде в кошик (рисунок 3.6), в якому буде добавлений обраний товар, для подальшого оформлення замовлення потрібно натиснути кнопку «Checkout» (рисунок 3.27).

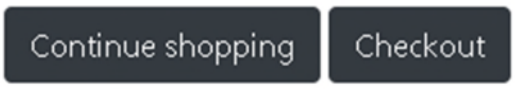


Рисунок 3.27 – Кнопки функціонування кошика.

Після чого потрібно правильно заповнити дані (рисунок 3.28) та натиснути кнопку «Buy» .

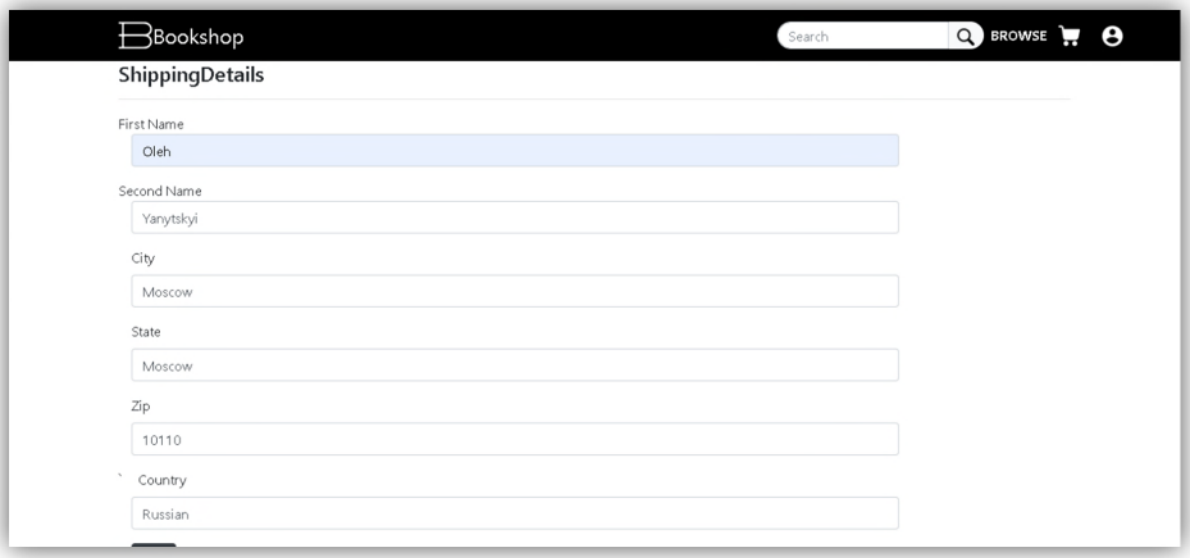


Рисунок 3.28 – Заповнена форма деталі замовлення.

Якщо покупка пройшла успішно клієнт побачить про це повідомлення (рисунок 3.9).

Крім сайту користувач може переглядати та купувати товар за допомогою бота, перейшовши за посилання та розпочавши його роботу, надається доступ до головного меню яке дозволяє використовувати основний функціонал цього бота (Рисунок 3.19).

Для покупки товару спочатку потрібно авторизуватися, в головному меню вибрати пункт «Account», а потім «Login» дальше потрібно ввести email, за допомогою якого попередньо було зареєстрований акаунт на сайті. Щоб бот правильно зчитав пошту її слід вводити таким чином «email useremail», де email це просто ключове слово, useremail це вже пошта користувача, якщо пошта правильна і такий користувач існує в базі даних, про це прийде повідомлення і

дальше з’явиться доступ для вводу паролю, в іншому випадку повідомлення про відсутність такого користувача (Рисунок 3.29). Введення паролю здійснюється аналогічно, лише з ключовим словом «password»

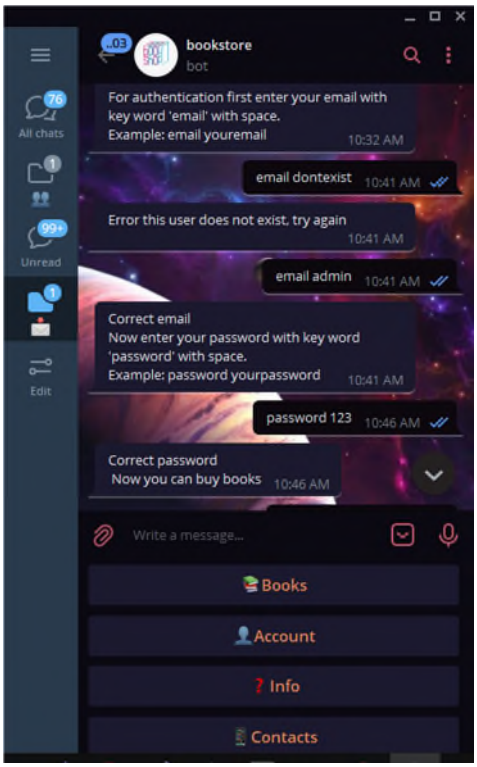


Рисунок 3.29 – Авторизація в телеграм боті.

Після автентифікації користувач може купити товар за допомогою кнопки «Buy», якщо все правильно, прийде повідомлення про успішність операції (Рисунок 3.30).

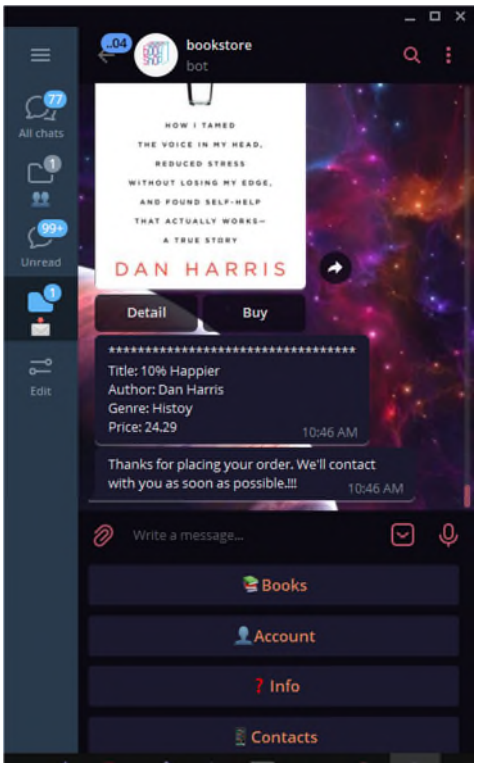


Рисунок 3.30 – Покупка книги в телеграм боті.

Отже, як користування сайтом так і ботом є інтуїтивно зрозумілим та не вимагає додаткових знань, умінь та навичок, окрім необхідних для звичайного користування комп’ютером.

3.5 Вимоги до розгортання інформаційної системи

Мінімальні системні вимоги:

Браузер:

- Mozilla Firefox 3.0 і вище;
- Google Chrome 3.0.195 і вище;

Вимоги до провайдера :

- Операційна система: Windows Server 2012
- Платформа Microsoft.AspNet Mvc Framework версії 5.2.7 і вище
- EntityFramework фреймворк версії 6.4 і вище
- Bootstrap версії 4.4.1 і вище
- База даних Microsoft SQL Server 2017

Додаткові вимоги до системи:

– для роботи вимагається підключення до Інтернету;

– не підходить Internet Explorer та інші застарілі браузери або старі версії сучасних браузерів;

– сайт не створений для перегляду на мобільних пристроях або інших пристроях з маленьким розміром екрану, але є альтернатива, використання телеграм бота.

Висновки

В результаті виконання даної кваліфікаційної роботи бакалавра було отримано веб-додаток Книгарня, який працює по бізнес-моделі B2C для продажі книжок, розроблений за допомогою засобів ASP .NET MVC 5, Entity framework, і MS SQL Server. Всі вимоги, поставленні до роботи, були виконані, а саме була створена база даних, та веб-додаток який реалізує всі бізнес-процеси та основні функції для ведення електронної комерції, а саме: автентифікація користувачів, управління товаром, система покупок, система управління. Для користувачів смартфонів був створений бот, за допомогою, якого можна переглядати та купувати товар, який наявний на сайт у форматі чату в месенджері «Телеграм». В перспективі для даного сайту та бота можна придбати хостинг, підняти сайт та налагодити обслуговування для реальних задач.

Перелік посилань

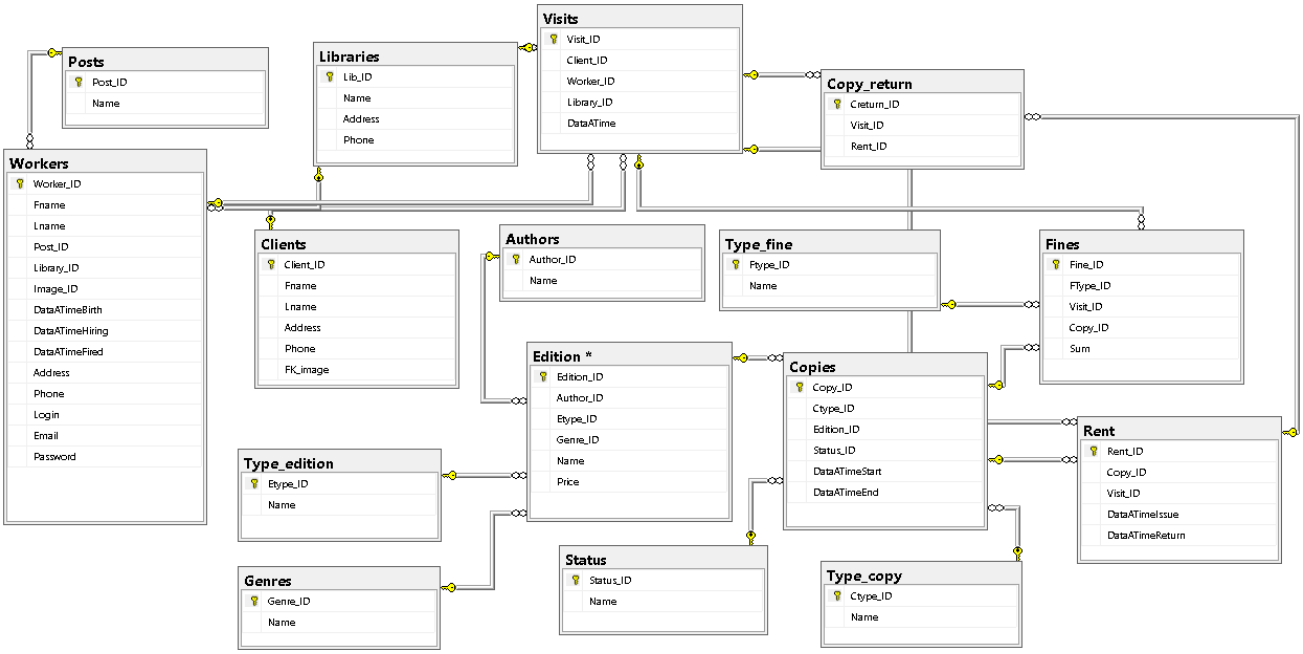
1. Wikipedia.com. Бібліотека [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Бібліотека
2. Wikipedia.com. Примірник [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ Примірник
3. Wikipedia.com. Видання [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ Видання
4. Wikipedia.com. Microsoft_SQL_Serve [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ Microsoft_SQL_Serve
5. Wikipedia.com. Бібліотека [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Бібліотека
6. Wikipedia.com. Бібліотечний_фонд [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ Бібліотечний_фонд
7. АРМ [Електронний ресурс]. – Режим доступу: http://ube.nlu.org.ua/article/
8. Бібліотека Конгресу [Електронний ресурс]. – Режим доступу: https://www.loc.gov
9. BOOK .RU [Електронний ресурс]. – Режим доступу: https://www.zara.ru
10. API [Електронний ресурс]. – Режим доступу: https://www.quality-assurance-group.com/korotko-pro-api-jogo-testuvannya/
11. Wikipedia.com. MVC [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/mvc
12. Wikipedia.com. SQL Server [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/sql_server
13. Wikipedia.com. MySQL [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/mysql
14. ASP.NET vs PHP [Електронний ресурс]. – Режим доступу: https://existek.com/blog/aspnet-vs-php-which-is-better-for-web-development/
15. Wikipedia.com. Entity Framework [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/entity_framework

16. Wikipedia.com. Visual Studio [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio

ДОДАТКИ

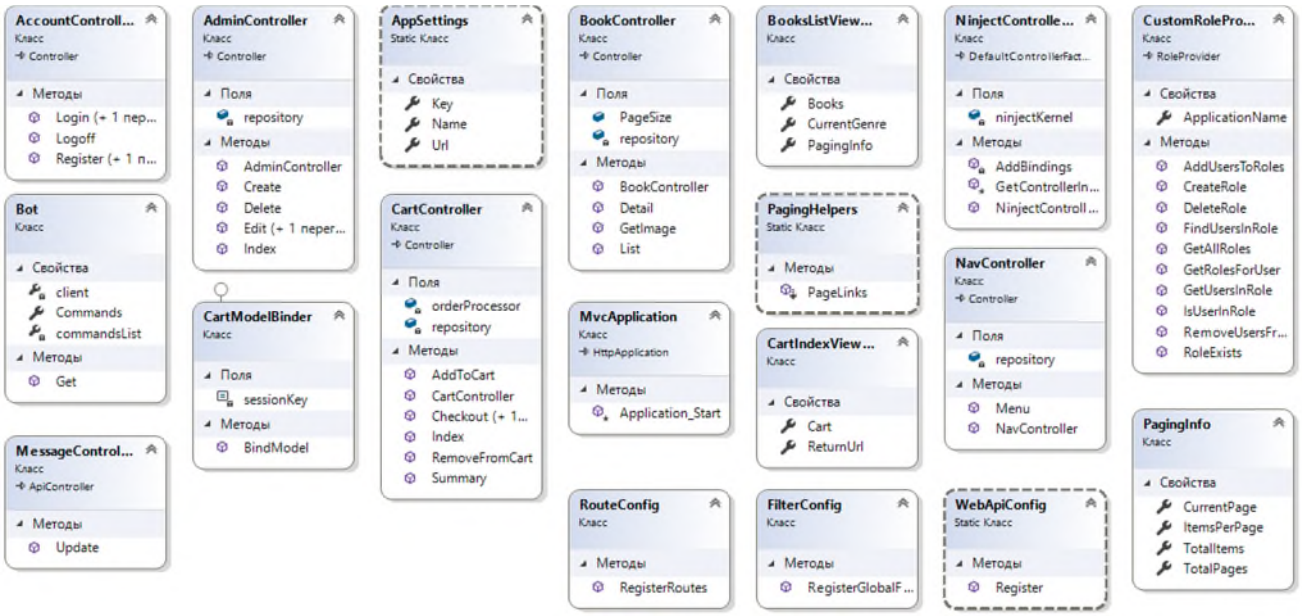
Додаток А

Структура бази даних інтернет-магазину



Додаток Б

Розгорнута структура класів інтернет-магазину



Додаток В

Програмні коди

Лістинг CopiesController.cs:

```
namespace Library.Controllers.Tables
{
    public class CopiesController : Controller
    {
        // GET: Copies
        public ActionResult Index()
        {
            return View();
        }

        LibCon db = new LibCon();
        public ActionResult ListC()
        {
            var copies = db.Copies.Include(c =>
c.Condition).Include(t => t.Tcopy).Include(e =>
e.Edition).ToList();
            return View(copies);
        }


        [HttpGet]
        public ActionResult Create()
        {
            SelectList conditions = new
SelectList(db.Conditions, "Id", "Name");
            SelectList tcopies = new
SelectList(db.Tcopies, "Id", "Name");
            SelectList editions = new
SelectList(db.Editions, "Id", "Name");
            ViewBag.Conditions = conditions;
            ViewBag.Tcopies = tcopies;
            ViewBag.Editions = editions;
            return PartialView();

        }

        [HttpPost]
        public ActionResult Create(Copy copy)
        {
            db.Copies.Add(copy);
            db.SaveChanges();
            return RedirectToAction("ListC");
        }


        [HttpGet]
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Copy copy = db.Copies.Find(id);
            if (copy != null)
            {
                SelectList conditions = new
SelectList(db.Conditions, "Id", "Name",copy.ConditionId);
                SelectList tcopies = new
SelectList(db.Tcopies, "Id", "Name",copy.TcopyId);
                SelectList editions = new
SelectList(db.Editions, "Id", "Name",copy.EditionId);
                ViewBag.Conditions = conditions;
                ViewBag.Tcopies = tcopies;
                ViewBag.Editions = editions;
                return PartialView(copy);
            }
            return RedirectToAction("ListC");
        }


        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Copy copy = db.Copies.Find(id);
            if (copy != null)
            {
                db.Copies.Remove(copy);
                db.SaveChanges();
            }
            return RedirectToAction("ListC");

        }


        [HttpPost]
        public ActionResult Edit(Copy copy)
        {
            db.Entry(copy).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("ListC");
        }

    }


}
```

Лістинг RentsController.cs:

```
namespace Library.Controllers.Tables
{
    public class RentsController : Controller
    {
        LibCon db = new LibCon();
        public ActionResult ListR()
        {
            var ren = db.Rents.Include(r =>
r.Copy).Include(l => l.Library).Include(l =>
l.Visit).ToList();
            return View(ren);
        }
        // GET: Rents
        public ActionResult Index()
        {
            ViewData["SubTitle"] = "Welcome in ASP.NET MVC
5 Library Project ";
            ViewData["Message"] = "It is Open Library.";
            return View();
        }


        [HttpGet]
        public ActionResult Create()
        {
            SelectList copies = new SelectList(db.Copies ,
"Id", "DataATimeStart");
            SelectList libraries = new
SelectList(db.Libraries, "Id", "Name");
            SelectList visits = new SelectList(db.Visits,
"Id", "Data");
            ViewBag.Libraries = libraries;
            ViewBag.Visits = visits;
            ViewBag.Copies = copies;
            return PartialView();

        }

        [HttpPost]
        public ActionResult Create(Rent rent)
        {
            db.Rents.Add(rent);
            db.SaveChanges();
            return RedirectToAction("ListR");
        }


        [HttpGet]
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Rent rent = db.Rents.Find(id);
            if (rent != null)
            {
```

```
                SelectList copies = new
SelectList(db.Copies, "Id", "DataATimeStart",
rent.CopyId);
                SelectList libraries = new
SelectList(db.Libraries, "Id", "Name",rent.LibraryId);
                SelectList visits = new
SelectList(db.Visits, "Id", "Data",rent.VisitId);
                ViewBag.Libraries = libraries;
                ViewBag.Copies = copies;
                ViewBag.Visits = visits;
                return PartialView(rent);
            }
            return RedirectToAction("ListR");
        }


        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Rent rent = db.Rents.Find(id);
            if (rent != null)
            {
                db.Rents.Remove(rent);
                db.SaveChanges();
            }
            return RedirectToAction("ListR");

        }

        [HttpPost]
        public ActionResult Edit(Rent rent)
        {

            db.Entry(rent).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("ListR");
        }

    }
}
```

Лістинг VisitsController.cs:

```
namespace Library.Controllers.Tables
{
    public class VisitsController : Controller
    {
        // GET: Visits
        public ActionResult Index()
        {
            return View();
        }

        LibCon db = new LibCon();
        public ActionResult ListV()
        {
            var vists = db.Visits.Include(c =>
c.Client).Include(w => w.Worker).Include(l =>
l.Library).ToList();
            return View(vists);
        }


        [HttpGet]
        public ActionResult Create()
        {
            SelectList clients = new
SelectList(db.Clients, "Id", "Name");
```

Лістинг WorkersController.cs:

```
namespace Library.Controllers.Tables
{
    public class WorkersController : Controller
    {

        LibCon db = new LibCon();
        public ActionResult List()
        {
            var wor = db.Workers.Include(r =>
r.Role).Include(p => p.Post).Include(l=>l.Library).ToList();
            SelectList workers = new
SelectList(db.Workers, "Id", "Name");
            SelectList libraries = new
SelectList(db.Libraries, "Id", "Name");
            ViewBag.Clients = clients;
            ViewBag.Workers = workers;
            ViewBag.Libraries = libraries;
            return PartialView();

        }

        [HttpPost]
        public ActionResult Create(Visit visit)
        {
            db.Visits.Add(visit);
            db.SaveChanges();
            return RedirectToAction("ListV");
        }


        [HttpGet]
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Visit visit = db.Visits.Find(id);
            if (visit != null) {

            SelectList clients = new
SelectList(db.Clients, "Id", "Name", visit.ClientId);
            SelectList workers = new
SelectList(db.Workers, "Id", "Name", visit.WorkerId);
            SelectList libraries = new
SelectList(db.Libraries, "Id", "Name",visit.LibraryId );
            ViewBag.Clients = clients;
            ViewBag.Workers = workers;
            ViewBag.Libraries = libraries;

            return PartialView(visit);
            }
            return RedirectToAction("ListV");
        }

        [HttpGet]
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Visit visit = db.Visits.Find(id);
            if (visit != null)
            {
                db.Visits.Remove(visit);
                db.SaveChanges();
            }
            return RedirectToAction("ListV");

        }

        [HttpPost]
        public ActionResult Edit(Visit visit)
        {
            db.Entry(visit).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("ListV");
        }

    }
}
```

```
            return View(wor);
        }
        // GET: Workers
        public ActionResult Index()
        {
            ViewData["SubTitle"] = "Welcome in ASP.NET MVC 5
Library Project ";
            ViewData["Message"] = "It is Open Library.";
            return View();
        }
```

```
        [HttpGet]
        public ActionResult Create()
        {
            SelectList posts = new SelectList(db.Posts, "Id",
"Name");
            SelectList libraries = new
SelectList(db.Libraries, "Id", "Name");
            SelectList roles = new SelectList(db.Roles, "Id",
"Name");
            ViewBag.Libraries = libraries;
            ViewBag.Posts = posts;
            ViewBag.Roles = roles;
            return PartialView();

        }

        [HttpPost]
        public ActionResult Create(Worker worker)
        {
            db.Workers.Add(worker);
            db.SaveChanges();
            return RedirectToAction("List");
        }


        [HttpGet]
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Worker worker = db.Workers.Find(id);
            if (worker != null)
            {
                SelectList posts = new SelectList(db.Posts,
"Id", "Name", worker.PostId);
                SelectList libraries = new
SelectList(db.Libraries, "Id", "Name", worker.LibraryId);
                SelectList roles = new SelectList(db.Roles,
"Id", "Name", worker.RoleId);
                ViewBag.Libraries = libraries;
                ViewBag.Posts = posts;
                ViewBag.Roles = roles;
                return PartialView(worker);
            }
            return RedirectToAction("List");
        }

        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return HttpNotFound();
            }
            Worker worker = db.Workers.Find(id);
            if (worker != null)
            {

                db.Workers.Remove(worker);
                db.SaveChanges();
            }
            return RedirectToAction("List");

        }


        [HttpPost]
        public ActionResult Edit(Worker worker)
        {
            db.Entry(worker).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("List");
        }

    }
}
```

Лістинг AccountController.cs:

```
namespace Library.Controllers
{
    public class AccountController : Controller
    {
        // GET: Account
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult Login()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Login(Worker worker)
        {
            if (ModelState.IsValid)
            {
                // поиск пользователя в бд
                Worker admin = null;
                using (LibCon db = new LibCon())
                {
                    admin = db.Workers.FirstOrDefault(u =>
u.Login == worker.Login && u.Password == worker.Password);

                }
                if (admin != null)
                {

FormsAuthentication.SetAuthCookie(worker.Login, true);
                    return RedirectToAction("Index", "Home");
                }
                else
                {
                    ModelState.AddModelError("", "User not
found");
                }
            }

            return View(worker);
        }

        public ActionResult Register()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Register(Worker worker)
        {
            if (ModelState.IsValid)
            {
                Worker admin = null;
                using (LibCon db = new LibCon())
                {
                    admin = db.Workers.FirstOrDefault(u =>
u.Login == worker.Login);
                }
                if (admin == null)
                {
                    // создаем нового пользователя
                    using (LibCon db = new LibCon())
                    {
                        db.Workers.Add(new Worker { Name =
worker.Name, Email = worker.Email, Password = worker.Password,
Login = worker.Login,Address =
worker.Address,Lname=worker.Lname,Phone=worker.Phone, RoleId =
2,LibraryId =1,PostId=1, });
                        db.SaveChanges();

                        admin = db.Workers.Where(u => u.Login
== worker.Login && u.Password ==
worker.Password).FirstOrDefault();
                    }
                    // если пользователь удачно добавлен в бд
                    if (admin != null)
                    {

FormsAuthentication.SetAuthCookie(worker.Email, true);
                        return RedirectToAction("Login",
"Account");
                    }
                }
                else
                {
                    ModelState.AddModelError("", "User is
already exist");
                }
            }

            return View(worker);
        }
        public ActionResult Logoff()
```

```
        {
            FormsAuthentication.SignOut();
            return RedirectToAction("Index", "Home");
        }
    }
}
```

Лістинг Worker.cs:

```
namespace Library.Models
{
    public class Worker
    {
        public int Id { get; set; }
        [Display(Name = "First Name")]
        public string Name { get; set; }
        [Display(Name = "Second Name")]
        public string Lname { get; set; }

        [Display(Name = "Birth date")]
        public string DataATimeBirth { get; set; }
        [Display(Name = "Hiring date")]
        public string DataATimeHiring { get; set; }
        [Display(Name = "Fired date")]
        public string DataATimeFired { get; set; }
        public string Address { get; set; }
        public string Phone { get; set; }
        public string Login { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }

        [Display(Name = "Library")]
        public int? LibraryId { get; set; }
        public Library Library { get; set; }

        [HiddenInput(DisplayValue = false)]
        public int RoleId { get; set; }
        public Role Role { get; set; }

        [Display(Name = "Post")]
        public int PostId { get; set; }
        public Post Post { get; set; }


    }
}
```

Лістинг Visits.cs:

```
namespace Library.Models
{
    public class Visit
    {
        public int Id { get; set; }

        [Display(Name = "Date")]
        public string Data { get; set; }

        [Display(Name = "Library")]
        public int LibraryId { get; set; }
        public Library Library { get; set; }

        [Display(Name = "Worker")]
        public int WorkerId { get; set; }
        public Worker Worker { get; set; }

        [Display(Name = "Client")]
        public int ClientId { get; set; }
        public Client Client{ get; set; }



    }
}
```

Лістинг Rent.cs:

```
namespace Library.Models
{
    public class Rent
    {
        public int Id { get; set; }

        [Display(Name = "Issue date")]
        public string DataIssue { get; set; }
        [Display(Name = "Return date")]
        public string DataReturn { get; set; }

        [Display(Name = "Library")]
        public int? LibraryId { get; set; }
        public Library Library { get; set; }

        [Display(Name = "Copy")]
        public int CopyId { get; set; }
        public Copy Copy{ get; set; }

        [Display(Name = "Visit date")]
        public int VisitId { get; set; }
        public Visit Visit{ get; set; }

    }
}
```

Лістинг Edition.cs:

```
namespace Library.Models
{
    public class Edition
    {

        public int Id { get; set; }
        public string Name { get; set; }
        public int Price { get; set; }

        [Display(Name = "Author")]
        public int? AuthorId { get; set; }
        public Author Author { get; set; }

        [Display(Name = "Type edition")]
        public int TeditionId { get; set; }
        public Tedition Tedition { get; set; }

        [Display(Name = "Genre")]
        public int? GenreId { get; set; }
        public Genre Genre { get; set; }


    }
}
```

Лістинг Login.cshtml:

```
@model Library.Models.Worker

@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Library | Login</title>
    <link href="~/Content/bootstrap.min.css" rel="stylesheet">
    <link href="~/Fonts/font-awesome/css/font-awesome.css"
rel="stylesheet">
    <link href="~/Content/animate.css" rel="stylesheet">
    <link href="~/Content/style.css" rel="stylesheet">
</head>
<body class="white-bg">
    <div class="middle-box text-center loginscreen  animated
fadeInDown">
        <div>
            <h1 class="logo-name">LIB+</h1>
            <div>
            </div>
            @using (Html.BeginForm("Login", "Account", new {
ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new { @class
= "form-horizontal", role = "form" }))
            {<small>Use a local account to log in.</small>

                @Html.AntiForgeryToken()
                <h4>Welcom in Library</h4>
                <hr />
                @Html.ValidationSummary(true)
```

```
                <form class="m-t" role="form" action="#">
                    <div class="form-group">
                        @Html.LabelFor(m => m.Login, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Login, new
{ @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Login)
                        </div>
                    </div>
                    <div class="form-group">
                        @Html.LabelFor(m => m.Password, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.PasswordFor(m => m.Password,
new { @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Password)
                        </div>
                    </div>
                    <button type="submit" class="btn btn-
primary block full-width m-b">Login</button>

                    <p class="text-muted text-
center"><small>Do not have an account?</small></p>
                    <a class="btn btn-sm btn-white btn-block"
href="@Url.Action("Register", "Account")">Create an
account</a>
                </form>
                <p class="m-t"> <small>Library &copy;
2015</small> </p>
            }
        </div>
    </div>
</body>
</html>


@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Лістинг Register.cshtml:

```
@model Library.Models.Worker

@{
    Layout = null;
}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Library | Register</title>
    <link href="~/Content/bootstrap.min.css" rel="stylesheet">
    <link href="~/Fonts/font-awesome/css/font-awesome.css"
rel="stylesheet">
    <link href="~/Content/animate.css" rel="stylesheet">
    <link href="~/Content/style.css" rel="stylesheet">
</head>
<body class="white-bg">
    <div class="middle-box text-center loginscreen  animated
fadeInDown">
        <div>
            <h1 class="logo-name">LIB+</h1>
            <div>
            </div>
            @using (Html.BeginForm("Register", "Account", new
{ ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
            {

                @Html.AntiForgeryToken()
                <h4>Welcom in Library</h4>
                <hr />
                @Html.ValidationSummary(true)

                <form class="m-t" role="form" action="#">
                    <div class="form-group">
                        @Html.LabelFor(m => m.Name, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Name, new
{ @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Name)
                        </div>
                    </div>
                    <div class="form-group">
                        @Html.LabelFor(m => m.Lname, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Lname, new
{ @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Lname)
                        </div>
                    </div>
                    <div class="form-group">
                        @Html.LabelFor(m => m.Email, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Email, new
{ @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Email)
                        </div>
                    </div>
                    <div class="form-group">
                        @Html.LabelFor(m => m.Address, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Address,
new { @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Address)
                        </div>
                    </div>
                    <div class="form-group">
                        @Html.LabelFor(m => m.Phone, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Phone, new
{ @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Phone)
                        </div>
                    </div>
                    <div class="form-group">
                        @Html.LabelFor(m => m.Login, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Login, new
{ @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Login)
                        </div>
                    </div>
                    <div class="form-group">
                        @Html.LabelFor(m => m.Password, new {
@class = "col-md-3 control-label" })
                        <div class="col-md-9">
                            @Html.TextBoxFor(m => m.Password,
new { @class = "form-control" })
                            @Html.ValidationMessageFor(m =>
m.Password)
                        </div>
                    </div>

                    <button type="submit" class="btn btn-
primary block full-width m-b">Register</button>
                    <p class="text-muted text-
center"><small>Already have an account?</small></p>
                    <a class="btn btn-sm btn-white btn-block"
href="@Url.Action("Login", "Account")">Login</a>
                </form>
                <p class="m-t"> <small>Library &copy;
2015</small> </p>
            }
        </div>
    </div>
</body>
</html>


@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Лістинг Create.cshtml:

```
@model Library.Models.Worker

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="modal-dialog">
        <div class="modal-content animated fadeIn">
            <div class="modal-header">
                <button type="button" class="close" data-
dismiss="modal"><span aria-hidden="true">&times;</span><span
class="sr-only">Close</span></button>
                <h4 class="modal-title">Add workers</h4>
            </div>
            <div class="modal-body">
                <p>
                    <div class="form-horizontal">

                        <small>
                            @Html.ValidationSummary(true, "",
new { @class = "text-danger" })
                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Name, htmlAttributes: new { @class = "control-label col-
md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Name, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Name, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Lname, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Lname, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Lname, "", new {
@class = "text-danger" })
                                </div>
                            </div>
                            <div class="form-group">
                                @Html.LabelFor(model =>
model.LibraryId, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">

@Html.DropDownListFor(model => model.LibraryId,
ViewBag.Libraries as SelectList)

@Html.ValidationMessageFor(model => model.LibraryId, "", new {
@class = "text-danger" })
                                </div>
                            </div>



                            <div class="form-group">

                                @Html.LabelFor(model =>
model.DataATimeBirth, htmlAttributes: new { @class = "control-
label col-md-2" })

                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.DataATimeBirth, new { htmlAttributes = new { @class =
"form-control" } })

@Html.ValidationMessageFor(model => model.DataATimeBirth, "",
new { @class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">

                                @Html.LabelFor(model =>
model.DataATimeHiring, htmlAttributes: new { @class =
"control-label col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.DataATimeHiring, new { htmlAttributes = new { @class =
"form-control" } })

@Html.ValidationMessageFor(model => model.DataATimeHiring, "",
new { @class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.DataATimeFired, htmlAttributes: new { @class = "control-
label col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.DataATimeFired, new { htmlAttributes = new { @class =
"form-control" } })

@Html.ValidationMessageFor(model => model.DataATimeFired, "",
new { @class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Address, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Address, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Address, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Phone, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Phone, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Phone, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Login, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Login, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Login, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Email, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Email, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Email, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Password, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Password, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Password, "", new {
@class = "text-danger" })
                                </div>
                            </div>
```

```
                            <div class="form-group">
                                @Html.LabelFor(model =>
model.RoleId, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">

@Html.DropDownListFor(model => model.RoleId, ViewBag.Roles as
SelectList, new { htmlAttributes = new { @class = "col-sm-2
control-label" } })

@Html.ValidationMessageFor(model => model.RoleId, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.PostId, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">

@Html.DropDownListFor(model => model.PostId, ViewBag.Posts as
SelectList)

@Html.ValidationMessageFor(model => model.PostId, "", new {
@class = "text-danger" })
                                </div>
                            </div>
                        </small>

                        <div class="form-group">
                            <div class="col-md-offset-8 col-
md-10">
                                <button type="button"
class="btn btn-danger" data-dismiss="modal">Close</button>
                                <input type="submit"
value="Create" class=" btn btn-primary" />
                            </div>
                        </div>
                    </div>
                </p>
            </div>
        </div>
    </div>
}
```

Лістинг Edit.cshtml:

```
@model Library.Models.Worker

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="modal-dialog">
        <div class="modal-content animated fadeIn">
            <div class="modal-header">
                <button type="button" class="close" data-
dismiss="modal"><span aria-hidden="true">&times;</span><span
class="sr-only">Close</span></button>
                <h4 class="modal-title">Edit worker</h4>
            </div>
            <div class="modal-body">
                <p>
                    <div class="form-horizontal">
                        @Html.HiddenFor(model => Model.Id)
                        <small>
                            @Html.ValidationSummary(true, "",
new { @class = "text-danger" })
                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Name, htmlAttributes: new { @class = "control-label col-
md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Name, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Name, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Lname, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Lname, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Lname, "", new {
@class = "text-danger" })
                                </div>
                            </div>
                            <div class="form-group">
                                @Html.LabelFor(model =>
model.LibraryId, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">

@Html.DropDownListFor(model => model.LibraryId,
ViewBag.Libraries as SelectList)

@Html.ValidationMessageFor(model => model.LibraryId, "", new {
@class = "text-danger" })
                                </div>
                            </div>


                            <div class="form-group">

                                @Html.LabelFor(model =>
model.DataATimeBirth, htmlAttributes: new { @class = "control-
label col-md-2" })

                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.DataATimeBirth, new { htmlAttributes = new { @class =
"form-control" } })

@Html.ValidationMessageFor(model => model.DataATimeBirth, "",
new { @class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">

                                @Html.LabelFor(model =>
model.DataATimeHiring, htmlAttributes: new { @class =
"control-label col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.DataATimeHiring, new { htmlAttributes = new { @class =
"form-control" } })

@Html.ValidationMessageFor(model => model.DataATimeHiring, "",
new { @class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.DataATimeFired, htmlAttributes: new { @class = "control-
label col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.DataATimeFired, new { htmlAttributes = new { @class =
"form-control" } })

@Html.ValidationMessageFor(model => model.DataATimeFired, "",
new { @class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Address, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Address, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Address, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
```

```
                                @Html.LabelFor(model =>
model.Phone, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Phone, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Phone, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Login, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Login, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Login, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Email, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Email, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Email, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.Password, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">
                                    @Html.EditorFor(model =>
model.Password, new { htmlAttributes = new { @class = "form-
control" } })

@Html.ValidationMessageFor(model => model.Password, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.RoleId, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">

@Html.DropDownListFor(model => model.RoleId, ViewBag.Roles as
SelectList)

@Html.ValidationMessageFor(model => model.RoleId, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                            <div class="form-group">
                                @Html.LabelFor(model =>
model.PostId, htmlAttributes: new { @class = "control-label
col-md-2" })
                                <div class="col-md-10">

@Html.DropDownListFor(model => model.PostId, ViewBag.Posts as
SelectList)

@Html.ValidationMessageFor(model => model.PostId, "", new {
@class = "text-danger" })
                                </div>
                            </div>

                        </small>
                        <div class="form-group">
                            <div class="col-md-offset-8 col-
md-10">
                                <button type="button"
class="btn btn-danger" data-dismiss="modal">Close</button>
                                <input type="submit"
value="Save" class="btn btn-primary" />
                            </div>
                        </div>
                    </div>
                </p>
            </div>
        </div>
    </div>
}
```

Лістинг List.cshtml:

```
@model IEnumerable<Library.Models.Worker>

@{
    ViewBag.Title = "Workers";
}
<script
src="https://ajax.gooleapis.com/ajax/libs/jquery/3.3.1/jquery,
min.js"></script>

<div class="row wrapper border-bottom white-bg page-heading">
    <div class="col-lg-10">
        <h2>Data Workers</h2>
        <ol class="breadcrumb">
            <li>
                <a href="@Url.Action("Index",
"Home")">Home</a>
            </li>
            <li>
                <a>Tables</a>
            </li>
            <li class="active">
                <strong>Data Workers</strong>
            </li>
        </ol>
    </div>
    <div class="col-lg-2">
    </div>
</div>
<div class="wrapper wrapper-content animated fadeInRight">
    <div class="row">
        <div class="col-lg-12">
            <div class="ibox float-e-margins">
                <div class="ibox-title">
                </div>


                <div class="ibox-content">
                    <button type="button" title="create"
class="button btn-block btn btn-primary"
style="display:inline-block;width:125px">Add</button>
                    <div class="modal fade" id="create"
role="dialog" data-url='@Url.Action("Create","Workers")'>
</div>



                    <div style="display:inline-block;
float:right ">
                        <label>Search</label>
                        <input type="text" id="filter">
                    </div>
                    <table class="footable  toggle-arrow-tiny
table   " data-filter="#filter" data-page-size="3">
                        <thead>
                            <tr>
                                <th data-toggle="true">
                                    @Html.DisplayNameFor(model
=> model.Name)
                                </th>
                                <th>
                                    @Html.DisplayNameFor(model
=> model.Lname)
                                </th>
                                <th data-hide="all">
                                    @Html.DisplayNameFor(model
=> model.LibraryId)
                                </th>
                                <th>
                                    @Html.DisplayNameFor(model
=> model.DataATimeBirth)
                                </th>
                                <th>
                                    @Html.DisplayNameFor(model
=> model.DataATimeHiring)
```

```
                                </th>
                                <th>
                                    @Html.DisplayNameFor(model
=> model.DataATimeFired)
                                </th>
                                <th>
                                    @Html.DisplayNameFor(model
=> model.Address)
                                </th>
                                <th>
                                    @Html.DisplayNameFor(model
=> model.Phone)
                                </th>
                                <th data-hide="all">
                                    @Html.DisplayNameFor(model
=> model.Login)
                                </th>
                                <th data-hide="all">
                                    @Html.DisplayNameFor(model
=> model.Email)
                                </th>
                                <th data-hide="all">
                                    @Html.DisplayNameFor(model
=> model.Password)
                                </th>
                                <th data-hide="all">
                                    @Html.DisplayNameFor(model
=> model.RoleId)
                                </th>
                                <th data-hide="all">
                                    @Html.DisplayNameFor(model
=> model.PostId)
                                </th>

                            </tr>
                        </thead>
                        <tbody>
                            @foreach (var item in Model)
                            {
                                <tr class="gradeX">
                                    <td>

@Html.DisplayFor(modelItem => item.Name)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Lname)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Library.Name)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.DataATimeBirth)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.DataATimeHiring)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.DataATimeFired)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Address)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Phone)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Login)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Email)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Password)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Role.Name)
                                    </td>
                                    <td>

@Html.DisplayFor(modelItem => item.Post.Name)
                                    </td>
                                    <td border="0">
                                        <small>
                                            <button
type="button" title="@item.Id" class="button btn btn-primary"
style="width:60px;height:30px">Edit</button>
                                            <div class="modal
fade" id="@item.Id" role="dialog" data-
url='@Url.Action("Edit","Workers",new { id = item.Id })'>
</div>
                                            <button class="btn
btn-danger" style="width:60px;height:30px;text-align:justify"
onclick="location.href='@Url.Action("Delete","Workers" , new {
id = item.Id })';return false;">Delete</button>
                                        </small>
                                    </td>
                                </tr>

                            }
                        </tbody>
                        <tfoot>
                            <tr>
                                <td colspan="8">
                                    <ul class="pagination
pull-right"></ul>
                                </td>
                            </tr>
                        </tfoot>
                    </table>

                </div>
            </div>
        </div>
    </div>
</div>



@section Styles {
    @Styles.Render("~/plugins/footableStyles")

}
@section Scripts {
    @Scripts.Render("~/plugins/footable")

<script type="text/javascript">
    $(document).ready(function () {
        $('.button').click(function () {
            var temp = '#' + this.title;
            var url = $(temp).data('url');
            $.get(url, function (data) {
                $(temp).html(data);
                $(temp).modal('show');
            });
        });
    });


</script>

    <script type="text/javascript">
        $(document).ready(function () {

            $('.footable').footable();
        });
    </script>

}
```

Лістинг _Navigation.cshtml:

```
<nav class="navbar-default navbar-static-side"
role="navigation">
    <div class="sidebar-collapse">
        <ul class="nav" id="side-menu">
            <li class="nav-header">
                <div class="dropdown profile-element">
                    <span>
                        <img alt="image" class="img-circle"
src="~/Images/profile_small.jpg" />
                    </span>
                    <a data-toggle="dropdown" class="dropdown-
toggle" href="#">
                        <span class="clear">
```

```
                            <span class="block m-t-xs">
                                <strong class="font-bold">Oleh
Yanytskyi</strong>
                                <span class="text-muted text-
xs block">Administrator<b class="caret"></b></span>
                            </span>
                        </span>
                    </a>
                    <ul class="dropdown-menu animated
fadeInRight m-t-xs">
                        <li><a href="#">Logout</a></li>
                    </ul>
                </div>
                <div class="logo-element">
                    Lib
                </div>
            </li>
            <li class="@Html.IsSelected(action: "Index")">
                <a href="@Url.Action("Index", "Home")"><i
class="fa fa-laptop"></i> <span class="nav-
label">Dashboards</span> </a>
            </li>
            <li class="@Html.IsSelected(action: "List")">
                <a href="@Url.Action("List", "Workers")"><i
class="fa fa-users"></i> <span class="nav-
label">Workers</span></a>
            </li>
            <li class="@Html.IsSelected(action: "ListC")">
                <a href="@Url.Action("ListC", "Copies")"><i
class="fa fa fa-book"></i> <span class="nav-
label">Copies</span></a>
            </li>
            <li class="@Html.IsSelected(action: "ListV")">
                <a href="@Url.Action("ListV", "Visits")"><i
class="fa fa-chevron-right"></i> <span class="nav-
label">Visits</span></a>
            </li>
            <li class="@Html.IsSelected(action: "ListR")">
                <a href="@Url.Action("ListR", "Rents")"><i
class="fa fa-usd"></i> <span class="nav-
label">Rents</span></a>
            </li>
        </ul>
    </div>
</nav>
```

Лістинг _Layout.cshtml:

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Library | @ViewBag.Title</title>
    <link
href='https://fonts.googleapis.com/css?family=Open+Sans:400,30
0,600,700' rel='stylesheet' type='text/css'>
    <!-- Add local styles, mostly for plugins css file -->
    @if (IsSectionDefined("Styles"))
        {@RenderSection("Styles", required: false)}

    @Styles.Render("~/Content/css")
    @Styles.Render("~/font-awesome/css")
</head>
<body>
    <!-- Wrapper-->
    <div id="wrapper">
        <!-- Navigation -->
        @Html.Partial("_Navigation")
        <!-- Page wraper -->
        <div id="page-wrapper" class="gray-bg">
            <!-- Top Navbar -->
            @Html.Partial("_TopNavbar")
            <!-- Main view  -->
            @RenderBody()
            <!-- Footer -->
            @Html.Partial("_Footer")
        </div>
        <!-- End page wrapper-->
    </div>
    <!-- End wrapper-->
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @Scripts.Render("~/plugins/metsiMenu")
    @Scripts.Render("~/plugins/pace")
    @Scripts.Render("~/plugins/slimScroll")

    @Scripts.Render("~/bundles/inspinia")

    @RenderSection("scripts", required: false)
</body>
</html>
```

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ

Направляється студент Яницький О. О. на захист дипломного проекту (роботи)
(прізвище, ініціали)

за спеціальністю 122 - Комп'ютерні науки

На тему: Інтернет магазин книжок з чат-ботом у телеграм

Дипломний проект (робота), рецензія і довідка про перевірку на плагіат додаються.

Декан факультету _____ САВЕНКО О.С.
(підпис) (прізвище та ініціали)

ДОВІДКА УСПІШНОСТІ

Яницький О. О. за період навчання на факультеті програмування та компютерних і телекомунікаційних систем з 2017 по 2021 роки. повністю виконав навчальний план спеціальності з такими розподілом оцінок за:
національною шкалою: відмінно 46,88 %, добре 37,50 %, задовільно 15,62 %.
шкалою ЄКТС: A 45,45 %, B 14,55 %, C 21,82 %, D 5,45 %, E 12,73 %.

Методист факультету _____ _____
(підпис) (прізвище та ініціали)

ВИСНОВОК КЕРІВНИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ) ТА ОБГРУНТУВАННЯ ОЦІНКИ

Студент Яницький О.О. виконав кваліфікаційну роботу бакалавра згідно поставленого завдання. За своєю структурою, ілюстративним матеріалом, поставлені мені на вирішення завдання роботи відповідає вимогам, що пред'являються до освітньо-кваліфікаційного рівня „бакалавр"

Оцінка дипломного проекту (роботи) „відмінно"

Керівник дипломного прокту (роботи) _____ Багрій Ф.О
(підпис) (прізвище та ініціали)

"09" 06 2021 р.

ВИСНОВОК КАФЕДРИ ПРО ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Дипломний проект (роботу) розглянуто. Студент Яницький О. О. допускається до захисту цього

Завідувач кафедри КНІТ
(назва)

_____ "09" 06 2021 р.
(підпис, прізвище, ініціали)

Tue Jun 08 09:45:00 EEST 2021, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 46.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 9%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 92621 Название: Інтернет-магазин книжок з чат-ботом у “Телеграм” Добавлено в БД: 2021-06-08 Авторы: О.О. Яницький Руководители: Р.О. Багрій Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	41793	409	20622 (49%)	214 (52%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы
90171	Название: ЗВІТ з професійної практики Добавлено в БД: 2021-05-11 Авторы: Яницький О.О. Руководители: Скрипник Т.К. Консультанты: Опоненты:	19152 (46.0%)	203 (50.0%)

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1008224515

Дата перевірки:
08.06.2021 10:49:00 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2021 10:58:04 EEST

ID користувача:
100005671

Назва документа: **Bachelor_Yanytskyi Lite**

Кількість сторінок: **47** Кількість слів: **6652** Кількість символів: **48336** Розмір файлу: **3.72 MB** ID файлу: **1008298363**

6.09%
Схожість

Найбільша схожість: **2.56%** з джерелом з Бібліотеки (ID файлу: **1005450704**)

4.83% Джерела з Інтернету	146	Сторінка 49
2.92% Джерела з Бібліотеки	41	Сторінка 50

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	1

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Інтернет-магазин книжок з чат-ботом у "Телеграм"*

Автор: *студент 4 курсу, група КН-17-1, Яницький О.О.*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *к.т.н., доцент кафедри КНІТ, Багрій Р.О.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні;
3) до запозичень входять фрагменти програмного коду, що на мають авторства і містять поширені конструкції;
4) серед запозичень знаходяться загальновідомі терміни, скорочення та визначення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 6.09% і адресується до першоджерел, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Р. О. Багрій

Гарант ОП _____ О. В. Мазурець

Завідувач кафедри КНІТ _____ О. В. Бармак

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *групи КН-17-1 Яницький Олег Олегович*

за темою *«Інтернет-магазин книжок з чат-ботом у "Телеграм».*

1. Актуальність і значення теми: *в наш час пандемії по всьому світі, все більше онлайн користувачів, які мають вільний час, також більше людей почали користуватися інтернет магазинами та замовляти товари онлайн, а покупка книжки є чудовим та корисним способом проведення часу. Для більш прогресивних користувачів наявність бота в такому месенджері як «Телеграм» є досить важливою, так як більшість покупців проводять час за смартфоном, а не комп'ютером.*

2. Оцінка запропонованих моделей, підходів, алгоритмів, інформаційної складової та засобів розробки: *використання такого шаблона, як MVC допомагає проєкту створити розширювану архітектуру яку легко тестувати, крім того робота сайту та бота не залежить один від одного, саме тому навіть, якщо один з них перестане працювати це не вплине на роботу іншого.*

3. Оцінка розробленої інформаційної системи, її практична цінність та економічна доцільність: *розроблений інтернет магазин, а також бот призначений для власників книжкових магазинів які хочуть розвинути свій бізнес, і для покупців які хочуть зекономити свій час.*

4. Загальний висновок: *вимоги поставленої задачі виконані в повному обсязі, інтернет-магазин книжок та чат-бот працює вірно.*

Робота заслуговує на оцінку «*відмінно*»

Рецензент *к. ф.-м. н. доц. Ярещук П. О.* *08.06.2021р.*