ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка веб-застосунку книжкового каталогу з застосуванням технології WPF»

(тема кваліфікаційної роботи українською мовою)

«Development of a book catalog web application using WPF technology»

(тема кваліфікаційної роботи англійською мовою)

Виконав: здобувач заочної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Петров Андрій Андрійович

(прізвище, ім'я, по-батькові здобувача)

Керівник к.т.н., доцент Гнатовська Г.А.

(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент к.т.н., доцент Волощук Л.А.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Захищено на засіданні ЕК № 13 .
протокол № 5 від 19 червня 2024 р.

Оцінка добре / С / 75 .
(за національною шкалою/шкалою ECTS/ бали)

Завідувачка кафедри

(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Голова ЕК

(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Хостинг – послуга з надання простору для розміщення сайтів в Інтернеті

Apache – вільний веб-сервер

CSS– Cascading Style Sheets – каскадні таблиці стилів

HTML– HyperText Markup Language – мова гіпертекстової розмітки

Microsoft .NET Framework – модель програмування з керованим кодом

MSSQL – система управління базами даних (СУБД)

XAML – Extensible Application Markup Language – мова розмітки, яка використовується для створення екземплярів об'єктів .NET

WPF– Windows Presentation Foundation – технологія створення інтерфейсів користувача

ВСТУП

Одним із важливих завдань завжди було передавання досвіду наступним поколінням, шляхом збереження інформації в часі та просторі. У сучасному суспільстві ключовим ресурсом, разом з матеріальними та енергетичними, є інформація, яка може приймати форму архівів, документів, музеїв та інших носіїв.

З часом обсяги інформації, що зберігаються у традиційній формі, постійно збільшувалися, а робота з ними (зберігання, розповсюдження, пошук, аналіз і так далі) стає все складнішою. Розвиток обчислювальної техніки дозволив зберігати і розповсюджувати інформацію у електронній формі, що відіграє революційну роль у історії, порівнювану з винаходом книгодрукування. Електронний формат на сьогоднішній день дозволяє зберігати книги надійно і компактно, розповсюджувати їх оперативніше і ширше, а також надає можливості маніпулювання, яких не можна було досягти за інших умов. У зв'язку з цим останнім часом кількість електронних публікацій інтенсивно зростає у всьому світі. Велика кількість різноманітних документів вже існує у електронній формі. Однією з ключових запорук розвитку будь-якого суспільства – є повний доступ кожному громадянину до необхідної інформації та культурних цінностей світу. Інвестування у розвиток інформаційних технологій також сприяє розвитку освіти, науки, національної свідомості та національної безпеки, зокрема створенню бібліотек для університетів та іншій сфері [1].

Останнім часом у суспільстві з'явилося нове розуміння значення бібліотеки. У зв'язку з переходом до інформаційного суспільства, бібліотеки розглядаються як активні учасники інформаційних процесів та постачальники інформаційних ресурсів. Це пов'язано з тим, що до традиційних ресурсів бібліотек додалися аудіо- та відеоматеріали, а також електронні ресурси. Глобальна мережа Інтернет взяла на себе більшість функцій бібліотек та стала основним джерелом інформації, яка має високий

пріоритет у суспільстві. Неперервний процес модернізації суспільства призвів до того, що онлайн-бібліотеки стали більш доступними, ніж традиційні бібліотеки у класичному розумінні [1].

Різноманітні інституціональні структури в мережі Інтернет створюють власні моделі інформаційного простору за допомогою веб-сайтів, формуючи конкретні інформаційно-економічні середовища з використанням інтелектуалізованого програмного забезпечення, такого як мультиагентні системи або програмні агенти, що виступають в ролі представників суб'єктів економічної діяльності, в глобальному електронному середовищі. Також використовується об'єктно-орієнтоване програмне забезпечення, таке як веб-сайти, портали, електронні поштові скриньки і т. д.

Веб-застосунок книжкового каталогу призначений для продажу книжкової продукції засобами мережі Інтернет. Зараз подібні веб-ресурси дуже популярні серед людей, які цінують читання, прагнуть до саморозвитку або хочуть поглибити свої знання з цікавих для них тем.

Тому, головною метою розробки веб-застосунку є створення зручного каталогу книжкової продукції з можливістю пошуку, легкої навігації по асортименту та швидкого оформлення замовлень. Реалізація цієї задачі передбачає автоматизацію процесів і забезпечення структурованого зберігання, обробки та пошуку даних з використанням сучасної технології WPF. Застосування веб-застосунку книжкового каталогу з застосуванням технології забезпечить присутність книгорозповсюджувальної компанії в мережі Інтернет та надання інформації про її товари якомога більшому числу потенційних партнерів і споживачів.

Дана кваліфікаційна робота бакалавра містить 71 сторінке, 9 таблиць, 28 рисунків, 12 посилань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предмета, цілей і особливостей веб-застосунку книжкового каталогу

Каталогом (від грецької katalogos) називається перелік якихось предметів (книг, експонатів, товарів), складений у певному порядку. Книжковим каталогом називається упорядкований перелік творів друку, наявних у фонді букіністичної компанії або декількох бібліотек, що розкриває зміст або порядок утримання бібліотечних фондів (алфавітний, систематичний, предметний тощо).

Книжкові каталоги, як будь-яка інформаційно-пошукова система, виконує знаково-комунікативну, фондовідображаючу, інформаційно-пошукову, пізнавальну і педагогічну функції. Знаково-комунікативна функція книжкового каталогу допомагає дистанційному спілкуванню у зв’язці «споживач – каталог – автор». Сутність фондовідображаючої функції полягає в розкритті складу та змісту бібліотечних фондів. Каталогізатор є інтерпретатором інформації, що міститься в документі, та володіє методами створення книжкового каталогу, за допомогою якого здійснюється дистанційне спілкування розділених в просторі, або в часі, автора і читача.

Інформаційно-пошукова функція реалізується в процесі пошуку документів у відповідь на запит читача. Пізнавальна функція ґрунтується на пізнавальних можливостях інформаційно-пошукового масиву. Водночас книжкові каталоги виконують і педагогічну функцію, сприяючи інтелектуальному та професійному розвитку особистості читача.

Останніми роками електронна комерція стрімко розвивається. Незважаючи на усі перешкоди, такі як високі податкові ставки та недосконалі закони, ситуація постійно покращується. Сьогодні все більше торговельних підприємств акцентують увагу на професіоналізмі своїх працівників та впроваджують нові технології для розвитку електронної

комерції. Торгівля надає найсприятливішу підґрунтя для впровадження нових комп'ютерних технологій. Багато завдань, що виникають у торгових підприємств, легко автоматизуються. Швидка та безперервна обробка інформації є однією з основних задач будь-якої великої інформаційної системи. Крім того, необхідно забезпечити простий доступ до інформації, зручність користування та зрозумілість для кінцевого користувача, який часто не має великого досвіду роботи з комп'ютерами та технікою. Всі ці фактори сприяли появі великої кількості інформаційних технологій та інструментів для їх створення [2].

У галузі книжкової справи основні тенденції досліджень визначив сучасний соціокультурний процес. Кожен з цих напрямів спрямований на вивчення взаємодії між книгою та споживачем, їхніх взаємозв'язків і взаємозалежностей. Ключовим питанням для видавництв є задоволення та вивчення потреб споживачів, книготорговельних площадок і бібліотек, оскільки воно впливає на комфортність та повноцінність обслуговування споживачів [3].

Однією з закономірностей формування видавничих портфелів, книготорговельних та бібліотечних колекцій є їхня відповідність потребам суспільства. У епоху інформаційного суспільства і соціально-економічних змін дослідження, аналіз та виявлення тенденцій розвитку книжкової справи, включаючи книгорозповсюдження, стає важливим і необхідним, оскільки вони є одними з найважливіших факторів цього процесу.

1.2 Книгорозповсюдження в структурі книжкової справи

Книготорговельні підприємства, порівняно з видавництвами та бібліотеками, виникли у відповідь на потребу суспільства у подоланні просторово-часових обмежень усної комунікації. Вони стали посередниками, що сприяють реалізації цієї функції. Видавництва допомагають подолати перешкоди на шляху передачі інформації від автора до читача [1].

Книжкова торгівля виконує соціально-необхідну функцію, доводячи книжкову продукцію до споживача. Головна мета її діяльності полягає в подоланні просторових обмежень у розповсюдженні авторської інформації, оскільки книготорговельні підприємства передбачають короткотермінове зберігання товару.

У поширенні авторської інформації для подолання часових обмежень існують як публічні, так і приватні бібліотеки, які створюють специфічний комунікативний простір. Це допомагає читачам засвоювати інформацію через консультації бібліотекарів, можливість використання довідково-бібліографічного апарату, а також участь у заходах, які сприяють спілкуванню з іншими читачами, наприклад, на читацьких конференціях.

Збільшення обсягу написаного матеріалу та його надрукованого варіанту викликало необхідність виникнення бібліографічної діяльності, яка сприяє розповсюдженню інформації про видання. Сутність книжкової торгівлі полягає у її місці та ролі, яку вона відіграє в розширеному відтворенні книжкової продукції. Книжкова торгівля є одним зі способів обігу нещодавно опублікованих видань, і саме вона здійснює цей процес. Обіг є фазою відтворення. Під час торговельного процесу відбувається обмін товарної книжкової продукції на гроші. Цей обмін поділяється на дві самостійні операції: купівлю та продаж [3].

Книготорговельні підприємства зазвичай спочатку закуповують товарну продукцію у виробників або посередників, а потім реалізовують її споживачам. Таким чином, книжкова торгівля виступає посередником між виробниками, такими як видавництва, видавничі організації, поліграфічні підприємства, та кінцевими споживачами (населенням та бібліотеками) книжкової продукції.

Україна спостерігає поступове збільшення кількості інтернет-книгарень, які зазвичай формуються як самостійні підприємства або розширюють діяльність традиційних книгарень чи видавництв.

Вплив науково-технічних факторів відчутний у загальносвітовому розвитку електронних засобів комунікації, що також має вплив на книжкову торгівлю. З одного боку, розвиток сучасних технологій дозволяє книготорговельним підприємствам використовувати прогресивні методи продажу та формувати асортимент книгарень за допомогою Інтернету. З іншого боку, науково-технічний прогрес призводить до зростання конкуренції в галузі книгопоширення через появу книжкових інтернет-магазинів та конкуруючих товарів, таких як електронні видання, аудіокниги та електронні книги [1].

У сучасних умовах неперіодичні видання продаються населенню через різноманітні торгові точки: традиційні магазини (незалежні або мережеві), магазини-кав'ярні, супермаркети, кіоски, відділи реалізації видавництв, книжкові відділи, виставки-ярмарки, інтернет-магазини, книжкові клуби і т.д. Дослідження показують, що понад 50% респондентів віддають перевагу покупці книг в магазинах. Тому проблема відповідності розміщення книжкової торгівельної мережі та її доступності для споживачів стає надзвичайно актуальною.

1.3 Огляд та аналіз існуючих веб-застосунків

Для визначення вимог, функцій та бізнес-логіки розроблюваного веб-застосунку книжкового каталогу з застосуванням технології WPF проведено аналіз існуючих аналогічних систем, що функціонують у мережі Інтернет. Під час аналізу були виявлені основні переваги та недоліки таких систем.

З великої кількості електронних каталогів книг, функціонуючих у мережі Інтернет були обрані ті, що найбільше відповідають потребам розроблюваного веб-застосунку книжкового каталогу. Першим серед них було розглянуто відомий Інтернет-магазин книг YAKABOO, головна сторінка якого наведена на рисунку 1.1.

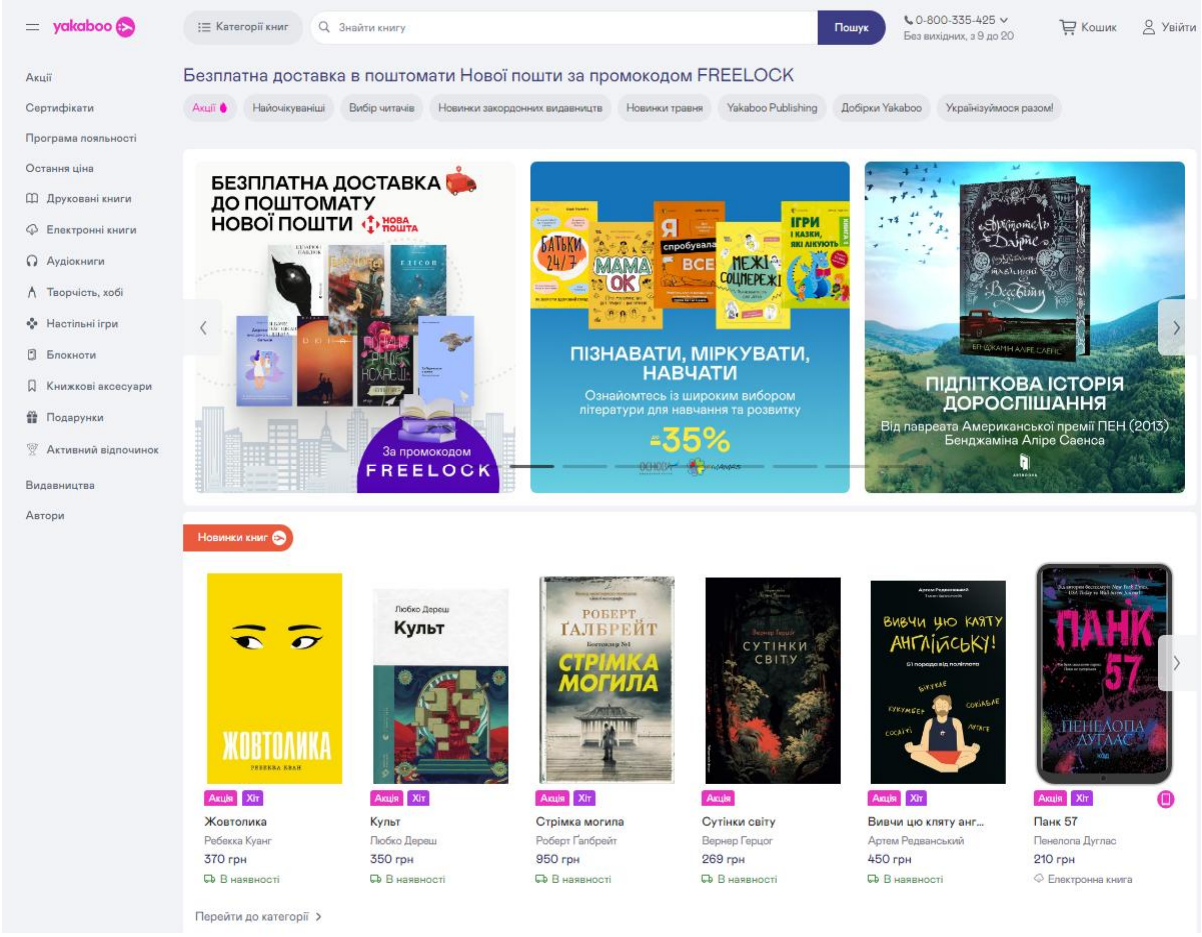

Рисунок 1.1 – Головна сторінка книжкового Інтернет-магазину «YAKABOO»

Yakaboo — пропонує не лише паперові, а й електронні та аудіокниги, налічує понад 300 тисяч видань. Привертає увагу красивий і зручний інтерфейс, реалізація основних функцій. Недолік в перевантаженості середній частині сторінки додатковою інформацією. Було б зручніше додаткову інформацію відображати на вимогу. Крім того, реалізація додаткових функцій (детальний перегляд, корзина) передбачає завантаження нової сторінки, що істотно сповільнює роботу. Перевагою даної системи є: можливість реєстрації для спрощення замовлення в майбутньому; контактний телефон для зворотного зв'язку доступний з будь-якої сторінки сайту; існує пошук літератури за допомогою каталогу.

Недоліками є: немає докладного опису товару; відсутня розширена інформація про книгу та її авторів; немає змісту книги, а це дуже важливо,

щоб зацікавити май бут-нього покупця; не передбачені фільтри для пошуку товарів; пошук товару здійснюється у системі за автором, чи за назвою книги, або за кодом ISBN. Але пошук за всіма переліченими критеріями здійснюється при умові надання повного найменування.

Наступною розглянута інформаційна система книжкового Інтернет-магазину «Наш формат» (Nashformat.ua) (рис. 1.2).

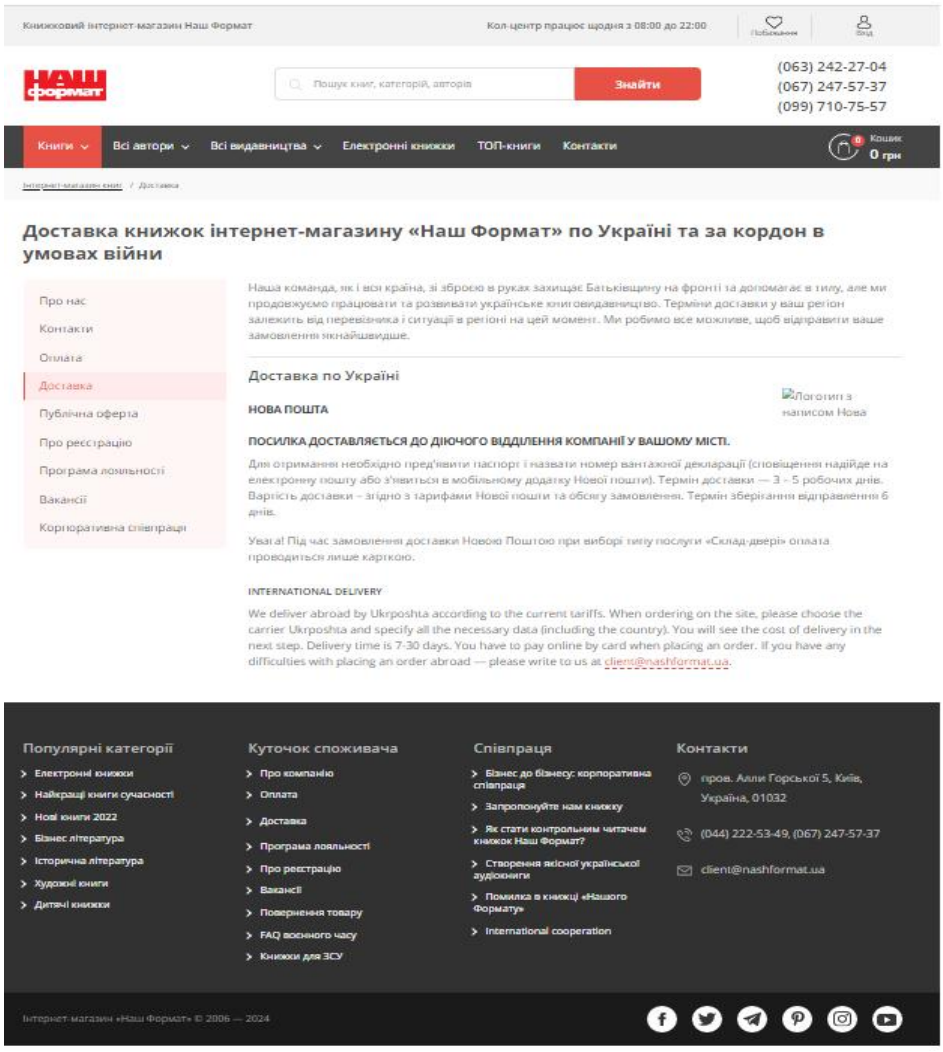

Рисунок 1.2 – Головна сторінка книжкового магазину «Наш формат»

У порівнянні з попередньою системою середня частина перевантажена неструктурованою інформацією. У цій системі відчувається брак попередньої інформації, наприклад, ціни книги. Відсутні пошук за критеріями: автор,

назва, тощо. При здійсненні пошуку по першим літерам, система повертає інформацію про видання, як у назві так і у прізвищі авторів, що ускладнює пошук. Переваги: можливість приймати електронні платежі. Недоліки: не дотримуються принципи usability або зручність використання або практичність; некоректний адаптивний дизайн; сайт може бути трохи повільним у відгуках.

Далі було розглянуто Інтернет-магазин книг Book Shop (book-shop.com.ua), який пропонує широкий вибір літератури для різноманітних уподобань та інтересів (рис. 1.3) .

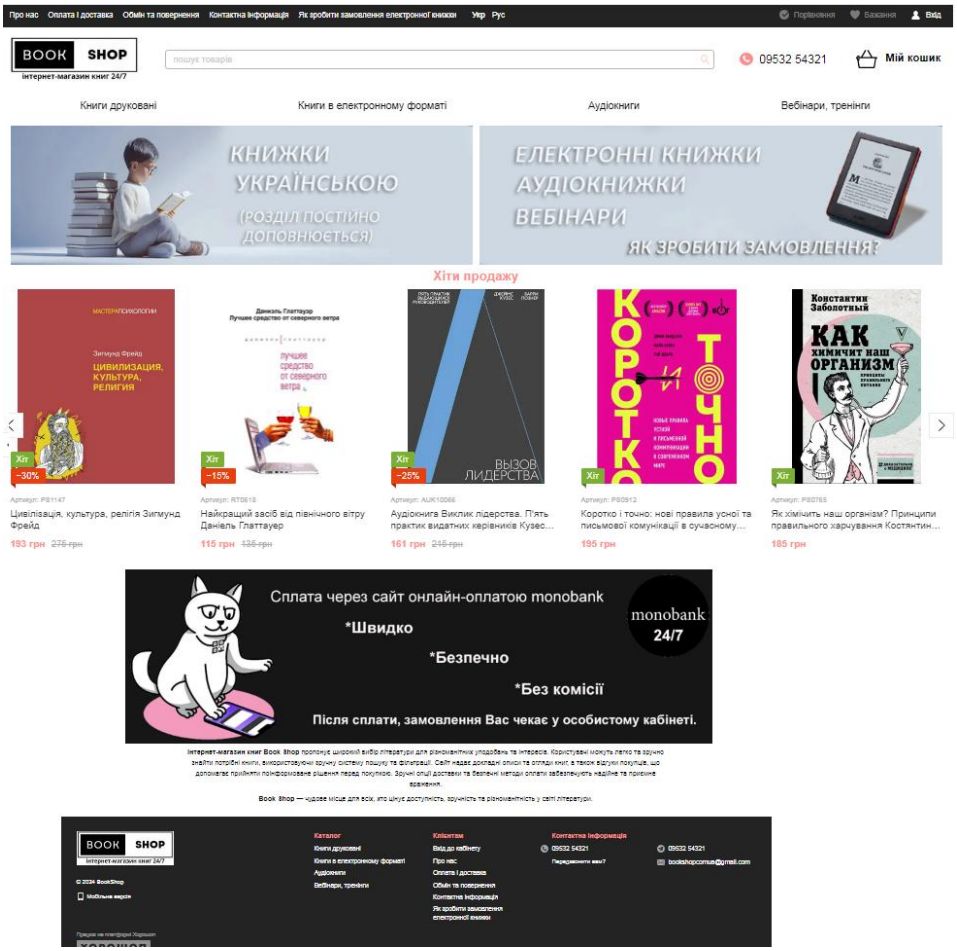

Рисунок 1.3 – Головна сторінка книжкового інтернет-магазину «Book Shop»

У книжковому магазині Book Shop користувачі можуть легко та зручно знайти потрібні книги, використовуючи систему пошуку та фільтрації. Сайт

надає лише стислі описи та огляди книг, передбачено відгуки покупців, що допомагає прийняти поінформоване рішення перед покупкою, є вибір варантів доставки та оплати. До недоліків можливо віднести наступне: немає докладного опису товару; для отримання знижок необхідно здійснити регистрацію; не дотримуються принципи usability або зручність використання або практичність при оформленні заказу у кошику.

Наступний приклад, головна сторінка якого на рисунку 1.4 – це система книжкового каталогу «КнигоЛенд» (knigoland.com.ua) .

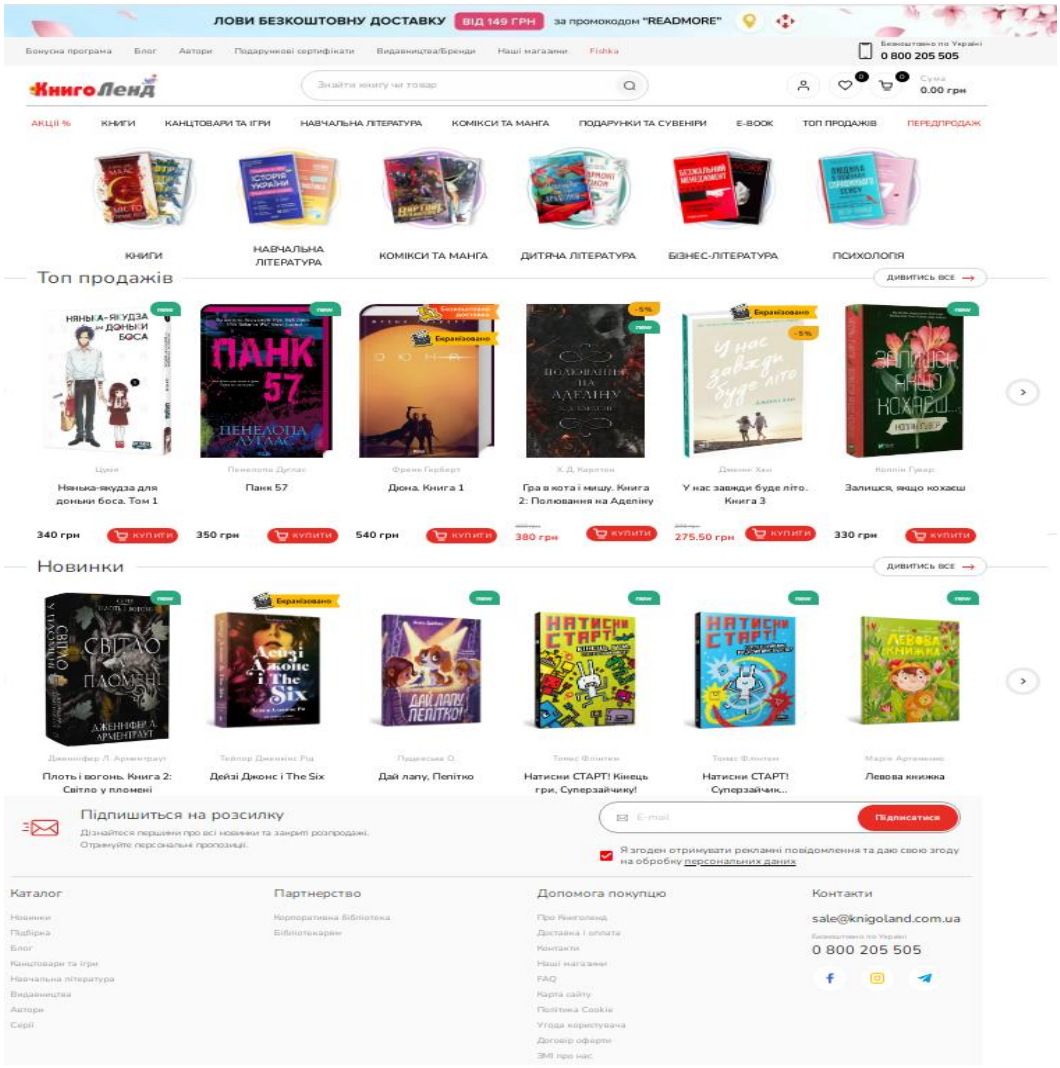

Рисунок 1.4 – Головна сторінка Ітернет-магазину «КнигоЛенд»

Ця система книжкового Інтернет-магазину каталогу є прикладом системи де користувачеві надається вся необхідна контактна інформація; безліч інформаційних додаткових послуг для клієнтів; надано достатньо широкий ассортимент; доставка и оплата товарів здійснюється лише за допомогою служби «Нова пошта»; адаптивний web-дизайн, який забезпечує відмінне відображення сайту на різних пристроях. Серед недоліків можливо зазначити наступне: при перегляді каталогу товарів відсутня можливість додаванням певного товару у кошик; відсутня фільтрація по розділах. Також недоліком є те, що використовується Flash-додатки та реклама, яка впливає на швидкість завантаження сторінок та нерозвинений інтерфейс формування пошукових запитів.

У результаті проведеного аналізу існуючих систем, з множини електронних реалізацій книжкових каталогів перевага віддається тій системі, яка найбільшою мірою задовольняє потреби користувача. При сучасній конкуренції і в Інтернет просторі фірм, які реалізують книжкову продукцію, необхідно при розробці веб-застосунку книжкового каталогу враховувати: зручність для користувача інтерфейсу, різноманітність надання послуг, обов'язково приділити увагу наявності зручних інформаційно-пошукових механізмів.

1.4 Постановка завдання

Завданням дипломної роботи є розробка веб-застосунку книжкового каталогу з застосуванням технології WPF, яку необхідно реалізувати у вигляді мережевої інформаційної системи комерційного підприємства, для публікації її в мережі Інтернет. Основним призначенням веб-застосунку книжкового каталогу з застосуванням технології WPF букіністичної фірми є створення офіційного представництва такої компанії в мережі Інтернет. Web-система створюється для всіх бажаючих отримати інформацію про комерційній пропозиції букіністичної фірми: ознайомитися з каталогом

продукції, здійснити пошук необхідного товару, отримати вичерпну та достовірну інформацію про пропонований товар, а так само дозволяє купити вибраний товар.

Метою створення веб-застосунку книжкового каталогу з застосуванням технології WPF є забезпечення інформаційної присутності компанії в мережі Інтернет та надання інформації про товари і послуги фірми якомога більшому числу потенційних партнерів і споживачів. При розробці веб-застосунку книжкового каталогу з застосуванням технології WPF визначені наступні користувачі: Користувач-Адміністратор; Користувач-Клієнт.

Користувачу-адміністратору повинні пропонуватися наступні функціональні можливості представлені на рисунку 1.5.

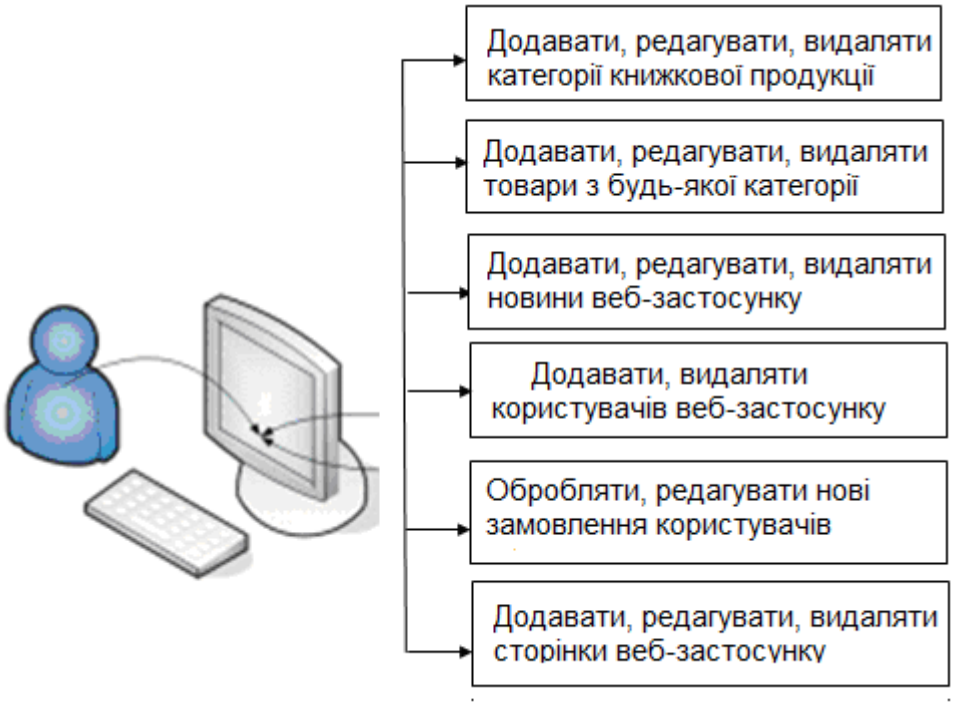

Рисунок 1.5 – Функціональні можливості Користувача-Адміністратора

Веб-застосунок книжкового каталогу з застосуванням технології WPF передбачає реалізацію наступних завдань для Користувача-Клієнта: перегляд каталогу книг; можливість вибірки книг за напрямком (жанром); можливості

вибору книг по розділу; можливість пошуку необхідної книги; можливість перегляду додаткової інформації по обраній книзі; можливість формувати купівельний кошик з необхідної кількістю обраного товару; можливість редагувати обрані товари у кошику; можливість сформувати замовлення на покупку; можливість підтвердити свій вибір і задати необхідні дані для формування замовлення і здійснення оплати у зручний спосіб (рис. 1.6).

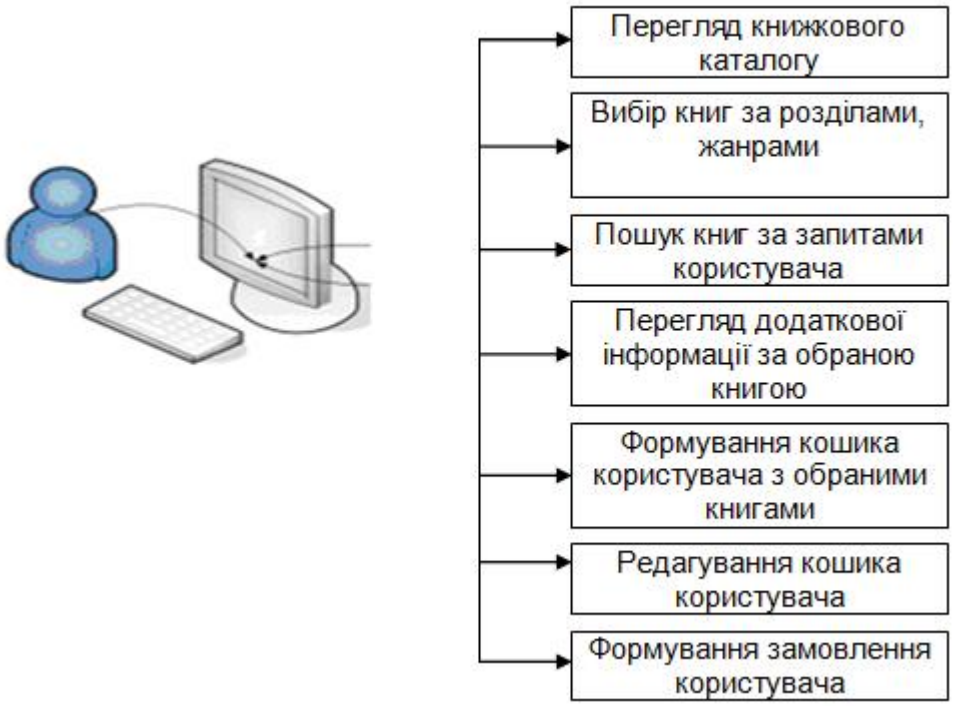

Рисунок 1.6 – Функціональні можливості Користувача-Клієнта

Дані замовлення Користувача-Клієнта повинні бути поміщені в текстовий файл, на підставі якого формується електронний лист, що містить платіжний документ. Подальшу взаємодію фірми, що реалізує книги, і покупця здійснюється засобами електронної пошти.

Веб-система книжкового каталогу з застосуванням технології WPF буде реалізовано у вигляді web-застосунку, доступного для браузерів, що підтримують Framework не нижче версії 4.1. Для редагування кошика користувача (покупця) необхідно створити додаткову Windows-форму. Необхідно організувати можливість паралельної роботи з обома формами,

тобто реалізувати можливість вибору книг в каталозі та забезпечити синхронне відображення вибору у купівельному кошику. Крім того, при роботі з кошиком повинен бути доступний каталог. За рахунок можливостей технології WPF передбачається реалізація зручного і привабливого для користувача інтерфейсу. Маніпуляція даними, що зберігаються в базі даних, повинна здійснюватися засобами технології ADO.NET.

2 ВИБІР АРХІТЕКТУРИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Основні принципи використання технології WPF

Windows Presentation Foundation (WPF) – це платформа для розробки клієнтських додатків для ОС Windows з візуально привабливим інтерфейсом користувача. Вона дозволяє створювати як автономні, так і веб-засновані застосунки. WPF базується на векторній системі відтворення, яка не залежить від роздільної здатності і призначена для сучасного графічного обладнання. Платформа розширює стандартні можливості розробки програм, включаючи мову розмітки XAML (Extensible Application Markup Language), компоненти керування, прив'язки даних, макет, двовимірну та тривимірну графіку, анімацію, стилі, шаблони, обробку документів, мультимедіа, текстові можливості та оформлення. WPF є частиною Microsoft .NET Framework і дозволяє створювати програми, що використовують інші компоненти бібліотеки класів .NET Framework [4].

Відразу після своєї появи платформа Microsoft .NET породила велику кількість нових технологій. Це був новий спосіб написання Web-додатків (ASP.NET), абсолютно новий спосіб підключення до баз даних (ADO.NET), нові мови програмування з безпекою щодо типів (зокрема, C # і VB.NET) і кероване виконуюе середовище (CLR). Не менш важної серед цих нововведень була Windows Forms – бібліотека класів, необхідних для створення Windows-додатків. Незважаючи на те, що Windows Forms є зрілим і повнофункціональним інструментальним засобом, вона жорстко пов'язана з основними конструктивними особливостями Windows, що не змінювалися протягом останніх десяти років. Більше того, Windows Forms ґрунтується на інтерфейсі Windows API при створенні зовнішнього виду стандартних елементів інтерфейсу користувача, та тих як кнопки, текстові вікна, прапорці і т. п. Як результат, ці складові фактично не піддаються налаштуванню. Все

змінилося завдяки новій моделі створення інтерфейсів користувачів, яку пропонує нова технологія створення інтерфейсів користувача Windows Presentation Foundation (WPF).

Незважаючи на включення стандартних елементів керування, WPF самостійно відтворює кожен текст, рамку і фон. Це дозволяє WPF надати значно більше потужних можливостей, що можуть змінити будь-який елемент вмісту, який відображається на екрані. Шляхом використання цих можливостей можна модифікувати стиль звичайних елементів керування, таких як кнопки, часто необхідно переписувати код. Крім того, можна використовувати об'єкти трансформації для обертання, розтягування, зміни масштабу та спотворення будь-якого елементу інтерфейсу користувача. Вбудовану систему анімації в WPF можна використовувати для здійснення цих дій у динаміці. Оскільки механізм WPF візуалізує вміст вікна як частину однієї операції, він здатний обробляти необмежену кількість шарів перекриття елементів керування, навіть якщо вони мають нестандартні форми або часткову прозорість. Основою для цих нових можливостей WPF є потужна нова інфраструктура, що базується на DirectX API - інтерфейсі апаратного прискорення графіки, який зазвичай застосовується в сучасних комп'ютерних іграх. Це означає, що можна застосовувати багаті графічні ефекти без втрати продуктивності, як це може бути у випадку з Windows Forms [4].

WPF покращує функції, які будуть представляти інтерес для розробників бізнес-додатків, включаючи істотно поліпшену модель прив'язки даних, новий набір класів для друку вмісту і управлення чергою друку, а також можливість використання потокових документів для відображення великих обсягів форматованого тексту. WPF надає нову модель для створення сторінкових додатків, що виконуються в браузері і які можуть запускаються з Web-сайту.

WPF існує як підмножини типів .NET Framework, які займають велику частину в просторі імен System.Windows. Користувачі, які раніше

створювали програми за допомогою .NET Framework, використовуючи такі керовані технології, як ASP.NET і Windows Forms, знайомі з основами програмування WPF; створення примірників класів, завдання властивостей, виклик методів і обробка подій здійснюється за допомогою одного з добре знайомих мов програмування .NET Framework, таких як C # або Visual Basic.

У WPF додатково вдосконалюється процес програмування для розробки клієнтських додатків Windows. Одним очевидним удосконаленням є можливість розробляти програми за допомогою розмітки та коду програмної частини.

Розмітка мовою XML (Extensible Application Markup Language) зазвичай використовується для реалізації зовнішнього вигляду додатка при реалізації його поведінки за допомогою керованих мов програмування (коду програмної частини).

Мова XAML – це заснована на XML мова розмітки, яка використовується для декларативної реалізації зовнішнього вигляду програми. Зазвичай вона використовується для створення вікон, діалогових вікон, сторінок користувача елементів управління, а також для їх заповнення елементами управління, фігурами і графікою.

Додаток в основному призначено для реалізації функціональних можливостей, які відповідають на взаємодії з користувачем, включаючи обробку подій (наприклад, натискання меню, панелі інструментів або кнопки) і виклик бізнес-логіки і логіки доступу до даних у відповідь на події. У WPF така поведінка зазвичай реалізується в коді, який пов'язаний з розміткою. Цей тип коду називається кодом програмної частини.

.NET Framework, System.Windows, розмітка і виділений код складають основу розробки додатків WPF. Крім того, WPF надає повний набір засобів для створення зручних і багатофункціональних елементів інтерфейсу користувача. Щоб упакувати розроблені елементи і надати їх користувачеві у вигляді додатків, WPF надає типи і служби, які називаються моделлю

додатків. Модель додатку підтримує розробку як автономних, так і розміщених в браузері додатків [3].

Для додатків, що розміщуються в браузері, також званих XAML-додатки веб-браузера, можна створювати сторінки (Page) і сторінкові функції (PageFunction), за якими можна переходити за допомогою гіперпосилань (класи Hyperlink). WPF комбінує краще зі світу Windows-розробки з новітніми технологіями для побудови сучасних, графічно розвинених користувальницьких інтерфейсів.

2.2 Визначення архітектури WPF

WPF представляє собою графічну систему, що ґрунтується на керованому коді. Ця технологія надбудовує над DirectX і забезпечує програмам, які створені з її використанням, розширені можливості для візуалізації. Вона об'єднує в собі три сфери: інтерфейс користувача (UI – user interface) для настільних і веб-додатків, а також UI для ігор і мультимедіа-додатків. WPF включає в себе різноманітні елементи управління, підтримку відео, анімації, тривимірних зображень тощо. На підґрунті нових можливостей WPF лежить потужна інфраструктура, побудована на базі DirectX – API для апаратно-прискореної графіки, що зазвичай використовується в сучасних комп'ютерних іграх. Це означає, що можна використовувати потужні графічні ефекти без втрат продуктивності, як це було б при використанні Windows Forms. Більше того, можна навіть отримати розширені можливості, такі як підтримка відеофайлів і тривимірних об'єктів. За допомогою цих можливостей (а також інструментів для їх розробки) можна створювати різноманітні інтерфейси і візуальні ефекти, чого неможливо досягти за допомогою Windows Forms. На рисунку 2.1 наведені складові архітектурного рішення технології WPF [4].

Рисунок 2.1 – Архітектура WPF

API WPF – це managed Фреймворк, тобто він надає програмний доступ до своїх можливостей через managed код Microsoft.NET. Треба відзначити, що WPF – це не тільки діалоги, картинки та відео. Крім іншого, WPF включає в себе також синтез і розпізнавання мови. WPF забезпечує інтерфейс користувача високого рівня і надає наступні можливості [4]:

– веб-подібну модель компоновки, яка забезпечує розміщення і впорядкування елементів управління по їх вмісту;
– багатофункціональну модель малювання на базі графічних приміти-вів (базових форм, текстових блоків, графічних інгредієнтів);
– модель з форматованим текстом, яка забезпечує відображення форматованого стилізованого тексту в будь-якій частині користу-вальницького інтерфейсу, комбінування тексту зі списками, малюн-ками та іншими інтерфейсними елементами;
– завдання анімації за допомогою декларативних дескрипторів;

- підтримка аудіовізуальної середовища для програвання будь-яких аудіо- та відеофайлів;
- стилі і шаблони, які дозволяють стандартизувати форматування і керування візуалізацією елементів управління, а також повторно використовувати ці рішення в різних місцях проекту;
- команди, які дозволяють визначати їх в одному місці і багаторазово пов'язувати з різними елементами управління у додатку;
- декларативний користувальницький інтерфейс, який дозволяє описувати вміст вікон або сторінок за допомогою мови XAML.

WPF покращує функції, які будуть представляти інтерес для розробників бізнес-додатків, включаючи істотно поліпшену модель прив'язки даних, новий набір класів для друку вмісту і управління чергою друку, а також можливість використання документів для відображення великих обсягів форматованого тексту. Нарешті, WPF комбінує кращі якості зі старого світу розробки додатків для Windows і нові інноваційні технології для створення сучасних, насичених якісною графікою користувальницьких інтерфейсів.

На зовнішньому рівні WPF представлена об'єктними моделями, що відповідають за користувальницький інтерфейс, маніпуляцію документами і відображення графіки. Вони, в свою чергу, базуються на цілому ряді служб і технологій, які можна умовно об'єднати в два принципових структурних елементи: Движка і WPF Framework (API).

WPF Engine. Включає векторний графічний движок Windows Graphics Foundation (WGF), що є розвитком DirectX і надає можливість використання всього потенціалу сучасних відеоакселераторів при відображенні графіки. Так, наприклад, коли WPF визначає наявність відеокарти, підтримуючої апаратне прискорення, він автоматично використовує ці можливості карти. Векторний рендерінг дає можливість використовувати переваги моніторів з високою роздільною здатністю, без будь-яких додаткових зусиль програміста або користувача. Інтерфейс користувача більш не залежить від конкретних

дозволів, введено поняття «віртуального пікселя». Движок також відповідає за відтворення аудіо та візуалізацію вмісту документів [4].

WPF Framework. Об'єктно-орієнтоване середовище для створення додатків, які застосовують WPF Engine. WPF Framework дозволяє програмістам створювати просунуті додатки з багатим інтерфейсом користувача, що працюють з мультимедіа, витончено працювати з документами: все це об'єднано в єдиній програмній моделі. WPF пропонує безліч різних елементів управління для: стандартних форм (такі як кнопки і елементи введення); документів; зображень і відео; графічних примітивів; 3D; різних контейнерів і панелей для розміщення контролів і т.д. Природно, у програмістів є можливість створювати власні контрили, базуючись на існуючих, так само як і створювати їх з нуля.

2.3 Особливості створення інтерфейсів користувача у WPF

Додатки на базі WPF діляться на дві цільові групи – традиційні настільні та Web-based. Web-based відрізняються від перших більш низькими доступними привілеями безпеки під час виконання, відсутністю необхідності інсталяції програми, і спеціальним сторінково-орієнтованим (схожим на Web) механізмом навігації. Для запуску браузер-орієнтованих додатків на цільової машині необхідна присутність .NET Framework.

XAML являє собою мову розмітки, яка використовується для створення екземплярів об'єктів .NET. Хоча мова XAML – це технологія, яка може бути застосовна до багатьох різних предметних областях, її основне призначення – конструювання інтерфейсів користувача в WPF. Іншими словами, документи XAML визначають розташування панелей, кнопок і інших елементів управління, складових вікна в додатку WPF. Застосування мови XAML вирішує цілий ряд завдань[4]:

– прив'язка обробників подій. Прикріплення обробників подій в найбільш поширених місцях – наприклад, до події Click для Button

легко зробити в Visual Studio. Однак XAML дозволяє створювати більш витончені з’єднання. Наприклад, можна встановити обробник події, реагуючий на подію Click в кожній кнопці вікна;

– визначення ресурсів. Ресурси – це об'єкти, які визначені одного разу в коді XAML, в спеціальному розділі, а потім повторно використовуються в різних місцях коду розмітки. Ресурси дозволяють централізувати і стандартизувати форматування і створення невізуальних об'єктів, таких як шаблони та анімації;
– визначення шаблонів елементів управління. Елементи управління WPF проектуються як позбавлені зовнішнього вигляду; це означає, що можна підставляти власні візуальні представлення елементів замість стандартних. Щоб зробити це, необхідно створити власний шаблон елементів та управління, який являє собою не що інше, як блок розмітки XAML;
– написання виразів прив'язки даних. Прив'язка даних дозволяє витягувати дані з об'єкта і відображати їх у прив'язаному елементі. Щоб встановити це відношення і конфігурувати його роботу, необхідно додати вираз прив'язки даних до коду розмітки XAML;
– визначення анімацій. Анімації – поширений компонент додатків XAML. Зазвичай вони визначаються як ресурси, конструюються з використанням розмітки XAML, а потім прив'язуються до інших елементів управління (або ініціюються в коді). Більшість розробників WPF використовують комбінацію прийомів, розробляючи частина користувацького інтерфейсу за допомогою інструменту проектування (Visual Studio або Expression Blend), а потім проводячи тонке налаштування за допомогою редагування коду розмітки вручну.

Використання XAML обумовлено декількома причинами. По-перше, є сенс описувати за допомогою XAML ті речі, для яких це є логічним і вигідним (наприклад, розмітка елементів управління, опис векторного і

тривимірного графічного матеріалу, елементи data binding, і так далі). Вигоди від застосування XAML в контексті засобів розробки я опишу окремим пунктом. Для створення WPF-додатку не обов'язково використовувати XAML. XAML є лише верхнім допоміжним шаром керованого API WPF. Тобто до будь функціональності середовища можна отримати доступ процедурним способом.

XAML дає великі можливості для розробників інструментальних засобів. Через наявність жорсткої відкритої специфікації формату XAML, сторонні розробники можуть створювати конкуруючі інструментальні засоби (GUI Designers, Graphics Designers і т.д.). Офіційними інструментальними засобами для обробки XAML від Microsoft є продукт під кодовою назвою Orcas, який надає функціональність GUI Designer у формі розширення для Visual Studio і системи під назвою Expression Interactive Designer, яка є GUI Designer і Graphic Designer одночасно. Перший продукт призначений для програмістів, а другий більшою мірою для графічних дизайнерів відповідно. Примітно те, що Expression Interactive Designer створений за технологією WPF і є функціонуючим прикладом досить великого додатка.

WPF пропонує розробникам нову модель створення елементів управління. На відміну від «замкнутих у собі» елементів управління MFC і Windows Forms с прошитим всередині кодом піксельної отрисовки, WPF пропонує елементи управління, створені за принципами відкритості та розширюваності. До будь-якого стандартного елементу управління WPF можна застосувати стилі і шаблони. Стилі є деяким аналогом CSS: вони дозволяють декларативно, централізовано, ґрунтуючись на типі цільового елемента, з урахуванням спадкування, встановлювати значення різних властивостей візуальних елементів. Шаблони дозволяють в значній мірі змінити стандартний зовнішній вигляд і поведінку візуальних елементів декларативними способами. Стандартна бібліотека елементів управління містить достатній набір засобів для організації GUI, але все ж помітно, що

Microsoft залишає велику нішу на ринку для сторонніх розробників бібліотек елементів управління [5].

Використання XAML, інструментальних засобів, Retained mode graphics і механізмів розширення елементів управління дає можливість більш структуровано організувати процес розробки presentation layer додатків. Таким чином, дизайнер може «намалювати» загальний вигляд програми за допомогою XAML, а програміст потім займається нескладної адаптацією такого макета до працюючого додатком. Цим потік робіт над оригінально виглядають додатками WPF схожий з діяльністю з розробки традиційних веб-сайтів (але з отсутсвием механічних проблем з нарізкою, верстkою і підтримкою переносимості).

Розробники ввели до складу WPF досить велику функціональність підтримки відображення та обробки різних документів. Тепер не потрібно використовувати ActiveX компонент веб-браузера або стандартний RichTextBox для відображення складного відформатованого документа з різними вбудованими об'єктами. Для цього в WPF входять стандартні елементи управління. У цьому контексті Microsoft пропонує для використання новий універсальний заснований на XML формат XPS (XML Paper Specification). Крім жорстко специфікованого загального відкритого формату документів, специфікація XPS надає можливості автоматичної архівації й передачі таких документів, і просунуту систему друку твердих копій. Існує так само вбудована підтримка анімації (зміни в залежності від різних факторів) практично будь-яких властивостей візуальних об'єктів. Така анімація може бути задана в XAML декларативно. Добре підтримуються та вбудовуються в документи XAML (потокові) відео і аудіо дані [5].

Програміст отримає можливість сформувати зображення на екрані, використовуючи техніку композиції описів векторних примітивів в пам'яті, а не зафарбовування матриці пікселів за запитом ОС. З цього виходять дуже багато сильні сторони WPF (наприклад, відкрита модель елементів управління і шаблони). Є можливість застосовувати всі можливості

векторних трансформацій, анімації, згладжування і так далі. Це не згадуючи абсолютно нового зовнішнього вигляду додатків для користувачів.

Відтворенням графіки займається єдиний некерований компонент WPF milcore.dll, робота якого базується на DirectX. Неможливо отримати доступ до API WPF з некерованих мов.

2.4 Механізми взаємодії дизайнера та розробника в WPF

Як правило, web-додаток являє собою програму, що реалізовує певні бізнес-завдання. Додаток повинен взаємодіяти з даними, які розташовуються в базі даних веб-застосунку книжкового каталогу. Елементи контролю повинні підтримувати візуальне уявлення функціональності системи для користувача, проводити верифікацію даних, що вводяться і взаємодіяти з бізнес-класами. Шар бізнес-логіки додатка повинен забезпечувати основну функціональність програми: формувати бізнес-класи, реалізовувати алгоритми обробки даних, забезпечувати з'єднання з даними та їх кешування. Реалізація даного шару програми може бути побудована на базі класів, що реалізують бізнес-логіку, методами класів інтерфейсних елементів або методами класів моделі даних. Шар даних повинен забезпечити взаємодію додатку з даними системи управління базами даних [5].

Як правило, дизайнер створює макет інтерфейсу користувача, а конкретну реалізацію мрій дизайнера все одно створює програміст. Це веде до того, що, по-перше, дизайнер може створити макет інтерфейсу користувача, який в принципі не реалізується або важко реалізується, а по-друге, програміст може помилитися в реалізації або ж по-своєму інтерпретувати ті чи інші побажання дизайнера. У підсумку дизайнер змушений переглядати творіння і висловлювати побажання до покращення інтерфейсів. Таким чином, розробка інтерфейсу користувача перетворюється в багатоетапний ітеративний процес, що гальмує основну розробку.

За допомогою WPF можливо виключити такі ситуації. Наявність у WPF загального знаменника – мови опису інтерфейсів XAML дозволить розділити роботу дизайнера і програміста. Справді, програмісту від XAML в основному потрібні тільки імена елементів управління і обробники їх подій, які він використовує в своїй програмі. Дизайнер же може спокійно розробляти інтерфейсу користувача в інструментах, більш пристосованих для цього завдання, а не в незрозумілих йому середовищах розробки додатків.

Треба визначити критерії для здійснення вибору технології для реалізації інтерфейсу користувача веб-застосунку книжкового каталогу на основі технології WPF.

Хоча ASP.NET бере свою назву від старої технології Microsoft ASP, вона значно від неї відрізняється. Microsoft повністю перебудувала ASP.NET, ґрунтуючись на Common Language Runtime (CLR), яка є основою всіх додатків Microsoft .NET. Розробники можуть писати код для ASP.NET, використовуючи практично будь-які мови програмування, що входять в комплект .NET Framework (C #, Visual Basic.NET і JScript .NET). ASP.NET має перевагу в швидкості в порівнянні зі скриптовими технологіями, так як при першому зверненні код компілюється і поміщається в спеціальний кеш, і згодом тільки виконується, не вимагаючи витрат часу на парсинг, оптимізацію.

DirectX слід вибирати, коли завданням являється реалізація спеціальних додатків з 3D графікою чи, наприклад, насичені графікою ігри. Технологія DirectX дозволяє програмістам створювати в Windows додатки з вбудованим доступом до апаратних засобів. При цьому їм не потрібно знати специфіку апаратної конфігурації певного комп'ютера – явного програмування конкретної плати не потрібно. Фактично DirectX виконує роль проміжної ланки між програмою і драйвером, перетворюючи узагальнені команди в команди, специфічні для того чи іншого пристрою.

Технологія Windows Forms є технологією інтелектуальних клієнтів для .NET Framework. Windows Forms – це набір керованих бібліотек, що

забезпечують поширені завдання додатків, наприклад читання і запис у файлову систему. За допомогою середовища розробки типу Visual Studio можна створювати додатки Windows Forms, які відображають інформацію, запитують введення від користувачів і обмінюються даними з віддаленими комп'ютерами по мережі. За допомогою такої технології розробляють бізнес-додатки, які повинні працювати у всіх версіях Windows.

Технологію Windows Presentation Foundation доцільно використовувати для розробки витонченого інтерфейсу користувача. Як і .NET Framework в цілому, WPF являє собою технологію, орієнтовану на Windows. Це означає, що додатки WPF можуть використовуватися тільки на комп'ютерах, що працюють під управлінням операційної системи Windows. Додатки WPF, засновані на браузерах, обмежені аналогічним чином – вони працюють тільки на комп'ютерах Windows, хоча підтримують різні браузери. Технологія Silverlight спроектована як підмножина платформи WPF, працює в будь-якому сучасному браузері за рахунок використання модуля, і відкрита для інших операційних систем [5].

WPF входить в операційну систему Microsoft Windows. Для того щоб запускати WPF-додатки, комп'ютер користувача повинен працювати під управлінням Microsoft Windows. Інтегрована середа розробки Microsoft Visual Studio володіє однією очевидною перевагою –підтримує середу часу проектування, яка дозволить створювати користувацькі інтерфейси за допомогою операцій, які виконуються за допомогою миші, як при розробці додатків із застосуванням Windows Forms. За допомогою Visual Studio можна попередньо переглядати отримані інтерфейси користувача на етапі проектування. З технічної точки зору, Visual Studio використовує .NET, яка включає WPF.

3 ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ КНИЖКОВОГО КАТАЛОГУ

3.1 Визначення архітектури

Архітектура програмного забезпечення – це комплекс технічних рішень, спрямованих на відповіді на вимоги до програмного забезпечення. Вона оперує програмними модулями, які є складовими частинами системної структури, вирішуючи певні завдання та взаємодіючи з навколишнім середовищем через визначений інтерфейс. Ці модулі є одиницями збірки та конфігураційного управління, включаючи файли з кодом на певній мові, бінарні файли та інші документи, що утворюють компоненти системи та структурні одиниці розгортання.

Веб-застосунок книжкового каталогу має клієнт-серверну архітектуру. Це означає наявність двох взаємодіючих процесів – клієнтського та серверного, які можуть виконуватися на різних комп'ютерах та обмінюватися даними через мережу. Така архітектура може бути використана для систем обробки даних на базі СУБД, пошукових систем та інших. На клієнтському комп’ютері додаток відповідає за формування інтерфейсу користувача, логічну обробку даних і безпосереднє управління ними. Файловий сервер забезпечує лише базові операції з файлами, такі як відкриття, закриття та модифікація, а маніпулювання даними відбувається різними незалежними процесами, які не завжди спільно працюють.

Веб-додатки використовують користувацький інтерфейс, побудований на динамічних сторінках XAML, що містять дані, запитані користувачем. Ці додатки мають можливість доступу до даних, які зазвичай зберігаються у вигляді таблиць та зв'язків між ними, подібно до набору пов'язаних об'єктів. Основною технологією, яка забезпечує функціонування додатку, є WPF [5].

Архітектура веб-застосунку книжкового каталогу може бути проілюстрована схемою, наведеною на рисунку 3.1. Програмні модулі

розробляються для двох рівнів: рівень інтерфейсу користувача (WebUI) і рівень доступу до даних.

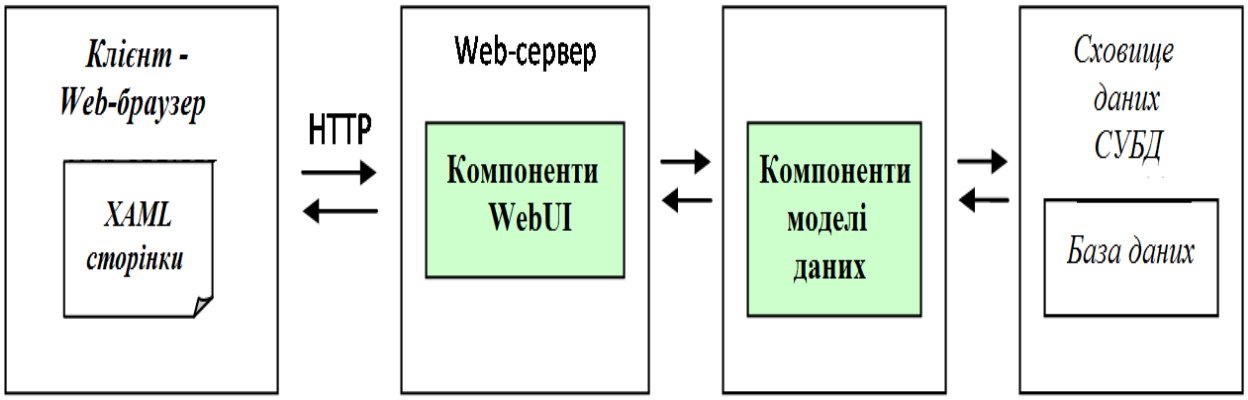

Рисунок 3.1 – Схема архітектури Web-застосунку книжкового каталогу

WPF виконується на Web-сервері і використовує функціональні можливості, що надаються середовищем .NET Framework. Однак на комп'ютері користувача WPF не виконується; замість цього вона динамічно генерує елементи, необхідні браузеру для візуалізації сторінки. До цих елементів, які відправляється браузеру, відносяться HTML-код, зображення і CSS-таблиці, які надають кольори, розміри і типи вирівнювання для різних елементів в HTML. WPF також генерує процедурний код, який відправляється браузеру, для забезпечення механізму перевірки коректності даних і вказівки браузеру, як має здійснюватися взаємодія з Web-сервером.

.NET Framework об'єднала логіку програми-клієнта і логіку відповідного додатку-сервера під час виконання. Оскільки та ж сама платформа використана для написання обох частин, результат полягає у взаємодії платформ. Разом з появою розширювальної мови розмітки (XML), мови, подібної HTML, але спеціалізованої для обробки даних які відображаються, розробка мережевих додатків досягла досконалості [6]. Схема взаємодії клієнта і сервераза технологією .NET Framework наведена на рисунку 3.2.

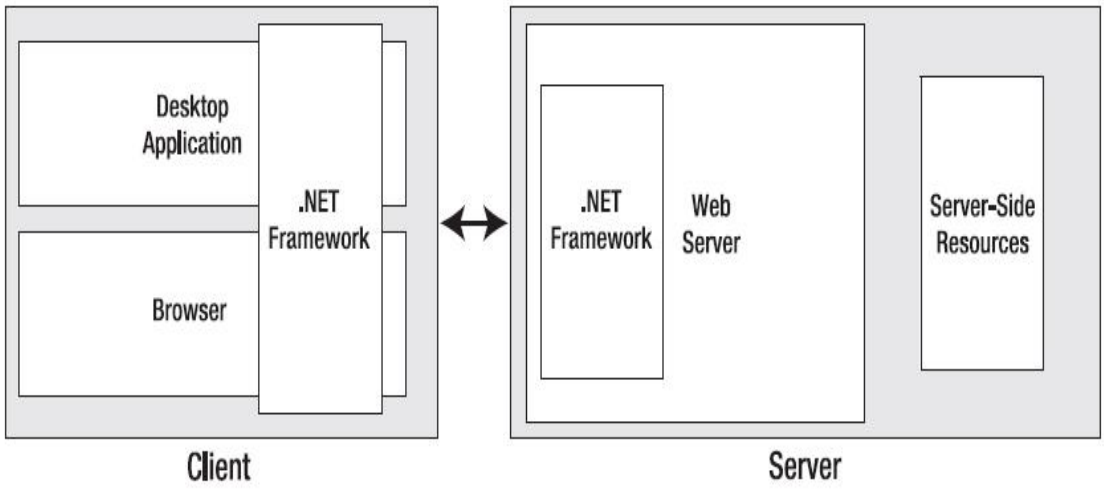

Рисунок 3.2 – Схема взаємодії клієнта і сервера у технології .NET Framework

WPF забезпечують два інтерфейси: автономні (Windows) і мережеві (Web). При виконанні розробки веб-застосунку книжкового каталогу з застосуванням технології WPF для публікації в подальшому у мережі Інтернет має сенс використовувати клієнтське додаток, що працює в Web-браузері, а не як у додатку Windows. Щоб дозволити цим клієнтам представляти сучасні інтерфейси, WPF забезпечує технологію XBAPs (Extended browser application). XBAP-файли виконаються програмами-браузерами. Багато з існуючих мережевих додатків є статичними і вимагають частих перевантажень сторінки і постійного переміщення для виконання завдання, тому вони не завжди привабливі. Їх можна вдосконалити технологіями JavaScript і XML.

XAML дозволяє оновлювати інформацію на сторінці без додаткової перевантаження. Додаток XAML проглядається, а не виконується. Браузер робить запит серверу на передачу додатку, який завантажується в програму-браузер. Створюється і встановлюється XAML-додаток також як і виконувані програми. Відмінності визначаються в часі компіляції, змінами параметрів настроювання компілятора [5].

XBAP може діяти як клієнт для Web-додатків, які побудовані, використовуючи ASP.NET, сторінки JSP, або інших мережевих технологій. Щоб мати доступ до цього Web-додатка, XBAP може використовувати гіпертекстовий транспортний протокол. Незалежно від того, яка серверна платформа використовується, XBAP завжди завантажується за технологією ClickOnce [7]. Схема взаємодії WPF-клієнта с іншими web-додатками наведена на рисунку 3.3.

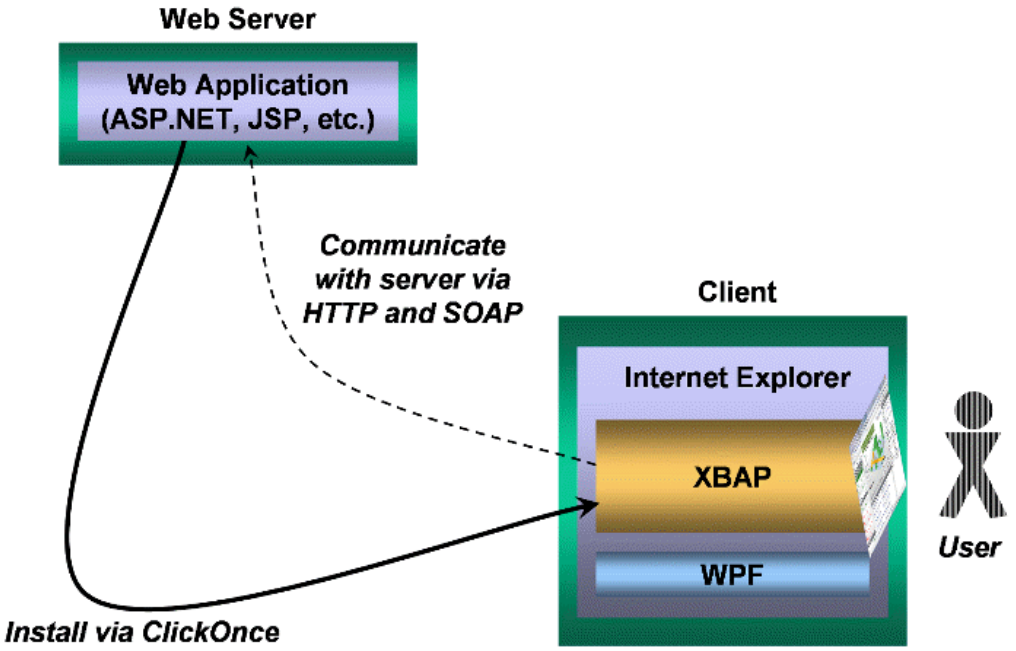

Рисунок 3.3 –Механізм взаємодії WPF-клієнта з web-застосунками

Під час установки XAML-додаток, створюється три файли: маніфест розгортання: файл з розширенням .xbap. Технологія Click-Once використовує це тип файлу для установки програми; маніфест додаток: файл з розширенням .exe.manifest. Тут міститься стандартні .NET метадані, які створюються для будь-якого керованого застосування; Виконуваний код: файл з розширенням .exe. Це виконуваний файл, який надалі буде загружатися.

Розгортання відбувається, коли користувач вказує індикатор ресурс (URI) .xbap файлу. Це викликає ClickOnce, завантажувати додатки. Треба зауважити, що користувачі можуть розгорнути додаток-XAML, лише використовуючи ClickOnce. Додаток не може бути завантажено .exe файлом, треба переглянути .xbap файл на web-сервері, щоб виконати додаток знову. Це гарантує прикладну послідовність версій бо може бути виконана тільки прийнята сервером прикладна версія. Якщо .xbap файл розміщується на локальній системі, то будуть використовуватися установки зони локального захисту. Це зручно для тестування. Але реальна робота додатка буде здійснюватися в Інтернет-дозволах зони безпеки. Оскільки додаток працює в Інтернет-зоні безпеки, багато можливостей не вирішені, тому що додаток виконується з частковим довірою [6].

3.2 Проектування веб-застосунку за допомогою методології SADT

При проектуванні веб-застосунку книжкового каталогу була обрана методологія функціонального моделювання SADT (Structured Analysis and Design Technique) з використанням стандарту IDEF0.

Елементи, які застосовуються при проектуванні за методологією SADT складаються з діаграм зі стрілками, що мають чотири сторони, кожна з яких має свій власний зміст: вхід, вихід, контроль та механізм. Посередині діаграми розташована діяльність. Визначимо терміни, які визначаються за методологією SADT [8]:

- поняття Діяльність – це функція або процес, який здійснює перетворення вхідних даних у вихідні;
- поняття Вхідні дані – це інформація або дані, необхідні для початку процесу перетворення;
- поняття Вихід – це дані або інформація, що генеруються діяльністю в результаті цього перетворення.

– поняття Контроль – це будь-яке обмеження, яке впливає на поведінку діяльності;
– поняття Механізм – це люди, ресурси або інші засоби, необхідні для виконання діяльності.

Розробка контекстної діаграми передбачає розробку високорівневої моделі, яка відображає систему або процес у контексті його зовнішнього середовища. Вона є першим кроком у представленні системи (за стосунку), що допомагає зрозуміти взаємодію з зовнішніми чинниками та визначити межі системи. Контекстна діаграма не надає детальної інформації про внутрішню структуру і функціонування системи, але забезпечує загальний огляд і контекст для розуміння системи та її взаємодії з навколишнім середовищем. Вона слугує вихідною точкою для більш детального моделювання та аналізу функцій і процесів, що відбуваються у системі [10].

Контекстна діаграма веб-застосунку книжкового каталогу наведена на рисунку 3.4.

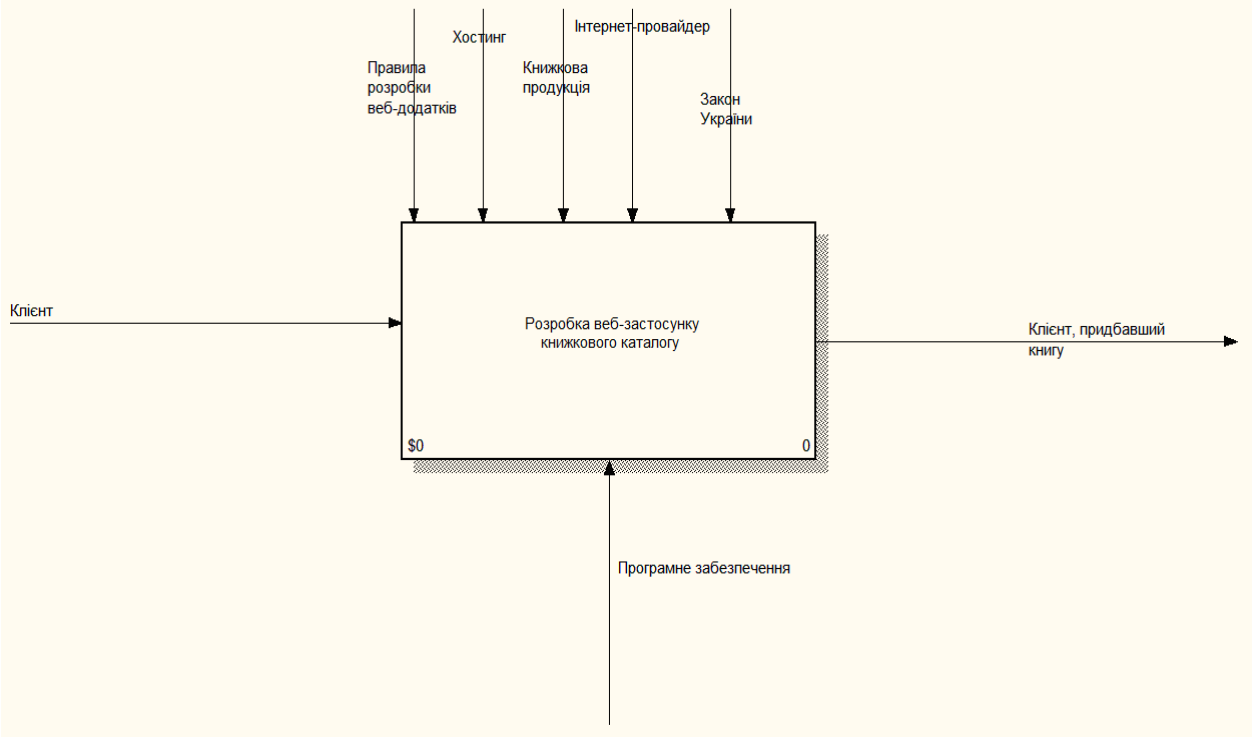

Рисунок 3.4 – Контекстна діаграма веб-застосунку книжкового каталогу

Головним процесом в контекстній діаграмі виступає «Розробка веб-застосунку книжкового каталогу», де на вході надається інформація про клієнта. Керування здійснюється правилами розробки веб-застосунків, наявністю книжкової продукції, хостингом, Інтернет-провайдером, та законами України щодо розповсюдження книжкової продукції без порушення авторських прав. Виходов в цієї контекстної діаграмі є клієнт, який засобами за стосунку придбав книгу.

Наступним кроком проектування за цією методологією є розбиття визначеної системи – декомпозиція на більш дрібні частини. Ці частини можуть бути визначені як компоненти або підзавдання, що дозволить провести детальний аналіз, визначити структуру та функції системи.

Такий етап дозволяє ще на етапі проектуванні здійснювати розробку уточнюючих моделей, які є більш деталізованими, що значно полегшує системний аналіз, проектування, документування окремих компонентів загальної системи. З кожним кроком деталізації саме механізм декомпозиції дозволяє більш глибоко з потрібною точністю ідентифікувати функції, процеси, взаємозв'язки у системі. Синтаксис опису функціональної моделі на різних рівнях декомпозиції системи може бути як однаковим, так і мати деяку модифікацію з метою отримання більш детальної інформації щодо компоненти системи.

Після декомпозиції контекстної діаграми виникли три блоки, які описують основні етапи робіт. Розглянемо кожний з них.

Блок «Аналіз вимог» передбачає аналіз існуючих систем та формулювання вимог, але на вході непередбачено жодних даних. Він керується правилами розробки веб-застосунків, а програмне забезпечення виступає у ролі механізму. На виході передбачено отримання завдання.

Блок «Розробка додатку» передбачає створення веб-застосунку книжкового каталогу, який має охопити розробку всіх компонентів системи: основних функцій, користувацького інтерфейсу, бази даних. На вході маємо постановку завдання, а на виході отримуємо готовий веб-застосунок

книжкового каталогу. Керується цей блок також правилами розробки веб-застосунків, а в якості механізму виступає програмне забезпечення.

Блок «Розміщення додатку» передбачає виконання наступних робіт: отримання доменного імені, публікація веб-застосунку на сервері, заповнення бази даних, виконання програмної реалізації. На вході отримуємо інформацію про клієнта та розроблений застосунок, а на виході маємо отримати клієнта, якому надана можливість засобами за стосунку отримати бажану книжкову продукцію. Діаграма декомпозиції блоку «Розробка веб-застосунку книжкового каталогу» наведена на рисунку 3.5.

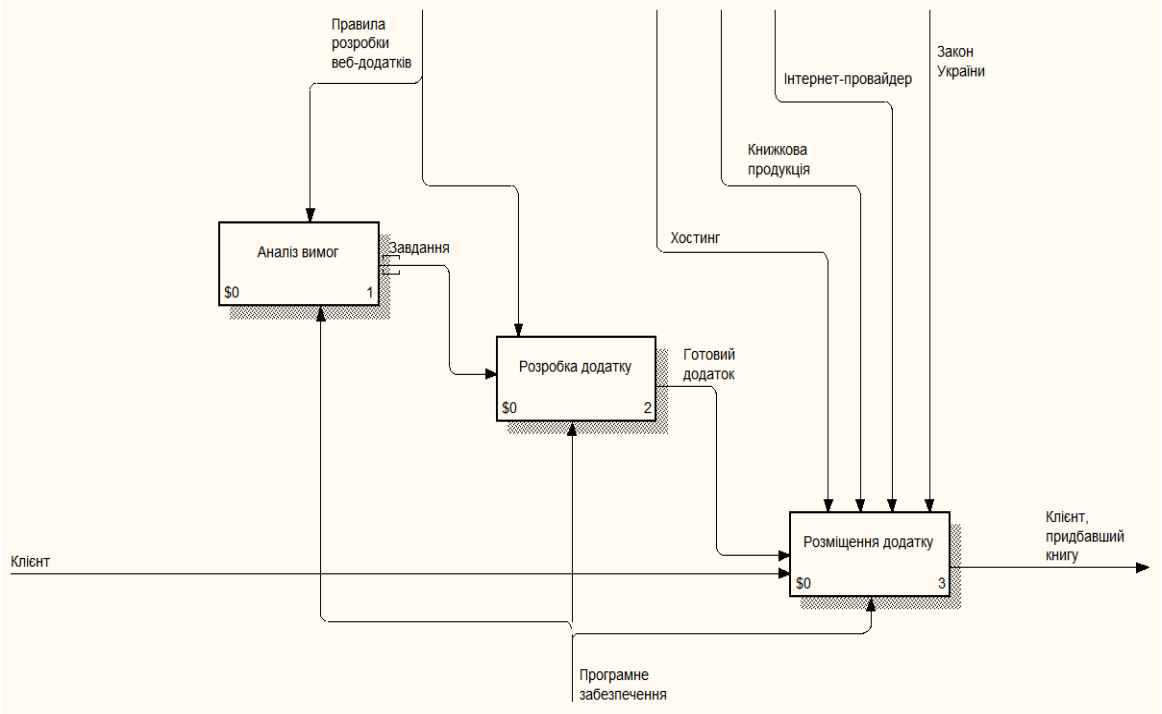

Рисунок 3.5 – Діаграма декомпозиції блоку «Розробка веб-застосунку книжкового каталогу»

При подальшій деталізації контекстної діаграми виконаємо декомпозицію блоку «Розробка додатку», що надасть наступні блоки:

Блок «Розробка інтерфейсу» передбачає розробку інтерфейсу користувачів застосунку, де в якості вхідних даних виступає постановка

завдання. Робота керується правилами розробки веб-застосунків, а механізмом є програмне забезпечення. В результаті на виході маємо отримати розроблений інтерфейс для всіх визначених постановкою завдання категорій користувачів. Блок «Розробка функціональності» відповідає за реалізацію всіх визначених функцій веб-застосунку, де вхідними даними є розроблений інтерфейс користувачів та постановка завдання. Робота також керується правилами розробки веб-застосунків, а механізм – програмне забезпечення. Виходом з цього блоку є застосування з зазначеним в специфікаціях функціоналом. Блок «Розробка бази даних» передбачає проектування як логічної та фізичної моделі даних, які необхідні в системі. Входом є розроблений функціонал додатку та завдання, які повинна виконувати система. Як і на попередніх етапах, робота керується правилами розробки веб-застосунків, а програмне забезпечення виступає механізмом. На виході отримуємо готовий програмний продукт – веб-застосунок книжкового каталогу. Діаграма декомпозиції блоку «Розробка додатку» наведена на рисунку 3.6.

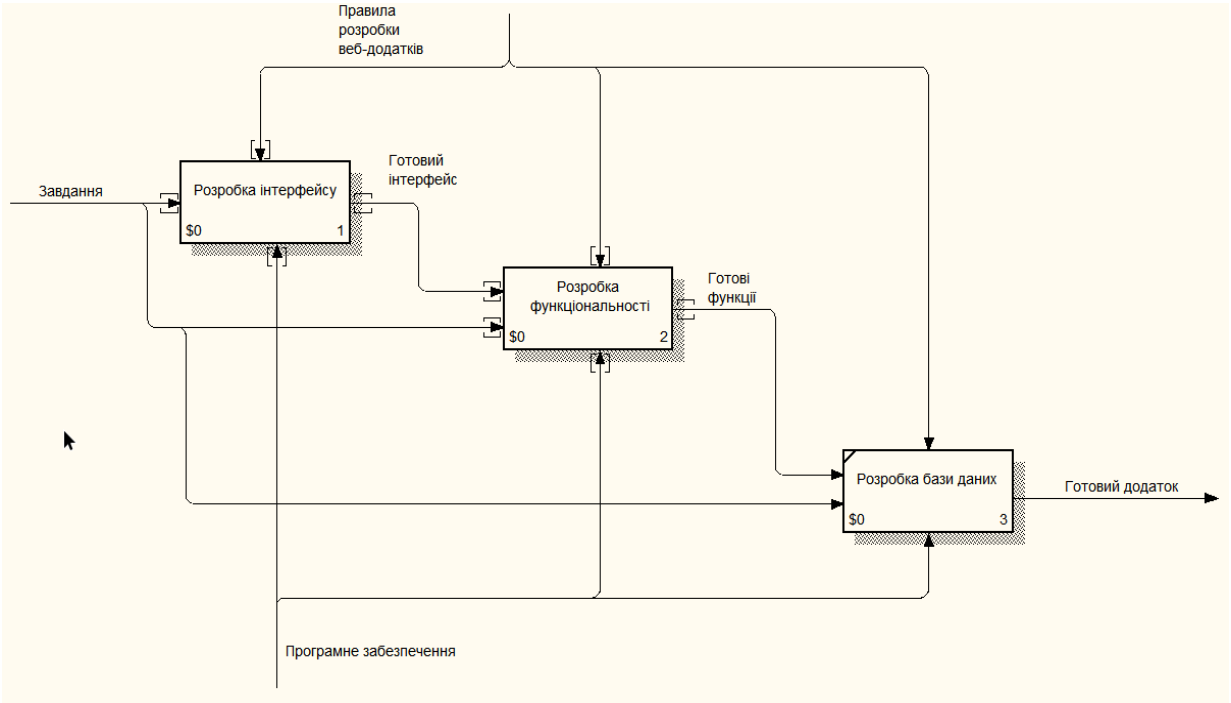

Рисунок 3.6 – Діаграма декомпозиції блоку «Розробка додатку»

Проведемо декомпозицію останнього блоку «Розміщення додатку» та отримаємо у результаті чотири нових блоки, які розглянемо більш детально.

Блок «Отримання доменного імені» має ціллю отримання доменного ім’я для розміщення веб-застосунку та робота керується хостингом та Інтернет-провайдером. Механізмом виступає спеціалізоване програмне забезпечення, а на виході є отримане доменне ім’я.

Блок «Публікація додатку на сервері» передбачає налаштування серверу на якому буде в подальшому розміщено веб-застосунок книжкового каталогу. Входом є розроблений веб-застосунок з отриманим доменним ім’ям. Керування роботою здійснюється хостингом та Інтернет-провайдером, а в якості механізму виступає програмне забезпечення. На виході маємо опублікований веб-застосунок книжкового каталогу на сервері.

Блок «Наповнення бази даних» повинен відповідати за заповнення бази даних інформацією, яка є необхідною для здійснення засобами створеного веб-застосунку продажі книжкової продукції. Робота керується хостингом, Інтернет-провайдером, та інформацією про книжкову продукцію, в якості механізма виступає програмне забезпечення, а на виході отримаємо веб-застосунок книжкового каталогу з навченою даними базою даних.

Блок «Реалізація» передбачає реалізацію етапу проектування впровадження та супровід. Входом є інформація про клієнта, та готовий веб-застосунок книжкового каталогу. Робота керується хостингом, Інтернет-провайдером та діючими законами України. Механізмом є програмне забезпечення. А виходом є зацікавлений клієнт, який засобами розробленого додатку, отримав бажану книжкову продукцію.

Розроблені діаграми за стандартом IDEF0 допомогли створити повну та докладну модель веб-застосунку книжкового каталогу, що сприяє кращому визначенню та розумінню його функцій та структури. На рисунку 3.7 наведена декомпозиції блоку «Розміщення додатку».

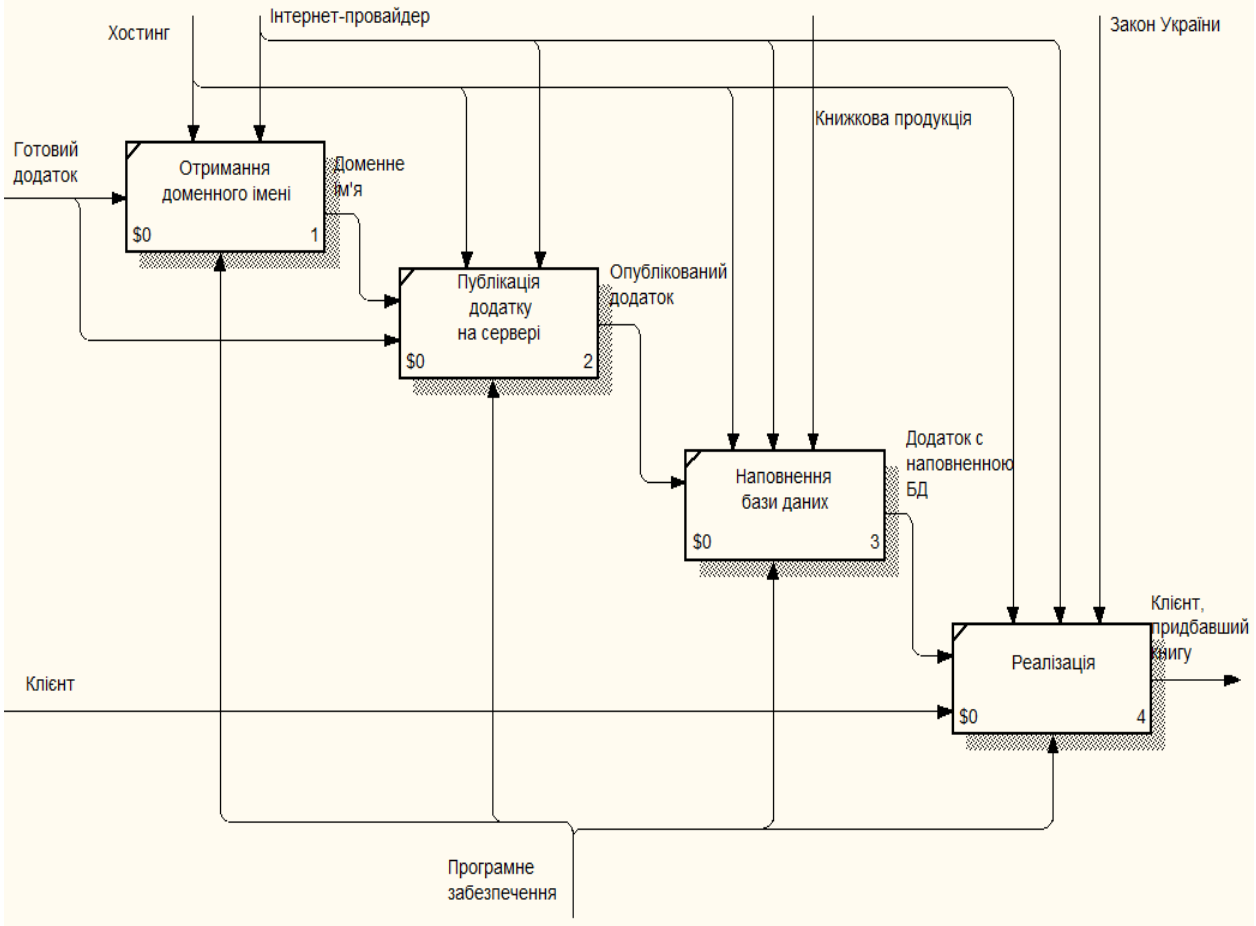

Рисунок 3.7 – Діаграма декомпозиції блоку «Розміщення додатку»

Під час проектування веб-застосунку книжкового каталогу були створені діаграми IDEF0 для представлення системи у вигляді структурованої моделі та опису її функцій та взаємодій між компонентами.

Діаграма IDEF0 забезпечила детальний опис функцій, взаємозв'язків та ієрархічної структури системи. Це дозволило зрозуміти, як кожна функція пов'язана з іншими, і як вони впливають на загальну роботу веб-застосунку.

3.3 Проектування інтерфейсу користувача

Веб-застосунок повинен надавати користувачам зручний, що має безліч функціональних можливостей, графічний інтерфейс. Першим кроком при створенні інтерфейсу веб-застосунку книжкового каталогу інформаційної є

розробка візуального дизайну, що передбачає проектування загальної схеми системи. Ця візуальна архітектура визначає «зовнішній вигляд і поведінку» інтерфейсу застосування з точки зору користувача. Одною з основних деталей, що впливають на враження користувача, є зручне меню і можливості навігації, наявність зображень і організація елементів на сторінці системи. Меню повинні бути інтуїтивно зрозумілими і підкріплюватися навігаційними підказками [8].

Головна сторінка веб-застосунку книжкового каталогу з застосуванням технології WPF повинна відображати основну функціональність,тому головна сторінка за стосунку, яка наведена на рисунку 3.8 має наступні розділи: область завдання напряму пошуку; область основного вмісту; меню навігації та пошуку.

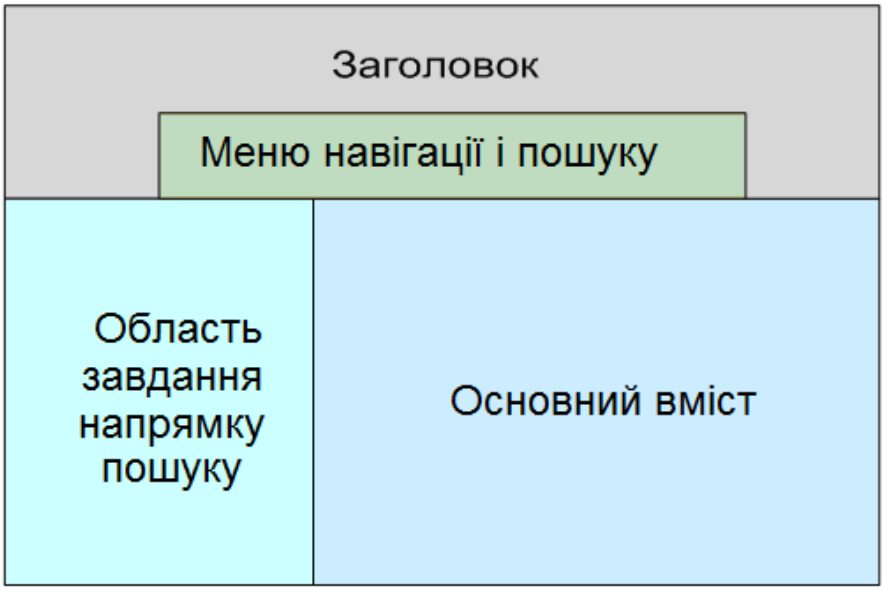

Рисунок 3.8 – Схема інтерфейсу користувача веб-застосунку книжкового каталогу

Процес проектування рівня інтерфейсу користувача не означає тільки проектування та розмітку головної сторінки системи, він також передбачає розробку навігаційної системи. Навігація визначається областю завдання

напряму пошуку і навігаційним меню. Області завдання напряму пошуку являє собою ієрархію книжкових класифікаторів. Меню навігації та пошуку складається з трьох кнопок: «Кошик», що реалізує функцію перегляду вмісту кошика; «Про нас», що реалізує функцію перегляду інформації про підприємство; «Пошук», що реалізує функцію пошуку певного видання. При зміні класифікатора змінюється наповнення основного вмісту (рис. 3.9).

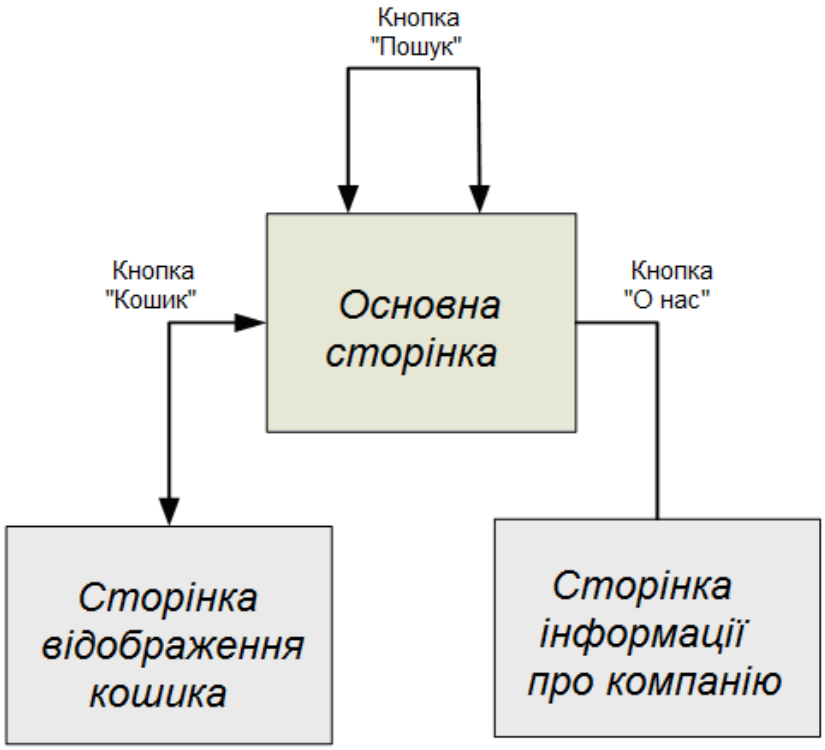

Рисунок 3.9 – Схема навігації веб-застосунку книжкового каталогу

У веб-застосунку книжкового каталогу передбачено інтерфейс адмін-частини за стосунку, який передбачає наявність тільки навігаційного меню та частини відображення основного вмісту. Відмінність розробки адміністраторській частини від користувальницької полягає в наступному: менше уваги приділяється графічної стороні інтерфейсу; зручність навігації інтерпретується інакше; доступ до даних передбачає не лише відображення, але і редагування; можливість редагування даних тягне за собою обмеження доступу до адміністраторських сторінок (рис.3.10).

Рисунок 3.10 – Схема інтерфейсу адміністраторській частини веб-застосунку книжкового каталогу

3.4 Проектування бази даних веб-застосунку

Для відображення логічної структури веб-застосунку книжкового каталогу була обрана модель представлення даних «сутність-зв'язок». Ця модель ґрунтується на важливій семантичній інформації про реальний світ та призначена для логічного опису даних. Вона визначає значення даних у контексті їх взаємозв'язку з іншими даними. Сутність фактично представляє собою набір атрибутів, які описують властивості всіх членів даного набору сутностей. Домен – це множина значень атрибуту. Зв'язок – це асоціація, яка встановлюється між кількома сутностями. Для бази даних веб-застосунку книжкового каталогу були визначені такі сутності: Замовлення, Список, Замовник, Книги, Автори, Замовлені книги.

Кожна сутність повинна містити атрибут або групу атрибутів, які будуть однозначно ідентифікувати кожен екземпляр сутності. Такий атрибут називають первинним ключем [9]. Під час аналізу предметної області веб-застосунку книжкового каталогу були визначені необхідні дані для її

функціонування. Перш за все, це інформація про книги: назва, автори, рік видання, титульна сторінка, коротка інформація про вміст, детальна інформація про вміст, ціна. Крім того, потрібно врахувати, що кожна книга відноситься до якого-небудь напрямку (жанру), а деякі книги можуть бути ідентифіковані по розділу. Тому в базі даних необхідно зберігати інформацію про напрями (жанри) та розділи.

При проектуванні веб-застосунку книжкового каталогу з застосуванням технології WPF були розглянуті наступні сутності та їх атрибути.

Сутність ЗАМОВЛЕННЯ («ORDERS») містить наступні атрибути (табл. 3.1): id_замовлення, id_замовника, дата_замовлення, статус, примітка.

Таблиця 3.1 – Сутність «ORDERS»

№	Атрибут	Тип
1	Id_order	int(11)
2	Id_customer	int(11)
3	date	datetime
4	status	enum(0,1)
5	prim	text

Сутність СПИСОК («LIST») містить атрибути (табл. 3.2), які визначають посилання на таблицю Books та посилання на таблицю Autors: id_книги, id_автора.

Таблиця 3.2 – Сутність «LIST»

№	Атрибут	Тип
1	Id_book	Int
2	Id_autor	Int

Сутність ЗАМОВНИК («CUSTOMERS») містить наступні атрибути (табл. 3.3): id_замовника, Призвище_Ім’я_По-батькові, e-mail, телефон, адреса, логін, пароль.

Таблиця 3.3 – Сутність «CUSTOMERS»

№	Атрибут	Тип
1	id_customers	int(11)
2	name	varchar(255)
3	email	varchar(255)
4	phone	varchar(255)
5	address	varchar(255)
6	login	varchar(25)
7	password	varchar(25)

Сутність КНИГИ («BOOKS») містить наступні атрибути (табл. 3.4): id_книги, назва_книги, титульна_сторінка, інформаційний_файл, класифікація_книги, ціна, коментарі.

Таблиця 3.4 – Сутність «BOOKS»

№	Атрибут	Тип
1	id _book	Int
2	Nazv	VarChar (50)
3	Pict	VarChar (25)
4	File_name	VarChar (25)
5	id_type	Int
6	Cost	Float (7, 2)
7	comment	VarChar(100)

Сутність АВТОРИ («AUTORS») містить атрибути (табл. 3.5): id_автора, им’я_ автора, коментарі.

Таблиця 3.5 –Сутність «AUTORS»

№	Атрибут	Тип
1	Id_autor	Int
2	Autor	VarChar (25)
3	comment	VarChar (100)

Сутність ЗАМОВЛЕНІ КНИГИ («ORDER_BOOKS») містить атрибути (табл. 3.6): id_замовленого_товару, id_замовлення, id_товару, кількість замовлених товарів, ціна, ім’я, логін.

Таблиця 3.6 – Сутність «ORDER_BOOKS»

№	Атрибут	Тип
1	id	int(11)
2	id_order	int(11)
3	id_book	int(11)
4	quantity	tinyint(4)
5	name	varchar(255)
6	login	varchar(25)
7	price	float

Сутність КАТЕГОРІЯ («TYPES») містить атрибути (табл. 3.7): id_категорії, назва_ напрямку, назва_розділу.

Таблиця 3.7 – Сутність «TYPES»

№	Атрибут	Тип
1	id_type	Int
2	Direct	VarChar (25)
3	part	VarChar (25)

Після здійснення опису всіх головних сутностей та їх атрибутів у базі даних для веб-застосунку книжкового каталогу, треба визначити зв'язки між цими сутностями. Після визначення таблиць, полів, індексів та зв'язків між таблицями, слід нормалізувати проектовану базу даних.

Важливість нормалізації полягає в тому, що вона дозволяє розбити відносини, які містять велику надмірність інформації, на дрібніші логічні одиниці, що групують лише дані, об'єднані «по природі» [10]. Уявімо базу даних у вигляді моделі «Сутність – Зв'язок», де використовується тип зв'язку «один-до-багатьох» у середовищі розробки Microsoft SQL Server (рис. 3.11).

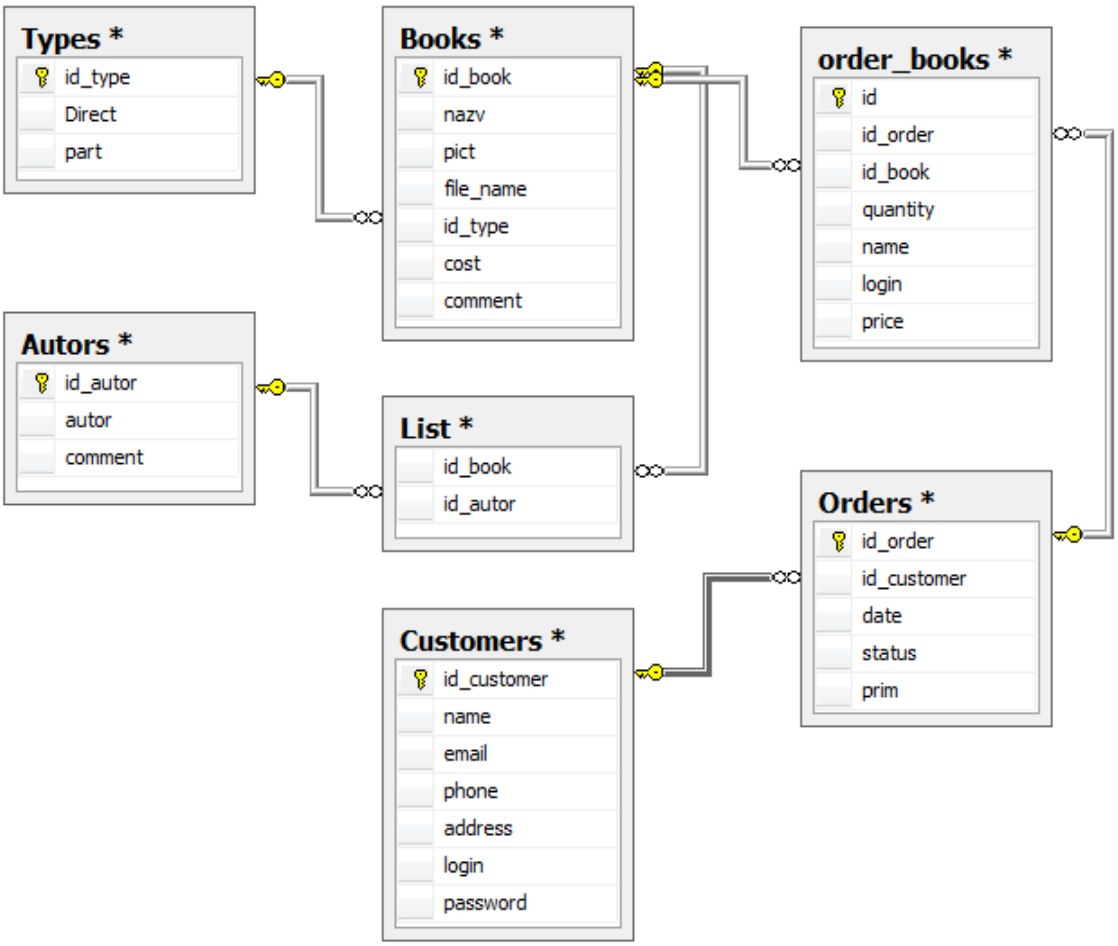

Рисунок 3.11 – Модель «Сутність – Зв'язок» бази даних веб-застосунку

У результаті проведеного, в ході дипломної роботи, проектування веб-застосунку книжкового каталогу з застосуванням технології WPF визначена

архітектура застосунку, розроблена база даних, розроблені структури і шаблони сторінок для різних категорій користувачів, розроблено механізм аутентифікації користувачів в системі і розмежування прав доступу до інформації. Основним результатом проектування стало реалізація вимог, функціональних можливостей і бізнес-логіки веб-застосунку книжкового каталогу з застосуванням технології WPF для визначених категорій користувачів.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

4.1 Застосування технології WPF для створення інтерфейсу користувача

Однією з ідей WPF є те, що програмування інтерфейсу зробити якомога більше декларативним. Для опису інтерфейсу користувача технологія WPF використає мову XML. Логіка додатка управляється процедурних кодом на мові C #. Важливою деталлю є легкість XAML для локалізації інтерфейсу користувача. Якщо раніше для цих цілей використовували спеціальні механізми схожі на таблиці рядків, завантаження рядків з ini-файлів, ресурсні dll або ж спеціальні утиліти «витягування» рядків, то з появою XAML, що зберігає всі рядки інтерфейсу користувача в текстовому вигляді, локалізація значно спрощується.

Кожен тег XAML відповідає класу об'єктної моделі WPF. Тег зазвичай має набір атрибутів, які відповідають властивостям цього класу. Під час компіляції парсер по XAML-опису породжує частковий (partial) клас, який містить еквівалентний код. Кожен тег XAML стає екземпляром відповідного класу об'єктної моделі, значення атрибутів тега присвоюються відповідним властивостям цього об'єкта. Потім частковий клас, породжений з XAML, об'єднується з code-behind кодом, написаним програмістом.

Все це надає можливість, крім усього іншого, породжувати інтерфейс на льоту. Як правило, дизайнер створює макет користувача інтерфейсу в графічному редакторі, на папері або за допомогою будь-якого іншого засобу. Як наслідок, конкретну реалізацію мрій дизайнера все одно створює програміст. Це веде до того, що, по-перше, дизайнер може створити макет інтерфейсу користувача, який в принципі не можливо реалізувати або важко реалізований, а по-друге, програміст може помилитися в реалізації або ж по-своєму інтерпретувати ті чи інші побажання дизайнера [11].

У підсумку, дизайнер змушений переглядати вийшло творіння чи ні, і висловлювати побажання до покращення інтерфейсів. Таким чином, розробка інтерфейсу користувача перетворюється в багатоетапний ітеративний процес, що гальмує основну розробку. За допомогою WPF стане можливим виключити такі ситуації. Наявність у ньому загального знаменника – мови опису інтерфейсів XAML – дозволить розділити роботу дизайнера і програміста. Справді, програмісту від XAML в основному потрібні тільки імена елементів управління і обробники їх подій, які він використовує в своїй програмі. Дизайнер же може спокійно розробляти інтерфейс користувача в інструментах, більш пристосованих для цього завдання, а не в незрозумілих йому середовищах розробки додатків (рис. 4.1).

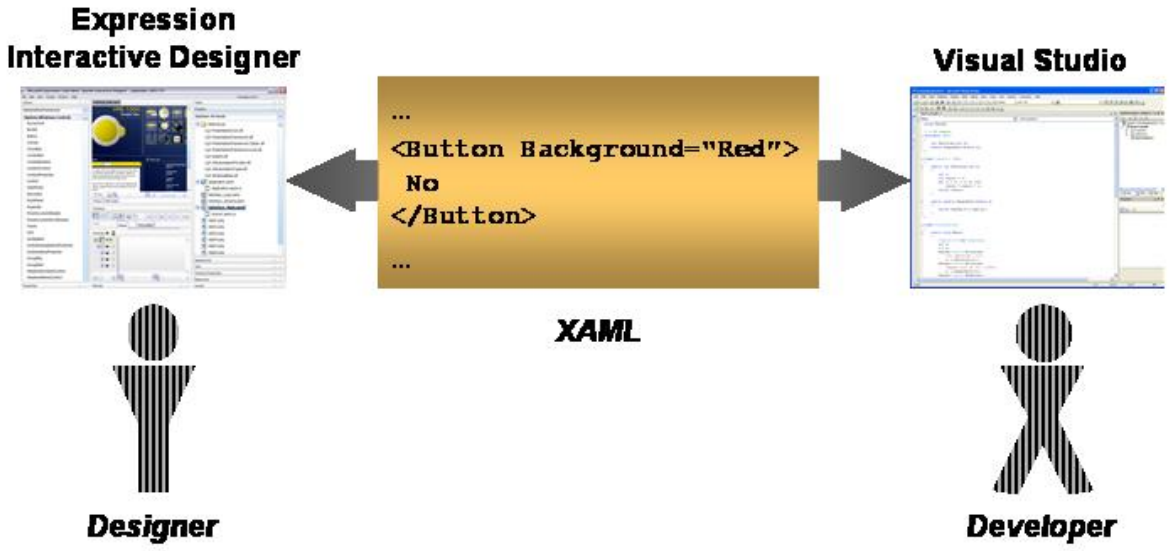

Рисунок 4.1 – Взаємодії дизайнера і розробника за технологіэю WPF

Інтерфейс користувача WPF-додатка задається мовою XAML. З мовою XAML може бути пов'язаний скомпільований code-behind код. Наприклад, якщо сторінка додатка описується файлом preved.xaml, то codebehind, як правило, буде зберігатися в preved.xaml.cs. Codebehind може містити реакцію на різні події, що генеруються інтерфейсом користувача (такі як натискання клавіші миші або «наїзд» покажчика миші на елемент управління). Однією з

цілей такого поділу є написання одного коду для всіх типів додатків (тобто, код можна було б скомпілювати і як настільний додаток, і як додаток, що переглядається за допомогою браузера, і як smart client-додаток).

Мова XAML підпорядковується всім правилам wellformed XML, зокрема, містить рівно одну кореневу вершину і є деревом. На вершині ієрархії знаходиться один з контейнерних об'єктів. Усередині цих об'єктів розташовуються елементи управління та інші контейнери [11].

Сучасний веб-додаток – це не просто сума складних компонентів на сторінці, це ще й їх взаємозв'язок. Основний розподіл елементів XAML такий: контейнери (панелі); елементи управління; служби документів (document services); графічні примітиви.

Розміщуються компоненти на основу дизайн-проекта для інтерфейсу користувача. Тому основним конфігураційним елементом є Grid, представити розміщення інших елементів можна за допомогою таблиці 4.1.

Таблиця 4.1 – Структура головної сторінки

DockPanel: Image	DockPanel: StackPanel: Image "OpenCage" Image "Hand" TextBox Image "Top"
TreeView: TreeView.ContextMenu GridSplitter	ListBox

Кожен пункт елемента ListBox являє собою складну структуру на основі Grid (2 рядки х 2стовбця), яка представлена в таблиці 4.2:

Таблиця 4.2 – Структура пункту елемента ListBox

Image	StackPanel: TextBlock TextBlock TextBlock	StackPanel: Label TextBlock
TextBlock	Expander: RichTextBox	Button: StackPanel: Image TextBlock

За допомогою панелей можливо розташовувати елементи, які містяться всередині них. Серед стандартних панелей є: Canvas – дочірні елементи розміщуються з використанням відносних координат); DockPanel – панель, в якій дочірні елементи стикуються; StackPanel – де елементи виводяться один під іншим; FlowPanel – де елементи виводяться в ряд один за одним; Grid – таблична організація синів. Елементи мови XAML складають логічне дерево, яке наведено на рисунку 4.2.

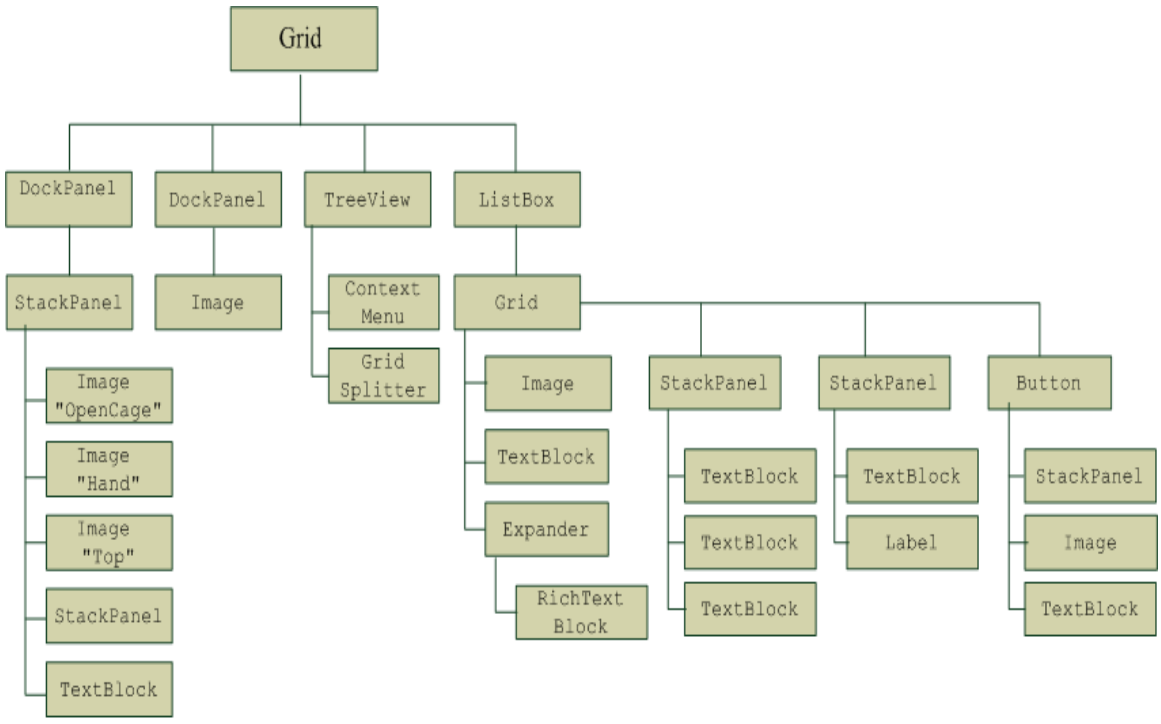

Рисунок 4.2 – Логічне дерево елементів XAML

Використання стилів і тригерів. Важливе питання – яким чином можна поліпшити зовнішній вигляд елементів. Тут XAML пропонує ідеї, схожі з ідеями CSS. Для початку вводиться поняття ресурсу: ресурс – це іменоване значення деякого елемента XAML:

```
     <Canvas
xmlns=”xmlns=”http://schemas.microsoft.com/2003/xaml”>
     <Canvas.Resources>
        <SolidColorBrush def:Name=”MyFavoriteColor”
Color=”Magenta”/>
     </Canvas.Resources>
     </Canvas>
```

Колекція ресурсів є у кожного елемента XAML. У випадку, якщо який-небудь елемент посилається на ресурс, але в ньому самому ресурси не визначені (або такий ресурс не знайдене), то пошук проводиться вгору по дереву вкладеності. Важливим типом ресурсу є стиль, який описується так:

```
     <Style def:Name=”MagentaButtonWithLargeFont”>
       <Button Background=”{MyFavoriteColor}” FontSize=”100”>
     </Style>
```

Прив'язка даних. Зв'язування даних не є чимось новим в сучасних середах розробки інтерфейсу користувача і давно існувало в різних UI-платформах. Це дозволяє зіставити візуальні елементи (контроли) інтерфейсу користувача і дані, які потрібно відобразити [12]. Кількість коду при цьому скорочується (рис.4.3).

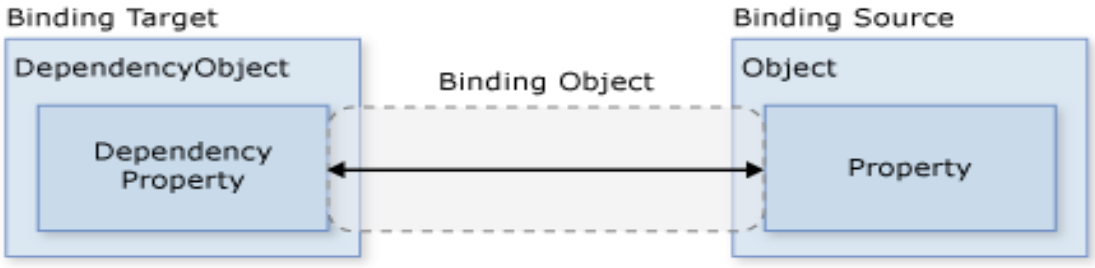

Рисунок 4.3 – Схема зіставлення візуальні елементів і даних

Інфраструктура, яка зв'язує, стежить за тим, щоб зміни в інтерфейсі приводили до змін даних, і навпаки. Зв'язування даних в WPF обертається навколо класу Binding. У цього класу досить простий і зрозумілий API [6].

Source – посилається на джерело даних. За замовчуванням посилання виробляється на значення властивості DataContext, яке може бути усадкована від батьківського елементу. Якщо встановити цю властивість в нульове значення, операція зв'язування даних буде сприймати це значення як місце обміну даними.

Path – використовується для того, щоб вказати, до якого властивості об'єкта-джерела приєднуватись. Тип цієї властивості PropertyPath, що дозволяє підтримувати складний діапазон виразів.

ElementName – може використовуватися, як альтернатива описаної раніше властивості Source. Дозволяє вказати ім'я елемента, який потрібно використовувати як джерело даних.

Converter – типу IValueConverter. Присвоївши цій властивості покажчик на екземпляр класу, який реалізує інтерфейс IValueConverter, отримаємо можливість перехоплювати будь-яке переміщення даних від джерела до одержувача, і навпаки. Конвертори значень дуже зручні і часто використовуються. Нижче наведено приклад заповнення об'єкта TreeView даними з використанням класу Binding.

```
     <TreeView FontSize="20"  SelectedItemChanged="SelectCateg"
Grid.Row="1"
          Grid.Column="0" Margin="1"Name="TreeList"
          ItemsSource="{Binding Path=TreeList}"
HorizontalAlignment="Stretch"
                Background="#80435467" BorderBrush="Wheat"
ForceCursor="True"
                Cursor="Hand" SnapsToDevicePixels="True">
          <TreeView.ContextMenu>
            <ContextMenu />
          </TreeView.ContextMenu>
          <TreeView.BitmapEffect>
             <BevelBitmapEffect EdgeProfile="CurvedOut" />
          </TreeView.BitmapEffect>
     </TreeView>
```

Об'єкт TreeList формується behind-кодом на основі технології ADO.NET доступу до бази даних. Виклик процедури обробки події здійснюється установкою відповідного параметра елемента XAML. Приклад завдання обробника події натискання кнопки «В кошик».

```
     <Button Grid.Column="2" Grid.Row="1" Click="AddToCage"
Name="ButToCage"
     Margin="15" Style="{StaticResource
butStyle}"HorizontalAlignment="Right"
     VerticalAlignment="Bottom"
HorizontalContentAlignment="Center"
     VerticalContentAlignment="Center" Width="auto" Height="auto"
     Cursor="Hand">
      </Button>
```

4.2 Реалізація функцій у веб-застосунку

Технологія WPF – це міст між C#-кодом і мовою XAML. Для обробки відповідної події в behind-коді формується процедура AddToCage, що реалізує обробку події. Наприклад, на вже зазначеної події натискання кнопки «В кошик» сформований наступний код:

```
private void AddToCage(object sender, RoutedEventArgs e)
        {
            Tbook bk = NazvBut.GetBook(sender,Books);
            if (!SelBooks.Contains(bk))
            {
                SelBooks.Add(bk);
                UpdateCage();
            }
        }
```

Для зручності маніпулювання даними про книги каталогу використовується глобальний клас Tbook. У цьому випадку доступ до даних про книгу не залежить від того, з якої сторінки або форми програми надійшов запит. Нижче наведено фрагмент C #-коду, що описує формування класу.

```
public class Tbook
{
 string name,autors,text,file_name,pict;
 double cost;
 int id,col;

public FlowDocument doc = new FlowDocument();
public string Name
  {
    set { name = value; }
    get { return name; }
  }
public string Pict
  {
    set { pict = value; }
    get { return pict; }
  }
public string File_name
  {
    set
    { file_name = value;doc = new FlowDocument();
      try{TextRange documentTextRange = new
TextRange(doc.ContentStart,
          doc.ContentEnd);
          using (FileStream fs = File.Open(File_name,
FileMode.Open))
          {
            documentTextRange.Load(fs, DataFormats.Rtf);
          }
        }
        catch { }
    }
    get { return "F:/book/files/"+file_name; }
 }
```

Поля класу заповнюються даними з бази даних. Заповнення «Кошика» закінчується процедурою формування замовлення. Замовлення зберігається в текстовому файлі на сервері. Нижче наведена процедура формування файлу замовлення.

```
private void Buy(object sender, RoutedEventArgs e)
{
  List<string> lns = new List<string>();
  double sum = 0;
  lns.Add("ФИО: " + name);
  lns.Add("Адрес: " + adress);
  lns.Add("E-Mail: " + mail);
  lns.Add("Телефон: " + phone);
```

```
        lns.Add("");
        lns.Add("-+-+-+-+-+-+-+-+-+-+-Зміст замовлення за  "
                + DateTime.Today.ToShortDateString().ToString() +
                +" -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-");
        lns.Add("");
        for (int q = 0; q < own.SelBooks.Count; q++)
        {
          lns.Add("Назва: " + own.SelBooks[q].Name);
          lns.Add("Id: " + own.SelBooks[q].Id);
      lns.Add("Сума(ціна/кіл-сть): "+own.SelBooks[q].Cost+" * " +
                  own.SelBooks[q].Col + " = " +
own.SelBooks[q].Col *
                  own.SelBooks[q].Cost);
          lns.Add("----------------------------------------");
          lns.Add("");
          sum += own.SelBooks[q].Col * own.SelBooks[q].Cost;
        }
        lns.Add("До сплати: "+sum);
        File.WriteAllLines("f:/qew.txt",lns.ToArray());
      }
```

Сформований файл замовлення передається по електронній пошті покупцю.

4.3 Реалізація доступу до даних

Для переміщення даних між їх постійним сховищем і додатком, в першу чергу необхідно створити з'єднання з джерелом даних (connection). В арсеналі ADO.NET для цих цілей є об'єкт SqlConnection – об'єкт, що дозволяє створити з'єднання з базами даних MS SQL Server версії 7.0 і вище.

Перша властивість, яку необхідно визначити для встановлення зв'язку з базою даних – ConnectionString. Це текстовий рядок, який містить набір елементів типу «атрибут = значення атрибута» і служить для визначення типу провайдера, імені джерела даних (DataSource), імені бази даних (Database), ідентифікатора користувача (UserId), пароля доступу (Password), системи безпеки, що використувана і т. п.

Перераховані атрибути задаються у властивості ConnectionString об'єкта, що забезпечує зв'язок з базою даних. ConnectionString можна створити за допомогою майстра конфігурації рядка підключення.

```
DataSource=.\SQLEXPRESS;
AttachDbFilename=F:\book\БД\Webbooks.mdf;
Integrated Security=True;
Connect Timeout=30;
User Instance=True
```

Рядок підключення зберігається у властивостях проекту. Доступ до рядка підключення з проекту:

```
SqlConnection conn = new
SqlConnection(Properties.Settings.Default.WebbooksConnectionStri
ng);
```

Об'єкти connection мають два базових методи для відкриття і закриття з'єднання (open і close). Метод open використовує інформацію з властивості ConnectionString, щоб звернутися до джерела даних і відкрити (встановити) зв'язок. Метод close закриває відкрите з'єднання. Закриття зв'язку додатка з базою даних є дуже важливою подією. У цей момент звільняються цінні системні ресурси, і база даних може обслуговувати нового користувача.

4.4 Виконання запитів до бази даних

Команда даних містить посилання на SQL-запит або збережену процедуру, які власне і реалізують конкретні дії. Команда даних – екземпляр класу OleDbCommand або класу SqlCommand.

Властивості об'єктів DataCommand містять всю інформацію, необхідну для виконання команд.

Connection. Посилання команди на об'єкт connection, який забезпечує встановлення зв'язку з базою даних. Ім'я або текст команди. Команда включає текст SQL-запиту або ім'я збереженої процедури.

Parameters. Команда може вимагати, щоб їй передали значення параметрів (вхідні параметри). Крім того, команда може мати і які повертаються параметри у вигляді одного або декількох значень. Кожна

команда має колекцію параметрів, які ви можете налаштувати індивідуально для передачі або отримання значень.

Для виконання команди даних потрібно викликати той метод, який відповідає результатами які повернені. Наприклад, для отримання набору записів, потрібно використовувати метод ExecuteReader, який повертає записи об'єкту DataReader.

Властивості об'єкта DataCommand. Name – ім'я, за допомогою якого можна звернутися до команди в програмному коді Connection. Ім'я об'єкта Connection, яке команда буде використовувати для зв'язку з базою даних. Можна вибрати ім'я існуючого з'єднання із списку.

CommandType – значення, що визначає тип виконуваної команди: Text – SQL-запит, StoredProcedure – збережена процедура.

TableDirect – отримання повного змісту таблиці. (Це значення доступно тільки для об'єктів OleDbCommand).

CommandText – текст (зміст) команди яка виконується. Значення даної властивості залежить від того, який тип команди був зазначений у властивості CommandType. Якщо Text, то вводиться текст SQL-запиту; якщо StoredProcedure, то вводиться ім'я збереженої процедури; якщо TableDirect, то вводиться ім'я таблиці.

Parameters Колекція об'єктів класу OleDbParameter або SqlParameter. Можливо передати дані команді, встановлюючи властивості індивідуальних параметрів. Повернений набір записів поміщається в об'єкт DataReader (OleDbDataReader АБО SqlDataReader). У циклі, шляхом перебору, можна отримати доступ до будь індивідуальної записи отриманого набору. Оскільки DataReader забезпечує доступ до даних у режимі forward-only, read-only, то це досить швидкий і ефективний спосіб отримання набору записів з бази даних.

Виконання команд, які повертають набір записів, відбувається наступним чином: створюється об'єкт datacommand; визначається у властивості об'єкта commandtext текст sql-запиту, який повертають набір записів; встановлюється у властивості commandtype значення commandtype.text (для

sql-запиту); якщо команда має параметри, то треба їх визначити; створюється об'єкт datareader як екземпляр класу sqldatareader; відкривається з'єднання, пов'язане з командою даних; викликається метод executereader, треба адресувати результати роботи команди об'єкту datareader; в циклі проглядаються отримані дані, використовуючи метод read; закривається об'єкт datareader. Якщо більше не передбачається використовувати метод Execute-Reader, то з'єднання закривається.

Нижче наведено приклад функції, що здійснює заповнення об'єктів класу TBook даними з бази даних. Щоб заповнити список книг Tbook, необхідно виконати запит, який поверне всі записи з таблиці list та пов'язаних з нею таблиць. Цю процедуру потрібно зробити один раз, при першому завантаженні Web-сторінки в браузер користувача. Після виконання запиту треба використовувати зчитувач даних (об'єкт SqlDataReader) для вибірки отриманих записів.

```
     public List<Tbook> LoadBooks(string cmd,string par,SqlDbType
type)
             {
                 List<Tbook> ll = new List<Tbook>();
                 SqlCommand red = new SqlCommand();
                 red.Connection = conn;
                 red.CommandType = CommandType.Text;
                 red.CommandText = cmd;
              red.Parameters.Add(new SqlParameter("@prm", type));
                 red.Parameters["@prm"].Value = par;
                 SqlDataReader rd = red.ExecuteReader();

     while (rd.Read())
                 {
                     try
                     {
                       ll.Add(new Tbook(rd["nazv"].ToString(),
     rd["autor"].ToString(),
     rd["comm"].ToString(),
     rd["file_name"].ToString(),
     rd["pict"].ToString(),
     double.Parse(rd["cost"].ToString()),
     int.Parse(rd["id"].ToString())));
                     }
                     catch
                     {
                         ll.Add(new Tbook("", "", "", "", "", 1,
```

```
        int.Parse(rd["id"].ToString())));

                    }
                }
                Prov(ll);
                rd.Close();
                return ll;
    }
    public void Prov(List<Tbook> lst)
            {
                for (int q = 0; q < lst.Count-1; q++)
                {
                    if (lst[q].Id == lst[q + 1].Id)
                    {
                 lst[q].Autors += " , " + lst[q + 1].Autors;
                        lst.RemoveAt(q + 1);
                        q--;
                    }
                }
    }
```

Для заповнення каталога необхідно здійснити визов функції:

```
    Books = LoadBooks("SELECT
n.nazv,n.id,n.comm,n.cost,n.file_name,n.pict, a.autor FROM" +
"(nazv_book as n inner join list as l on n.id=l.id_book)"+
    +"inner join authors as a on a.id=l.id_author");
```

Заповнення каталога обраними книгами здійснюється таким чином:

```
                   SqlDbType.Int));
                    mySqlCmd.Parameters["@prm"].Value = idrr[q];
                   SqlDataReader str = mySqlCmd.ExecuteReader();
                    while (str.Read())
                        it.Items.Add(str["nazv"].ToString());
                    str.Close();
                    TreeList.Items.Add(it);
                }
```

Щоб отримати запис який вимагає користувач, необхідно виконати SQL-запит з WHERE, який повинен повернути з таблиці обраний користувачем запис. Щоб передати в запит ідентифікатор категорії, вибраний користувачем, необхідно присвоїти значення параметру запиту. Іншими словами, необхідно визначити значення параметра (об'єкт SqlParameter), який є частиною команди даних.

4.5 Програмна реалізація загальних функції інтерфейсу веб-застосунка

Головна сторінка веб-застосунку книжкового каталогу з застосуванням технології WPF наведена на рисунку 4.4.

Рисунок 4.4 – Головна сторінка веб-застосунку книжкового каталогу

У верхній частині сторінки розміщено логотип веб-застосунку, у центральній частині розташований основний інформаційний блок (контент) веб-застосунку. На головній сторінки представлене функціональне горизонтальне меню де можна переглянути дані про книжковий каталог, замовлення у кошику, всю необхідну контактну інформацію. Головна функція головної сторінки – не бути перевантаженою додатковою інформацією та мати зручний інтерфейс навігації: дерево вибору, кнопки переходу до додаткових функцій.

Ліва панель меню містить елемент TreeView, тобто дерево вибору. Розділи дерева вибору можна розкрити. Дані заповнення дерева знаходяться в базі даних системи (рис. 4.5).

Рисунок 4.5 – Пункти меню веб-застосунку книжкового каталогу

У будь-якому веб-застосунку, тим більше, в тім, що міститься безліч однорідної інформації, організація сортування по розділу – необхідна функція. Це зручно для користувача, якому зручно здійснювати пошук потрібної книги за обраною тематикою. Якщо товар присутні у цьому розділі у списку, про нього виводиться вся доступна інформація.

На рисунку 4.5 відображені системою всі доступні книги після виконання сортування по розділу. Ця функція можлива, якщо задати конкретну гілку дерева вибору.

Після виконання сортування за назвою системою будуть відображені всі доступні книги. Додаткова інформація про книгу з'являється на цій же сторінці за допомогою компонента Expander – кнопка «Детальніше».

Для здійснення покупки книги треба натиснути кнопку «Кошик». Для редагування кошику створена додаткова форма. Організована можливість паралельної роботи з обома формами, тобто результати вибору книг в каталозі синхронно відображаються в купівельному кошику (рис. 4.6).

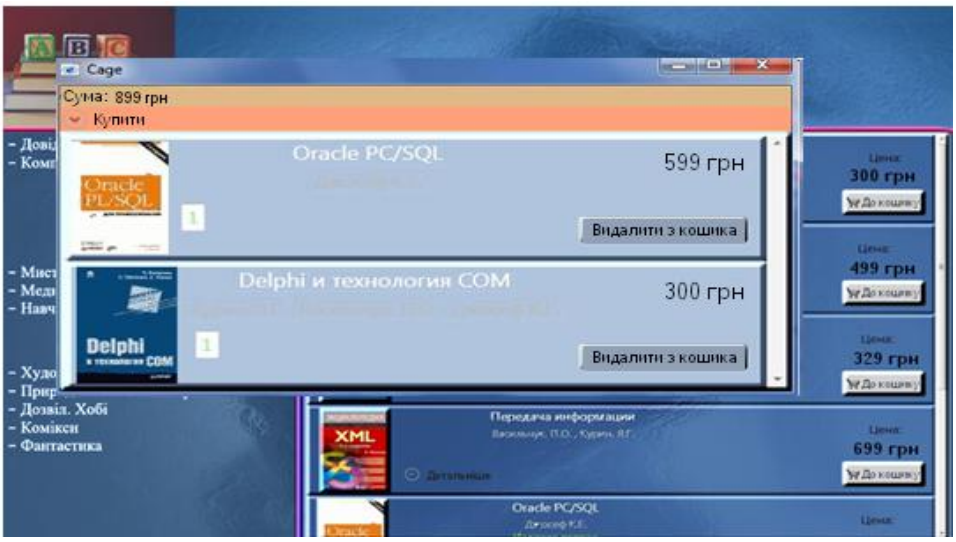

Рисунок 4.6 – Сторінка роботи з Кошиком користувача-покупця

У веб-застосунку книжкового каталогу реалізована можливість управління кількістю книг в кошику (додавання, видалення) (рис.4.7).

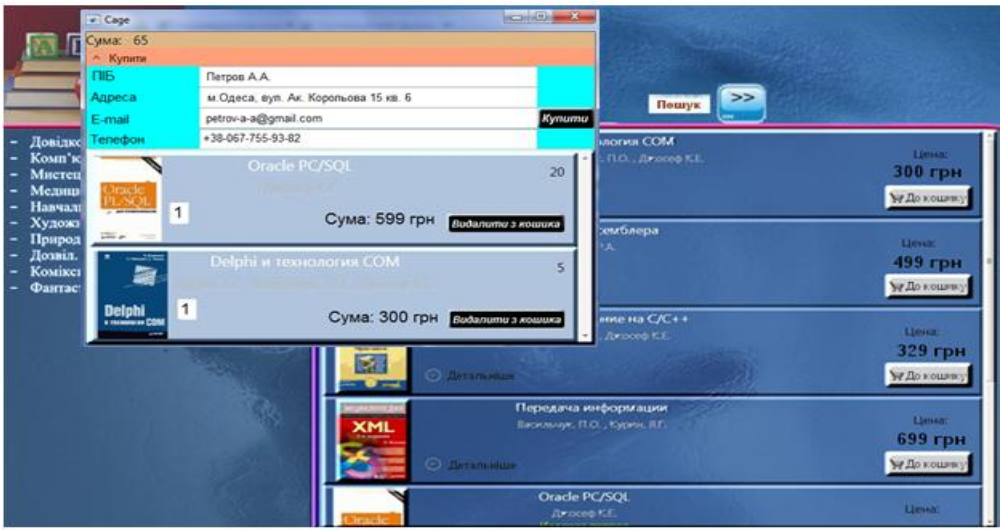

Рисунок 4.7 –Форма формування замовлення

Користувач має можливість підтвердити свій вибір і задати необхідні особисті дані для формування замовлення. Особисті дані користувача для подальшого оформлення замовлення будуть поміщені в текстовий файл, на підставі якого формується електронний лист, що містить платіжний документ. Подальшу взаємодію фірми, що реалізує книги, і покупця здійснюється засобами електронної пошти (рис. 4.8).

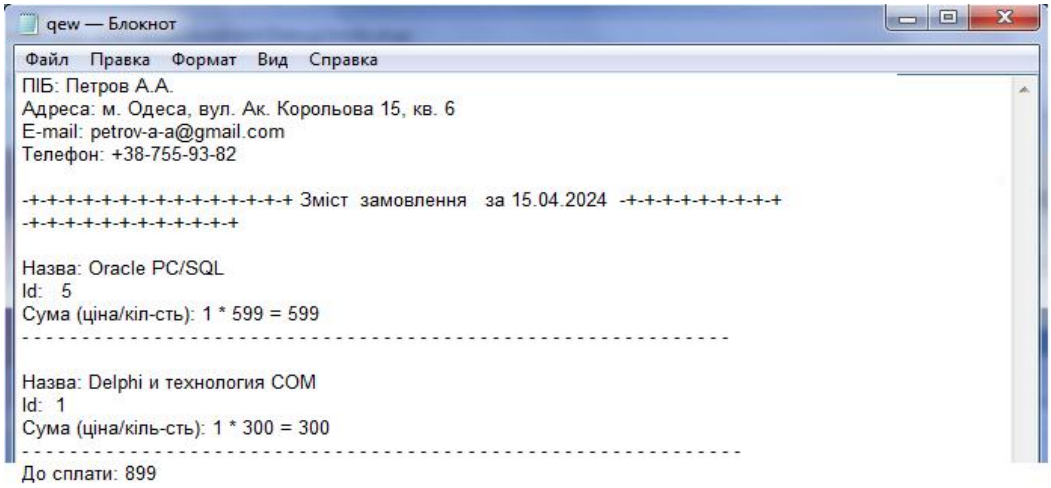

Рисунок 4.8 – Вміст файлу замовлення

Для певного класу Web-застосунків інтерфейс адміністратора є обов'язковою частиною інфраструктури. Інтерфейс адміністратора, читаючи метадані з моделі, надає потужний і повністю готовий інтерфейс. Користувачу-адміністратору для входу в адміністраторську панель, необхідно буде ввести унікальний логін і пароль (рис. 4.9).

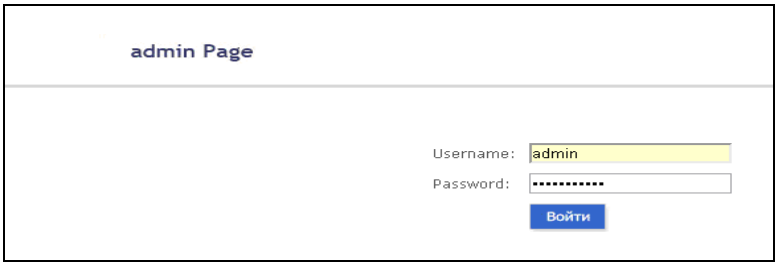

Рисунок 4.9 – Сторінка ідентифікації для Користувача-адміністратора

Якщо логін і пароль введений правильно, то адміністратору дозволяється увійти в веб-застосунку книжкового каталогу. У правій частині вікна знаходиться меню, в якому відображаються: основні сторінки веб-застосунку, інформери, основні категорії та під категорії, замовлення і користувачі. При перегляді основних сторінок системи: "Каталог», «Кошик», «О нас» адміністратору надається можливість переглянути, змінити, видалити, додати або сортувати дані сторінки. Якщо адміністратору необхідно створити нову сторінку, тоді потрібно натиснути на кнопку «Додати сторінку» (рис. 4.10).

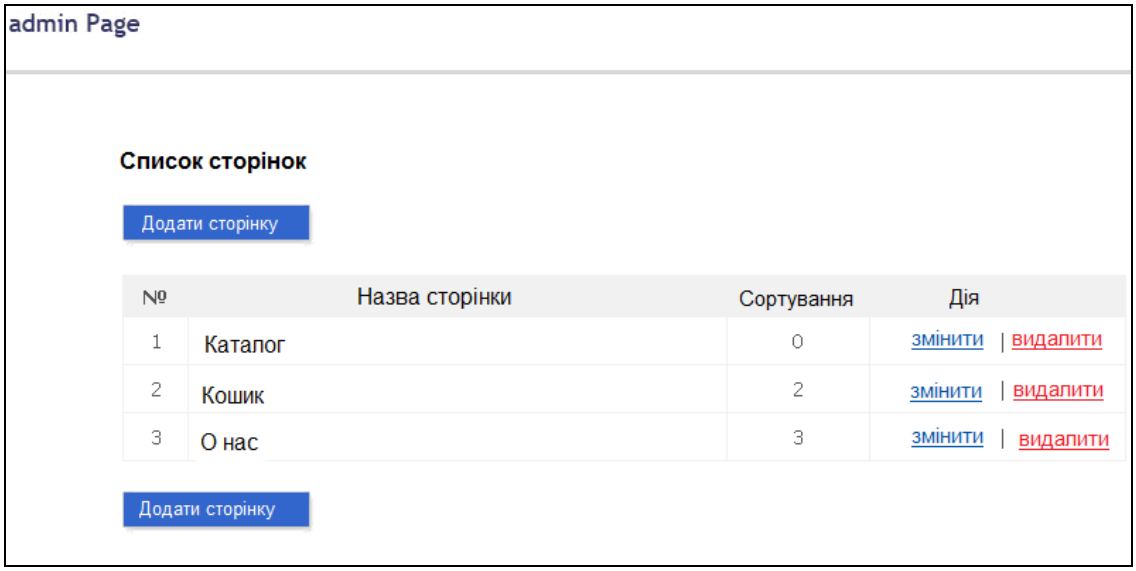

Рисунок 4.10 –Сторінка в адміністраторській частини веб-застосунку

Частина інтерфейсу, що відповідає за роботу з користувачами, дозволяє адміністратору переглянути всіх зареєстрованих користувачів в системі, їх логін, e-mail, яку роль виконують у системі.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра здійснено розробку веб-застосунку книжкового каталогу з застосуванням технології WPF, який призначений для розповсюдження книжкової продукції засобами мережі Інтернет, що дозволяє забезпечити присутність та здійснення комерційної діяльності книжкового каталогу з можливістю пошуку, легкої навігації по асортименту та швидкого оформлення замовлень у мережі Інтернет.

Створений веб-застосунок букіністичної компанії дозволяє клієнтам отримувати інформацію про комерційній пропозиції книжкового асортименту компанії: ознайомитися з книжковим каталогом запропонованих книг, здійснити пошук необхідної книги, отримати вичерпну та достовірну інформацію про обрану книгу, а так само дозволяє здійснювати покупку обраного товару.

В ході дипломного проектування здійснено розробку та програмну реалізацію веб-застосунку книжкового каталогу з застосуванням технології WPF: визначена архітектура, детально розглянуто і обґрунтовано застосування технології WPF для реалізації веб-застосунку, здійснено вибір програмних засобів реалізації, розроблена база даних, проведено проектування за стосунку; здійснено програмну реалізацію. В дипломній роботі було проаналізовано та використано основні можливості та переваги технології WPF щодо створення веб-застосунків, що дозволило автоматизувати процеси і забезпечити структуроване зберігання, обробку та пошук даних з використанням сучасної технології WPF.

Розроблений веб-застосунок книжкового каталогу забезпечить присутність книгорозповсюджувальної компанії в мережі Інтернет та надання інформації про її товари якомога більшому числу потенційних партнерів і споживачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Афонін О.В., Сенченко М. І. Українська книга в контексті світового книговидання. – Київ: Книжкова палата України, 2009. – 277 с.
2. Краус К.М., Краус Н.М., Манжура О.В. Електронна комерція та Інтернет-торгівля: навчально-методичний посібник. – Київ: Аграр Медіа Груп, 2021. – 454 с.
3. Низовий М.А. Вступ до книгознавства: Навчальний посібник – К.: Кондор, 2009. – 144 с.
4. Алісса Вокер. Підручник WPF для початківців: Як створити програму [Приклад]. URL: https://www.guru99.com/uk/wpf-tutorial.html#quick-guide (дата звернення: 25.04.2024)
5. Chris Sells, Ian Griffiths. Programming WPF 2nd Edition. O'REILLY, 2007. – 871 p.
6. Joydip Kanjilal. ASP.NET Web API: Build RESTful web applications and services on the .NET framework. – Packt Publishing, 2013. – 224 p.
7. Adam Freeman. Pro ASP.NET Core 7, Tenth Edition 10th ed. Edition. Manning Publications Co., 2023. – 1256 p.
8. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд). : Львівська політехніка, 2018. – 248 с.
9. Лосєв М.Ю., Федько В.В. Бази даних: навчально-практичний посібник для самостійної роботи студентів. – Харків: ХНЕУ ім. С.Кузнеця, 2018. – 233 с.
10. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань: навчальний посібник. «Магнолія-2006», 2013. – 680с.
11. Mark J. Price. C# 11 and .NET 7. Modern Cross-Platform Development Fundamentals. – Packt Publishing, 2022. – 818 p.
12. Голуб Б.М. С#. Концепція та синтаксис. Навч. посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2006. – 136 с.