Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнес-технологій «УАБС»
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему «ПРОГРАМНА РЕАЛІЗАЦІЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ РОЗДРІБНОГО БІЗНЕСУ»

Виконала студентка 2 курсу, групи Ек.м.-71а
(номер курсу) (шифр групи)
Спеціальності 051 «Економіка» («Економічна кібернетика»)
Вовк В.В .
(прізвище, ініціали студента)
Керівник професор, д.е.н. Олійник В.М. .
(посада, науковий ступінь, прізвище, ініціали)

Суми – 2018 рік

РЕФЕРАТ

кваліфікаційної магістерської роботи на тему

«ПРОГРАМНА РЕАЛІЗАЦІЯ ЕЛЕКТРОННОЇ КОМЕРЦІЇ РОЗДРІБНОГО БІЗНЕСУ»

студентки Вовк Валерії Володимирівни

На сьогоднішній день сфера електронної торгівлі знаходиться у стані активного розвитку. Як в Україні, так і в усьому світі постійно зростає кількість Інтернет- користувачів. Поряд з цим, збільшується використання безготівкових розрахунків, а витрати на відкриття Інтерне-магазину значно менші, ніж звичайного. До того ж, підприємства, що все мають магазини, все частіше звертаються до засобів електронної комерції, що відкривають нові шляхи розвитку. Саме тому, питання використання засобів електронної комерції в роздрібному бізнесі є актуальним.

Метаю даної кваліфікаційної магістерської роботи є створення системи для автоматизації діяльності підприємства роздрібного бізнесу в сфері електронної комерції. Це передбачає створення веб-додатку та реляційної бази даних. Користувачі будуть взаємодіяти із базою даних шляхом взаємодії із веб-інтерфейсом розробленої системи.

Об’єкт визначається, як діяльність підприємства роздрібної торгівлі в сфері електронної комерції.

Предметом є сучасні засоби та методології автоматизації діяльності підприємства в сфері електронної комерції.

У відповідності до завдань даної роботи було здійснено: дослідження суті поставленої задачі та предметної області, проаналізовано існуючі інформаційні системи, досліджено бізнес-процеси підприємства, сформовано вимоги до інформаційної системи, надано функціональну характеристику системи, визначено економічний ефект від впровадження системи.

Результатом даної роботи є створений прототип автоматизованої системи, що містить в собі всі необхідні функціональні модулі та ергономічний інтерфейс. Даний прототип рекомендується використовувати на малих та середніх підприємствах.

Апробація кваліфікаційної магістерської роботи здійснена на III Всеукраїнській науково-практичній on-line конференції «Сучасні тенденції в економіці та управлінні» 22-23 листопада 2018 року, а в рамках публікації в електронному науковому фаховому виданні Мукачівського державного університету «Економіка та суспільство» у 2018 роців, випуск № 14.

Ключові слова: електронна комерція, автоматизація, технологія JNDI, веб-сайт, Інтернет-магазин, Weblogic, Oracle xe, Java, ефективність, бізнес-процес, товар.

Основний зміст кваліфікаційної магістерської роботи викладено на 82 сторінках, зокрема список використаних джерел із 71 найменуваня, розміщених на 6 сторінках. Робота містить 32 таблиці, 43 рисунка, а також 4 додатка, розміщених на 97 сторінках.

Рік виконання кваліфікаційної роботи – 2018 рік

Рік захисту роботи – 2018 р

Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнес-технологій «УАБС»
Кафедра економічної кібернетики

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
(науковий ступінь, вчене звання)

_____ _____
(підпис) (ініціали, прізвище)
"___" _____20__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ
(спеціальність 051 «Економіка» («Економічна кібернетика»))
студенту 2 курсу, групи Ек.м.-71а
(номер курсу) (шифр групи)

Вовк Валерії Володимирівні .
(прізвище, ім'я, по батькові студента)

1. Тема роботи «Програмна реалізація електронної комерції роздрібного бізнесу» .
затверджена наказом по університету від «____» _____ 20___ року №____
2. Термін подання студентом закінченої роботи «____» _____ 20___ року
3. Мета кваліфікаційної роботи створення системи для автоматизації діяльності підприємства роздрібного бізнесу в сфері електронної комерції .
4. Об'єкт дослідження діяльність підприємства роздрібної торгівлі в сфері електронної комерції .
5. Предмет дослідження сучасні засоби та методології автоматизації діяльності підприємства в сфері електронної комерції .
6. Кваліфікаційна робота виконується на матеріалах .

7. Орієнтовний план кваліфікаційної роботи, терміни подання розділів керівникові та зміст завдань для виконання поставленої мети

Розділ 1 дослідження стану об'єкта автоматизації та формування вимог до системи .

.
(назва – термін подання)

У розділі 1 дослідити предметну область та суть поставленої задачі та проаналізувати існуючі інформаційні системи .
(зміст конкретних завдань до розділу, які повинен виконати студент

Розділ 2 розробка проекту автоматизованої системи .

.

У розділі 2 дослідити бізнес-процесі підприємства,. сформувати вимоги до інформаційної системи та надати функціональну характеристику системи .

(зміст конкретних завдань до розділу, які повинен виконати студент)

Розділ 3 реалізація прототипу автоматизованої системи на прикладі книжкового Інтернет-магазину

(назва – термін подання)

У розділі 3 розробити прототип системи, розробити та реалізувати систему автоматизації, визначити економічний ефект впровадження системи .

(зміст конкретних завдань до розділу, які повинен виконати студент)

8. Консультації з роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

9. Дата видачі завдання: «___»_____ 20__ року

Керівник кваліфікаційної роботи _____ Олійник В.М.
(підпис) (ініціали, прізвище)

Завдання до виконання одержав _____ Вовк В.В. .
(підпис) (ініціали, прізвище)

ЗМІСТ

ВСТУП

В жорсткій конкуренції підприємств роздрібної торгівлі важливим завданням є вчасне інформування покупців про свої товари чи послуги. Своєчасне донесення інформації є запорукою успішного продажу. Другим важливим питанням є розширення ринків збуту та організація доступу до послуг підприємства в режимі реального часу. Десять років тому, це здавалося неможливим, але сучасні телекомунікаційні технології дозволяють розширювати можливості функціонування підприємств.

Традиційна фізична торгівля товарами стає менш популярною та не ефективною. Лише в комбінуванні із застосуванням засобів електронної комерції, підприємство набуває нових можливостей розвитку та масштабування своєї діяльності.

На сьогоднішній день сфера електронної торгівлі знаходиться у стані активного розвитку. Як в Україні, так і в усьому світі постійно зростає кількість Інтернет- користувачів. Поряд з цим, збільшується використання безготівкових розрахунків, а витрати на відкриття Інтерне-магазину значно менші, ніж звичайного. До того ж, підприємства, що все мають магазини, все частіше звертаються до засобів електронної комерції, що відкривають нові шляхи розвитку. Саме тому, питання використання засобів електронної комерції в роздрібному бізнесі є актуальним.

Метаю даної кваліфікаційної магістерської роботи є створення системи для автоматизації діяльності підприємства роздрібного бізнесу в сфері електронної комерції. Це передбачає створення веб-додатку та реляційної бази даних. Користувачі будуть взаємодіяти із базою даних шляхом взаємодії із веб-інтерфейсом розробленої системи.

Об’єкт визначається, як діяльність підприємства роздрібної торгівлі в сфері електронної комерції.

Предметом є сучасні засоби та методології автоматизації діяльності підприємства в сфері електронної комерції.

На основі мети було сформовано перелік завдань:

- дослідити предметну область та суть поставленої задачі;
- проаналізувати існуючі інформаційні системи;
- дослідити бізнес-процесі підприємства;
- сформувати вимоги до інформаційної системи;
- надати функціональну характеристику системи;
- розробити прототип системи;
- розробити та реалізувати систему автоматизації;
- визначити економічний ефект впровадження системи.

Апробація кваліфікаційної магістерської роботи здійснена на III Всеукраїнській науково-практичній on-line конференції «Сучасні тенденції в економіці та управлінні» 22-23 листопада 2018 року, а в рамках публікації в електронному науковому фаховому виданні Мукачівського державного університету «Економіка та суспільство» у 2018 роців, випуск № 14.

1 ДОСЛІДЖЕННЯ СТАНУ ОБ'ЄКТА АВТОМАТИЗАЦІЇ ТА ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

1.1 Характеристика об'єкта автоматизації: основні поняття та перспективи розвитку електронної комерції

Електронна комерція – це будь-яка транзакція, яка здійснюється через комп'ютерну мережу, внаслідок якої право власності або право використання товаром або послугою було передано від однієї особи до іншої. Електронний бізнес – це будь-який процес, який будь-яка організація проводить за допомогою мережі пов'язаних між собою терміналів (комп'ютерів, телефонів). З огляду визначень, можна стверджувати, що електрона комерція існує давно і, вже на сьогоднішній день, стрімко розвивається. [1 – 3].

Розвиток електронної комерції йде поряд із розвитком інформаційних технологій, що почали розвиватися із середини 20 століття. Етапи еволюції інформаційних технологій та їх характеристика за ознаками розподілу наведена у таблиці 1.1.

Таблиця 1.1 – Характеристика етапів еволюції інформаційних технологій за ознаками розподілу [3, 4 – 7]

№ п/п	Ознака розподілу	Етап	Період	Характеристика періоду
1	По виду завдань і процесів обробки інформації	1-й	60 – 70 рр.	Автоматизація рутинних дій людини, як основний напрямок розвитку. Відбувається опрацювання даних в обчислювальних центрах. Робота з ЕОМ є колективною.
		2-й	з 80-х рр.	Розробка інформаційних технологій направлена на вирішення стратегічних завдань.

Продовження таблиці 1.1

№ п/п	Ознака розподілу	Етап	Період	Характеристика періоду
2	По проблемам, які вирішують інформаційні технології	1-й	до кінця 60-х рр.	Спостерігаються обмеження можливостей апаратних засобів та необхідність опрацювання великих обсягів даних.
		2-й	до кінця 70-х рр.	Характеризується відставанням рівня програмного забезпечення від розвитку апаратних засобів.
		3-й	до кінця 80-х рр.	Основною проблемою стає задоволення потреб користувача та розробка інтерфейсу для роботи у комп'ютерному середовищі. На даному етапі з'являється персональний комп'ютер.
		4-й	з початку 90-х рр.	Характеризується активним створенням сучасних інформаційних систем, технологій та міжорганізаційних зв'язків. Спостерігається поява численних проблем, зокрема слід виділити проблеми встановлення стандартів і протоколів комп'ютерного зв'язку, забезпечення надійного захисту інформації та організація доступу до інформації.
3	По перевагам, які надають комп'ютерні технології	1-й	з початку 60-х рр.	Орієнтація роботи обчислювальних центрів на централізоване колективне використання ресурсів при опрацюванні даних шляхом виконання рутинних операцій. Критерій оцінки ефективності інформаційних систем розраховувався як різниця між витраченими на розробку і зекономленими в результаті впровадження коштами. Не зважаючи на достатньо великі можливості обчислювальних систем, вони використовувалися не повною мірою, через погану взаємодію користувачів та розходження поглядів розробників на розуміння розв'язуваних проблем.
		2-й	з середини 70-х рр.	Відбуваються зміни у підході до створення інформаційних систем через появу персонального комп'ютера. Подальші інформаційні системи орієнтуються на підтримку прийняття рішень індивідуального користувача.
		3-й	з початку 90-х рр	Орієнтація роботи інформаційних систем на стратегічному аналізі переваг бізнесу та базується на основі телекомунікаційних технологіях розподіленої обробки даних.

Характеристика етапів еволюції електронної комерції наведена у таблиці 1.2 [7 – 8].

Таблиця 1.2 – Характеристика етапів еволюції електронної комерції

№ п/п	Назва періоду	Період	Характеристика
1	Період становлення	50 - 60-ті рр.	Характеризується впровадженням додатків «Mainframe-based», що застосовувалися для транспортної системи: замовлення квитків та обмін даними між різними службами підготовки рейсів. Подальші розробки проводилися у напрямку автоматизації процедури резервування місць для авіакомпаній. Таким чином, починають з'являтися системи SABRE - Semi – Automatic Business Research Environment. Автоматизація процесу розрахунків тарифів при резервуванні місць зменшила вартість послуг.
2	Період фрагментаризації	70-ті рр.	Характеризується створенням спеціального комітету TDCC - Transportation Data Coordination Committee, для вирішення питань узгодження чотирьох індустріальних стандартів для різних транспортних систем: залізничної, автомобільної та авіаційної. Результати діяльності комітету стали основою для нового стандарту організації електронного обміну даними між різними організаціями - EDI - Electronic Data Interchange. Створення стандарту ANSI X.12 (host - based) та набору стандартів для міжнародної торгівлі - Tradacoms. Європейська економічна комісія – UNECE (United Nations Economic Commission for Europe) приймає даний набір стандартів як міжнародні стандарти GTDI (General – purpose Trade Data Interchange standards).
3	Період розвитку	80 - 90-ті рр.	Створення стандарту EDIFACT (Electronic Data Interchange for Administration, Commerceand Transport), що ґрунтувався на стандартах GTDI та був прийнятий ISO як ISO 9735. Як транспортне середовище EDIFACT використовує стандарт електронної пошти X.400 (підмножина X.435).
4	Період кастомізації	1990 - 1995 рр.	Поява більш актуальної проблеми електронної комерції – можливості проведення операцій через мережу Інтернет. З появою web-технологій з'являється новий тип бізнесу – роздрібна торгівля через мережу Інтернет. Створення стандарту EDIINT (EDIFACTover Internet) на базі стандартів для електронної пошти Інтернет – SMTP/S – MIME.

Продовження таблиці 1.2

№ п/п	Назва періоду	Період	Характеристика
5	Період стандартизації	1996 – 2000 рр.	Проведення низки тестів перевірки сумісності різних комерційних реалізацій EDIINT співтовариством Commerce Net (організація по сприянню розвитку бізнесу в мережі Інтернет. При перевірці виявлена велика кількість недоліків. У 1997 році з'являється новий стандарт – OBI (OpenBuyingonthe Internet). В даному стандарті закріплені принципи, яким повинене відповідати програмне забезпечення електронної комерції з підтримкою Інтернет – стандартів.
6	Період мультиатрибутивності та конвергенції	з 2000-х рр.	Характеризується комбінування елементів електронної та традиційною торгівлі; використанням всіма суб'єктами взаємодії усіх можливостей електронної комерції.

Розвиток Інтернету спонукав за собою зниження вартості обміну інформації і, як результат, зниження витрат на використання засобів електронної комерції.

Усе це стимулювало подальший розвиток технологій і появу вже відомих на сьогоднішній день видів електронної комерції: «бізнес - бізнес» (B2B), – «бізнес - споживач» (B2C), «споживач - бізнес» (C2B), «споживач - споживач»(C2C), «мобільна комерція» (M-Commerce) [9] . Детальна характеристика видів електронної комерції за рівнями призначення наведена у таблиці 1.3 [9 – 12].

Таблиця 1.3 – Класифікація електронної комерції

№ п/п	Вид електронної комерції	Характеристика
1	B2B – «бізнес - бізнес»	Тип комерційної транзакції, що існує між підприємствами або транзакцією, яка відбувається між компанією та іншою компанією для передачі послуг та продуктів.
2	B2C– «бізнес - споживач»	Тип комерційної транзакції між бізнесом та його кінцевим споживачем. Характеризується створенням електронних вітрин.

Продовження таблиці 1.3

№ п/п	Вид електронної комерції	Характеристика
3	C2B – «споживач - бізнес»	Передача послуг, товарів або інформації від людей до бізнесу; бізнес-модель, де кінцеві користувачі створюють товари та послуги, що використовуються бізнесом та установами.
4	C2C – «споживач - споживач»	Характеризується операціями між користувачами; є бізнес-моделлю, за якою двоє споживачів безпосередньо взаємодіють між собою.
5	M-Commerce – «мобільна комерція»	Передбачає «купівлю та продаж продуктів, інформації та послуг» за допомогою бездротових портативних пристроїв, таких як стільникові телефони, ноутбуки та персональні цифрові помічники.

В умовах роздрібного бізнесу одна з переваг застосування електронної комерції може стати проблемою для іншого виробника. Тому необхідно класифікувати переваги та недоліки засобів електронної комерції із сторін споживача, підприємства та держави. Детальна класифікація наведена у таблицях 1.4 – 1.5 [13 – 22].

Таблиця 1.4 – Класифікація переваг електронної комерції

Суб'єкт, для якого розглядаються переваги	№ п/п	Назва переваги	Характеристика
Споживач	1	Зменшення цін на продукцію	Використовуючи засоби електронної комерції підприємство зменшує невиробничі витрати і завдяки цьому може оптимізувати цінову політику.
	2	Доступність інформації	Завдяки роботі Інтернет – магазинів користувачі отримують доступ до необхідної інформації про товари цілодобово.
	3	Вивчення ринків товарів та послуг	Завдяки відкритому доступу до Інтернет – магазинів, споживачі отримують можливість порівнювати ціни та інформацію на різних ресурсах. Завдяки цьому, продавці не можуть необґрунтовано завищувати ціни, бо споживачі з легкістю зможуть це зрозуміти і замовити аналогічний товар у конкурентів.

Продовження таблиці 1.4

Суб'єкт, для якого розглядаються переваги	№ п/п	Назва переваги	Характеристика
Споживач	4	Доступ до зарубіжних ринків	Мережа Інтернет надає споживачам можливість користуватися іноземними ресурсами та замовляти іноземні товари, приймати участь в аукціонах та замовляти рідкісні товари.
	5	Конфіденційність покупок	Споживачі можуть конфіденційно здійснювати покупки.
	6	Перегляд відгуків	Споживачі мають можливість переглядати відгуки інших споживачів та залишати свої, що надає можливість попередньо оцінити товар перед його покупкою.
	7	Цифрові покупки	Споживачі отримують можливість замовляти нематеріальні товари, які одразу доставляються споживачу через мережеві канали зв'язку.
Підприємство (виробник)	1	Зниження первісних вкладень та нематеріальних витрат	Первісні вкладення зменшуються за рахунок того, що для початку роботи не обов'язково орендувати торгівельний майданчик. Підприємство може обмежитись створення веб-сторінки. Також, зменшуються невиробничі витрати пов'язані з сервісним обслуговуванням чи рекламними витратами.
	2	Зменшення чисельності персоналу	Завдяки процесам автоматизації, підприємство має змогу зменшити чисельність працівників та відповідно фонд оплати праці.
	3	Зручність проведення маркетингових досліджень	Підприємці отримують змогу проводити маркетингові дослідження у режимі Інтернет, зокрема використовувати сервіси Google Analitycs, Customer Relationship Management,, тощо.
	4	Розширення ринків	Підприємці отримують доступ до нових сегментів ринку та можуть розширювати асортимент товарів та послуг.
	5	Рівність доступу	Підприємці отримують рівний доступ до ринків, не зважаючи на розміри підприємства.
	6	Цілодобовий доступ	Завдяки Інтернет мережі, споживачі отримують доступ до веб-сторінки підприємства цілодобово.
	7	Підвищення довіри до своєї марки	Завдяки маркетинговим та рекламним засобам на веб-сторінці виробника, підприємство може збільшувати прихильність споживачів.
Держава (суспільство)	1	Широкий асортимент продукції	Завдяки Інтернет мережі та засобам електронної комерції споживачам надається широкий асортимент найрізноманітніших товарів та послуг.

Продовження таблиці 1.4

Суб'єкт, для якого розглядаються переваги	№ п/п	Назва переваги	Характеристика
Держава (суспільство)	2	Нарощення потужності національної економіки	Завдяки Інтернет мережі та засобам електронної комерції збільшується обсяг надходження інвестицій, відбувається розвиток науки та техніки, підвищується рівень життя населення, зменшується «цифровий розрив».
	3	Підвищення рівня національної безпеки	Завдяки сучасним засобам захисту підвищується рівень національної безпеки.

Таблиця 1.5 – Класифікація недоліків електронної комерції

Суб'єкт, для якого розглядаються недоліки	№ п/п	Назва недоліку	Характеристика
Споживач	1	Контроль за споживачем	Можна здійснювати контроль за споживачем через засоби ідентифікації особи.
	2	Поширення мережі Інтернет	В країні все ще існує сегмент населення, що не має доступу до мережі Інтенет.
	3	Необхідність передплати	Частіше за все споживачі повинні сплачувати часткову або повну вартість товару в момент його замовлення.
	4	Оцінка якості	В повній мірі, споживачі не мають змоги пересвідчитися у якості товару до його отримання.
	5	Заплутаність мережі Інтернет	Мережа Інтернет є громіздкою та заплутаною, що в деяких випадках ускладнює роздрібний бізнес в мережі Інтернет.
Підприємство (виробник)	1	Посилення конкуренції	Посилення конкуренції та перехід її на глобальний рівень.
	2	Потужна технічна та технологія основа	Робота в цілодобовому режимі вимагає надійної технічної та технологічної бази та постійних фінансових вкладень в обновлення системи.
	3	Ціноутворення	Завдяки відкритому доступі до інформації, підприємство вже не може істотно завищувати ціни.
	4	Розробка онлайн-вітрин	Більш складна організація роботи онлайн – вітрин у порівнянні зі звичайним магазином.

Продовження таблиці 1.5

Суб'єкт, для якого розглядаються недоліки	№ п/п	Назва недоліку	Характеристика
Підприємство (виробник)	5	Захист інформації	Завдяки відкритості інформація можливі випадки порушення прав інтелектуальної власності, плагіату. Також, необхідно забезпечувати надійний захист веб-ресурсу від вірусів та хакерських атак.
	6	Додаткові працівники	Для функціонування веб-ресурсу необхідно наймати та залучати спеціалістів, що будуть здійснювати адміністрування та оновлення контенту веб-ресурса.
	7	Ускладнення бухгалтерського обліку	Переорієнтація традиційних паперових носіїв в електроні вимагає залучення трудових та фінансових ресурсів, також із застосуванням засобів електронної комерції ускладняється і бухгалтерського облік.
Держава (суспільство)	1	Нерівномірність розвитку	Спостерігається нерівномірний розвиток засобів електронної комерції у різних регіонах та галузях виробництва.
	2	Ускладнення товарно-грошових відносин	Завдяки засобам електронної комерції ускладняються товарно – грошові відносини в країні. Спостерігаються сприятливі умови для здійснення протиправної діяльності та ухилення від оподаткування зі сторони недобросовісних громадян.
	3	Збільшення безробіття	Електронна комерція як створює нові робочі місця, так і зменшує їх внаслідок процесів автоматизації, що може збільшувати рівень безробіття в країні.

Ми визначили переваги та недоліки застосування електронної комерції, але треба розглянути питання доцільності її застосування.

В першу чергу, необхідно з'ясувати чи дійсно люди активно користуються мережею Інтернет та з якою метою. На рис. 1.1 – 1.2. [23 – 27] бачимо, що відсоток людей в Україні, які користуються Інтернетом з кожним роком збільшується. Також, спостерігаємо, що 2017 році відбулося зменшення відсотка людей, що користуються мережею Інтернет щодня, порівняно з 2016 роком. Далі слід дослідити тип пристрою, яким користуються споживачі для доступу в мережу Інтернет.

На рис. 1.3 [23 – 25] наведені відсотки користування такими пристроями як: комп’ютер, смартфон та планшет. Бачимо, що лідером є комп’ютер, але з кожним роком зростає відсоток людей, які користуються мережею Інтернет за допомогою смартфона. Популярність планшету також зростає, але незначними темпами. Тому новому підприємству при розробці свого веб-ресурсу слід орієнтуватися на користувачів, що будуть використовувати комп’ютери та смартфони для доступу до мережі Інтернет.

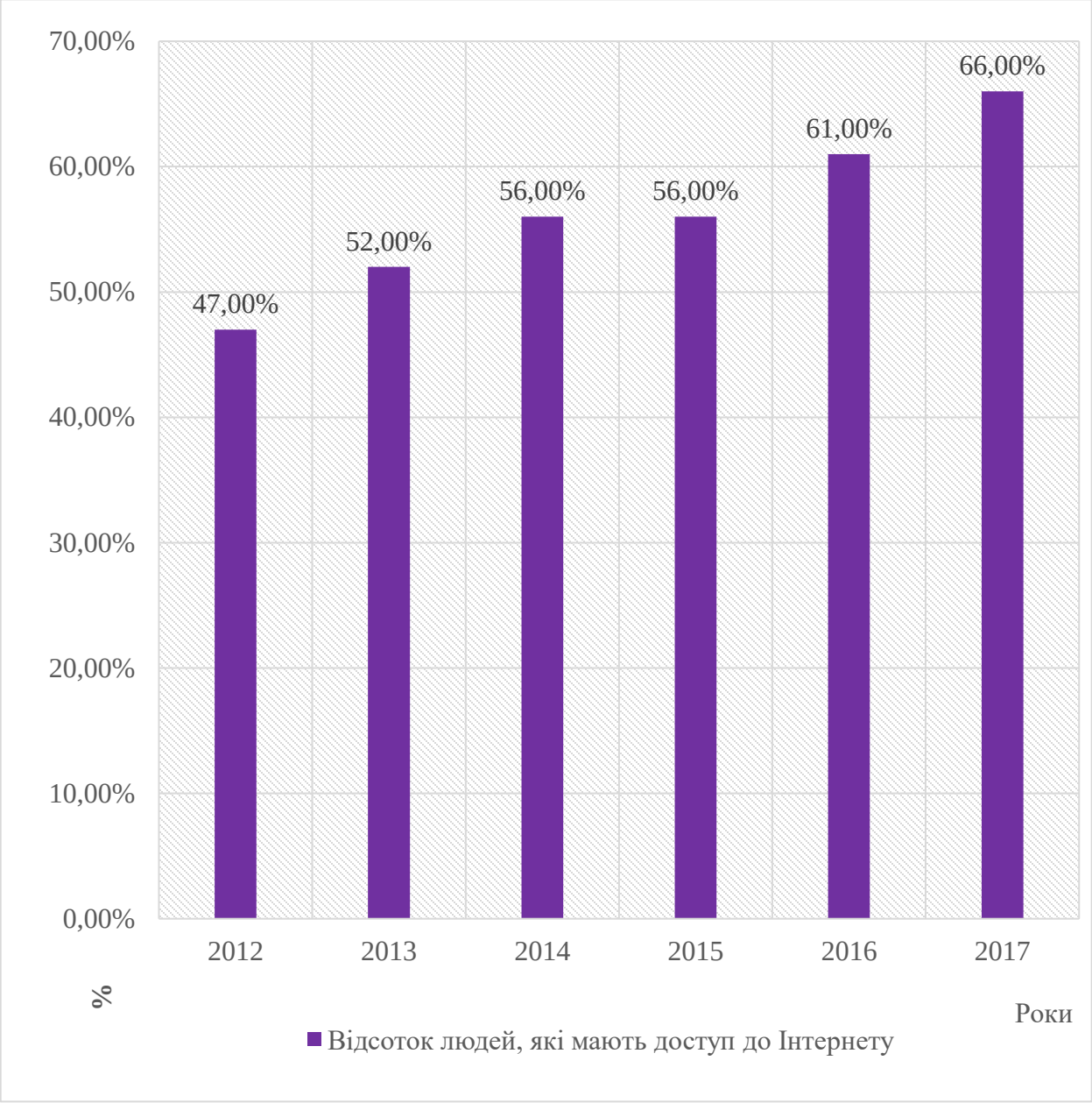


Рисунок 1.1 – Відсоток людей в Україні, які мають доступ до Інтернету, %

Рисунок 1.2 – Відсоток людей в Україні, які користуються мережею Інтернет щодня, %

Рисунок 1.3 – Розподіл за типом пристроїв для доступу в мережу Інтернет в Україні, %

За даними звіту GSMA Intelligence «Глобальні тенденції мобільного зв'язку 2017 року» [23 – 26] у 2020 році загальна кількість пристроїв у світі перевищить 20 млрд, а найбільшого розповсюдження набудуть IoT системи. Також з огляду на інформацію, наведену на рис. 1.4 – 1.5 [23 – 27] слід звернути увагу, що у 2020 році майже 1 млрд користувачів у світі, для доступу до мережі Інтернет почне використовувати мобільні телефони.

Дані статистичні дані, підтверджують те, що на сьогоднішній день при створенні веб-ресурсів, підприємствам слід орієнтуватися не тільки на користувачів, що використовують персональні комп'ютери, а і на користувачів із смартфонами.

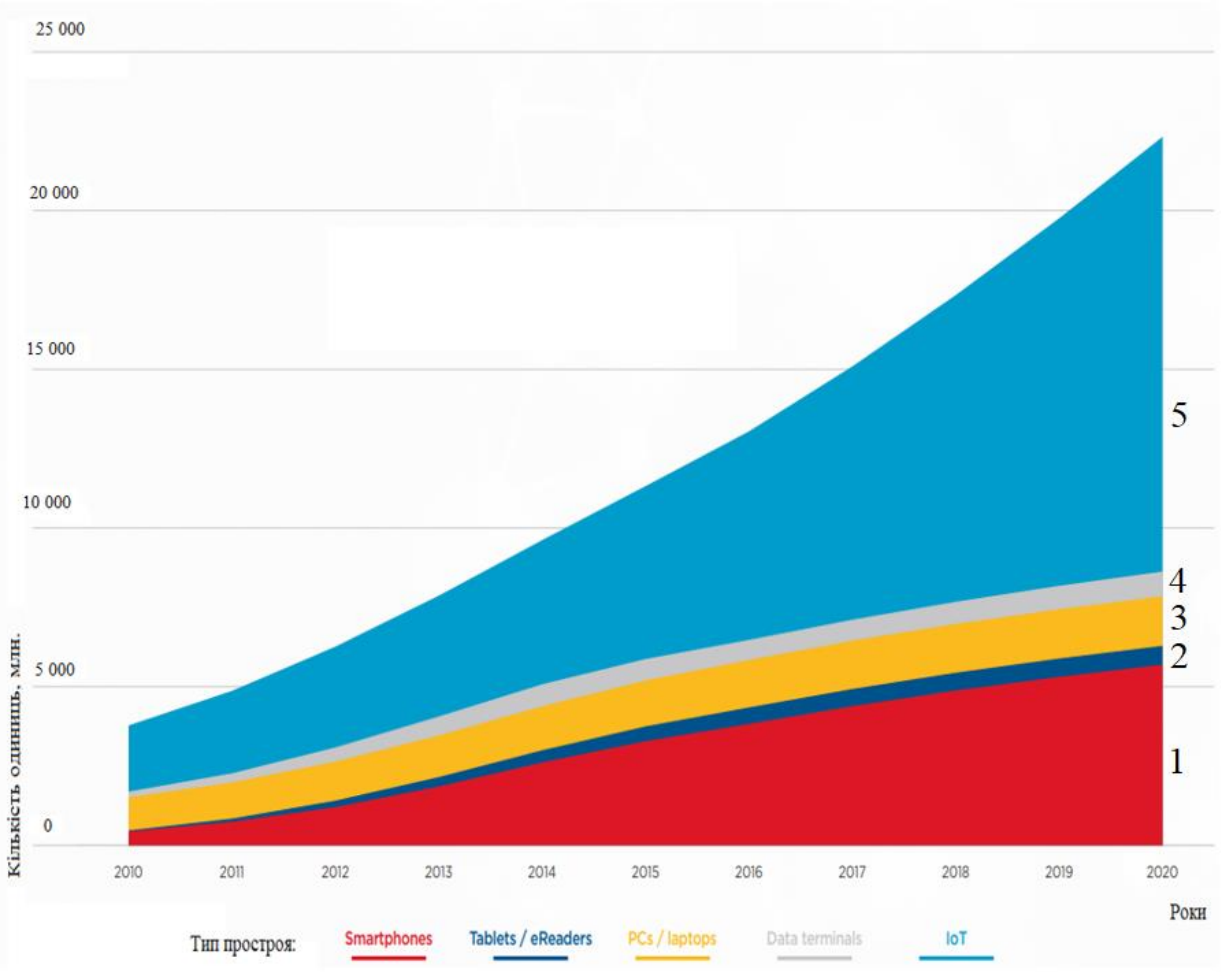


Рисунок 1.4 – Розподілення пристроїв для користування мережею Інтернет у світі, млн одиниць: 1 – Smartphones, 2 – Tablets/eReaders, 3 – Pcs/laptops, 4 – Data terminals, 5 – IoT

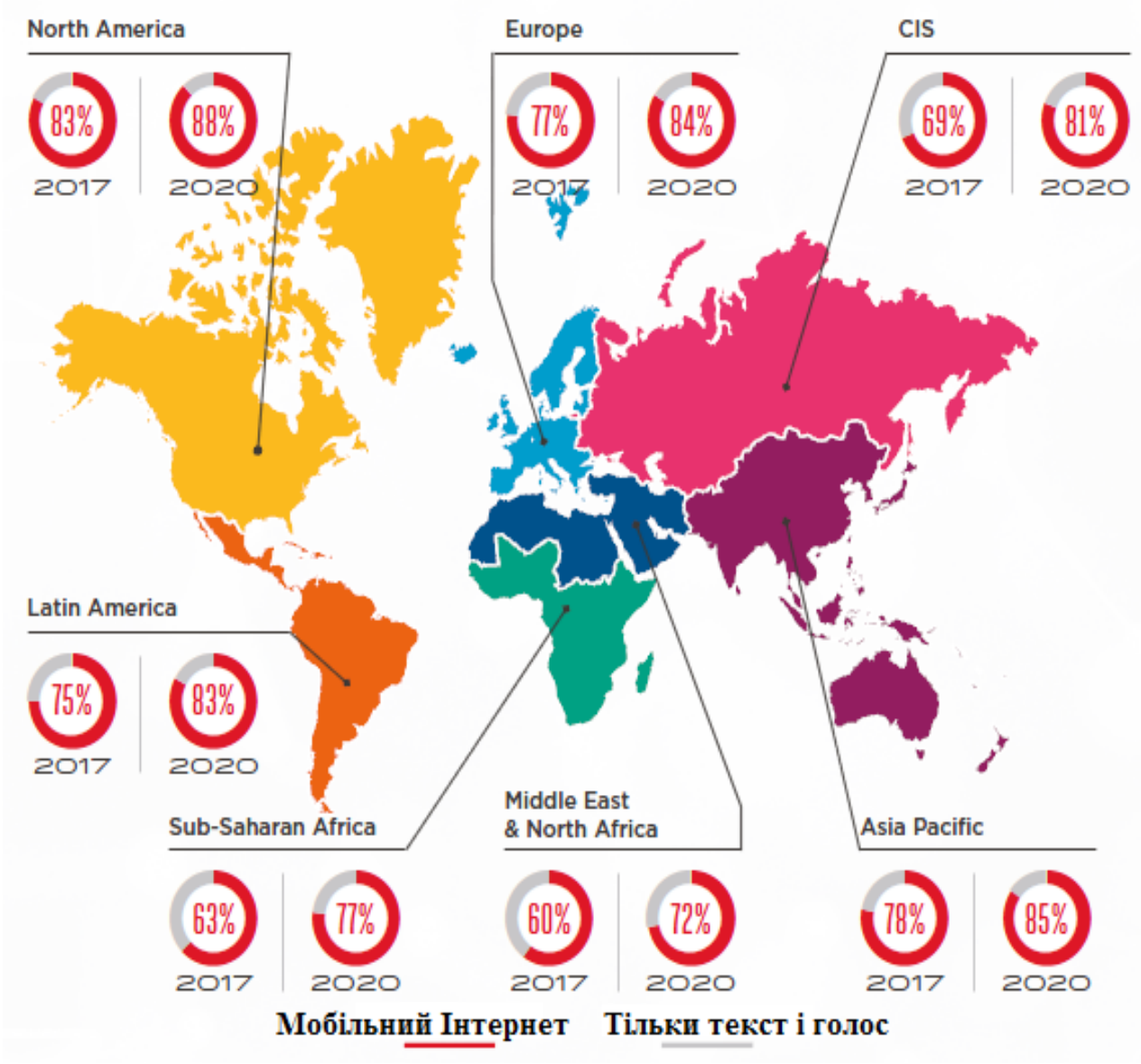


Рисунок 1.5 – Частка власників мобільних телефонів, які мають доступ до Інтернету

Далі слід дослідити популярність Інтернет-покупок та визначити найпопулярніші сфери. Для цього дослідимо популярність Інтернет – покупок у наступних країнах: Норвегія, Аргентина, Швейцарія, Бразилія, Туреччина, Чилі, Польща, США, Греція, Канада, Китай, Мексика, Австрія, Болгарія, Фінляндія, Україна, Румунія, Угорщина, Австралія, Японія, Індія.

Розглянемо середні витрати на електронну покупку на одного покупця (рис. 1.6) [23 – 26]. Бачимо, що з 2014 року до 2016 середні витрати зросли с 5,7% до 7,40%.

Рисунок 1.6 – Середні витрати на електронну покупку на одного покупця, $ та %

Далі розглянемо, що саме шукають по купують інтернет-користувачі. За даними, наведеними на рис. 1.7 – 1.8 [23 – 26], видно, що найпопулярнішими категоріями є мода, матеріальні розважальні продукти та подорожі.

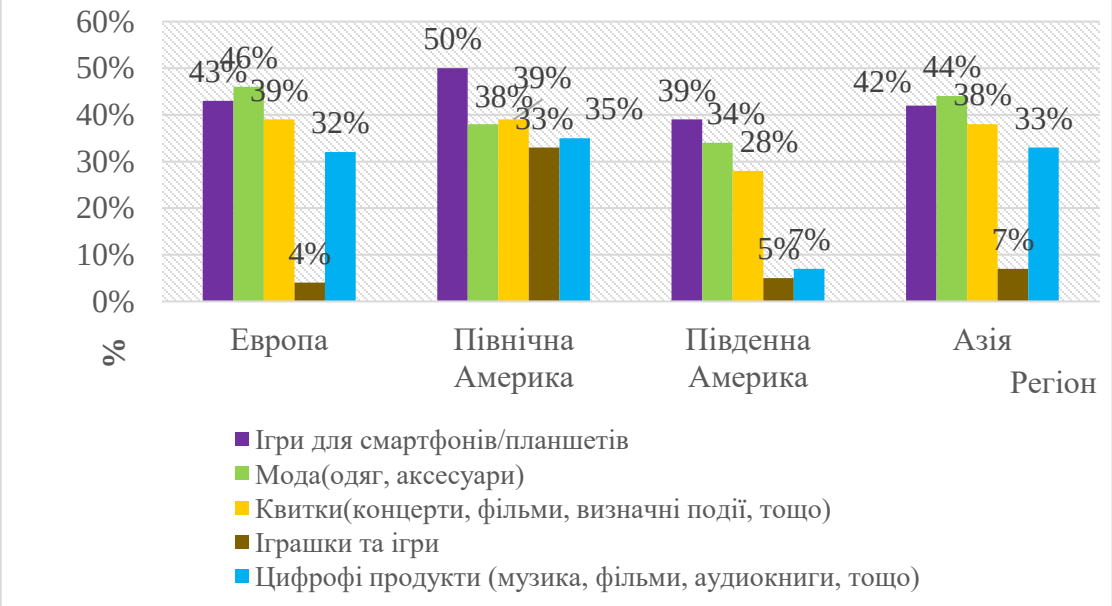


Рисунок 1.7 – Найпопулярніші мобільні категорії інтернет – покупок, %

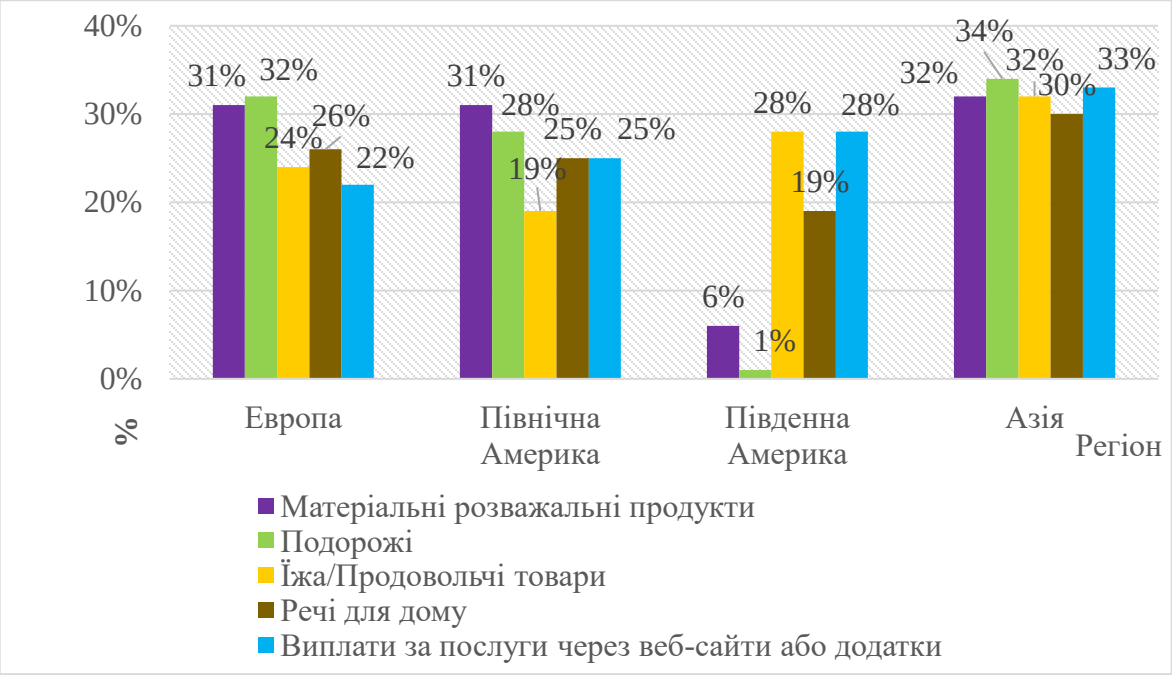


Рисунок 1.8 – Частка продуктів придбаних через смартфони та планшети, %

Досліджуючи інтернет-покупки за способом здійснення (рис. 1.9) [23 – 26], також спостерігається тенденція збільшення кількості покупок за допомого смартфонів.

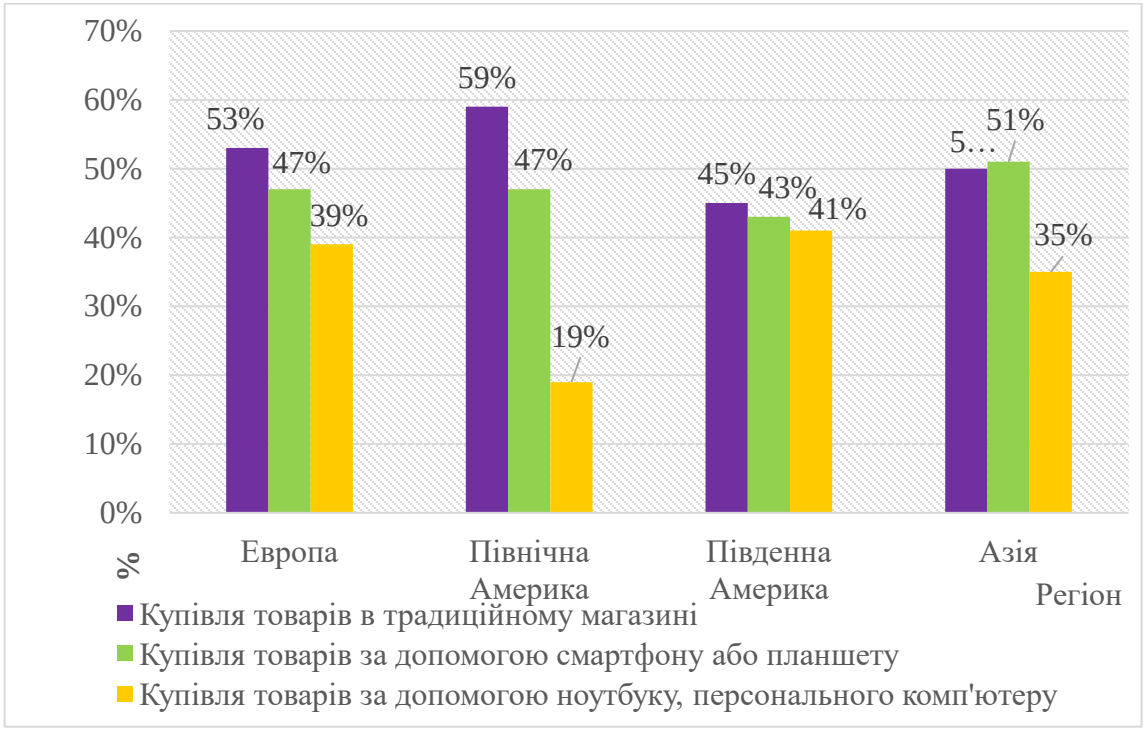


Рисунок 1.9 – Класифікація покупок за способом здійснення у 2016 році, %

Дослідивши причини покупок у мережі Інтернет, що наведені на рис. 1.10 [23 – 25] можна зробити висновок про те, що засобами електронної комерції користуються через зручність; кращу ціну товарів в мережі Інтернет, порівняно з традиційними магазинами; більшу безпеку, ніж готівковий розрахунок.

З огляду на вище зазначену інформацію українським підприємствам у своєму розвитку слід орієнтуватися на розвиток веб-ресурсів за допомогою засобів електронної комерції.

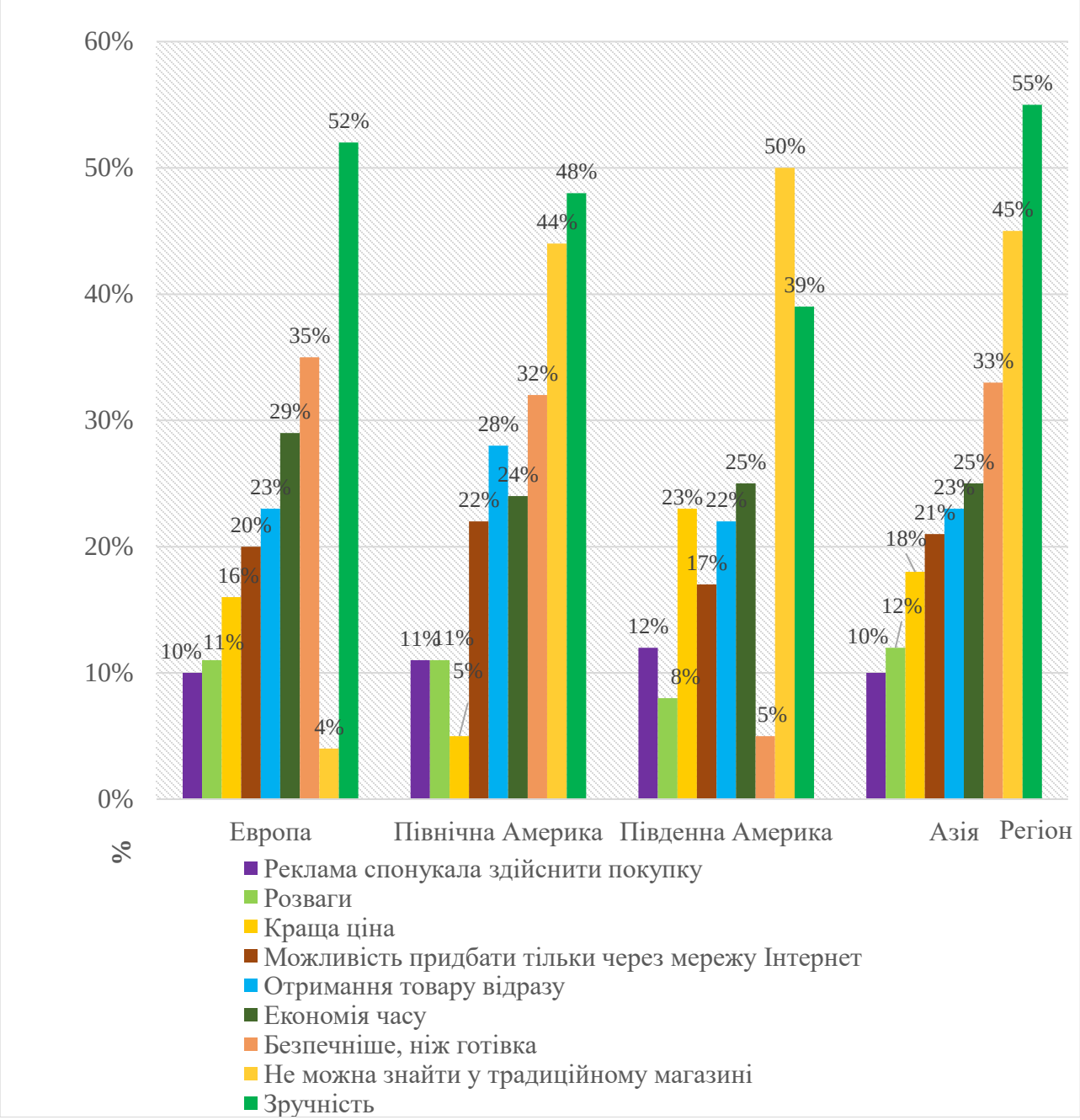


Рисунок 1.10 – Причини здійснення покупок через мережу Інтернет

Місто Суми не перенасичене книжковими магазинами, більшість з яких не мають веб-ресурсу. Основними напрямками діяльності таких підприємств є оптова та роздрібна торгівля книгами, канцелярськими засобами, іноді іграшками. Більшість книжкових магазинів мають широкий асортимент продукції: книжки від історичної літератури до книжок про психологію та навчальну літературу. Усі підприємства займаються дослідженням ринку та оновленням асортименту. Завдяки цьому, підприємства ефективно функціонують та мають насичену базу клієнтів, понад тисячі постійних покупців. Також, в магазинах проводяться акції та існує система знижок для постійних клієнтів.

Не всі магазини мають веб-сторінки, а тому підприємства не можуть ефективно конкурувати на ринку. Для ефективної конкуренції потрібне створення веб – ресурсу. Розглянемо організаційну структуру потенційного книжкового магазину, що наведена на рис. 1.11. Організаційна структура книжкового магазину є лінійною, в основі якої лежить поєднання виробничих та управлінських функцій у керівників (директор, заступники). Усі повноваження є лінійними та спрямовані від вищої ланки управління до нижчої.

У торговому залі магазину займаються безпосередньо продажем товарів. Даний відділ складається із одного завідуючого магазином, касира, прибиральника та декількох продавців – консультантів. Завідуючий магазином займається питаннями організації праці, забезпеченням належних умов праці та облік товарно-матеріальних цінностей. У кінці звітного періоду завідуючий подає відповідну звітність директорам про обсяги продажів.

Відділ продажів займається розширенням асортименту продукції, пошуком нових клієнтів та підтримкою зав’язків із постійними клієнтами. Відділ маркетингу та реклами займається розробленням ефективних рекламних кампаній. ІТ-відділ займається усуненням технічних несправностей та обслуговуванням і видачі техніки на підприємстві.

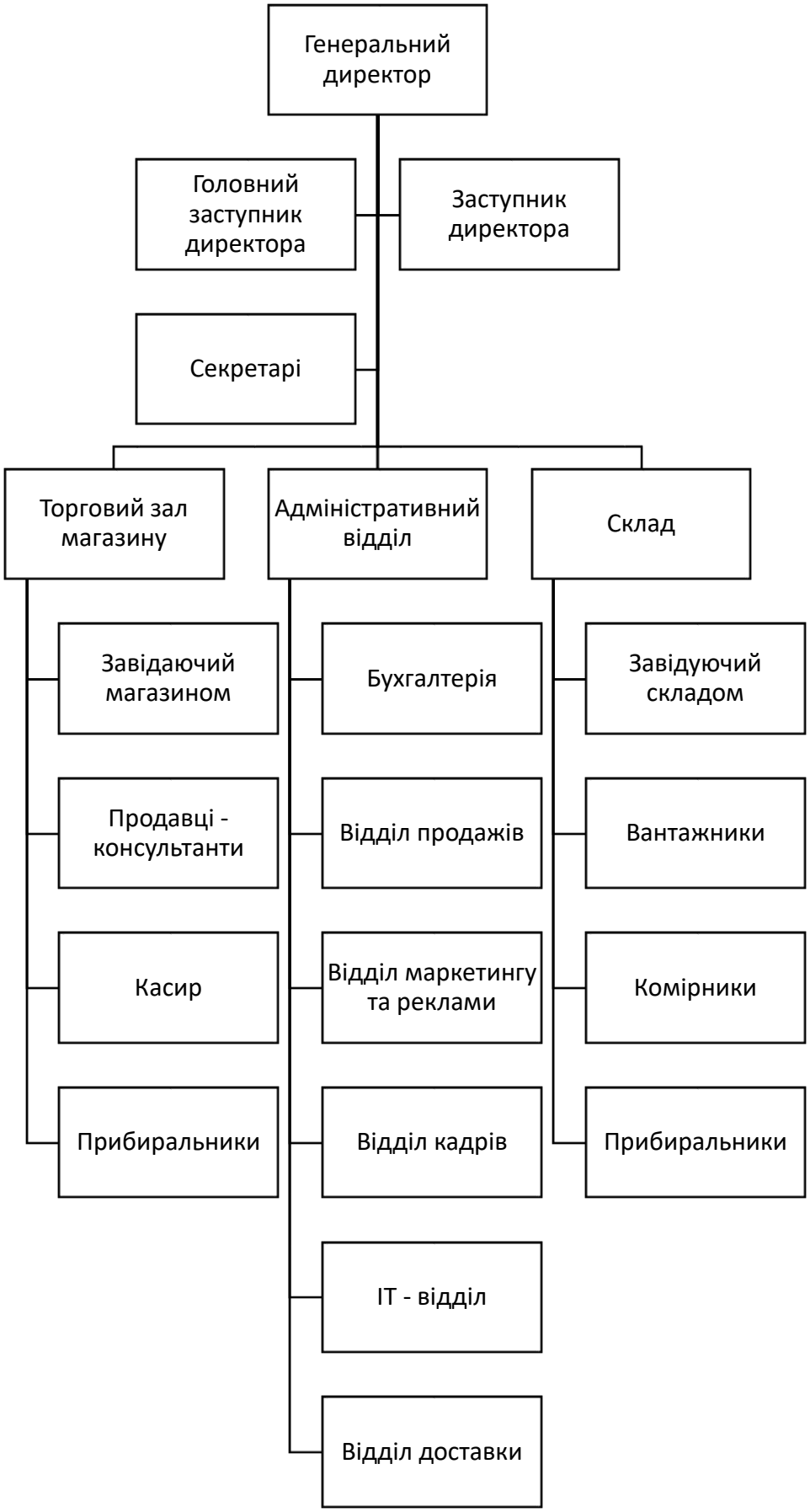


Рисунок 1.11 – Організаційна структура потенційного книжкового магазину

Найчастіше у книжкових магазинах інформаційна база зберігається у документах програми «MS Excel». Це ускладнює обробку даних та створює проблеми при розширенні клієнтської бази. Дане програмне забезпечення не дозволяє здійснити автоматизацію діяльності підприємства в сфері електронної комерції.

Техніко-економічна сутність полягає у розробці та реалізації механізму для автоматизованого проведення операцій, що пов’язані із замовленням та оплатою товарів, веденням та зберіганням інформації про клієнтів, складського обліку та створення зручних механізм формування та подальшої обробки звітної інформації.

1.2 Аналіз існуючих систем для автоматизації діяльності підприємства в сфері електронної комерції

Оптимальна система управління веб-ресурсом має відповідати не тільки зовнішнім критеріям краси веб-сторінки, а й низки важливих технічних вимог, без яких ефективне функціонування сайту буде неможливим.

Найважливіші характеристики, яким має відповідати веб-сторінка книжково магазину наведена у таблиці 1.6 [28 – 29].

Таблиця 1.6 – Характеристика найважливіших ознак веб-ресурсу

№ п/п	Назва ознаки	Характеристика
1	Безпека	Одна із найважливіших характеристик, що забезпечує надійне функціонування системи без витоків та пошкодження інформації.
2	Продуктивність	Різні додатки, написані на одній і тій же мові програмування можуть вимагати різну кількість ресурсів.
3	Розширення	Система повинна супроводжуватися документаціє, що полегшить у майбутньому процеси розширення.

Продовження таблиці 1.6

№ п/п	Назва ознаки	Характеристика
4	Інтеграція	Розроблена система виконує лише частину адміністративних функцій підприємства. Питаннями ціноутворення, розробки каталогу товарів чи іншими функціями може займатись корпоративна система підприємства. Тому важливо, щоб розроблена система веб-ресурсу могла інтегруватися із корпоративною системою підприємства.
5	Зручний та приємний інтерфейс	Інтерфейс веб-ресурсу повинен бути зрозумілий, простий та приємний для користування. Завдяки цьому, споживачі зможуть здійснювати покупки зручним для себе способом без зайвих, непотрібних дій.
6	Необхідність обов'язкової реєстрації	Обов'язкова реєстрація відштовхує багато споживачів, які для здійснення однієї покупки повинні заповнювати складну реєстраційну форму.

Далі розглянемо найпопулярніші системи для автоматизації діяльності підприємств роздрібної торгівлі. У таблиці 1.7 [30 – 35] наведена характеристика платних систем.

Таблиця 1.7 – Характеристика платних систем для створення веб-ресурсів та автоматизації

№ п/п	Назва системи	Характеристика
1	DIAFAN.CMS	Дана система є універсальною системою управляння сайтами, направлена на створення веб-ресурсів різного рівня складності та подальшу їх підтримку. За допомогою «DIAFAN.CMS» можна створювати наступні веб-ресурси: інтернет-магазини, корпоративні сайти та інформаційні портали. Панель управління надає можливість управління та редагування майже всіх елементів веб-ресурсу. Додатково можна розділяти права керування веб-ресурсом. Панель управління можна відображати на російській або англійській мовами, публічні частини сайту можна відображати на будь-якій мові. «DIAFAN.CMS» постійно вдосконалюється та виправляє дрібні недоліки функціонування. Система пропонує безкоштовний період користування в 21 день, приблизна вартість пакету для розробки та підтримки веб-ресурсу – 300$.

Продовження таблиці 1.7

№ п/п	Назва системи	Характеристика
2	Shop-Script	Одна із популярних комерційних систем створення веб-ресурсів у формі інтернет – магазину, із відкритим вхідним кодом. Базується на мові програмування PHP та фреймфорку Webasyst. Підтримується інтеграція із додатками, що також написані на базі Webasyst. Також, система підтримує один із популярних шаблонів проектування – MVC. Існує можливість додаткової організації продажів через соціальні мережі. До недоліків слід віднести незначну кількість безкоштовних шаблонів для створення свого сайту, складна система навігації для адміністраторів сайту та обмежений навігаційний функціонал для редагування зовнішнього вигляду веб-ресурсу. Приблизна вартість пакету складає 240$.
3	PrestaShop	Одна із популярних комерційних систем створення веб-ресурсів у формі інтернет – магазину, із відкритим вхідним кодом. Система написана на мові програмування PHP і використовує базу даних MySQL. Панель управління сайтом є продуманою, простою та зручною у користуванні. Серед недоліків слід віднести неможливість інтеграції із програмним забезпеченням «1С: Підприємство». Цінова політика пакетів починається від 70$.
4	«1С -Бітрікс: Управління сайтом»	Являє собою професійну систему створення, управління веб-ресурсами. За допомогою даної системи можна створювати наступні веб-ресурси: інтернет-магазин, корпоративний сайт та інформаційний портал. «1С-Бітрікс: Управління сайтом» розрахована на створення сайтів різного типу складності. Система, також, характеризується високим рівнем безпеки та інтеграцією із програмним забезпеченням компанії 1С. Цінова політика пакетів для управління сайтом починається від 1500 грн.
5	UMI.CMS	Являє собою просту, функціональну та економічну систему для управління веб-ресурсами. Для розробки та управління сайтом не вимагає від користувача знань у даній області. Система не накладає обмежень на дизайн веб-ресурсу. Цінова політика пакетів для управління сайтом починається від 2000 грн.
6	NetCat	Являє собою універсальну систему для створення та управління веб-ресурсами з потужною платформою електронної комерції та соціальних проектів. Цінова політика пакетів для управління сайтом починається від 2500 грн.

У таблиці 1.8 [35 – 38] наведена характеристика безкоштовних систем для автоматизації діяльності підприємств роздрібної торгівлі. Але даний вид систем не може в повній мірі задовольнити поставлені вимоги до системи автоматизації діяльності підприємства роздрібної торгівлі.

Таблиця 1.8 – Характеристика безкоштовних систем для створення веб-ресурсів та автоматизації

№ п/п	Назва системи	Характеристика
1	WordPress	Являє собою систему для розробки та керування вмістом веб-ресурсу. Застосовується як для створення блогів, так і складних сайтів. Система написана на мові програмування PHP та використовує базу даних MySQL.
2	OpenCart	Популярна безкоштовна система для розробки веб-ресурсів. Характеризується легкістю доопрацювання створених шаблонів та функціонування.
3	Commercebox	Готова збірна інтернет-магазину на базі Drupal.
4	uCoz	Являє собою просту систему для розробки та керування вмістом веб-ресурсу. Для створення сайту не потрібно навичок роботи в даній області. Головним недоліком є реклама.

При виборі системи для розроблення автоматизованої системи підприємства слід орієнтуватися на основні характеристики, що має забезпечувати веб-ресурс. Для створення надійного сайту слід звернутися до послуг комерційних систем, бо вони забезпечують технічну підтримку створених сайтів та додаткові модулі функціоналу.

1.3 Формування вимог до автоматизованої системи

Інформаційна система - це сукупність засобів збору, зберігання, передачі та оброблення інформації в певному програмному середовищі для досягнення поставленої мети у процесі управління. Автоматизована інформаційна система – це сукупність інформації, різних методів і моделей, апаратних, програмних, організаційних, технологічних засобів і відповідних фахівців. Тому в широкому розумінні інформаційна система - це організаційно впорядкована сукупність фахівців, інформаційних ресурсів та інформаційних технологій, зокрема з використанням засобів обчислювальної техніки і зв'язку, що реалізовують такі інформаційні процеси як: отримання вхідних даних, обробка даних, зміна

власного внутрішнього стану (внутрішніх зв’язків/відносин), видача результату або зміна свого зовнішнього стану (зовнішніх зв'язків/відносин) [39 – 40].

Вимоги до інформаційної системи обумовлюються вхідними даними, на підставі яких проектуються та створюються автоматизовані інформаційні системи. Система повинна вміти працювати з базою клієнтів, товарів та співробітників підприємства, вміти створювати та обробляти замовлення клієнтів, тощо. До найголовніших вимог, яким має відповідати розроблена автоматизована система відноситься:

- програмне забезпечення має орієнтуватися на вимоги користувача та задовольняти його потреби;
- розроблена система має бути економічно ефективною;
- розроблена система повинна вміти працювати із замовленнями, ефективно оперувати із даними, що зберігаються у віддаленій базі даних;
- розроблена система має виконувати функцію автоматизації роботи підприємства.

Далі необхідно визначитися із функціональними вимогами, вимогами надійності та зручності до системи автоматизації, що з часом можуть видозмінюватися. Перелік вимог наведений у таблиці 1.9 [39 – 40].

Таблиця 1.9 – Характеристика вимог системи автоматизації

№ п/п	Тип вимоги	Назва вимоги	Характеристика
1	Функціональні	Швидкість роботи	Розроблена система має швидко оброблювати запити та представляти необхідну інформацію користувачеві.
2		Гнучкість архітектури	Розроблена архітектура має підтримувати можливість змінення певних частин системи, їх доопрацювання та вдосконалення.
3		Здатність обробляти запити	Система повинна обробляти запити роботи з замовленнями, інформацією про клієнтів, працівників підприємства та інформацією про товари.

Продовження таблиці 1.9

№ п/п	Тип вимоги	Назва вимоги	Характеристика
1	Вимоги надійності	Частота збоїв	Значення даної характеристика має бути зведеною до мінімуму. В разі виникнення помилки користувач має отримувати повідомлення про свої подальші дії.
2		Стійкість бази даних	Функціонування системи має забезпечуватися безперебійним з'єднанням з базою даних без втрати інформації та виникнення взаємних блокувань.
3		Доступність	Веб-ресурс має функціонувати у режимі реального часу з можливістю цілодобового доступу користувачів.
4		Захист бази даних	Веб-ресурс має бути захищений від sql-ін'єкцій.
1	Вимоги зручності	Зрозумілість інтерфейсу	Інтерфейс веб-ресурсу має бути зрозумілим та простим. Користувач на інтуїтивному рівні має розуміти, що необхідно роботи.
2		Зручність управління контентом	Процедуру додавання, видалення чи змінення контенту сайту мають бути зручними.
3		Швидка реєстрація	На здійснення реєстрації користувачі повинні витрачати мінімум часу.

Перший пакет вимог до розробки системи автоматизації для підприємства, що займається роздрібною торгівлею в сфері електронної комерції, був сформований. В процесі подальшого аналізу та розробки системи вимоги до програмного забезпечення будуть доповнюватися та деталізуватися.

2 РОЗРОБКА ПРОЕКТУ АВТОМАТИЗОВАНОЇ СИСТЕМИ

2.1 Розробка моделі бізнес-процесів

Основною складовою управління бізнес-процесами є модель даного бізнес-процесу. Для кращого розпізнавання, порівняння, проведення аналізу та, найголовніше, управління, необхідно розділити бізнес-процес на множину ознак, що будуть характеризувати його кожну властивість чи здатність. Таким чином, відбувається моделювання - процес відображення суб'єктивного бачення потоку робіт у вигляді формальної моделі, що складається з взаємопов'язаних операцій. Бізнес-модель - це формалізований (графічний, табличний, текстовий, символьний) опис бізнес-процесів, що відображає реально існуючу або передбачувану діяльність підприємства [41 – 43].

У якості методів моделювання використовуються різні підходи: як структурний, так і об'єктно-орієнтовний. Розподіл методів на структурні та об'єктні є умовним, бо найбільш розвинені методологія використовують поєднання вище описаних методів. На сьогоднішній день до найбільш поширених методів відносяться [41 – 43]:

- метод функціонального моделювання SADT (IDEF0);
- моделювання потоків даних DFD;
- метод моделювання процесів IDEF3;
- метод ARIS;
- метод Ericsson-Penker.

У ході розробки моделі бізнес-процесів будемо використовувати методології IDEF0, DFD, IDEF3.

IDEF0 – методологія функціонального моделювання. За допомогою даної методології відбувається формалізація і опис бізнес-процесів у вигляді ієрархічного представлення об'єктів [45].

В методології IDEF0 важливим є положення процесу та стрілок, характеристика наведена у таблиці 2.1.

Таблиця 2.1 – Значення положення бізнес-процесу у методології IDEF0

№ п/п	Розташування бізнес-процесу	Значення
1	Ліва сторона	Позначає інформацію, товарно-матеріальні цінності, які подається на вхід бізнес процесу для подальшого перетворення у ході виконання бізнес-процесу.
2	Права сторона	Позначає інформацію, товарно-матеріальні цінності, які були отримані у ході виконання бізнес-процесом, тобто на виході бізнес-процесу.
3	Верхня сторона	Позначає інформацію, товарно-матеріальні цінності, які визначають яким чином має виконуватися бізнес-процес, тобто є елементом управління бізнес-процесом.
4	Нижня сторона	Позначає те, що перетворює вхідну інформації, товарно-матеріальні цінності на вихідну. Прикладом можуть бути співробітники підприємства, інтерфейс взаємодії з користувачем, тощо.

Діаграми потоків даних (Data Flow Diagram) використовуються для документування механізмів передачі і обробки інформації в модельованій системі. Найчастіше дані діаграми будуються для відображення документообігу підприємства і є доповненням до діаграм іншої нотації. Характеристика основним елементів для даної нотації наведена у таблиці 2.2 [41, 45 – 48].

Таблиця 2.2 – Характеристика об'єктів у методології DFD

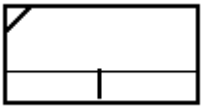
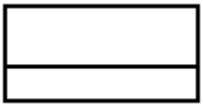
№ п/п	Назва об’єкту	Характеристика
1	Зовнішня сутність (External entity)	Представляє сутність поза контекстом системи, яка являє собою джерело або приймач даних системи.
2	Сховища даних (Data Store)	Представляють дані, до яких здійснюється доступ. Дані сховища, використовують, створюють та змінюють дії (activities).
3	Дії (Activities)	Представляють процеси обробки та зміни інформації.
4	Стрілки (Arrows)	Відображають потоки інформації.

Розробляючи діаграми у даній нотації слід дотримуватися наступним правил:

- кожен процес повинен мати щонайменше один вхід та вихід;
- кожне сховище даних повинно мати хоча б один вхідний або вихідний потік;
- всі процеси йдуть в інший процес або в сховище даних [43 – 44, 46 – 48].

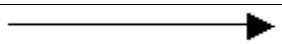
Нотація IDEF3 частіше застосовується для побудови процесів нижнього рівня, наприклад для декомпозиції блоків процесу IDEF0. На відміну від нотації IDEF0, IDEF3 нотація не підтримує відображення «механізмів» і «управління»,. В свою чергу, дана метрологія зосереджує увагу на відображенні черговість виконання робіт [43 – 44, 46 – 48]. Характеристика основних об’єктів нотації IDEF3 наведена у таблиці 2.3 [43 – 44, 46 – 48].

Таблиця 2.3 – Характеристика об'єктів у методології IDEF3

№ п/п	Назва об’єкту	Зовнішній вигляд	Характеристика
1	Модель роботи		Об'єкт використовується для опису функцій (процедур, робіт), які виконуються підрозділами чи співробітниками підприємства.
2	Об’єкт посилання		Об'єкт використовується для опису посилань на інші діаграми моделі, циклічні переходи в рамках однієї моделі або різні коментарі до функцій.
3	Логічне «і»		Логічний оператор, що визначає зв'язок між функціями в рамках процесу. Застосовується для розгалуження процесу. Позначає виконання логічної умови «і».
4	Логічне «та»		Логічний оператор, що визначає зв'язок між функціями в рамках процесу. Застосовується для розгалуження процесу. Позначає виконання логічної умови «та».
5	Логічне виключне «та»		Логічний оператор, який визначає зв'язку функціями в рамках процесу. Застосовується для розгалуження процесу. Позначає виконання виключної логічної умови «та».

Всі зв’язки в IDEF3 є односпрямованим і організовуються зліва направо. У моделях можуть використовуватися стрілки трьох видів, характеристика яких наведена у таблиці 2.4 [43 – 44, 46 – 48]. Контекстна діаграма діяльності Інтернет-магазину зображена на рисунку 2.1, пояснення елементів наведено у таблиці 2.5.

Таблиця 2.4 – Характеристика типів стрілок у нотації IDEF3

№ п/п	Тип стрілки	Графічне представлення	Характеристика
1	Стрілка передування		Поєднує послідовно виконувані функції.
2	Стрілка відношення		Використовується для прив'язки об'єктів-коментарів до функцій.
3	Стрілка потоку об’єктів		Показує потік об'єктів від однієї функції до іншої.

Таблиця 2.5 – Опис основних елементів контекстної діаграми, нотації IDEF0

№ п/п	Назва стрілки	Тип	Характеристика
1	Дані про клієнтів	Input	Отримання даних про клієнта.
2	Дані про замовлення	Input	Отримання даних про замовлення.
3	Інформація про стан книжок та їх наявність	Input	Інформація про наявність товару на складі.
4	Надходження нових товарів	Input	Дані щодо надходження нових товарів та інформація про них.
5	Законодавство	Control	Нормативні та законодавчі акти України відповідно до яких здійснюється діяльність.
6	Внутрішні інструкції підприємства	Control	Документи, в яких визначено основні завдання, обов’язки, права і відповідальність співробітників підприємства при здійсненні своєї роботи.
7	Оформлення та відправка замовлення	Output	Оформлення замовлення та доставки товару клієнтові.
8	Інформація про залишки товарів	Output	Дані щодо залишків товарів на складі.
9	Оновлення інформації на сайті/базі даних	Output	Здійснення оновлення інформації про стан об’єкту, його наявності чи інших характеристик.
10	Клієнти	Mechanism	Користувач Інтернет-магазину.
11	Працівники підприємства	Mechanism	Співробітники підприємства.
12	Інтерфейс взаємодії із системою онлайн - магазину	Mechanism	Інтерфейс за допомогою якого клієнти, взаємодіють із системою.

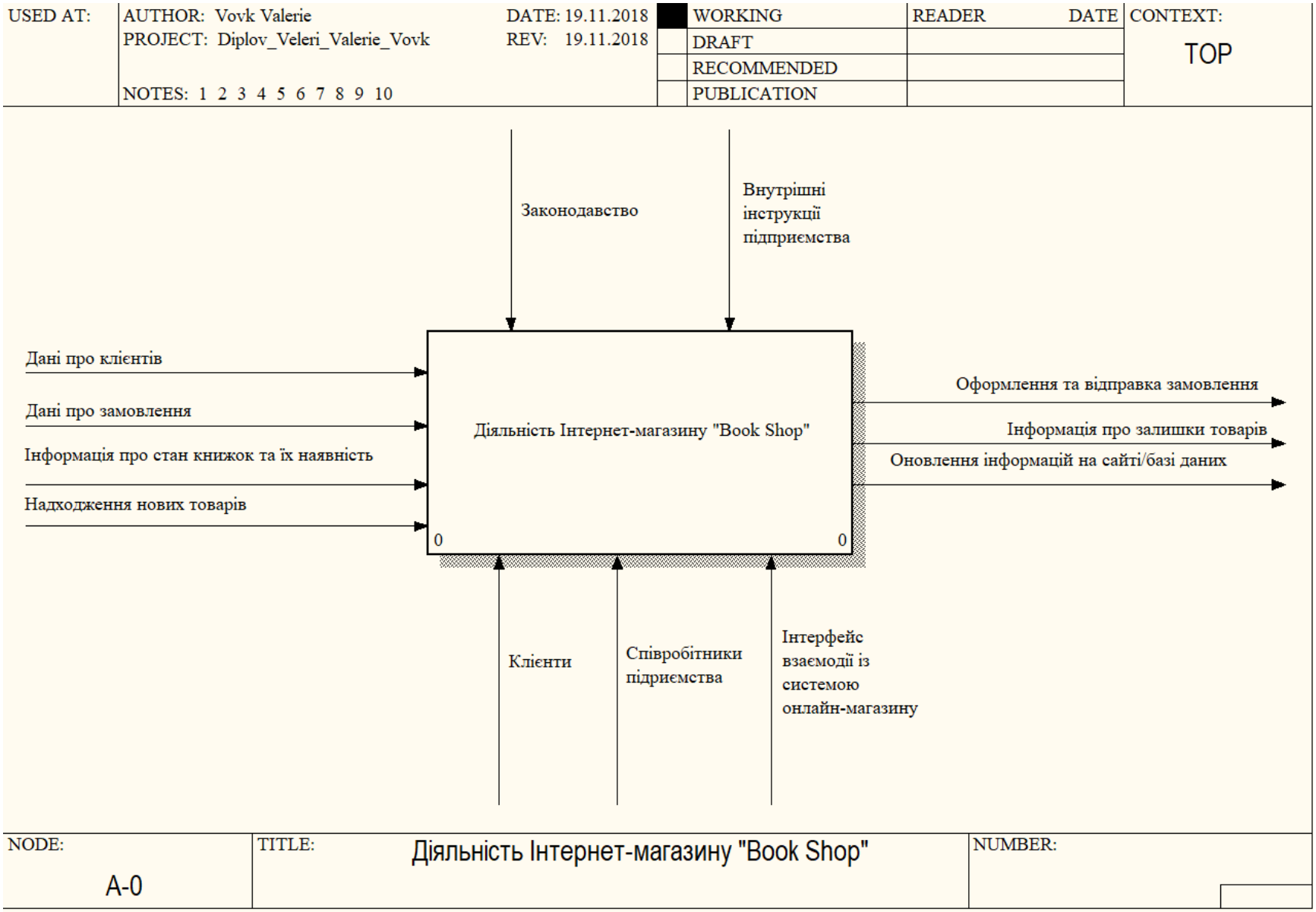


Рисунок 2.1 – Контекстна діаграма діяльності Інтернет-магазину, нотація IDEF

Для подальшої розробки бізнес-процесу була виконана декомпозиція контекстної діаграми у нотації IDEF0, що наведена на рис. 2.2. Пояснення основним елементів контекстної діаграми наведено у таблицях 2.6 – 2.7. відповідно. Дана діаграма більш детально описує процес діяльності Інтернет – магазину, та включає в себе вже чотири елементи типу «робота», а саме: «Замовлення товару клієнтом» та «Оформлення замовлення клієнта», «Доставка товару».

В подальшому дослідженні розглянемо декомпозиції основних «робіт» даної діаграми, а саме: «Замовлення товару клієнтом» та «Оформлення замовлення клієнта». Досліджувати декомпозицію роботи «Доставка товару» не потрібно. Виконання даної роботи виходить за рамки функціонування веб-інтерфейсу розроблювальної системи. Дана функція виконується шляхом заведенням відповідних запитів до поштових служб, які у свою чергу виконуються доставку товару безпосередньо клієнту, зокрема до одного із своїх поштових відділень.

Таблиця 2.6 – Опис робіт декомпозиції контекстної діаграми діяльності Інтернет-магазину у нотації IDEF0

№ п/п	Назва стрілки	Статус	Характеристика
1	Прийняття запиту на прийом товару	WORKING	Отримання інформації про новий товар на підготовка запиту на внесення змін до системи.
2	Опрацювання запиту додавання нового товару, внесення змін до бази даних	WORKING	Внесення змін про новий товар до бази даних.
3	Замовлення товару клієнтом	WORKING	Процес взаємодії клієнта із системою при створенні замовлення.
4	Оформлення замовлення клієнта	WORKING	Прийом замовлення від клієнта, уточнення інформації та фіксація сформованого замовлення.
5	Доставка товару	WORKING	Організація та створення запиту на доставку товару клієнтові.

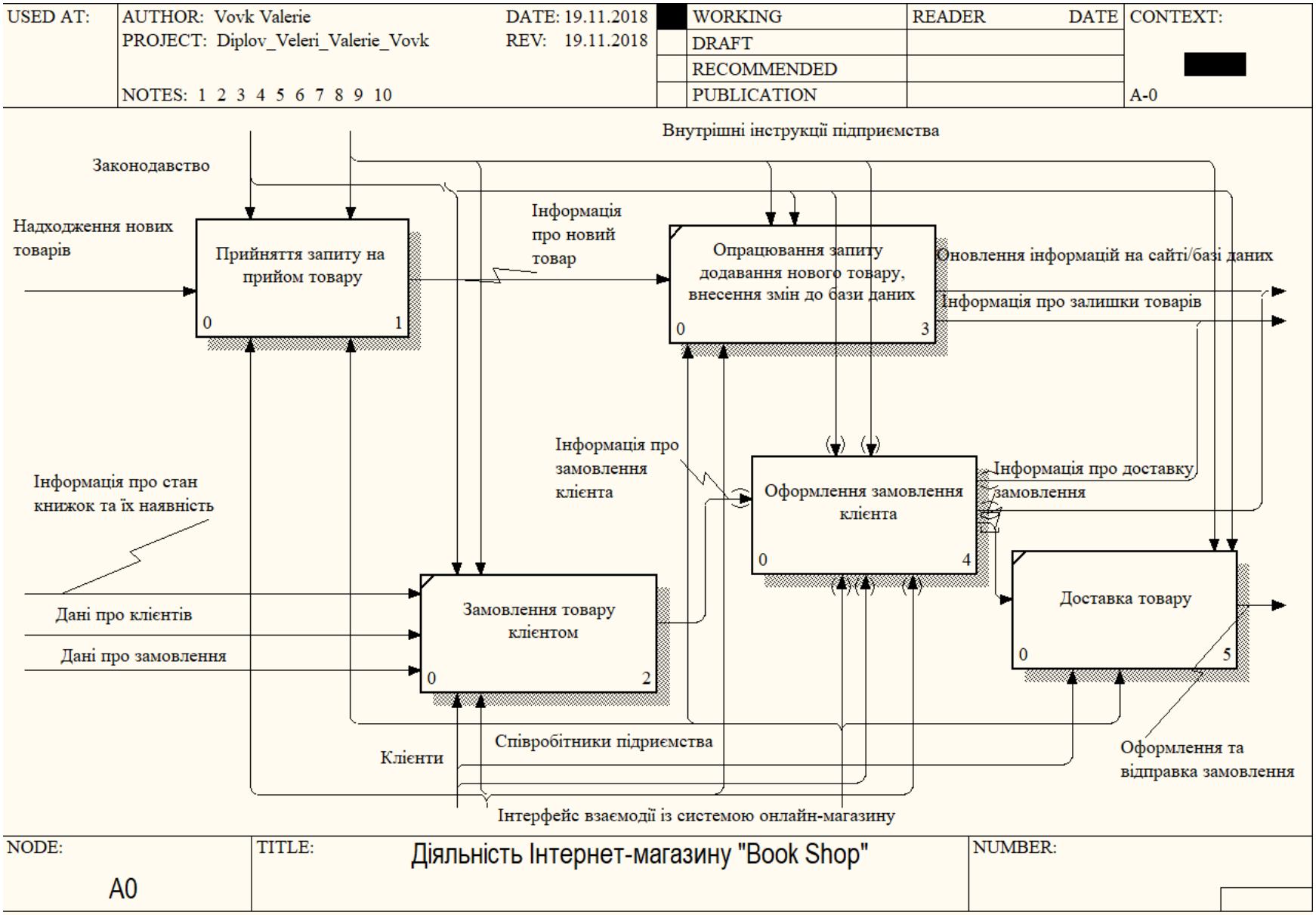


Рисунок 2.2 – Декомпозиція контекстної діаграми діяльності Інтернет-магазину у нотації IDEF0

Таблиця 2.7 – Опис зв’язків між роботами декомпозиції контекстної діаграми діяльності Інтернет-магазину у нотації IDEF0

№ п/п	Стрілка	Джерело	Тип	Призначення	Тип
1	Дані про клієнтів	Контекстна діаграма	----	«Замовлення товару клієнтом»	Input
2	Дані про замовлення	Контекстна діаграма	----	«Замовлення товару клієнтом»	Input
3	Інформація про стан книжок та їх наявність	Контекстна діаграма	----	«Замовлення товару клієнтом»	Input
4	Надходження нових товарів	Контекстна діаграма	----	«Прийняття запиту на прийом товару»	Input
5	Законодавство	Контекстна діаграма	----	«Прийняття запиту на прийом товару», «Опрацювання запиту додавання нового товару, внесення змін до бази даних», «Замовлення товару клієнтом», «Доставка товару»	Control
6	Внутрішні інструкції підприємства	Контекстна діаграма	----	«Прийняття запиту на прийом товару», «Опрацювання запиту додавання нового товару, внесення змін до бази даних», «Замовлення товару клієнтом», «Доставка товару»	Control
7	Клієнти	Контекстна діаграма	----	«Замовлення товару клієнтом», «Оформлення замовлення клієнта», «Доставка товару»	Mechanism
8	Співробітники підприємства	Контекстна діаграма	----	«Доставка товару», «Опрацювання запиту додавання нового товару, внесення змін до бази даних», «Оформлення замовлення клієнта»	Mechanism
9	Інтерфейс взаємодії із системою онлайн - магазину	Контекстна діаграма	----	«Замовлення товару клієнтом», «Опрацювання запиту додавання нового товару, внесення змін до бази даних», «Оформлення замовлення клієнта»	Mechanism

Продовження таблиці 2.7

№ п/п	Стрілка	Джерело	Тип	Призначення	Тип
10	Оформлення та відправка замовлення	«Доставка товару»	Output	{Border}	----
11	Інформація про залишки товарів	««Опрацювання запиту додавання нового товару, внесення змін до бази даних»»	Output	{Border}	----
12	Оновлення інформації на сайті/базі даних	«Доставка товару», «Опрацювання запиту додавання нового товару, внесення змін до бази даних»	Output	{Border}	----
13	Інформація про новий товар	«Прийняття запиту на прийом товару»	Output	«Опрацювання запиту додавання нового товару, внесення змін до бази даних»	Input
14	Інформація про замовлення клієнта	«Замовлення товару клієнтом»	Output	«Оформлення замовлення клієнта»	Input
15	Інформація про доставку замовлення	«Оформлення замовлення клієнта»	Output	«Доставка товару»	Input

Декомпозицію функціонального блоку «Замовлення товар клієнтом» виконаємо у нотації IDEF3, яка наведена на рисунку 2.3. Опис робіт діаграми декомпозиції даного функціонального блоку наведено в таблиці 2.8.

Таблиця 2.8 – Опис робіт діаграми декомпозиції функціонального блоку «Замовлення товару клієнтом» у нотації IDEF3

№ п/п	Функціональний блок	Характеристика	Статус
1	Перегляд асортименту продукції та вибір книги	Клієнт обирає книгу, які він хоче придбати	WORKING

Продовження таблиці 2.8

2	Створення запиту на придбання книги шляхом натискання кнопки "Buy"	Клієнт ініціює запит на придбання	WORKING
3	Перевірка авторизації клієнта у системі	Перевірка системою авторизації клієнта	WORKING
4	Клієнт надає інформацію про себе, необхідну для замовлення: телефон, e-mail, бажану кількість книжок	Неавторизований клієнт заповнює форму для здійснення покупки	WORKING
5	Клієнт вказує бажану кількість книжок	Авторизований клієнт вказує лише кількість книжок	WORKING
6	Додання замовлення до персонального кабінету клієнта	Замовлення клієнта додається до БД	WORKING
7	Додання інших книжок до замовлення	Додавання до замовлення інших книжок	WORKING
8	Перегляд списку замовлень, внесення змін при необхідності	Перегляд існуючих книжок у замовленні, внесення змін	WORKING
9	Каталог товарів	База даних каталогу товарів. містить інформацію про товари	DATABASE
10	Перелік замовлень	База даних замовлень	DATABASE

Слід звернути увагу, на те, що при замовленні товару, в залежності від того, авторизований користувач чи ні, змінюється алгоритм подальшої обробки запиту. Авторизованим користувач необхідно лише заповнити інформацію про бажану кількість книжок. Неавторизованим користувачам необхідно заповнювати додаткову інформацію, зокрема: номер мобільного телефону, електрону адресу та бажану кількість книжок.

Також, авторизований користувач має можливість додавання нових товарів до замовлення, а також можливість редагування свого замовлення. При додаванні нового товару, неавторизованим користувачам необхідно заповнювати довідкову інформацію кожного разу, бо це єдиний спосіб ідентифікації неавторизованих користувачів.

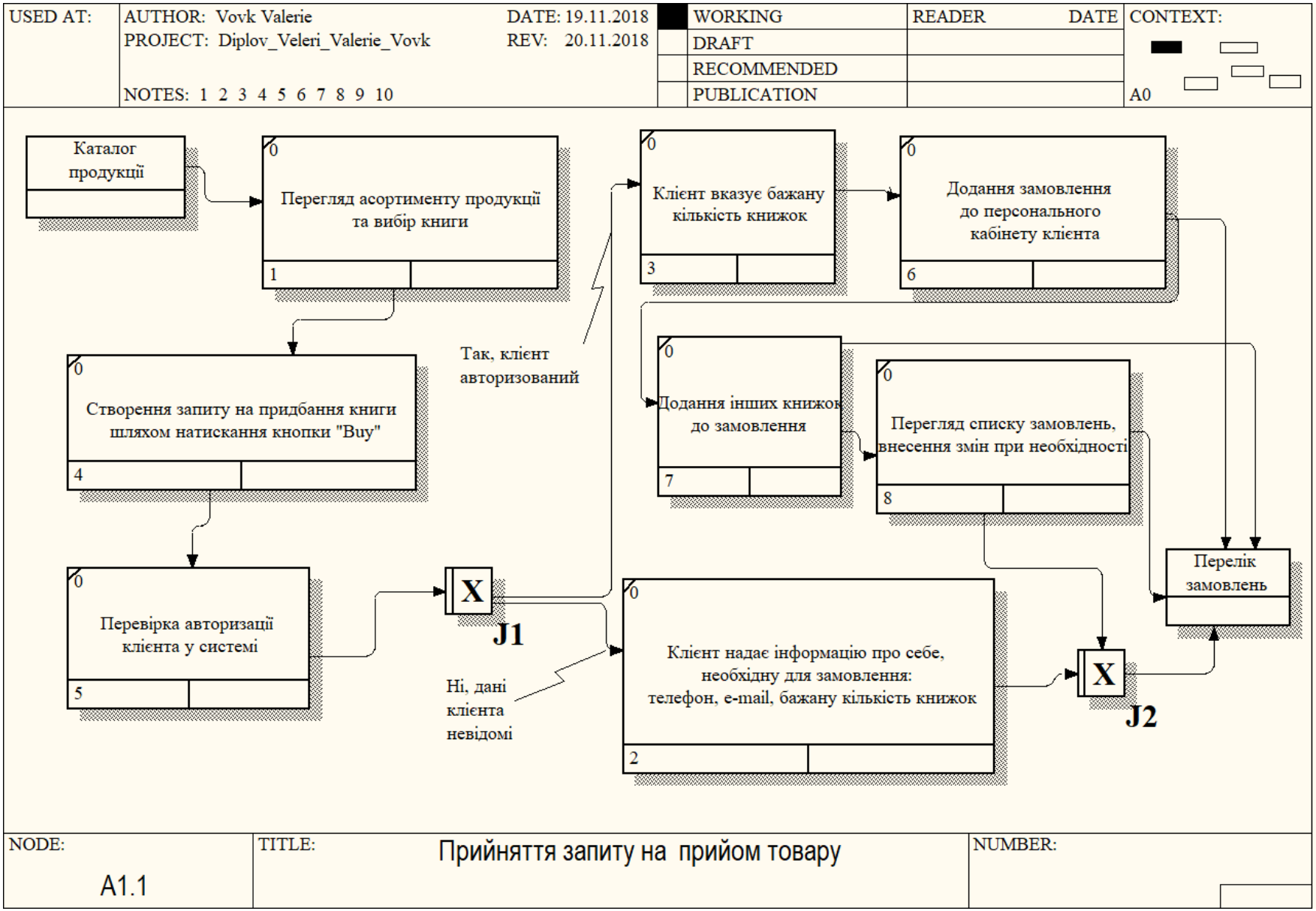


Рисунок 2.3 – Декомпозицію функціонального блоку «Замовлення товар клієнтом» у нотації IDEF3

Декомпозицію функціонального блоку «Оформлення замовлення клієнта» виконаємо у нотації DFD, яка наведена на рисунку 2.4. Опис робіт діаграми декомпозиції даного функціонального блоку наведено в таблиці 2.9.

Таблиця 2.9 – Опис функціональних блоків декомпозиції функціонального блоку «Оформлення замовлення клієнта» у нотації DFD

№ п/п	Функціональний блок	Характеристика	Статус
1	Прийняття заявки на обробку даних	Співробітник підприємства перевіряє замовлення клієнта.	WORKING
2	Перевірка наявності товару	Співробітник підприємства перевіряє наявність товару на складі.	WORKING
3	Уточнення даних у клієнта, перевірка замовлення	Співробітник підприємства уточняє у клієнта статус замовлення та іншу інформацію у разі необхідності.	WORKING
4	Оформлення замовлення	Співробітник підприємства фіксує остаточно фіксує замовлення та відправляє його на доставку.	WORKING
5	Інформація клієнтів	Список клієнтів, що користуються Інтернет-магазином та їхньою персональною інформацією.	DATASTORE
6	Перелік замовлень	Список замовлень, з їхніми характеристиками	DATASTORE
7	Каталог продукції	Товари, які являються асортиментом продукції Інтернет-магазину, з інформацію про характеристики даних товарів.	DATASTORE
8	Замовлення клієнта	Замовлення, створене клієнтом для обраного ним товару.	EXTERNAL ENTITY
9	Клієнти	Клієнти, що користуються Інтернет-магазином.	EXTERNAL ENTITY

Таким, чином ми сформували детальний бізнес-процес функціонування Інтернет-магазину «Book Shop» у якості однієї головної діаграми у нотації IDEF0 та необхідних декомпозицій основним функціональних блоків у нотаціях IDEF3 та DFD.

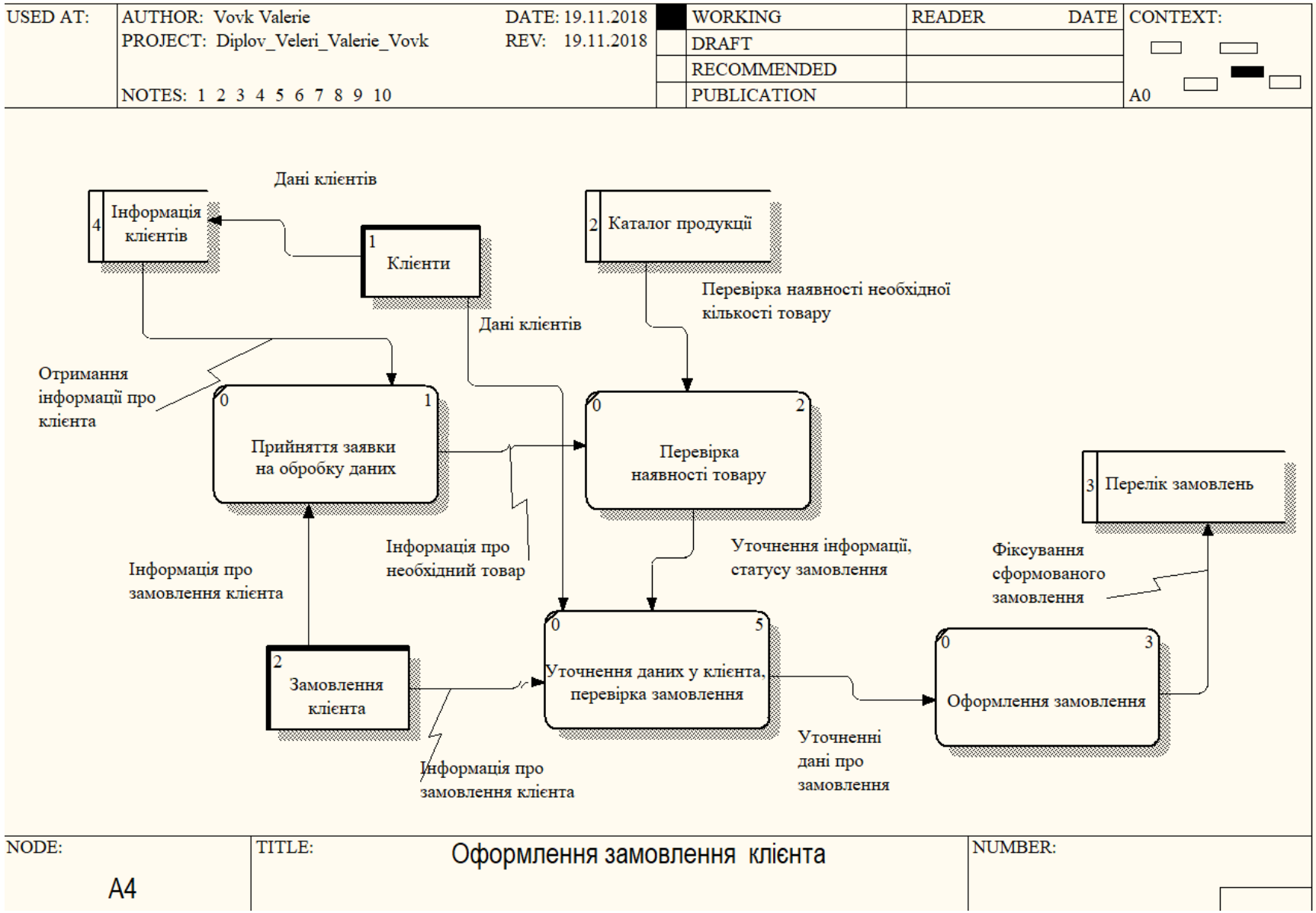


Рисунок 2.4 – Декомпозицію функціонального блоку «Оформлення замовлення клієнта» у нотації DFD

2.2 Вибір архітектури та технології автоматизованої системи

У процесі створення, проектування та реалізації автоматизованої інформаційної системи є вибір архітектури. Важливим елементом є веб-сервер, який представляє собою сервер, що приймає HTTP-запити від клієнтів та повертає їм HTTP-відповіді, разом із HTML-сторінкою або іншими даними [49].

Веб-додатки представляють собою особливий тип програм, які базуються на архітектурі «клієнт-сервер». Головною особливістю є те, що веб-додаток знаходиться на веб-сервері і, усі операції виконується на стороні сервера. Клієнт, у свою чергу, оперує запитами і відповідями з веб-серверу та відображенням відповіді на стороні клієнта, зокрема за допомогою HTML-сторінки.

Для реалізації автоматизовані системи була обрана трирівнева клієнт-серверна структура, яка передбачає наявність трьох компонент: клієнт, сервер – додатків, тобто веб-сервер та сервер бази даних. Структура даної архітектури наведена на рис. 2.5.

Рисунок 2.5 – Трирівнева архітектура клієнт-сервер

Клієнт представляє перший рівень та виконує функцію коректного відображення результатів запитів до веб-серверу, тобто це графічний інтерфейс, завдяки якому користувач взаємодіє із веб-сервером та базою даних. Даний рівень не передбачає прямих зв’язків із базою даних [49].

На другому рівні розташовується сервер додатків, на якому зосереджуються уся бізнес-логіка додатку. Третій рівень представляє собою реляційну базу даних або об’єктно-орієнтовану СУБД та виконує функцію збереження, оновлення і доступ до даних [49].

Для реалізації простої конфігурації, сервер додатків та бази даних фізично суміщують на одному комп’ютері. До цього комп’ютера за допомогою мережі підключаються термінали [49].

Для забезпечення безпеки, надійності та можливості масштабування використовують іншу конфігурацію. Сервер бази даних розміщується на віддаленому кластері чи комп’ютері, до якого за допомогою мережі підключається один чи декілька серверів додатків. Термінали, також, за допомогою мережі підключаються до серверів додатків.

Дана конфігурація забезпечує більшу надійність та безпеку, бо клієнт не має безпосереднього доступу до бази даних. Також, дана архітектура передбачає наступні переваги: логічне розділення між рівнями, можливість паралельного розвитку кожного з рівнів, завдяки їхній незалежності, можливість масштабування. До недоліків архітектури можна віднести: високі вимоги до продуктивності серверів додатків і бази даних та швидкості каналу мережі, який забезпечує передачу даних між сервером додатків та бази даних, ускладнення проектування додатків, необхідність досвіду в об’єктно-орієнтованій концепції.

У якості веб-серверу обрано «Weblogic». Даний веб-сервер підтримує більшість сучасних операційних систем, зокрема Linux, Unix та Windows, та баз даних: Microsoft Sql Serverl, Sybase, DB2, Oracle [50–53].

Запити до веб-сервісу будемо обробляти за допомогою сервлетів та допоміжних класів. Сервлет – компонент на стороні сервера в моделі клієнт-сервер. Розташовується в контейнері сервлетів, якому присвоєно певний шаблон URL [50–53].

Алгоритм роботи сервлетів та веб – серверу представлений на рис 2.6 та складається із наступних пунктів [50–53]:

1. Клієнт робить запит на сервер.
2. Сервер запускає сервлет.
3. Сервлет обчислює результат для сервера і не завершує роботу.
4. Сервер повертає відповідь клієнту.
5. Інший клієнт робить запит на сервер.
6. Сервер звертається до сервлету знову і обробляє інформацію (пункт 3).

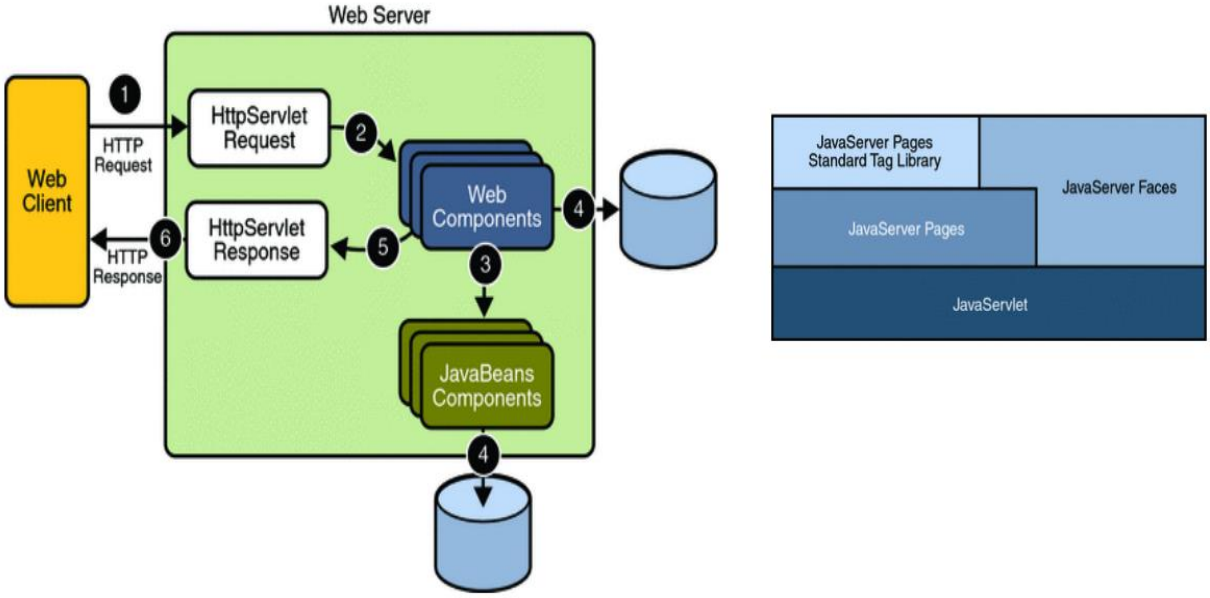


Рисунок 2.6 – Взаємодія веб–серверу та сервлетів

У якості бази даних обрано «Oracle». Дана СУДБ передбачена для створення проектів будь-якої складності. Завдяки засобам масштабування, в базі можна зберігати необмежену кількість інформації. Важливою перевагою є підтримка ієрархічних запитів та багатоплатформність.

У якості мови програмування обрано «Java», яка представляє собою об’єктно-орієнтовану мову програмування, що була розроблена компанією «Sun Microsystems». Додатки, написані на даній мові компілюються в спеціальний байт-код, який може працювати на будь-якій віртуальній Java-машині та не залежати від комп’ютерної архітектури [54 – 57].

До переваг «Java» можна віднести:

- незалежність від платформи, на якій виконується додаток;

- об’єктно-орієнтоване спрямування мови програмування;
- синтаксис «Java» схожий на мови програмування «C#» та «C++»;
- є простою мовою програмування для розуміння та навчання;
- підтримка багатопотокових додатків;
- висока надійність, яка забезпечується:

a) виключена можливість явного виділення та звільнення пам’яті через механізм автоматичного збору та очищення сміття;

b) виключення можливості переплутання між собою операторів присвоювання та порівняння на рівність;

c) виключення множинного спадкування класів, в «Java» можливе лише одне успадкування від класу, але безліч успадкувань від інтерфейсів.

2.3 Функціональна частина

Система повинна забезпечувати механізм автоматизованого проведення операцій, що пов’язані із переглядом асортименту товару та його замовленням, веденням обліку клієнтської бази та стану складу підприємства.

Наведемо характеристику функціоналу розроблювальної системи більш детально. поділивши функції на дві групи: для клієнтів системи, співробітників підприємства та самої системи.

Для клієнтів, система повинна забезпечувати наступні функції:

- перегляд асортименту продукції підприємства та детальних відомостей про товари;
- оформлення, перегляд відомостей та редагування замовлення;
- створення власного профілю у системі та його редагування;
- перегляд інформаційних сторін системи;
- можливість зворотнього зв’язку;
- функція швидкого оформлення замовлення для авторизованих користувачів. Авторизований користувач повинен заповнювати лише відомості про бажану кількість товару.

Для співробітників, система повинна забезпечувати наступні функції:

- додавання та редагування відомостей про товар;
- видалення товарів;
- створення і видалення рубрик та категорій;
- перегляд замовлень клієнтів;
- змінення профілів клієнтів;
- редагування власного профілю.

Система повинна забезпечувати наступні функції:

- доступ до системи у режимі 24 години на добу;
- оновлення даних у базі даних та коректне відображення наявної інформації;
- розподілення ролів користувачів та в залежності від ролі відображення доступної для даного користувача інформації і функцій.

Вхідною інформацією для запуску системи є інформація про перелік клієнтів, список товарів, перелік рубрик та категорій, перелік замовлень клієнтів. Результатом роботи є вихідна інформація у виглядів інформації-документу: «Перелік товарів», «Перелік замовлень», «Профілі клієнтів».

2.4 Підсистеми забезпечення функціональної частини

Для реалізації системи автоматизації діяльності підприємства роздрібного бізнесу з електронної комерції обрано наступні програмні продукти: «IntelliJ IDEA Ultimate», «Java SE Development Kit 8», «Oracle WebLogic Server 12c», «Oracle Database XE 12c». В процесі розробки додатку, для організації доступу до бази даних будемо використовувати програмне забезпечення «SQL Developer».

«IntelliJ IDEA Ultimate» - це провідне інтегроване середовище для швидкого та ефективного програмування на мові «Java». Комерційна версія даного програмного забезпечення забезпечує виконання наступного функціоналу:

- розумне автодоповнення коду, засоби аналізу якості коду, проста та зручна навігація, форматування для наступним мов: «Java», «Groovy», «HTML», «JavaScript», «XML», «CSS», «LESS» та ін.;

- підтримка популярних фреймворків та платформ, зокрема: «Java EE», «Spring Framework», «Play Framework», «Android», «Grails» та ін.;

- засоби роботи з базами даних та sql-файлами$

- інструменти для тестування та аналіз покриття коду;

- інтеграція з комерційними системами управління версіями «Team Foundation Server», «Perforce» та ін.;

- підтримка роботи із системами контролю версій, зокрема: «SVN», «GitHub» та ін.

«Java SE Development Kit 8» представляє собою безкоштовний комплект розробника на мові «Java», що включає в себе компілятор «Java», стандартні бібліотеки класів на мові «Java», документацію, виконавчу систему «Java» (JRE) та різноманітні утиліти.

«Oracle WebLogic Server 12c» - це хмарова корпоративна Java-платформа, що дозволяє в повній мірі реалізувати переваги хмарових обчислень. В дану платформу входить сервер додатків J2EE, портал, інтеграційні продукти, середовище для розробки додатків та JRockit – власна віртуальна машина Java.

«Oracle Database XE 12c» - об’єктно-реляційна система управління базами даних компанії «Oracle».

«SQL Developer» - інтегроване середовище розробки на мовах «Sql» та «Pl/sql», з можливістю адміністрування баз даних та орієнтована на застосування в середовищі «Oracle Database».

Для успішного функціонування системи для автоматизації діяльності підприємства роздрібного бізнесу з електронної комерції необхідно встановлення відповідного технічного обладнання, до якого відноситься: обчислювальні машини, периферійні технічні засоби, мережеві комунікації.

Для реалізації системи необхідно два потужних сервера для сервера додатків та бази даних, і робочі станції. Рекомендована конфігурація для сервера додатків наведена у таблиці 2.10, сервера бази даних – таблиця 2.11.

Таблиця 2.10 – Рекомендована конфігурація сервера додатків

№ п/п	Компонента	Характеристика
1	Процесор	Intel® Core™ i3-4130 CPU 3.40 ГГц
2	Операційна система	Linux
3	Оперативна пам’ять	64 Гбайт
4	Жорсткий диск	SSD Samsung 860 EVO 2.5 500 GB (MZ-76E500BW)
5	Мережевий адаптер	Intel E1G44ETBLK

Таблиця 2.11 – Рекомендована конфігурація сервера бази даних

№ п/п	Компонента	Характеристика
1	Процесор	Intel® Core™ i3-4130 CPU 3.40 ГГц
2	Операційна система	Linux
3	Оперативна пам’ять	32 Гбайт
4	Жорсткий диск	SAS 500GB
5	Мережевий адаптер	Intel E1G44ETBLK

Достатніми умова функціонування робочих станцій є наявність операційної системи «Windows 8» або «Windows 10», браузер «Google Chrome», доступ до мережі Інтернет та підключення до локальної мережі.

Після впровадження автоматизованої системи організаційна структура підприємства залишиться без змін. Необхідно замінити вже існуючий сервер на два нових сервера: сервер додатків та бази даних; виділи одного співробітника з ІТ- відділу для підтримки та наповнення сайту інформацією та двох працівників із відділу продажів, що будуть займатися обробкою замовлень від клієнтів здійснених через мережу Інтернет.

3 РЕАЛІЗАЦІЯ ПРОТОТИПУ АВТОМАТИЗОВАНОЇ СИСТЕМИ НА ПРИКЛАДІ КНИЖКОВОГО ІНТЕРНЕТ-МАГАЗИНУ

3.1 Структура та особливості реалізації інформаційного забезпечення

Реалізація інформаційного забезпечення, безпосередньо, починається із формування вимог до реляційної бази даних. Створена база має бути надійною та масштабованою. В ній зберігатиметься інформація про замовлення користувачів, їх персональні дані, дані працівників магазину, тощо. Інформація, що зберігатиметься у базі даних з кожним днем буде зростати.

Модель бази даних побудуємо за допомогою моделі «сутність – зв'язок». Сутність представляє собою деяку модель об'єкту, що існує в реальному світі. Кожна сутність має набір властивостей та атрибутів, що її характеризують. За допомогою даної моделі спроектуємо базу даних.

База даних включатиме у себе наступні сутності:

- «Item» (Загальна сутність, від якої відбувається наслідування інших сутностей);
- «Author» (Автор);
- «Customer» (Користувачі системи, зокрема клієнти);
- «Orders» (Замовлення);
- «Properties» (Властивості книжок);
- «Content_order» (Зміст замовлення).

База даних матиме ієрархічну структуру, сутність «Item» міститиме у собі три сутності: розділ, рубрика та книга. Також, між сутностями існують зв'язки – це віртуальний зв'язок, заснований на взаємодіях реального світу. Відношення між сутностями, які відображаються у базі даних, як таблиці, відбувається за допомогою зовнішніх ключів. Існують три типи відношення зовнішніх ключів: «один до одного», «один до багатьох», «багато до багатьох» [53 – 54, 58 – 63].

При створенні реляційної бази даних необхідно [53 – 54, 58 – 63]:

- для кожної сутності створити відповідну таблицю з переліком всіх атрибутів у вигляді полів таблиці;
- задати первинні ключі для таблиць, які мають приймати унікальне та ненульове значення;
- створити зовнішні ключі, звертаючи увагу на те, що для таблиць із зв'язком «багато до багатьох» необхідно створити нову асоціативну таблицю, що включатиме первинні ключі даних таблиць і зовнішні ключі на дані таблиці;
- при необхідності створити лічильники для додавання записів у таблиці.

Розглянемо сутність утворених таблиць бази даних та наддамо характеристику. Структура таблиці, що описує сутність «Item» наведена у таблиці 3.1.

Таблиця 3.1 – Структура таблиці "Item"

№ п/п	Назва атрибуту	Тип даних	Обмеження		Призначення атрибуту
			Пусте значення	Значення за замовчуванням	
1	ID_ITEM	NUMBER	No	–	Первинний ключ.
2	NAME	VARCHAR2 (50 BYTE)	Yes	–	Назва сутності.
3	PARENT_ID	NUMBER	Yes	–	ID батьківського об'єкту.
4	DESCRIPTION	VARCHAR2 (150 BYTE)	Yes	–	Опис сутності.
5	TYPE	NUMBER	No	–	Тип сутності: 0, якщо це книга, 1-рубрика, 2-категорія.
6	ID_PROPERTIES	NUMBER	Yes	–	Ключ для посилання на таблицю із властивостями книги.
7	PICTURE	VARCHAR2 (250 BYTE)	Yes	–	Зберігається назва на зображення сутності.

Атрибут «ID_ITEM» є первинним ключом даної таблиці, має унікальне, ненульове значення та призначений для ідентифікації рядків таблиці. Атрибут «TYPE» визначає тип сутності: якщо це книга, вказуємо значення даного атрибуту «0», «1» для рубрики та «2» для категорії. Даний атрибут надає можливість ефективно оперувати сутностями в sql-запитах. Атрибут

«PARENT_ID» є зовнішнім ключом, що посилається на батьківський об’єкт: для книги – це рубрика, для рубрики – категорія, для категорії батьківського об’єкту не існує і значення даного атрибуту має дорівнювати «null». Атрибут «ID_PROPERTIES» є зовнішнім ключом на таблицю «Properties», що містить атрибути властивостей книжок. Даний атрибут заповнюється лише для об’єктів із типом «книга», тобто значенням атрибуту «type», що дорівнює «0».

Таблиця «Properties» описує властивості книжок, характеристики наведені у таблиці 3.2. Атрибут «ID_BOOK» є первинним ключом даної таблиці, має унікальне, ненульове значення та призначений для ідентифікації рядків таблиці.

Таблиця 3.2 – Структура таблиці "Properties"

№ п/п	Назва атрибуту	Тип даних	Обмеження		Призначення атрибуту
			Пусте значення	Значення за замовчуванням	
1	ID_BOOK	NUMBER	No	–	Первинний ключ, на який посилається поле «ID_PROPERTIES» таблиці «Item».
2	PAGES	NUMBER	Yes	–	Вказує на кількість сторінок книги.
3	PRICE	FLOAT	Yes	–	Вказує на ціну книги.
4	AMOUNT	NUMBER	Yes	–	Вказує на кількість наявних книжок на складі.
5	ID_AUTHOR	NUMBER	Yes	–	Атрибут є зовнішнім ключом, що посилається на автора книги.

Таблиця «Author» описує сутність «Автор», характеристика наведена у таблиці 3.3. Атрибут «ID_ AUTHOR» є первинним ключом даної таблиці.

Таблиця 3.3 – Структура таблиці "Author "

№ п/п	Назва атрибуту	Тип даних	Обмеження		Призначення атрибуту
			Пусте значення	Значення за замовчуванням	
1	ID_AUTHOR	NUMBER	No	–	Первинний ключ, на який посилається поле «ID_PROPERTIES» таблиці «Item».

Продовження таблиці 3.5

№ п/п	Назва атрибуту	Тип даних	Обмеження		Призначення атрибуту
			Пусте значення	Значення за замовчуванням	
2	SURNAME	VARCHAR2 (50 BYTE)	Yes	–	Вказує на кількість сторінок книги.
3	NAME	VARCHAR2 (50 BYTE)	Yes	–	Вказує на ціну книги.

Таблиця «Customer» описує сутність «Користувач», характеристика наведена у таблиці 3.4. Дана сутність описує користувачів системи, зокрема системних користувачів (адміністраторів, тощо) та клієнтів магазину. Атрибут «ID_ CUSTOMER» є первинним ключом даної таблиці, має унікальне, ненульове значення та призначений для ідентифікації рядків таблиці. Для визначення ролі користувача використовується атрибут «ROLE», що приймає різні значення в залежності від ролі користувача: «1»-звичайний користувач, «0»-системний користувач, «10»-адміністратор.

Таблиця 3.4 – Структура таблиці " Customer "

№ п/п	Назва атрибуту	Тип даних	Обмеження		Призначення атрибуту
			Пусте значення	Значення за замовчуванням	
1	ID_CUSTOMER	NUMBER	No	–	Первинний ключ.
2	LOGIN	VARCHAR2 (50 BYTE)	Yes	–	Логін користувача.
3	PASSWORD	VARCHAR2 (20 BYTE)	Yes	–	Пароль користувача.
4	E_MAIL	VARCHAR2 (50 BYTE)	Yes	–	Електрона адреса користувача.
5	PHOME_NUBMER	VARCHAR2 (12 BYTE)	Yes	–	Телефон користувача
6	ROLE	NUMBER	No	1	Роль користувача: «1»-звичайний користувач, «0»-системний користувач, «10»-адміністратор.

Таблиця «Orders» описує сутність «Замовлення», характеристика наведена у таблиці 3.5. Дана сутність описує замовлення, що створюють користувачі інтернет-магазину.

Таблиця 3.5 – Структура таблиці " Orders "

№ п/п	Назва атрибуту	Тип даних	Обмеження		Призначення атрибуту
			Пусте значення	Значення за замовчуванням	
1	ID_ORDER	NUMBER	No	–	Первинний ключ.
2	ID_CUSTOMER	NUMBER	No	–	Зовнішній ключ на таблицю «Customers».
3	DATA	DATE	Yes	–	Пароль користувача.

Атрибут «ID_ ORDER» є первинним ключом даної таблиці, має унікальне, ненульове значення та призначений для ідентифікації рядків таблиці.

Таблиця «Content_order» описує сутність «Зміст замовлення», характеристика наведена у таблиці 3.6.

Таблиця 3.6 – Структура таблиці " Content_order"

№ п/п	Назва атрибуту	Тип даних	Обмеження		Призначення атрибуту
			Пусте значення	Значення за замовчуванням	
1	ID_CONTENT	NUMBER	No	–	Первинний ключ.
2	ID_ORDER	NUMBER	No	–	Зовнішній ключ на таблицю «Orders».
3	ID_BOOK	NUMBER	No	–	Зовнішній ключ на таблицю «Item».
4	AMOUNT	NUMBER	Yes		Кількість книжок

Повна структура бази даних із усіма зв’язками, наведена на рис. 3.1.

Далі необхідно переконатися, що база даних відповідає формам нормалізації. Нормалізація представляє процес поділу даних на окремі пов'язані таблиці. Нормалізація усуває надмірність даних і, тим самим, надає можливість уникнути порушення цілісності даних при їх зміні, а саме уникнути аномалій зміни [53 – 54, 58 – 63].

Характеристика нормальних форм наведена у таблиці 3.7 [53 – 54, 58 – 63].

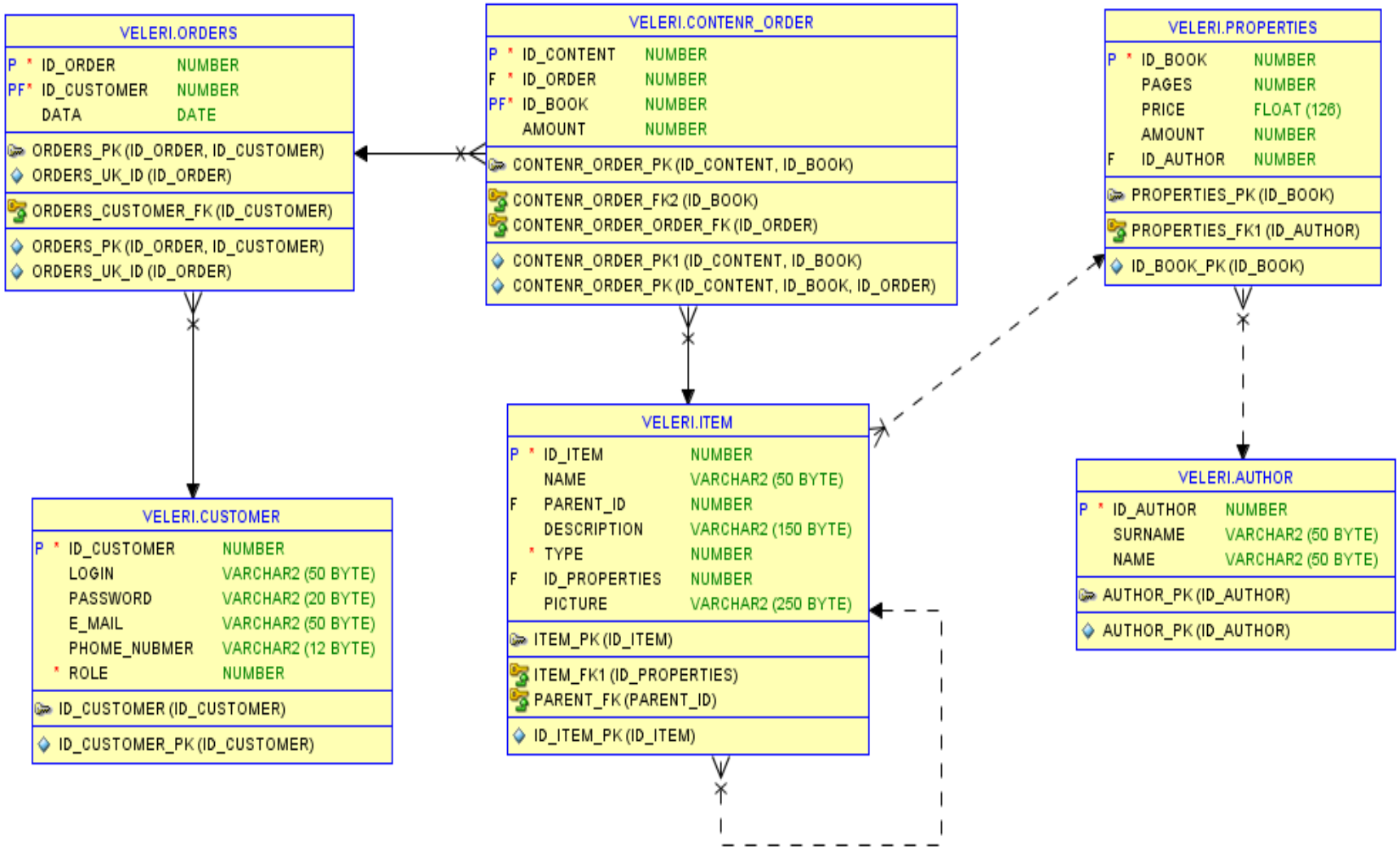


Рисунок 3.1 – Структура бази даних

Таблиця 3.7 – Характеристика нормальних форм нормалізації

№ п/п	Назва форми	Абревіатура	Характеристика
1	Перша нормальна форма	1NF	Дані, що зберігаються на перетині рядків і стовпців повинні представляти скалярний значення; таблиці не повинні містити повторюваних рядків.
2	Друга нормальна форма	2NF	Кожен стовпець, який не є ключем, повинен залежати від первинного ключа.
3	Третя нормальна форма	3NF	Кожен стовпець, який не є ключем, повинен залежати тільки від первинного ключа.
4	Нормальна форма Бойса-Кодда	BCNF	Являє собою більш сувору версією третьої нормальної форми і застосовується у наступних випадках відношень: відношення має два або більше потенційних ключа; два і більше потенційних ключа є складовими; вони перетинаються, тобто мають хоча б один атрибут.
5	Четверта нормальна форма	4NF	Застосовується для усунення багатозначних залежностей – залежностей, де стовпчик з первинним ключем має зв'язок один-до-багатьох зі стовпцем, який не є ключем. Також усуває некоректні відносини багато-до-багатьох.
6	П’ята нормальна форма	5NF	Форма розділяє таблиці на менші для усунення надмірності даних. Розбиття йде до тих пір, поки не можна буде відтворити оригінальну таблицю шляхом об'єднання малих таблиць.
7	Шоста нормальна форма	6NF (domain key normal form)	Кожне обмеження в зв'язках між таблицями має залежати тільки від обмежень ключа і обмежень домену, де домен являє набір допустимих значень для стовпця. Ця форма запобігає додавання неприпустимих даних шляхом установки обмеження на рівні відносин між таблицями, але не на рівні таблиць або стовпців.

Запит створення та наповнення бази даних наведений у додатку Б.

3.2 Структура та особливості реалізації алгоритмічного забезпечення

В залежності від ролі користувачів системі, відрізняються функції, які система повинна підтримувати. Розглянемо алгоритм системи для роботи з клієнтами, який наведений на рис. 3.2. [47, 49]

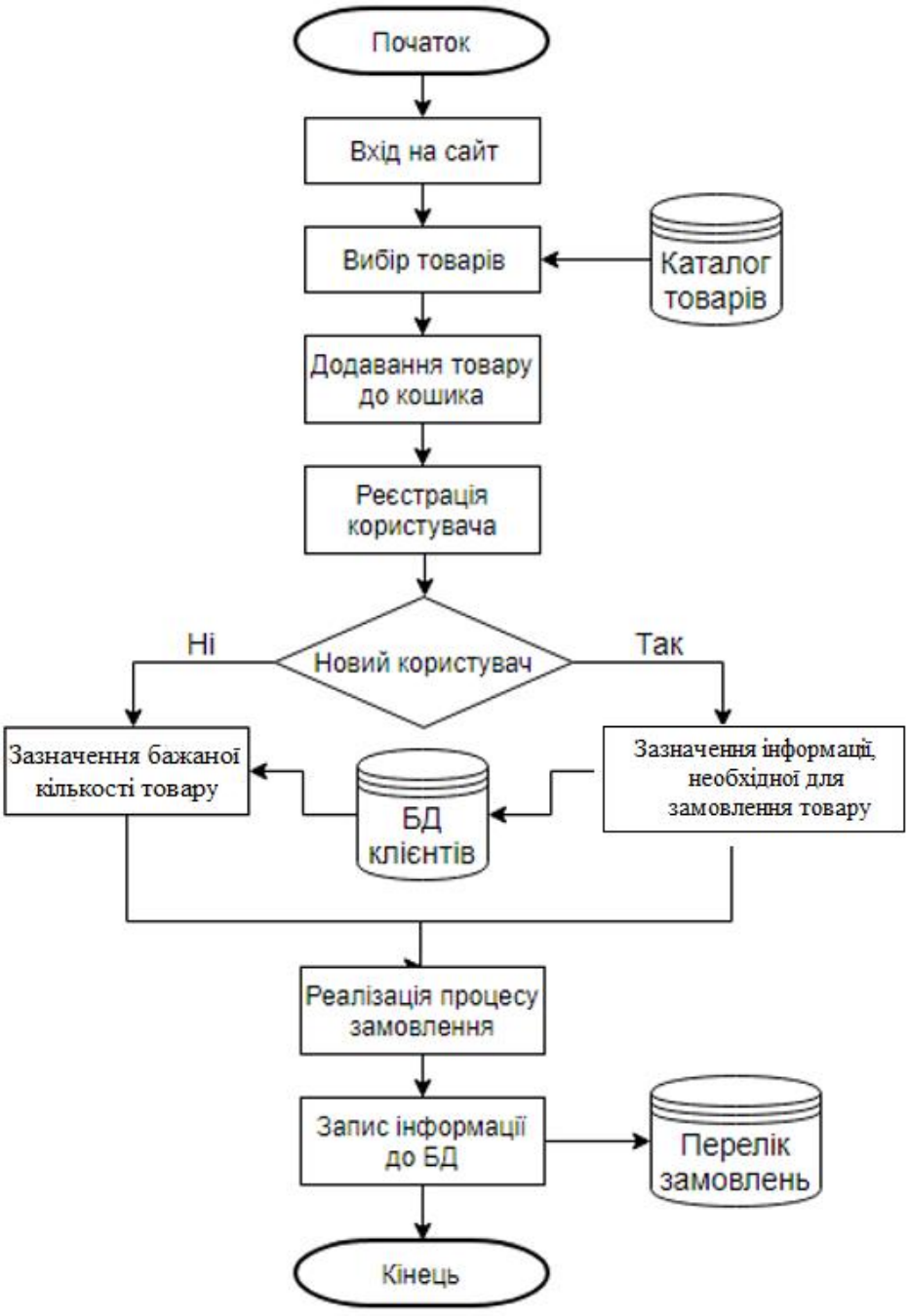


Рисунок 3. 2 – Блок схема роботи додатку для клієнтів

Після запуску системи, клієнт потрапляє на головну сторінку, де має змогу переглянути асортимент продукції. Після перегляду, клієнт обирає товар, який він хоче купити за ініціює процес оформлення замовлення. При оформленні замовлення, якщо клієнт є авторизованим користувачем, він має вказати лише бажану кількість товару. Неавторизованим користувачам необхідно вказувати додаткову інформацію (номер телефону та электрону адресу). Далі замовлення фіксується у базі даних та передається для оброблення співробітниками підприємства.

Алгоритм системи для роботи із співробітниками наведений на рис. 3.3.

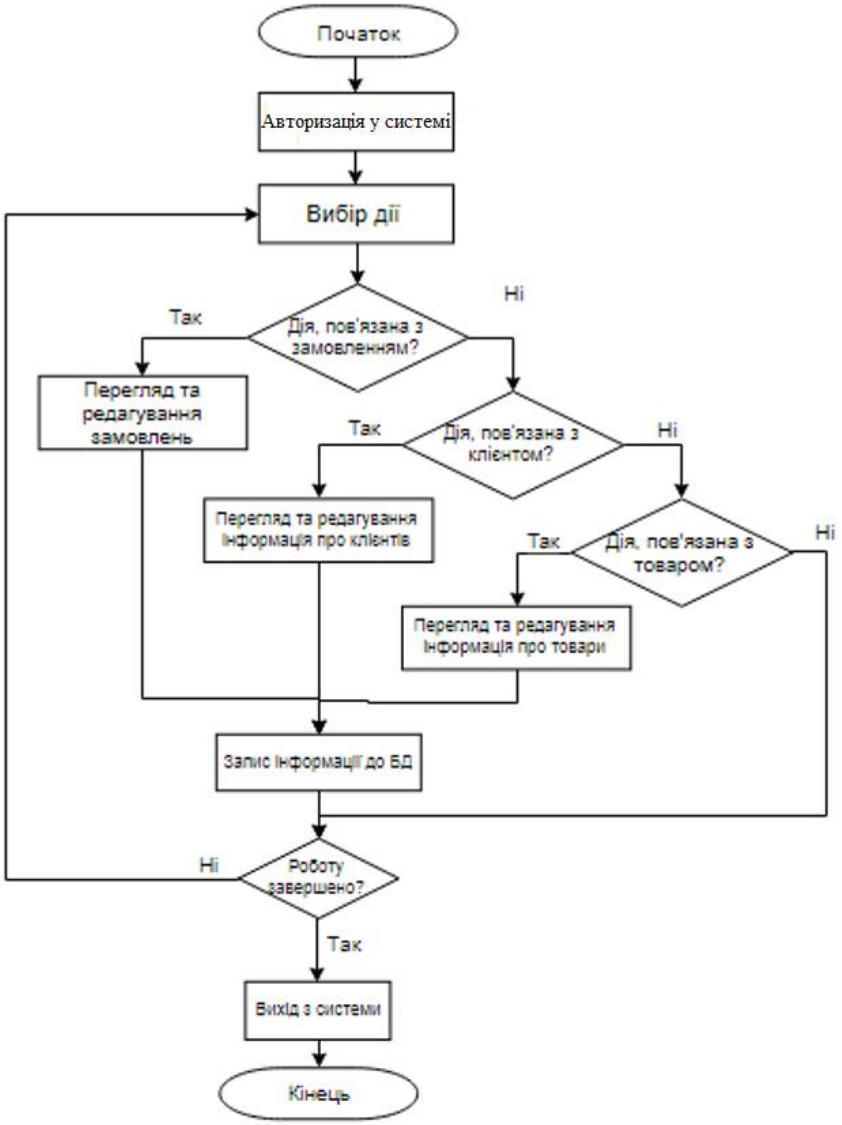


Рисунок 3.3 – Блок схема роботи програми для співробітників

Для співробітників підприємства, в залежності від виконуваної дії, системою передбачений різний набір функцій. Процеси пов’язані із замовленням товарів, дають можливість співробітнику переглядати стан замовлення та інформацію про клієнта, що зробив дане замовлення. Після уточнення усією інформація, співробітник підтверджує замовлення. Процес, що пов’язаний із клієнтами підприємства дозволяє переглядати наявну інформацію про клієнтів та вносити зміни. Процес, що пов’язаний із товаром дозволяє переглядати детальну інформацію про товар, змінювати її, додавати чи видаляти товари, рубрики та категорії. Також, співробітник має змогу вести облік товарів в системі. Після завершення роботи, співробітник виходить із системи.

3.3 Контрольний приклад та інструкція з використання

Як зазначалося раніше, додаток буде реалізований за допомогою мови програмування «Java» в середовищі «IntelliJ IDEA Ultimate» з використанням база даних «Oracle Database XE 12c» та веб-серверу «Oracle WebLogic Server 12c».

Розглянемо структуру проекту, що наведена на рис. 3.4. Проект розподілений на декілька частин. В директорії «java» міститься програмний код класів проекту, в директорії «webapp» містяться об’єкти, що відносяться до веб-частини проекту. Проект реалізований із дотриманням шаблону «Модель–представлення–контролер» («MVC»). «Модель» відображає поведінки програмного додатку, «представлення» реалізує зовнішній вигляд, а «контролер» забезпечує взаємодію «представлення» із «моделлю», перетворюючи вхідні дані на команди для «моделі» чи «представлення» [64 – 70].

Зв’язок додатку з базою даних організовується за допомогою технології JNDI через веб-сервер. Програмний код, що забезпечує даний зв’язок знаходиться у класі «OracleDataAccess» та наведений у лістингу 3.1.

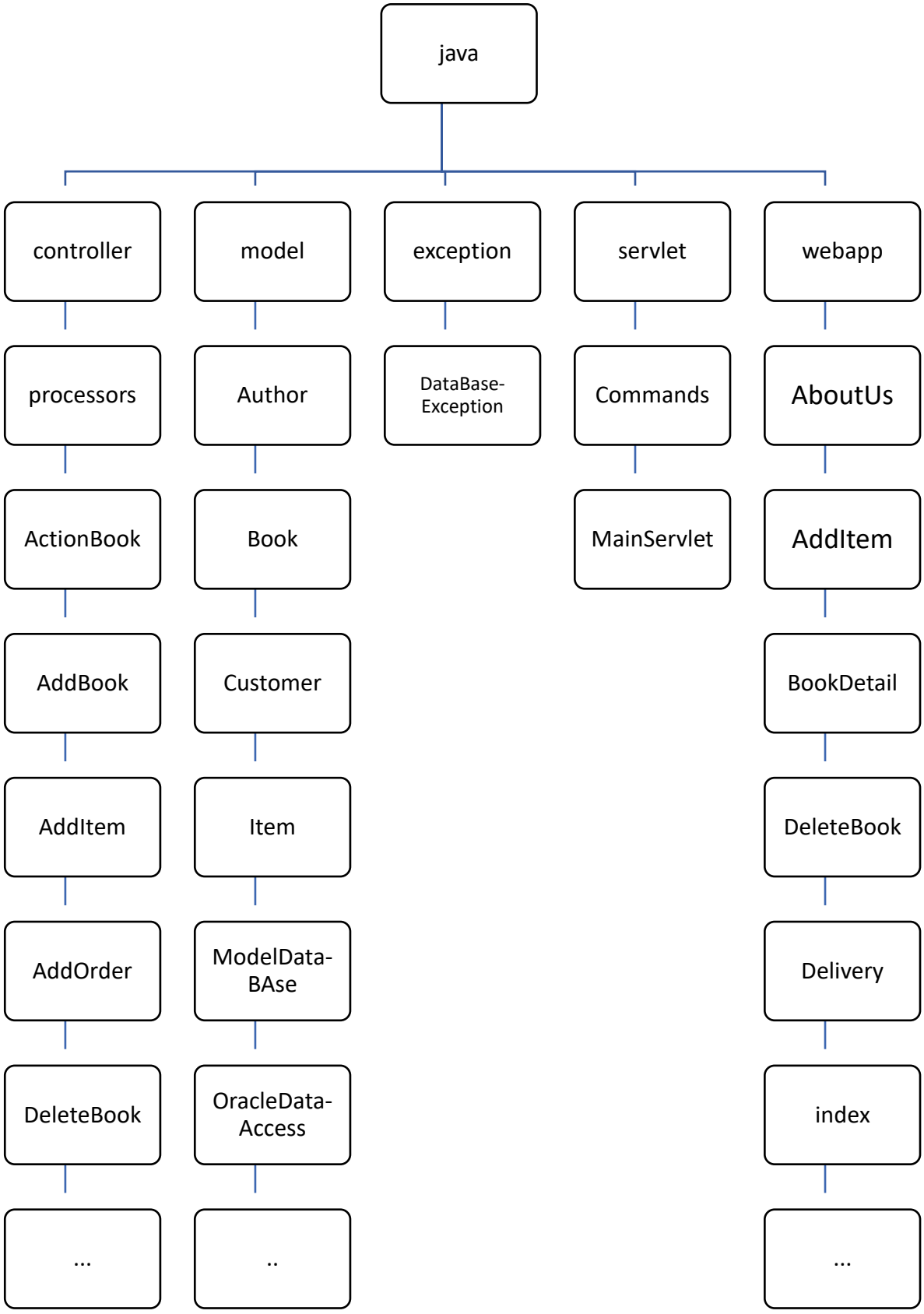


Рисунок 3.4 – Структура проекту

Лістинг 3.1. – Код класу «OracleDataAccess», що забезпечує доступ до з’єднання з базою даних

```
1:      protected static class Singleton {
2:         public static final OracleDataAccess _INSTANCE = new
2a:               OracleDataAccess();
3:       }
4:      public static OracleDataAccess getInstance() {
5:         return Singleton._INSTANCE;
6:      }
7:      private DataSource ds;
8:      private Context ctx;
9:      private Hashtable ht = new Hashtable();
10:     private OracleDataAccess(){
11:     ht.put(Context.INITIAL_CONTEXT_FACTORY,
12:           "weblogic.jndi.WLInitialContextFactory");
13:            ht.put(Context.PROVIDER_URL, "t3://localhost:7001");
14:     try {
15:       ctx = new InitialContext(ht);
16:       ds = (javax.sql.DataSource) ctx.lookup("JNDI_MPLS");
17:     } catch (NamingException e) {
18:       LOG.error("InitialContext or DataSource error", e);
19:     }finally {
20:     try {
21:        if (ctx != null) {ctx.close();}
22:     } catch (NamingException e) {
23:        LOG.error("error of close connection", e);
24:      }
25:     }
26:     }
```

Запити до веб-сервісу обробляємо за допомогою сервлетів та допоміжних класів. Створення сервлету наведено у лістингу 3.2, а допоміжного класу початкової сторінки у лістингу 3.3.

Лістинг 3.2.Створення сервлету

```
1:    @WebServlet(name = "MainServlet")
2:    public class MainServlet extends HttpServlet {
3:    private         Map<String,         Object>         map         =
3a:controller.servlets.Commands.getInstance().getCommands
3b:Map();
4:    protected void doPost(HttpServletRequest request,
4a:    HttpServletResponse
4b:    response) throws ServletException, IOException {
5:    processRequest(request, response);
6:    }
7:    protected void doGet(HttpServletRequest request,
7a:     HttpServletResponse
7b:     response) throws ServletException, IOException {
8:    processRequest(request, response);
```

Продовження лістингу 3.2

```
9:    }
10:   protected void processRequest(HttpServletRequest
10a:request,
11:   10b:   HttpServletResponse response) throws
10c:ServletException, IOException { GeneralProcess process;
12:   process = (GeneralProcess) map.get(action);
13:   if (process == null) {
14:   process = (GeneralProcess)
19a:map.get(controller.servlets.Commands.ACTION_WELCOME);
15:   }
16:   if (process != null) {
17:   try {
18:   process.process(request, response);
19:   } catch (DataBaseException e) {
20:   // LOG.error(e);
21:   }
22:   }
23:   }
24:  }
```

Лістинг 3.3. Створення допоміжного класу початкової сторінки

```
1:    public class Welcome implements  GeneralProcess {
2:    public void process(HttpServletRequest request,
2a:           HttpServletResponse
2b:           response) throws DataBaseException {
3:     код методу
4:    Commands.forward("/index.jsp", request, response);
5:      }
6:    }
```

Розглянемо роботу клієнта із системою. Головна сторінка додатку зображена на рис. 3.5. На даній сторінці розміщено асортимент продукції, яку може переглядати клієнт. Також, дана сторінка містить навігаційну панель, інформаційний блок, що містить інформацію про графік роботи магазину та номер гарячої лінії, а також блок «пошуку», за допомогою якого клієнт може знайти необхідну інформацію чи товар.

Верхня навігаційна панель містить декілька активних посилань, за допомогою яких клієнт може: переглянути асортимент продукції, переглянути контактну інформацію, інформацію про доставку замовлень, залишити відгук, авторизуватися чи створити новий профіль.

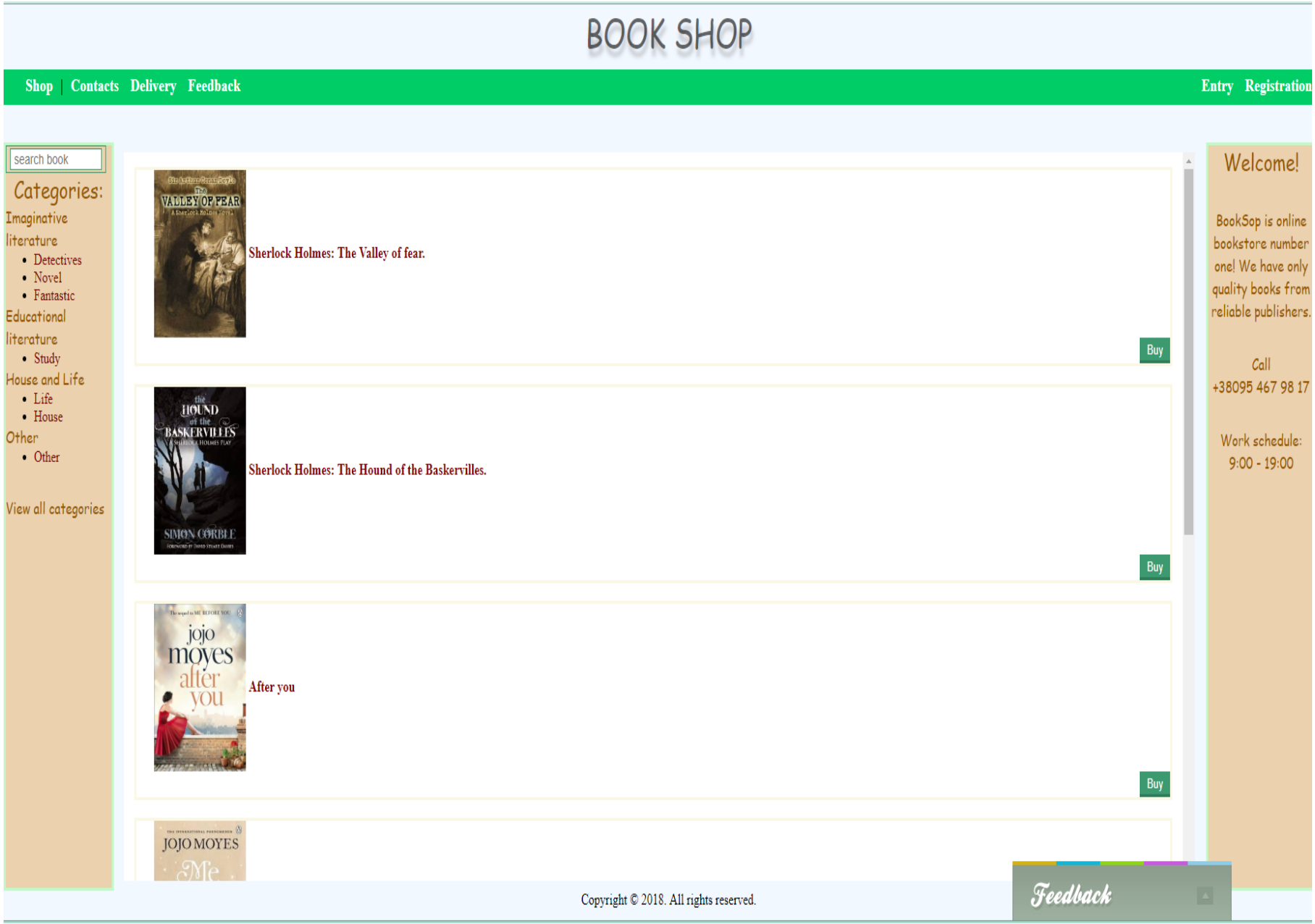


Рисунок 3.5 – Головна сторінка додатку

Асортимент продукції на головній сторінці наведений не повністю. Клієнт може переглянути декілька книжок та потім розширювати список, за допомогою команди «view more books», яка представлена на рис. 3.6. Після натискання на дану команду, перелік товарів збільшується.

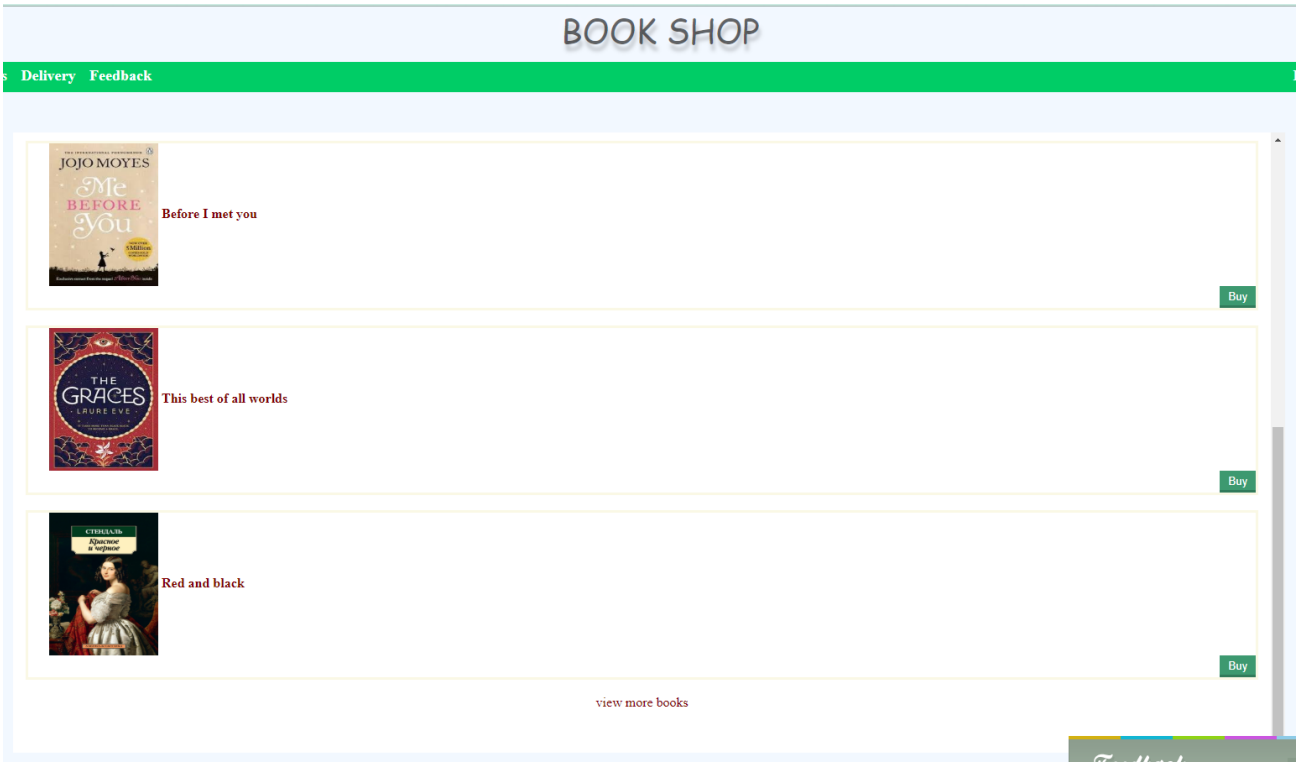


Рисунок 3.6 – Команда для розширення переліку товарів

Дана можливість реалізовується за допомогою методу «getAmountOfBooks» (лістинг 3.4), який у свою чергу отримує дані за допомогою sql-запиту, що наведено у лістингу (3.5)

Лістинг 3.4. – Код методу «getAmountOfBooks»

```
1:    /**
2:    * Method return amount of books that you need.
3:    * @param amount of books.
4:    * @return List<Book>.
5:    * @throws DataBaseException Exception with data.
6:    */
7:    public List<Book> getAmountOfBooks(int amount) throws
```

Продовження лістингу 3.4

```
7a:  DataBaseException {
8:   int page = 1;
9:   Connection connection       = getConnection();
10:  ResultSet result            = null;
11:  PreparedStatement statement = null;
12:  List<Book> listBooks = new ArrayList<Book>();
13:  try {
14:  statement =
14a:connection.prepareStatement(SqlScripts.SELECT_PAGE_OF_
14b:LIST_BOOKS); //задання sql-запиту для виконання
15:  statement.setInt(1, amount); //встановлення 1-го параметру
16:  statement.setInt(2, page); //встановлення 2-го параметру
17:  result    = statement.executeQuery(); //виконання запиту
18:       while (result.next()) {
19:         listBooks.add(getBook(result));//додавання результатів
20:       }
21:  } catch (Exception e) {
22:     throw new DataBaseException("Exception with data from
22a: database", e);//обрабка можливої помилки
23:  } finally {
24:  disconnect(connection, result, statement); //закриття ресурсів
25:  }
26:  return listBooks;//повернення результату у вигляді змінної
27:  }
```

Лістинг 3.5. – Код sql-запиту для отримання інформації про часткову кількість об’єктів

```
1:   SELECT * FROM
2:   (
3:    SELECT
4:      items.*,
5:      ROWNUM as rnum
6:    FROM
7:      (SELECT
8:        i.ID_ITEM,
9:        i.NAME,
10:       rub.ID_ITEM AS RUBRIC,
11:       a.ID_AUTHOR AS AUTHOR,
12:       p.PAGES,
13:       p.PRICE,
14:       p.AMOUNT,
15:       i.DESCRIPTION,
16:       i.PICTURE
17:      FROM
18:       ITEM i,
19:       PROPERTIES p,
20:       AUTHOR a,
```

Продовження лістингу 3.5

```
21:        ITEM rub
22:       WHERE
23:        i.TYPE = 0
24:        AND i.ID_PROPERTIES = p.ID_BOOK
25:        AND p.ID_AUTHOR = a.ID_AUTHOR
26:        AND i.PARENT_ID = rub.ID_ITEM
27:        AND rub.TYPE = 1
28:       )items
29:    WHERE
30:     ROWNUM <=  ?
31:  )
32:  WHERE rnum  >=?
```

Пошук реалізовано за допомогою форми, що наведена у лістингу 3.6. та методу process класу «ViewListBooks», який представлено у лістингу 3.7.

Лістинг 3.6. – Код форми, що реалізовує систему пошуку

```
1:   <form class="find_input" method="POST" action="<%=
2:   "MainServlet?action=" + Commands.ACTION_VIEW_LIST_BOOKS +
3:   "&" + ViewListBooks.ACTION_VIEW_LIST_ATTRIBUTE + "=" +
4:   ViewListBooks.NAME_ACTION_FOR_SEARCH%>">
5:     <input class = "find_input" type = "text" name =
6:   "<%=ViewListBooks.PARAMETER_ACTION_FOR_LIST_ATTRIBUTE%>"
7:   placeholder="search book" required />
8:   </form>
```

Лістинг 3.7. – Код методу process класу «ViewListBooks», що реалізовує систему пошуку

```
1:   public      void      process(HttpServletRequest      request,
1a: HttpServletResponse response) throws DataBaseException {
2:   String                      requestAction                      =
2a: request.getParameter(ACTION_VIEW_LIST_ATTRIBUTE);
3:   String nameForSearch;
4:   ArrayList books = null;
5:   if (requestAction != null) {
6:   switch (requestAction) {
7:     case NAME_ACTION_FOR_ALL:
8:       books = getListOfAlLBooks(request);
9:        break;
10:     case NAME_ACTION_FOR_RUBRIC:
11:       books = getListOfRubric(request, true);
12:        break;
13:      case NAME_ACTION_FOR_SEARCH:
14:  request.getSession().setAttribute(ACTION_VIEW_LIST_ATTRIBUTE,
14a: NAME_ACTION_FOR_SEARCH);
```

```
15:      nameForSearch =
15a:request.getParameter(PARAMETER_ACTION_FOR_LIST_ATTRIBUTE);
16:  request.getSession().setAttribute(PARAMETER_ACTION_FOR_LIST_AT
16a: TRIBUTE, nameForSearch);
17:       books = (ArrayList)
71a: OracleDataAccess.getInstance().getBooksByName(nameForSearch);
18:        break;
19:     default:
20:        books = getListOfAlLBooks(request);
21:         break;
22:  }
23:  Commands.AMOUNT_OF_BOOKS_ON_LIST =
23a: Commands.START_OR_PLUS_BOOKS_TO_LIST
24:    } else {
25:          requestAction = (String)
25a: request.getSession().getAttribute(ACTION_VIEW_LIST_ATTRIBUTE);
26:     switch (requestAction) {
27:         case NAME_ACTION_FOR_ALL:
28:             Commands.AMOUNT_OF_BOOKS_ON_LIST +=
28a: Commands.START_OR_PLUS_BOOKS_TO_LIST
29:             books = getListOfAlLBooks(request);
30:             break;
31:         case NAME_ACTION_FOR_RUBRIC:
32:             books = getListOfRubric(request, false);
33:             break;
34:         case NAME_ACTION_FOR_SEARCH:
35:             nameForSearch = (String)
35a:request.getSession().getAttribute(PARAMETER_ACTION_FOR_LIST_ATT
35b:RIBUTE);
36:                 books = (ArrayList)
37:  OracleDataAccess.getInstance().getBooksByName(nameForSearch);
38:             break;
39:         default:
40:               books = getListOfAlLBooks(request);
41:                 break;
42:      }
43:      }
44:  request.getSession().setAttribute(ATTRIBUTE_LIST_OF_ALL_BOOKS,
44a: books);
45:  Commands.forward("/index.jsp", request, response);
46:  }
```

Клієнт може переглянути детальну інформацію про книгу, натиснувши на її посилання. Приклад можливості перегляду детальної інформації про книгу наведений на рис. 3.7. Переглядаючи детальну інформацію, користувач має можливість його замовлення, за допомогою кнопки «Buy». Також дана можливість доступна з головної сторінки.

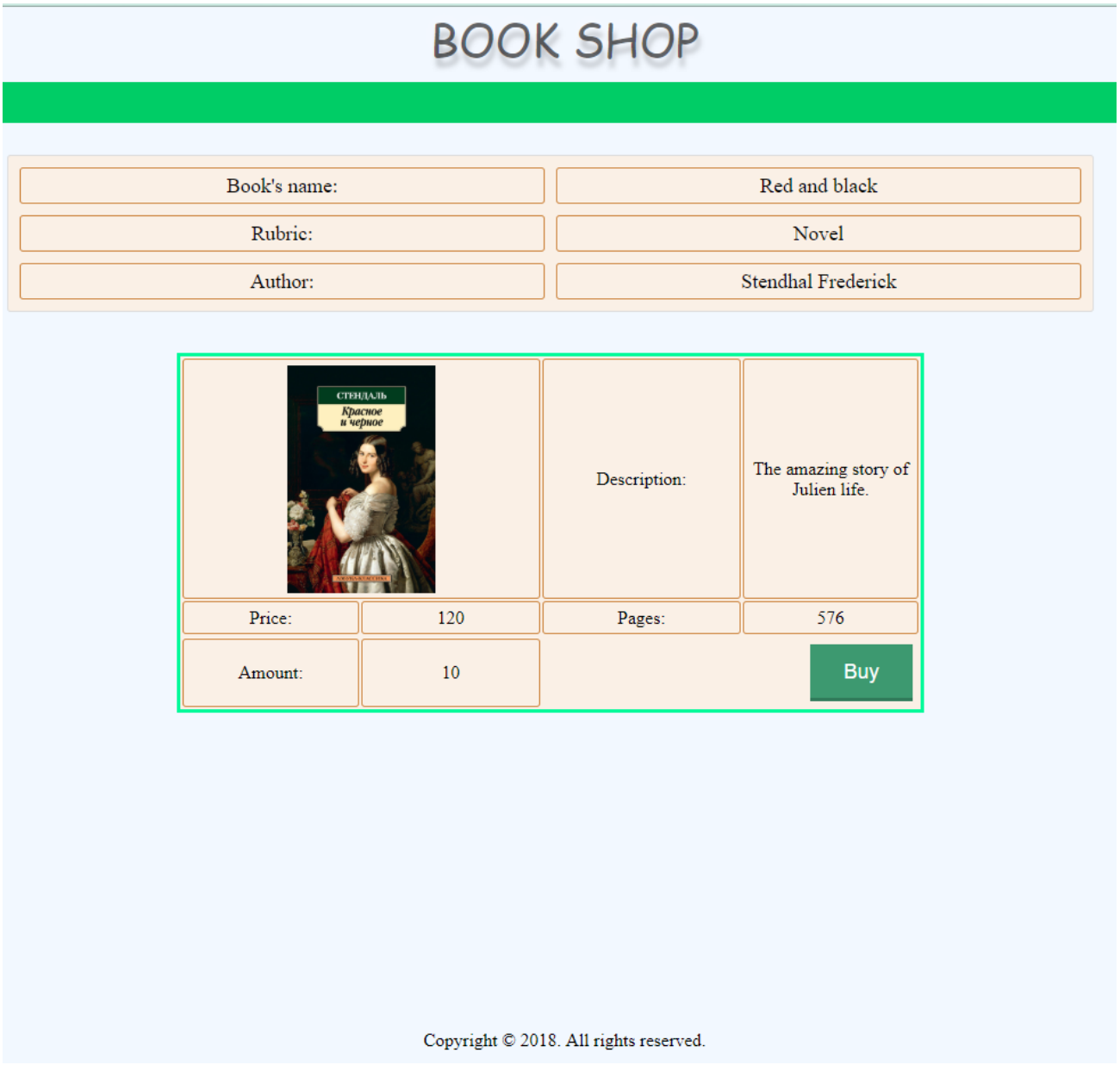


Рисунок 3.7 – Сторінка перегляду детальної інформації про книгу наведений

Вікно оформлення клієнтом замовлення для авторизованого користувача представлене на рис. 3.8, для неавторизованого на рис. 3.9. Неавторизованому користувачу необхідно вводити додаткові поля.

Авторизуватися клієнт може за допомогою посилання в верхній навігаційні панелі «Entry», зареєструватися – «Registration». На рис. 3.10 представлено вікно авторизації, на рис. 3.11 – реєстрації нового клієнта.

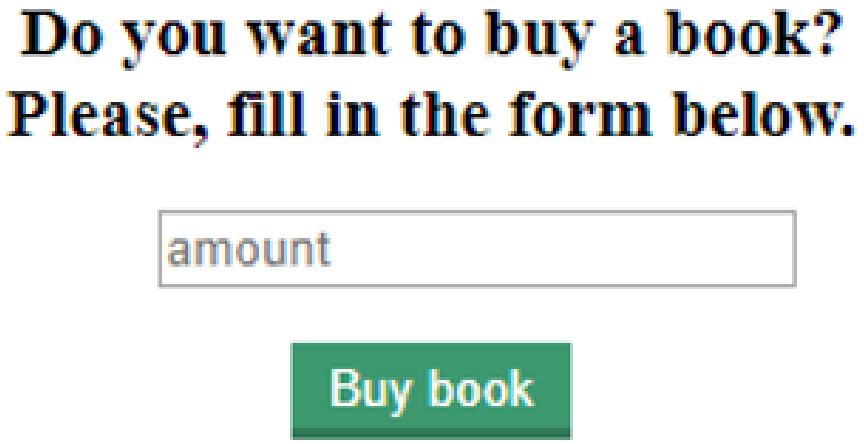


Рисунок 3.8 – Вікно замовлення товару для авторизованого користувача

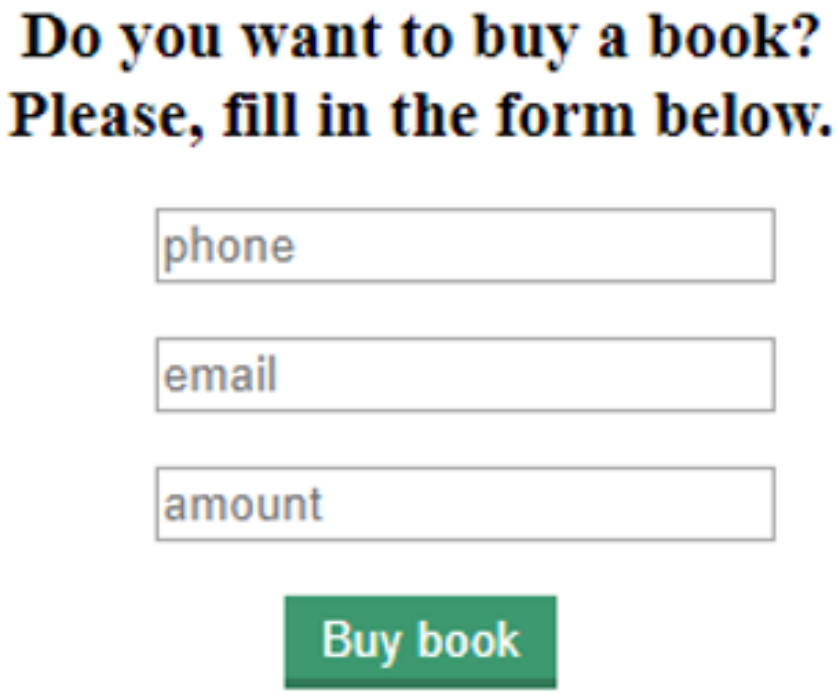


Рисунок 3.9 – Вікно замовлення товару для неавторизованого користувача

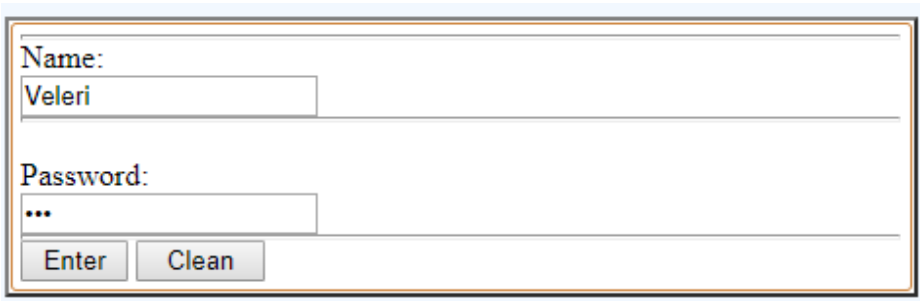


Рисунок 3.10 – Вікно авторизації користувача

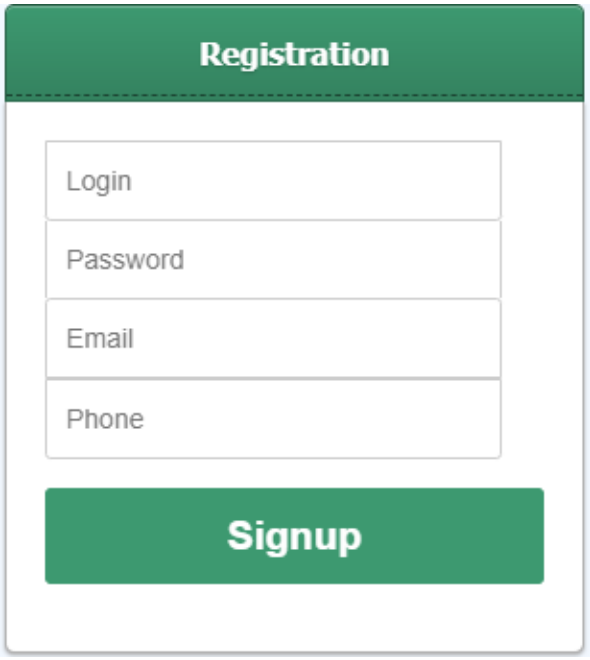


Рисунок 3.11 – Вікно реєстрації нового клієнта

Після успішної реєстрації чи авторизації, назва посилання «Entry» в верхній навігаційні панелі, змінюється на логін користувача та надає змогу перейти у профіль користувача. У своєму профілі, клієнт має можливість змінити інформацію про себе або переглянути і змінити відомості про своє замовлення. На рис. 3.12 зображено змінення даних профілю.

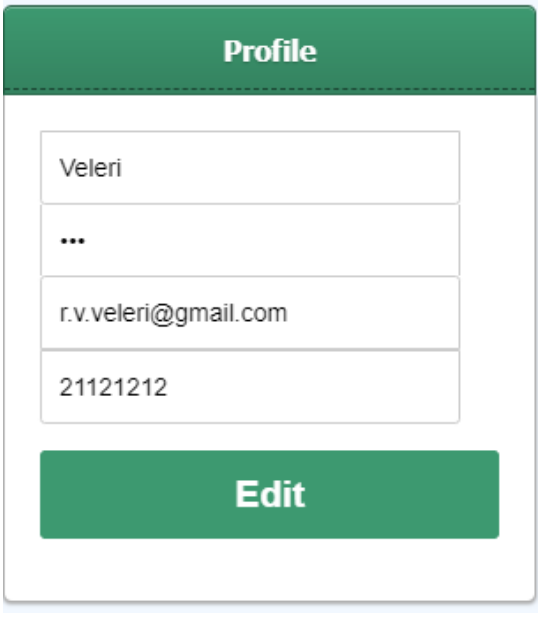


Рисунок 3.12 – Вікно внесення змін до профілю користувача

Вікно перегляду та редагування замовлення наведено на рис. 3.13. При видалені товару із замовлення, система вимагає додаткового підтвердження дії для переконання, що дана опція обрана не випадковим чином, приклад даного запиту наведено на рис. 3.14. Також. користувач має змогу змінити кількість книжок замовлення, приклад такої команди наведений на рис. 3.15.

Book name	Amount	Price	Author		
Sherlock Holmes: The Hound of the Baskervilles.	2	150	Arthur Conan Doyle	Delete	Update
After you	1	105	Jojo Moyes	Delete	Update
Sherlock Holmes: The Valley of fear.	3	60	Arthur Conan Doyle	Delete	Update
Red and black	5	120	Stendhal Frederick	Delete	Update
Sherlock Holmes: The Hound of the Baskervilles.	2	150	Arthur Conan Doyle	Delete	Update

Рисунок 3.13 – Вікно перегляду та редагування замовлення клієнта

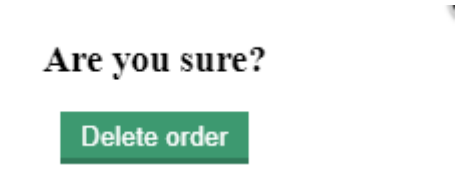


Рисунок 3.14 – Вікно додаткового підтвердження видалення товару із замовлення

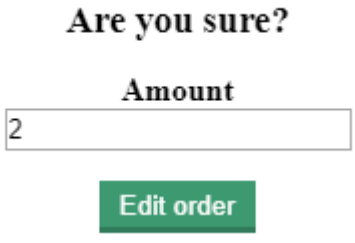


Рисунок 3.15 – Вікно внесення змін до замовлення шляхом змінення кількості товару

Сторінка «Delivery» надає користувачу інформацію щодо доставки замовлення, дана сторінка наведена на рис. 3.16. Сторінка «Contacts» відображає інформацію про графік роботи фізичного магазину, його адресу та номер гарячої лінії, вікно сторінки "Contacts" з відображення контактної інформації представлено на рис. 3.17.

Рисунок 3.16 – Сторінка "Delivery"

Рисунок 3.17 – Вікно сторінки "Contacts" з відображення контактної інформації

У процесі роботи із системою, клієнт має змогу залишити відгук. Для цього необхідно натиснути на кнопку «Feedback», що знаходиться у нижньому правому куті кожної із сторінок. Після натискання, розгортається вікно відгуку, що представлено на рис. 3.18.

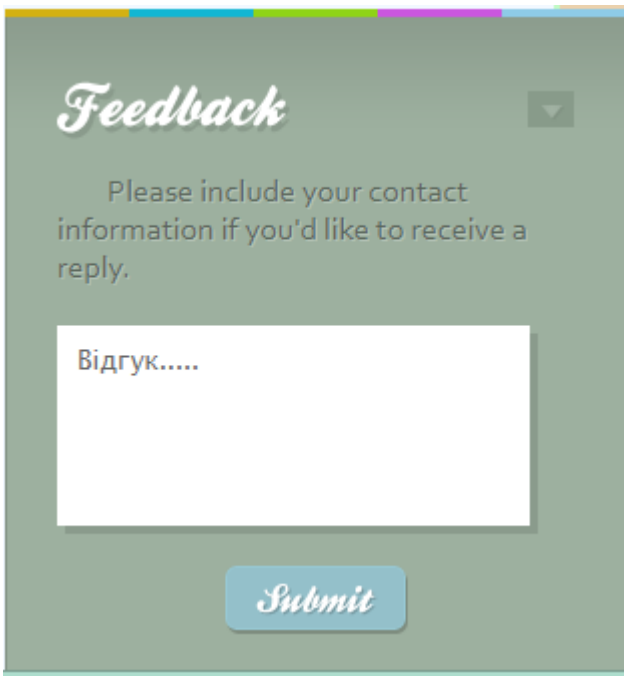


Рисунок 3.18 – Вікно відгуку

Розглянемо роботу системи зі сторони співробітників підприємства. Для співробітників змінюється відображення деяких вікон. Кнопка «Buy», при перегляді асортименту продукції, змінюється на «Delete» (рис. 3.19) та позначає видалення книги.

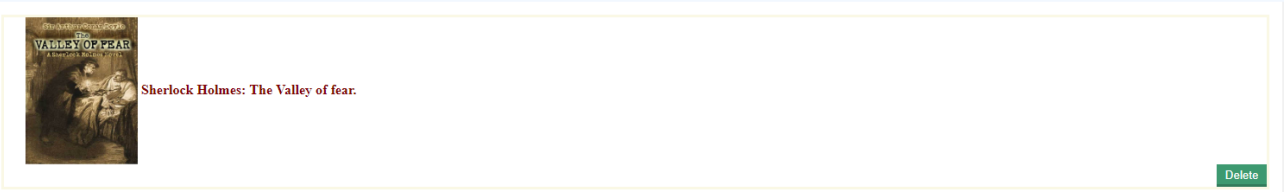


Рисунок 3.19 – Кнопка «Delete» при перегляді асортименту продукції для співробітників підприємства

На лівій навігаційній панелі, для співробітників з’являються нові кнопки «Add» та «Remove» для створення та видалення рубрик і категорій (рис. 3.20). Видалення та створення рубрики чи категорії наведено на рис. 3.21 – 3.22 відповідно.

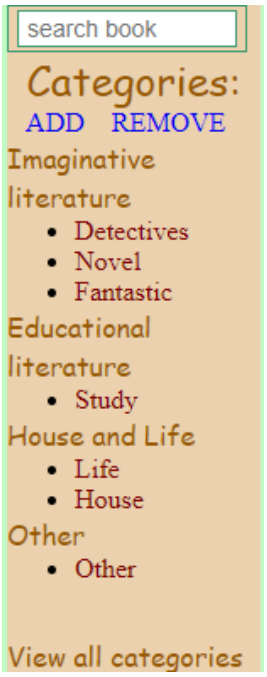


Рисунок 3.20 – Додаткові кнопки для співробітників підприємства

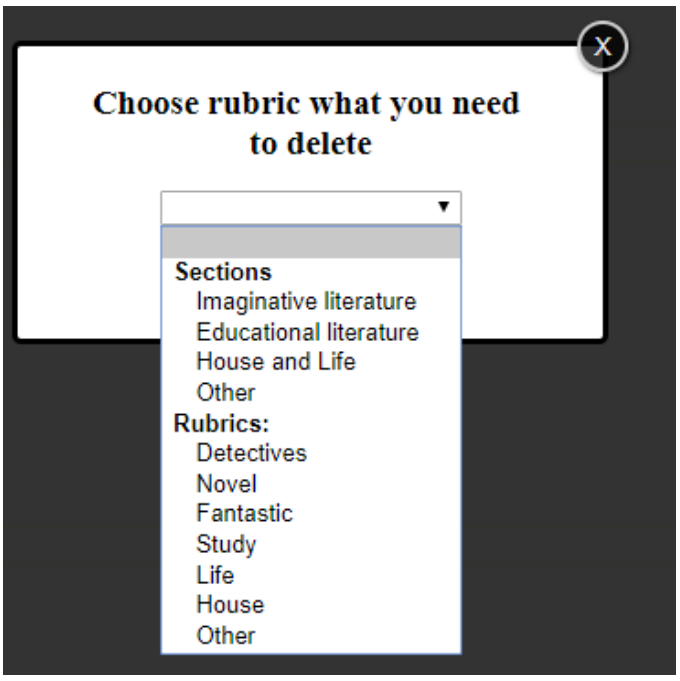


Рисунок 3.21 – Вікно видалення рубрики чи категорії

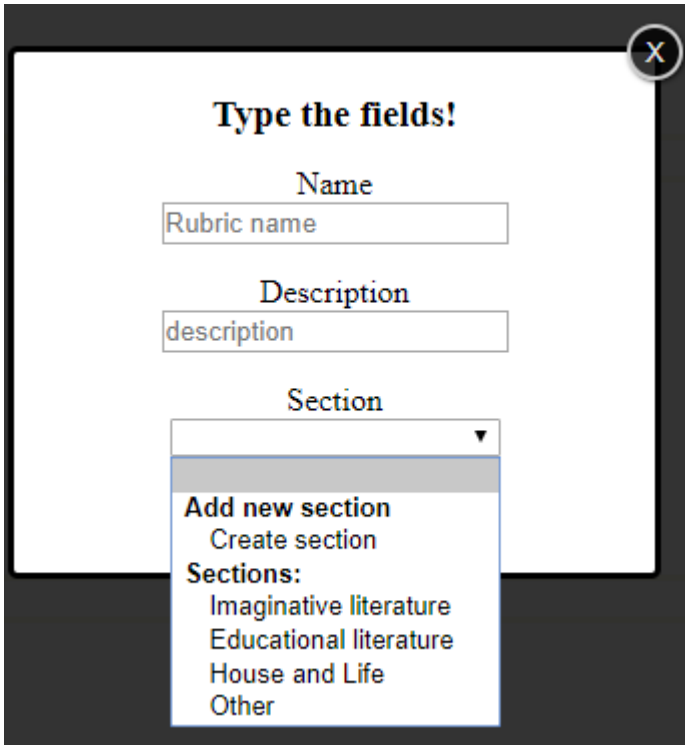


Рисунок 3.22 – Вікно створення нової рубрики чи категорії

При перегляді детальних відомостей про товар, кнопка «Buy» змінюється на «Edit», що представлено на рис. 3.23. Вікно редагування відомостей про товар наведено на рис.3.24.

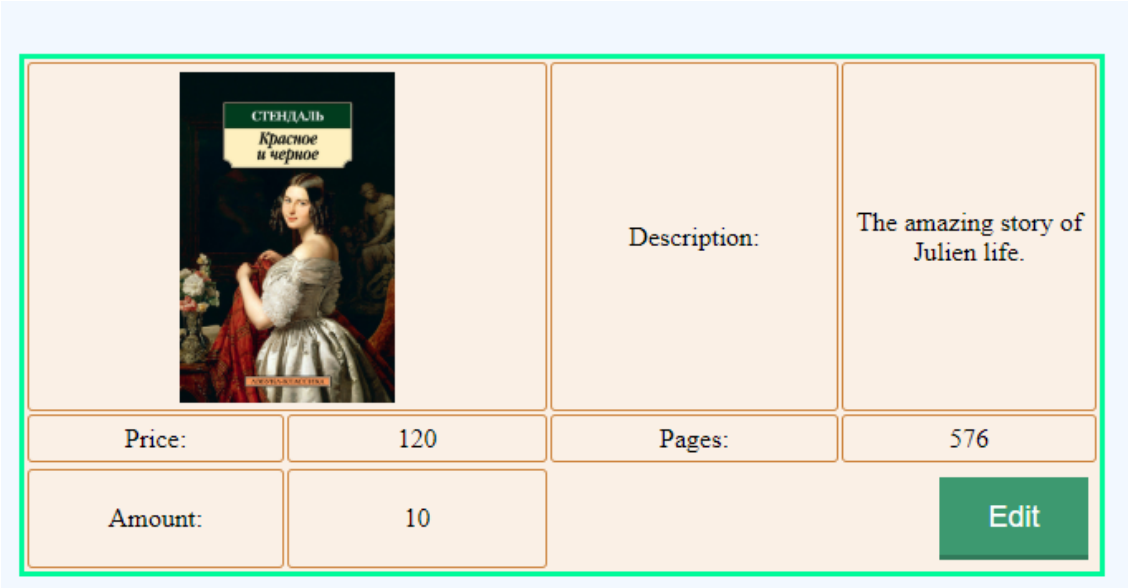


Рисунок 3.23 – Відображення кнопки "Edit" для співробітників при перегляді детальних відомостей про товар

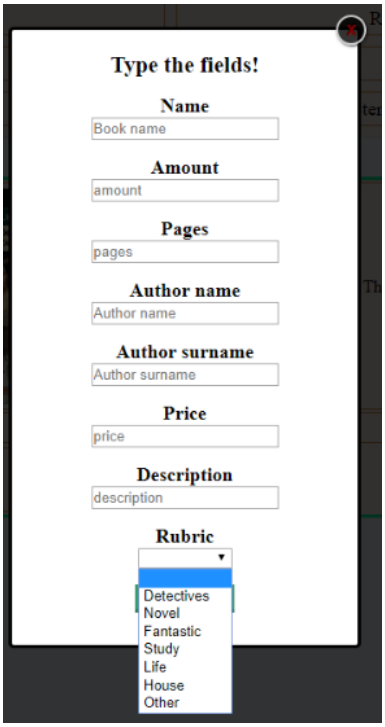


Рисунок 3.24 – Вікно редагування відомостей про товар

Для даного типу користувачів на головній панелі управління з’являється додаткове посилання «Add book» для створення нових книжок. Вікно створення нового товару наведено на рис. 3.25.

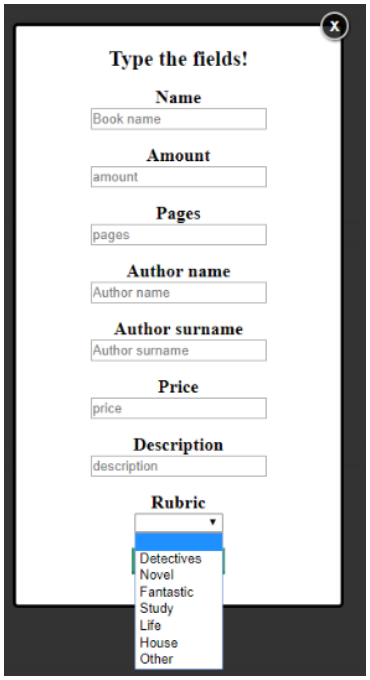


Рисунок 3.25 – Вікно створення нового товару

Віно перегляду замовлень змінюється для даних користувачів. В даному вікні відображається інформація про всіх користувачів та замовлення (рис. 3.26).

Рисунок 3.26 – Вікно відображення відомостей про користувачів та їх замовлення

3.4 Оцінка очікуваного ефекту від впровадження системи автоматизації

Ефективність від впровадження автоматизованої системи визначається шляхом порівняння результатів її функціонування та затрат ресурсів, що були витраченні на її проектування, створення та розвиток.

Першим кроком визначення ефективності автоматизованої системи є визначення капітальних витрат, які включають у себе витрати на проектування проекту, придбання необхідного устаткування і його установку, програмування комплексу задач та його налагодження. Капітальні витрати ($К$) будемо розраховувати за наступною формулою:

$$К = К_1 + К_2 + К_3, \quad (3.1)$$

де $К_1$ - витрати на проектування та програмування комплексу задач, грн.;

$К_2$ - витрати на ліцензійні програмні продукти, грн.;

$К_3$ - витрати на устаткування, грн.

Витрати на проектування та програмування системи у часовому вираженні наведені у таблиці 3.8.

Таблиця 3.8 – Витрати на проектування та програмування системи у часовому вираженні у часовому вираженні

№ п/п	Найменування задачі	Кількість витрачених годин	Уповноважена особа
1	Аналіз предметної області автоматизації та процесів діяльності підприємства.	10	Аналітик
2	Визначення функціональних вимог до системи.	3	Аналітик
3	Визначення необхідного програмного та технічного забезпечення, необхідну кількість ресурсів для проектування системи та її забезпечення.	2	Аналітик, технічний спеціаліст
4	Проектування системи.	20	Аналітик, програміст-розробник
5	Створення прототипу проекту.	5	програміст-розробник
6	Демонстрація прототипу, визначення недоліків.	3	Аналітик, програміст-розробник
7	Корегування вимог до системи.	2	Аналітик
8	Написання програмного додатку.	60	програміст-розробник
9	Тестування додатку.	10	тестувальник
10	У разі необхідності, внесення змін до додатку та повторне тестування.	10	програміст-розробник, тестувальник
11	Впровадження системи.	10	технічний спеціаліст

Таким чином загальний час на проектування та програмування системи складає 135 години. Для реалізації необхідно 4 трудових ресурси: аналітик, технічний спеціаліст, програміст-розробник та тестувальник. Витрати на оплату даних працівників знаходиться за наступною формулою:

$$K_1 = \sum_{k=1}^{k} N_k \cdot r_k \cdot T_k, \quad (3.2)$$

де N_k - кількість працівників, чол.;

r_k - годинна заробітна плата працівника, грн.;

T_k - трудомісткість працівника (кількість витраченого часу), год.;

Систематизуємо дані та розрахуємо годину заробітної плати працівників у таблиці 3.9. Також, розрахуємо витрат на проектування та програмування для кожного із працівників, результати наведені у таблиці 3.10 [71].

Таблиця 3.9 – Розрахунок годинної заробітної плати працівників

№ п/п	Працівник	Кількість чоловік	Фонд часу роботи в середньому	Середня заробітна плата (грн)	Формула розрахунку години заробітної плати	Годинна заробітна плата (грн)
1	Аналітик	1	184 години (23 робочих дня по 8 годин).	10 000	$\frac{\text{Ср.}^{з}/_{п}}{\Phi_{ч}}$, де з/п – середня заробітна плата, $\Phi_{ч}$ – фонд часу роботи в середньому	54,34
2	Програміст-розробник	2		10 000		
3	Технічний спеціаліст	1		8 000		43,47
4	Тестувальник	1		8 000		

Таблиця 3.10 – Визначення витрат на проектування та програмування для працівників

№ п/п	Працівник	Кількість затрачених годин	Проміжна розрахунки	Витрати
1	Аналітик	28	$1 \cdot 54{,}34 \cdot 28$	1 521,52
2	Програміст-розробник	98	$2 \cdot 54{,}34 \cdot 98$	10 650,64
3	Технічний спеціаліст	12	$1 \cdot 43{,}47 \cdot 12$	521,64
4	Тестувальник	20	$1 \cdot 43{,}47 \cdot 20$	869,40

Отже, витрати на проектування та програмування складають:

$$К_1 = 1521{,}52 + 10650{,}64 + 521{,}64 + 869{,}40 = 13563{,}20 \text{ (грн)}.$$

Витрати на ліцензійні програмні продукти наведена у таблиці 3.11. Витрати на технічне устаткування наведені у таблиці 3.12.

Таблиця 3.11 – Витрати на ліцензійні програмні продукти

№ п/п	Назва програмного продукту	Вартість ліцензії, грн
1	IntelliJ IDEA Ultimate	13 350
2	Oracle WebLogic Server 12c	35 000
3	Oracle Database XE 12c	30 000
Разом		78 350

Таблиця 3.12 – Витрати на устаткування

№ п/п	Найменування устаткування	Вартість, грн
1	Сервер бази даних	40 785,17
2	Сервер додатків	25 987,32
Разом		66 772,49

Отже, капітальні витрати складають:

$$К_1 = 13\,563{,}20 + 78\,350 + 66\,772{,}49 = 158\,685{,}69 \text{ (грн)}.$$

Наступним етапом є розрахунок річного економічного ефекту від впровадження автоматизованої системи, а також термін її окупності. Річний економічний ефект наведений у формулі 3.3 та визначається як різниця суми річної економії(річного приросту прибутку) та нормативним значенням повернення одноразових витрат на впровадження системи. Сума річної економії знаходиться за формулою 3.4.

$$E_y = S - K \cdot r_n, \tag{3.3}$$

де E_y - річний економічний ефект;

S - річна економія, яку отримає підприємство в результаті впровадження АСУ;

K - капітальні вкладення, які було витрачено в результаті впровадження

АСУ;

r_n - нормативний коефіцієнт окупності капітальних вкладень, узятий для конкретної галузі (для даного прикладу r_n = 0,33).

$$S = \text{Вб} - \text{Вп}., \tag{3.4}$$

де *Вб* - приведені до одного року витрати на обробку інформації при базовому варіанті організації обробки;

Вп - приведені до одного року витрати на обробку інформації при впроваджуваному (пропонованому) варіанті організації системи обробки.

В середньому, менеджеру необхідно 4 години для обробки усіх замовлень. Середня заробітна плата менеджера складає 10 000 грн, фонд часу роботи в середньому 184 години (23 робочих дня по 8 годин). Знайдемо погодинну заробітну плату менеджера:

$$\text{Погодинна заробітна плата} = \frac{\text{Середня заробітна плата}}{\text{Фонд часу роботи}} = \frac{10\,000}{184} = 54{,}34 \text{ грн.}$$

За рік даний працівник працює 250 днів. На підприємстві в середньому працює від двох до трьох працівників, порахуємо витрати на обробку до впровадження системи із двома працівниками:

$$\text{Вб} = 4 \cdot 2 \cdot 54{,}34 \cdot 250 = 108\,680 \text{ грн.}$$

В результаті впровадження автоматизованої системи час, для обробки усіх замовлень скоротиться до 1 години. Витрати на обробку після впровадження складають:

$$\text{Вп} = 1 \cdot 2 \cdot 54{,}34 \cdot 250 = 27\,170 \text{ грн.}$$

Розрахуємо суму річної економії за формулою 3.4, підставивши відповідні значення:

$$S = 869\,440 - 217\,360 = 81\,510 \text{ грн.}$$

Розрахуємо річний економічний ефект за формулою 3.3, підставивши відповідні значення:

$$\text{E}_\text{у} = 81\,510 - 158\,685{,}69 \cdot 0{,}33 = 29\,143{,}72 \text{ грн.}$$

Розрахуємо коефіцієнт економічної ефективності капітальних витрат за формулою:

$$R_{ce} = \frac{S}{C} = \frac{S}{K} \qquad (3.5)$$

Отримаємо:

$$R_{ce} = \frac{81\,510}{158\,685{,}69} = 0{,}51$$

Розрахуємо термін окупності витрат на впровадження проекту за наступною формулою:

$$P_p = \frac{C}{S} = \frac{K}{S} = \frac{1}{R_{ce}} \quad (3.6)$$

Отримаємо:

$P_p = \frac{1}{0,51} = 1,96$ (роки).

Проаналізувавши показники можна стверджувати, що система автоматизації електронної комерції роздрібного бізнесу повністю окупиться і вже через 2 роки буде приносити економічний ефект. Отже, після впровадження, дана програмна реалізація системи дозволить підвищити продуктивність підприємства та ефективність роботи працівників, а також зменшити витрати на обробку інформації.

ВИСНОВКИ

В ході написання кваліфікаційної магістерської роботи було досліджено предметну область, яка потребує автоматизації. Оброблена інформація стосовно поточного стану та розвитку засобів електронної комерції, як в Україні, так і в світі в цілому. Також, визначені внутрішні та зовнішні фактори впливу на сферу електронної комерції.

У ході дослідження платних та безкоштовних інформаційних систем створення та авторизації діяльності підприємств роздрібної торгівлі, виявлено, що безкоштовні системи не можуть в повній мірі задовольнити вимоги системи, а платні мають велику вартість та потребують багато часу для вивчення.

Досліджено бізнес-процеси підприємства та сформовано діаграми взаємодії системи з користувачами на кожному з етапів роботи у нотаціях IDEF0, DFD та IDEF3. Також побудовані блок-схеми алгоритму взаємодії системи з клієнтами та співробітниками підприємства. Розподіллено функції системи на дві групи: функції, що пов’язані із взаємодією системи із клієнтами та функції, що пов’язані із взаємодією зі співробітниками.

Результатом написання кваліфікаційної магістерської роботи став прототип автоматизованої системи для підприємств роздрібної торгівлі в сфері електронної комерції. Розроблено зручний та простий веб-орієнтований інтерфейс взаємодії клієнта з базою даних.

Для створення додатку було обрано трирівневу клієнт-серверну архітектуру. У якості веб-серверу було обрано «Oracle WebLogic Server 12c», у якості бази даних обрано «Oracle Database XE 12c». В процесі розробки додатку, для організації доступу до бази даних використовувалося програмне забезпечення «SQL Developer».

Додаток розроблено за допомогою програмного забезпечення «IntelliJ IDEA Ultimate» з використанням об’єктно-орієнтованої мови програмування Java.

Для використання системи, на кінцевій робочій станції необхідно наявність підключення до мережі Інтернет, операційної системи «Windows 8» або «Windows 10» та браузеру «Google Chrome».

Для оцінки якості розробленого додатку було сформовано інструкцію користувача та здійснено тестування. У ході аналізу тестових даних, з’ясовано, що додаток виконує усі функції системи без помилок. Отже, можна стверджувати, що система придатна для використання.

До можливостей, що не були реалізовані у даній системи, слід віднести доступ до системи за допомогою мобільного додатку, а також система не враховую привілеї постійних клієнтів підприємства.

Розрахунок економічної ефективності показав, що розробка та використання додатку є економічно доцільним. В результаті впровадження автоматизованої системи час, для обробки усіх замовлень відповідним співробітником підприємства скоротиться до 1 години, замість чотирьох. Термін окупності капіталовкладень складає приблизно 2 роки.

Таким чином мету кваліфікаційної магістерської роботи можна вважати досягнутою, а поставлені завдання для даної роботи повністю виконаними.

СПИСОК ВИКОРСТАНИХ ДЖЕРЕЛ

1. Царьов Р.Ю. Електронна комерція: навч.посіб.Одеса, 2010. 112 с.

2. Олійник В.М., Вовк В.В. Впровадження електронної комерції в роздрібний бізнес: матеріали III Всеукраїнської науково-практичної on-line конференції (22-23 листопада 2018, м.Суми).Суми, 2018. С. 298-303.

3. Scheneider Gary P. Electronic commerce. Boston, 2015. 598 p.

4. Стрелец И. А. Новая экономика и информационные технологии. Москва, 2003. 254 с.

5. Клейнер Я. С., Древицкая И.Ю., Дорофиенко В.В. Новые информационные технологии в менеджменте. Харьков, 2002. 106 с.

6. Самойленко С. В. Реклама в Интернете: реалии и «виртуалии». *Маркетинг и реклама*. 2004. N1. С. 35-39.

7. Етапи розвитку інформаційних технологій [Електронний ресурс]. - Режим доступу: https://sites.google.com/site/lvsereda13/etapi-rozvitku-informacijnih-tehnologij

8. Глобальные компьютерные сети [Електронний ресурс]. - Режим доступу: http://bourabai.ru/einf/chapter116.htm

9. Холмогоров В. Интернет-маркетинг. 2-е издание. Санкт-Петербург, 2005. 272 с.

10. B2B продажи: 3 шага к успешному бизнесу [Електронний ресурс]. - Режим доступу: https://geniusmarketing.me/lab/b2b-prodazhi-3-shaga-k-uspeshnomu-biznesu

11. Yuthayotin S. Access to Justice in Transnational B2C E-Commerce. A multidimensional Analysis of Consumer Protection Mechanisms. Thailand, 2014. 332 p.

12. Muhammad A., Tanzila S. Advanced SWOT Analysis of E-Commerce. *IJCSI International Journal of Computer science Issues*. Vol. 9. Issue 2. No 2. pp. 569-574

13. Денисова А.Л., Молоткова Н.В., Блюм Н.В., Уляхин Т.М., Гуськов А.В. Электронная коммерция: основы организации и ведения бизнеса : учеб. пос.Тамбов, 2012. 88 с.

14. Філіппова Л. Л. Електронна комерція: за і проти. *Вісник Нац. техн. ун-туХПІ*. Харків, 2013. № 44 (1017). С. 58-65.

15. Ховрак І. В. Електронна комерція в Україні: переваги та недоліки. *Економіка. Фінанси. Право*. 2013. № 4. С. 16-20.

16. Возний М. І. Міжнародна електронна торгівля. Проблеми та перспективи розвитку в Україні. *Збірник наук. праць Буковинського університету. Економічні науки*. Вип. 7. С. 243-252.

17. Креденець О. В. Стан і тенденції розвитку електронної роздрібної торгівлі в українському секторі мережі Internet. *Вісник Львівської комерційної академії*. 2011. № 34. С. 268-272.

18. Кудіна О. Ю. Розвиток електронної торгівлі в умовах становлення глобального інформаційного простору. *Бюлетень Міжнародного Нобелівського економічного форуму*. 2011. № 1(4). С. 196-202.

19. Легенчук С. Ф., Скакун А.С. Сутність електронної комерції: обліковий вимір. *Вісник ЖДТУ. Серія: Економічні науки*. 2011. № 4 (58). С. 59-65.

20. Laudon Kenneth C., Guercio Traver C. E-commerce 2016: Business, Technology, Society. *Global Edition*. 2016. 12th edition. 909 p.

21. Laudon Kenneth C., Guercio Traver C. E-commerce 2017: Business, Technology, Society. *Global Edition*. 2017. 13th edition. 909 p.

22. Патраманська Л.Ю. Електронна комерція: переваги та недоліки. *Ефективна економіка*. 2015. №11. С. 55-60.

23. Consumer Barometer Survey [Електронний ресурс]. - Режим доступу: https://www.consumerbarometer.com/en

24. Global Mobile Trends 2017. *GSMA* Intelligence. 2017. [Electronic resource]. - Available: https://www.gsmaintelligence.com/research/?file=3df1b7d57b1e63a0cbc3d585feb82dc2&download

25. Global B2C E-commerce Report 2016. *Ecommerce Foundation Raadhuisstraat*. 2016. [Electronic resource]. - Available: https://www.ecommercewiki.org/wikis/www.ecommercewiki.org/images/5/56/Global_B2C_Ecommerce_Report_2016.pdf

26. Global Ecommerce Report 2017. *Ecommerce Foundation*. 2017. [Electronic resource]. – Available: https://www.ecommercefoundation.org/reports

27. Олійник В. М., Речембей В.В. Сучасні тенденції розвитку телекомунікаційних технологій. *Економіка та суспільство*. 2018. № 14. С. 1016-1022.

28. Laudon Kenneth C., Laudon Jane P. Management Information Systems. 13th edition. USA, 2013. 648 p.

29. Задвірний Я., Орловська А. Використання можливостей електронної комерції у процесі ведення бізнесу. *Формування ринкової економіки в Україні*. 2008. Вип. 18. С. 70-75.

30. Система управления сайтами DIAFAN.CMS: офіційний сайт [Електронний ресурс]. - Режим доступу: https://www.diafan.ru

31. Shop-Script: офіційний сайт [Електронний ресурс]. - Режим доступу: http://www.shop-script.ru

32. PrestaShop: офіційний сайт [Електронний ресурс] . - Режим доступу: https://www.prestashop.com/en

33. Бітрікс: Управління сайтом: офіційний сайт [Електронний ресурс] . - Режим доступу: https://www.bitrix.ua

34. UMI.CMS: офіційний сайт [Електронний ресурс] . - Режим доступу: https://www.umi-cms.ru

35. NetCat: офіційний сайт [Електронний ресурс]. - Режим доступу: http://nc110.sourceforge.net

36. WordPress: офіційний сайт [Електронний ресурс] . - Режим доступу: https://wordpress.com

37. OpenCart: офіційний сайт [Електронний ресурс]. - Режим доступу: https://www.opencart.com

38. Ucoz: офіційний сайт [Електронний ресурс]. - Режим доступу: https://www.ucoz.com

39. Ганин А.Н. Применение информационных технологий для моделирования бизнес-процессов на предприятии радиоэлектронного комплекса. *Российское предпринимательство*. 2016. Т. 17. № 22. С. 3171-3184.

40. Избачков Ю.С., Петров В.Н., Васильев А.А., Телина И.С. Информационные системы: учебник для вузов. 3-е изд. Санкт-Петербург, 2011. 544 с.

41. Волков О. Стандарты и методологии моделирования бизнес-процессов [Электронный ресурс]. - Режим доступа: http://www.connect.ru/article.asp?id=5799

42. The Content Council Spending Study: A Look at How Corporate America Invests in Branded Content for 2013 [Electronic resource]. - Available: http://www.customcontentcouncil.com/research/2013-spending-study

43. Мінеев Є.І. Моделювання бізнес-процесів [Електронний ресурс]. - Режим доступу: http://zavantag.com/docs/663/index-1248743.html

44. Репин В.В. Бизнес-процессы. Моделирование, внедрение, управление. Москва, 2013. 512 с.

45. Introduction to IDEF0/3 for Business Process Modelling [Electronic resource]. - Available: http://businessprocessagility.com/wp content/uploads/2015/06/IDEF03-guidebook.pdf.

46. Тищенко Г. Моделирование бизнес-процессов предприятия [Электронный ресурс]. - Режим доступа: http://iteam.ru/publications/it/section_51/article_1335.

47. Мінеев Є.І. Моделювання бізнес-процесів [Електронний ресурс]. - Режим доступу: http://zavantag.com/docs/663/index-1248743.html

48. Buede Dennis M., Miller William D. The Engineering Design of Systems: Models and Methods. 3rd edition. New Jersey, 2016. 584 p.

49. Архитектура клиент-сервер [Электронный ресурс]. - Режим доступа: http://www.mstu.edu.ru/study/materials/zelenkov/ch_7_1.html

50. Chapter 25 Getting Started Securing Web [Electronic resource]. - Available:Applications»https://docs.oracle.com/cd/E19226-01/8207627/bncat/index.html

51. Murach J., Urban M. Java Servlets and JSP. 3rd edition. USA, 2014. 744 p.

52. Alapati Sam R. Oracle Weblogic Server 12c Administration Handbook. USA, 2014. 519 p.

53. Анісімов А.В., Кулябко П.П. Інформаційні системи та бази даних: навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. Київ. 2017. 110 с.

54. Oracle Database 11g: руководство администратора баз данных. Москва, 2010. 1440 с.

55. Schildt H. Java: A Beginner's Guide. 7th edition. New York, 2017. 729 p.

56. Goncalves A. Beginning Java EE 7. New York, 2013. 608 p.

57. Evans Benjamin J., Flanagan D. Java in a Nutshell: A desktop quick reference. 7th edition. USA, 2018. 438 p.

58. Oracle для профессионалов. Москва, 2016. 960 с.

59. Harrington Jan L. Relational database design and implementation. USA, 2016. 712 p.

60. Greenwald R., Stackowiak R., Stern J. Oracle Essentials. USA, 2013. 432 p.

61. Feuertein S., Pribyl B. Oracle PL/sql programming. USA, 2014. 1392 p.

62. Date C.J. Database Design and Relational Theory. USA, 2012. 2878 p.

63. PL/SQL Inherits Database Robustness, Security, and Portability [Electronic resource]. - Available: https://www.oracle.com/database/technologies/appdev/plsql.html

64. MVC [Електронний ресурс]. - Режим доступу: http://www.tutorialsteacher.com/mvc/mvc-architecture

65. Marla S. Learn MVC Project in 7 Days. India, 195p.

66. Schildt H. Java: The complete Reference. 10th edition. New York, 2017. 1309 p.

67. Krochmalski J. Intellij IDEA Essentials. UK, 2014. 263 p.

68. Orsine Assumpção H. Getting started with Intellij IDEA, UK. 2013. 114 p.

69. Intellij Idea Tutorial [Electronic resource]. - Available: https://www.tutorialspoint.com/intellij_idea/index.htm

70. Getting started [Electronic resource]. - Available: https://www.jetbrains.com/idea/documentation/

71. DOU: офіційний сайт [Електронний ресурс]. - Режим доступу: https://dou.ua/

ДОДАТКИ

Додаток А

SUMMARY

Vovk V.V. Software implementation of e-commerce for retail business - Masters-level Qualification Thesis. Sumy State University, Sumy, 2018.

The master's thesis focuses on the essence of the definition e-commerce, the analysis of existing information systems for the automation. Uses of the main business processes in the company. The description of the main business processes at the enterprise is implemented. The technology of creating an automated system was substantiated. Economic efficiency is calculated from the implementation of the developed system at the enterprise.

Keywords: e-commerce, automation, JNDI technology, Online Store, Weblogic, Oracle xe, Java, efficiency, business process, product.

АНАТОЦІЯ

Вовк В.В. Програмна реалізація електронної комерції в роздрібній торгівлі. Кваліфікаційна магістерська робота. Сумський державний університет, Суми, 2018 р.

У роботі досліджено сутність електронної комерції, проведений аналіз існуючих інформаційних системи для автоматизації. Здійснений опис основних бізнес-процесів на підприємстві. Була обґрунтована технологія створення автоматизованої системи. Розрахована економічна ефективність від впровадження розробленої системи на підприємстві.

Ключові слова: електронна комерція, автоматизація, технологія JNDI, Інтернет-магазин, Weblogic, Oracle xe, Java, ефективність, бізнес-процес, товар.

Додаток Б

Лістинг Б.1. – Код створення таблиць бази даних, зв’язків між ними та наповнення таблиць даними

```
--------------------------------------------------------
--  File created - воскресенье-декабря-16-2018
--------------------------------------------------------
--------------------------------------------------------
--  DDL for Sequence DEMO_CUST_SEQ
--------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."DEMO_CUST_SEQ"      MINVALUE   1   MAXVALUE
99999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER  NOCYCLE
;
--------------------------------------------------------
--  DDL for Sequence DEMO_ORDER_ITEMS_SEQ
--------------------------------------------------------

   CREATE   SEQUENCE    "VELERI"."DEMO_ORDER_ITEMS_SEQ"    MINVALUE  1  MAXVALUE
99999999999999999999999999999 INCREMENT BY 1 START WITH 61 CACHE 20 NOORDER  NOCYCLE
;
--------------------------------------------------------
--  DDL for Sequence DEMO_ORD_SEQ
--------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."DEMO_ORD_SEQ"      MINVALUE   1   MAXVALUE
99999999999999999999999999 INCREMENT BY 1 START WITH 11 CACHE 20 NOORDER  NOCYCLE
;
--------------------------------------------------------
--  DDL for Sequence DEMO_PROD_SEQ
--------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."DEMO_PROD_SEQ"      MINVALUE   1   MAXVALUE
99999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER  NOCYCLE
;
--------------------------------------------------------
--  DDL for Sequence DEMO_USERS_SEQ
--------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."DEMO_USERS_SEQ"      MINVALUE   1   MAXVALUE
99999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER  NOCYCLE
;
--------------------------------------------------------
--  DDL for Sequence LESSON_SEQUENCE
--------------------------------------------------------

   CREATE   SEQUENCE     "VELERI"."LESSON_SEQUENCE"      MINVALUE   1   MAXVALUE
99999999999999999999999999  INCREMENT  BY  1  START  WITH  120  CACHE  20  NOORDER
NOCYCLE ;
--------------------------------------------------------
--  DDL for Sequence MARK_SEQUENCE
--------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."MARK_SEQUENCE"      MINVALUE   1   MAXVALUE
99999999999999999999999999  INCREMENT  BY  1  START  WITH  140  CACHE  20  NOORDER
NOCYCLE ;
--------------------------------------------------------
```

```
--  DDL for Sequence OBJECT_SEQUENCE
------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."OBJECT_SEQUENCE"      MINVALUE   1   MAXVALUE
99999999999999999999999999999  INCREMENT  BY  1  START  WITH  140  CACHE  20  NOORDER
NOCYCLE ;
------------------------------------------------------
--  DDL for Sequence SEQUENCE_AUTHOR
------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."SEQUENCE_AUTHOR"      MINVALUE   1   MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER  NOCYCLE
;
------------------------------------------------------
--  DDL for Sequence SEQUENCE_BOOK
------------------------------------------------------

   CREATE   SEQUENCE       "VELERI"."SEQUENCE_BOOK"       MINVALUE   1   MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER  NOCYCLE
;
------------------------------------------------------
--  DDL for Sequence SEQUENCE_CONTENT
------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."SEQUENCE_CONTENT"      MINVALUE   1   MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 61 CACHE 20 NOORDER  NOCYCLE
;
------------------------------------------------------
--  DDL for Sequence SEQUENCE_CUSTOMER
------------------------------------------------------

   CREATE   SEQUENCE     "VELERI"."SEQUENCE_CUSTOMER"     MINVALUE   1   MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 61 CACHE 20 NOORDER  NOCYCLE
;
------------------------------------------------------
--  DDL for Sequence SEQUENCE_ITEMS
------------------------------------------------------

   CREATE   SEQUENCE       "VELERI"."SEQUENCE_ITEMS"      MINVALUE   1   MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 41 CACHE 20 NOORDER  NOCYCLE
;
------------------------------------------------------
--  DDL for Sequence SEQUENCE_ORDERS
------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."SEQUENCE_ORDERS"      MINVALUE   1   MAXVALUE
9999999999999999999999999 INCREMENT BY 1 START WITH 61 CACHE 20 NOORDER  NOCYCLE
;
------------------------------------------------------
--  DDL for Sequence SUBJECT_SEQUENCE
------------------------------------------------------

   CREATE   SEQUENCE      "VELERI"."SUBJECT_SEQUENCE"      MINVALUE   1   MAXVALUE
99999999999999999999999999999  INCREMENT  BY  1  START  WITH  140  CACHE  20  NOORDER
NOCYCLE ;
------------------------------------------------------
--  DDL for Table AUTHOR
------------------------------------------------------

  CREATE TABLE "VELERI"."AUTHOR"
   (  "ID_AUTHOR" NUMBER,
      "SURNAME" VARCHAR2(50 BYTE),
```

```
      "NAME" VARCHAR2(50 BYTE)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Table CONTENR_ORDER
--------------------------------------------------------

  CREATE TABLE "VELERI"."CONTENR_ORDER"
   (  "ID_CONTENT" NUMBER,
      "ID_ORDER" NUMBER,
      "ID_BOOK" NUMBER,
      "AMOUNT" NUMBER
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Table CUSTOMER
--------------------------------------------------------

  CREATE TABLE "VELERI"."CUSTOMER"
   (  "ID_CUSTOMER" NUMBER,
      "LOGIN" VARCHAR2(50 BYTE),
      "PASSWORD" VARCHAR2(20 BYTE),
      "E_MAIL" VARCHAR2(50 BYTE),
      "PHOME_NUBMER" VARCHAR2(12 BYTE),
      "ROLE" NUMBER DEFAULT 1
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;

   COMMENT ON COLUMN "VELERI"."CUSTOMER"."ROLE" IS 'if  is system - 0, admin -10,
user -1';
--------------------------------------------------------
--  DDL for Table ITEM
--------------------------------------------------------

  CREATE TABLE "VELERI"."ITEM"
   (  "ID_ITEM" NUMBER,
      "NAME" VARCHAR2(50 BYTE),
      "PARENT_ID" NUMBER,
      "DESCRIPTION" VARCHAR2(150 BYTE),
      "TYPE" NUMBER,
      "ID_PROPERTIES" NUMBER DEFAULT NULL,
      "PICTURE" VARCHAR2(250 BYTE)
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
```

```
   COMMENT ON COLUMN "VELERI"."ITEM"."PARENT_ID" IS 'if is rubric-ID_section, if
is book - ID_rubric';
   COMMENT ON COLUMN "VELERI"."ITEM"."TYPE" IS '0- if it is book,1-rubric,2-
section';
--------------------------------------------------------
--  DDL for Table ORDERS
--------------------------------------------------------

  CREATE TABLE "VELERI"."ORDERS"
   (  "ID_ORDER" NUMBER,
      "ID_CUSTOMER" NUMBER,
      "DATA" DATE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Table PROPERTIES
--------------------------------------------------------

  CREATE TABLE "VELERI"."PROPERTIES"
   (  "ID_BOOK" NUMBER,
      "PAGES" NUMBER,
      "PRICE" FLOAT(126),
      "AMOUNT" NUMBER,
      "ID_AUTHOR" NUMBER
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
REM INSERTING into VELERI.AUTHOR
SET DEFINE OFF;
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values
('1','Stendhal','Frederick');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values ('2','Arthur
Conan','Doyle');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values ('3','von
Goethe','Johann Wolfgang');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values
('4','Forsch','Tatiana');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values
('5','Tokarev','Victoria');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values
('6','Sergey','Garagulya');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values
('7','Savina','Lyudmila');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values ('8','Book
Club','"Family Leisure Club"');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values
('9','Julia','Vysotsky');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values ('10','Jojo','Moyes');
Insert into VELERI.AUTHOR (ID_AUTHOR,SURNAME,NAME) values
('11','Veronica','Roth');
REM INSERTING into VELERI.CONTENR_ORDER
SET DEFINE OFF;
Insert into VELERI.CONTENR_ORDER (ID_CONTENT,ID_ORDER,ID_BOOK,AMOUNT) values
('1','1','13','2');
```

```
Insert into VELERI.CONTENR_ORDER (ID_CONTENT,ID_ORDER,ID_BOOK,AMOUNT) values
('42','42','13','2');
Insert into VELERI.CONTENR_ORDER (ID_CONTENT,ID_ORDER,ID_BOOK,AMOUNT) values
('43','43','28','1');
Insert into VELERI.CONTENR_ORDER (ID_CONTENT,ID_ORDER,ID_BOOK,AMOUNT) values
('45','45','12','5');
Insert into VELERI.CONTENR_ORDER (ID_CONTENT,ID_ORDER,ID_BOOK,AMOUNT) values
('21','21','13','2');
REM INSERTING into VELERI.CUSTOMER
SET DEFINE OFF;
Insert into VELERI.CUSTOMER
(ID_CUSTOMER,LOGIN,PASSWORD,E_MAIL,PHOME_NUBMER,ROLE) values ('3','Common
user','1112','fdsgd@fdgd.com','ппп','1');
Insert into VELERI.CUSTOMER
(ID_CUSTOMER,LOGIN,PASSWORD,E_MAIL,PHOME_NUBMER,ROLE) values ('41','Common
user','1112','fdsjfkdsjfk@gmail.com','5665655665','1');
Insert into VELERI.CUSTOMER
(ID_CUSTOMER,LOGIN,PASSWORD,E_MAIL,PHOME_NUBMER,ROLE) values
('1','Admin','root','admin@yandex.ua','0951223335','10');
Insert into VELERI.CUSTOMER
(ID_CUSTOMER,LOGIN,PASSWORD,E_MAIL,PHOME_NUBMER,ROLE) values
('2','System','system','system@yandex.ua','0951221115','0');
Insert into VELERI.CUSTOMER
(ID_CUSTOMER,LOGIN,PASSWORD,E_MAIL,PHOME_NUBMER,ROLE) values
('21','Veleri','123','r.v.veleri@gmail.com','21121212','1');
REM INSERTING into VELERI.ITEM
SET DEFINE OFF;
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('1','Imaginative literature',null,'Wonderful leisure. Everyone will find their
rubric of this section.','2',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('2','Educational literature',null,'Literature for study.','2',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('3','House and Life',null,'Literature for improving you and your
house.','2',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('4','Other',null,'Other interesting book.','2',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('5','Detectives','1','A fascinating story with a lot of
secrets.','1',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('6','Novel','1','A fascinating love/life story.','1',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('7','Fantastic','1','Supernatural adventure.','1',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('8','Study','2','Literature for study.','1',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('9','Life','3','Literature for improving you.','1',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('10','House','3','Literature for improving your house.','1',null,null);
```

```
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('11','Other','4','Other interesting book.','1',null,null);
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('12','Red and black','6','The amazing story of Julien
life.','0','1','black_and_red.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('13','Sherlock Holmes: The Hound of the Baskervilles.','5','Tireless Sherlock
Holmes is still investigating complex cases, and Dr. Watson in this strongly
helps him.','0','2','sherlock_holmes_baskervilles.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('14','Johann Wolfgang von Goethe. Small Works.','11','Johann Wolfgang von
Goethe - a poet, is the eternal pride and glory of
Germany.','0','3','johann_Wolfgang_small_works.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('15','How to find Phoenix','7','The amazing
story.','0','4','forsch_tatiana_phoenix.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('16','This best of all worlds','6','Almost everyone is in the life of the main
love and not a few major non-principal -. Are forgotten and most
importantly.','0','5','tokarev_victoria_best.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('18','English language for students','8','Designed for undergraduate and
specialty architecture universities.','0','7','Sergey_Garagulya_english.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('19','ABC_BOOK','8','ABC_BOOK for your litle child.','0','8','abc.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('20','Exercise book for ABC-book','8','Exercise book for ABC-book of your litle
child.','0','9','abc.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('21','Beads','9','Learn to make amazing jewelry from
beads.','0','10','family_bead_1.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('22','Cross-stitch.','9','Scheme for embroidery.','0','11','family_bead.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('23','New puzzles Sherlock Holmes','11','New puzzles of amazing Sherlock
Holmes. A lot of new secrets and
mysteries.','0','12','sherlock_holmes_puzzles.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('24','Interior Design','10','Even a small apartment can be comfortable! In the
book you will find helpful tips that can help you dramatically change your
house.','0','13','interior_design.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('25','Best recipes','9','Best recipes of Julia
Vysotsky.','0','14','julia_vysotsky_recipes_1.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('26','Best recipes. Part II','9','Best recipes of Julia
Vysotsky.','0','15','julia_vysotsky_recipes_2.jpg');
```

```
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('27','Before I met you','6','Lou Clark do not knows what is about to lose his
job and that in the near future, it will need all the strength to overcome her
problems.','0','16','jojo_moyes_before_met_you.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('28','After you','6','Now Lou Clark is not just an ordinary girl, that living
an ordinary life. Six months spent with Will Traynor, changed her
forever.','0','17','jojo_moyes_after_you.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('29','Divergent','7','In a world where lives Beatrice Prior, people are divided
into five factions. Read continued.','0','18','divergent.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('30','Insurgents','7','New part of Divergent','0','19','insurgents.jpg');
Insert into VELERI.ITEM
(ID_ITEM,NAME,PARENT_ID,DESCRIPTION,TYPE,ID_PROPERTIES,PICTURE) values
('31','Four. History divergents','7','New part of
Divergent','0','20','Four_Divergent.jpg');
REM INSERTING into VELERI.ORDERS
SET DEFINE OFF;
Insert into VELERI.ORDERS (ID_ORDER,ID_CUSTOMER,DATA) values
('1','3',to_date('09.11.18','DD.MM.RR'));
Insert into VELERI.ORDERS (ID_ORDER,ID_CUSTOMER,DATA) values
('42','21',to_date('19.11.18','DD.MM.RR'));
Insert into VELERI.ORDERS (ID_ORDER,ID_CUSTOMER,DATA) values
('43','21',to_date('19.11.18','DD.MM.RR'));
Insert into VELERI.ORDERS (ID_ORDER,ID_CUSTOMER,DATA) values
('44','21',to_date('19.11.18','DD.MM.RR'));
Insert into VELERI.ORDERS (ID_ORDER,ID_CUSTOMER,DATA) values
('45','21',to_date('19.11.18','DD.MM.RR'));
Insert into VELERI.ORDERS (ID_ORDER,ID_CUSTOMER,DATA) values
('21','21',to_date('16.11.18','DD.MM.RR'));
REM INSERTING into VELERI.PROPERTIES
SET DEFINE OFF;
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('1','576','120','10','1');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('2','680','150','5','2');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('3','300','140','10','3');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('4','684','60','10','4');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('5','256','130','7','5');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('6','288','60','6','2');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('7','368','310','3','6');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('8','112','90','5','7');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('9','128','21','4','7');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('10','48','58','5','8');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('11','65','90','4','8');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('12','256','450','3','8');
```

```
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('13','160','110','7','8');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('14','128','370','3','9');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('15','128','370','2','9');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('16','210','107','3','10');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('17','210','105','3','10');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('18','416','160','9','11');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('19','384','120','11','11');
Insert into VELERI.PROPERTIES (ID_BOOK,PAGES,PRICE,AMOUNT,ID_AUTHOR) values
('20','352','150','5','11');
--------------------------------------------------------
--  DDL for Index CONTENR_ORDER_PK1
--------------------------------------------------------

  CREATE UNIQUE INDEX "VELERI"."CONTENR_ORDER_PK1" ON "VELERI"."CONTENR_ORDER"
("ID_CONTENT", "ID_BOOK")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Index ID_ITEM_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "VELERI"."ID_ITEM_PK" ON "VELERI"."ITEM" ("ID_ITEM")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Index ID_CUSTOMER_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "VELERI"."ID_CUSTOMER_PK" ON "VELERI"."CUSTOMER"
("ID_CUSTOMER")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Index CONTENR_ORDER_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "VELERI"."CONTENR_ORDER_PK" ON "VELERI"."CONTENR_ORDER"
("ID_CONTENT", "ID_BOOK", "ID_ORDER")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Index ID_BOOK_PK
--------------------------------------------------------
```

```
  CREATE UNIQUE INDEX "VELERI"."ID_BOOK_PK" ON "VELERI"."PROPERTIES" ("ID_BOOK")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Index ORDERS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "VELERI"."ORDERS_PK" ON "VELERI"."ORDERS" ("ID_ORDER",
"ID_CUSTOMER")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Index AUTHOR_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "VELERI"."AUTHOR_PK" ON "VELERI"."AUTHOR" ("ID_AUTHOR")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Index ORDERS_UK_ID
--------------------------------------------------------

  CREATE UNIQUE INDEX "VELERI"."ORDERS_UK_ID" ON "VELERI"."ORDERS" ("ID_ORDER")
  PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS" ;
--------------------------------------------------------
--  DDL for Trigger TR_NEW_AUTHOR
--------------------------------------------------------

  CREATE OR REPLACE TRIGGER "VELERI"."TR_NEW_AUTHOR"
BEFORE INSERT ON AUTHOR
FOR EACH ROW

BEGIN
SELECT SEQUENCE_AUTHOR.NEXTVAL INTO :NEW.ID_AUTHOR FROM dual;
END;

/
ALTER TRIGGER "VELERI"."TR_NEW_AUTHOR" ENABLE;
--------------------------------------------------------
--  DDL for Trigger TR_NEW_CON
--------------------------------------------------------

  CREATE OR REPLACE TRIGGER "VELERI"."TR_NEW_CON"
  BEFORE INSERT ON CONTENR_ORDER
FOR EACH ROW
BEGIN
  SELECT SEQUENCE_CONTENT.NEXTVAL INTO :NEW.ID_CONTENT FROM dual;
END;
```

```
/
ALTER TRIGGER "VELERI"."TR_NEW_CON" ENABLE;
--------------------------------------------------------
--  DDL for Trigger TR_NEW_CUSTOMER
--------------------------------------------------------
  CREATE OR REPLACE TRIGGER "VELERI"."TR_NEW_CUSTOMER"
BEFORE INSERT ON CUSTOMER
FOR EACH ROW
BEGIN
SELECT SEQUENCE_CUSTOMER.NEXTVAL INTO :NEW.ID_CUSTOMER FROM dual;
END;/
ALTER TRIGGER "VELERI"."TR_NEW_CUSTOMER" ENABLE;
--------------------------------------------------------
--  DDL for Trigger TR_NEW_ITEM
--------------------------------------------------------

  CREATE OR REPLACE TRIGGER "VELERI"."TR_NEW_ITEM"
BEFORE INSERT ON ITEM
FOR EACH ROW
DECLARE id_t NUMBER;

type_t NUMBER;
BEGIN
SELECT SEQUENCE_ITEMS.NEXTVAL INTO :NEW.ID_ITEM FROM dual;
IF inserting then
type_t:=:new.TYPE;
IF type_t =0 THEN
id_t:=SEQUENCE_BOOK.NEXTVAL;

SELECT id_t INTO :NEW.ID_PROPERTIES FROM dual;

INSERT INTO PROPERTIES(ID_BOOK) values(id_t);
END IF;
END IF;
END;

/
ALTER TRIGGER "VELERI"."TR_NEW_ITEM" ENABLE;
--------------------------------------------------------
--  DDL for Procedure ADDBOOK
--------------------------------------------------------
set define off;

  CREATE OR REPLACE PROCEDURE "VELERI"."ADDBOOK"
    (Name_p IN varchar2,
     DESCRIPTION_p IN varchar2,
     RUBRIC_p IN NUMBER,
     AUTHOR_ID IN NUMBER,
     PAGES_p IN NUMBER,
     PRICE_p IN FLOAT,
     AMOUNT_p IN NUMBER
    )
  IS

    BEGIN
      INSERT INTO ITEM(NAME,TYPE,DESCRIPTION,PARENT_ID)
values(Name_p,0,DESCRIPTION_p,RUBRIC_p);
      UPDATE  PROPERTIES
      SET PAGES=PAGES_p,PRICE=PRICE_p,AMOUNT=AMOUNT_p,ID_AUTHOR=AUTHOR_ID
      WHERE PROPERTIES.ID_BOOK=(SELECT ID_PROPERTIES  FROM ITEM i,PROPERTIES
      WHERE i.ID_PROPERTIES=PROPERTIES.ID_BOOK
            AND  i.NAME=Name_p
```

```
            AND i.TYPE=0
            AND i.DESCRIPTION = DESCRIPTION_p
            AND i.PARENT_ID = RUBRIC_p);

      COMMIT;
      EXCEPTION
      WHEN OTHERS THEN
      ROLLBACK;
      RETURN;
    END;

/
-----------------------------------------------------------
--  DDL for Procedure ADDORDER
-----------------------------------------------------------
set define off;

  CREATE OR REPLACE PROCEDURE "VELERI"."ADDORDER"
  (
  CUSTOMER IN NUMBER,
  BOOK IN NUMBER,
  AMOUNT_c IN NUMBER,
  DATA_C IN DATE
  )
  IS
   id_or NUMBER;
      BEGIN
  id_or:=SEQUENCE_ORDERS.NEXTVAL;
INSERT INTO ORDERS(ID_ORDER,ID_CUSTOMER,DATA) values(id_or,CUSTOMER,DATA_C);
 INSERT INTO CONTENR_ORDER(ID_ORDER,ID_BOOK,AMOUNT) values(id_or,BOOK,AMOUNT_c);

  COMMIT;
 EXCEPTION
 WHEN OTHERS THEN
 ROLLBACK;
 RETURN;
  END;


/
-----------------------------------------------------------
--  DDL for Procedure DELETEBOOK
-----------------------------------------------------------
set define off;

  CREATE OR REPLACE PROCEDURE "VELERI"."DELETEBOOK"
(
Book_id IN NUMBER
)

IS
BEGIN

DELETE FROM ORDERS WHERE ID_ORDER=(
SELECT ID_ORDER
FROM CONTENR_ORDER
WHERE CONTENR_ORDER.ID_BOOK=Book_id
AND ROWNUM=1);
DELETE FROM ITEM WHERE ID_ITEM=Book_id;
COMMIT;

EXCEPTION
```

```
WHEN OTHERS THEN
ROLLBACK;
RETURN;
END;

/
------------------------------------------------------
--  DDL for Function CUSTOM_AUTH
------------------------------------------------------

  CREATE OR REPLACE FUNCTION "VELERI"."CUSTOM_AUTH" (p_username in VARCHAR2,
p_password in VARCHAR2)
return BOOLEAN
is
  l_password varchar2(4000);
  l_stored_password varchar2(4000);
  l_expires_on date;
  l_count number;
begin
-- First, check to see if the user is in the user table
select count(*) into l_count from demo_users where user_name = p_username;
if l_count > 0 then
  -- First, we fetch the stored hashed password & expire date
  select password, expires_on into l_stored_password, l_expires_on
   from demo_users where user_name = p_username;

  -- Next, we check to see if the user's account is expired
  -- If it is, return FALSE
  if l_expires_on > sysdate or l_expires_on is null then

    -- If the account is not expired, we have to apply the custom hash
    -- function to the password
    l_password := custom_hash(p_username, p_password);

    -- Finally, we compare them to see if they are the same and return
    -- either TRUE or FALSE
    if l_password = l_stored_password then
      return true;
    else
      return false;
    end if;
  else
    return false;
  end if;
else
  -- The username provided is not in the DEMO_USERS table
  return false;
end if;
end;

/
------------------------------------------------------
--  DDL for Function CUSTOM_HASH
------------------------------------------------------

  CREATE OR REPLACE FUNCTION "VELERI"."CUSTOM_HASH" (p_username in varchar2,
p_password in varchar2)
return varchar2
is
  l_password varchar2(4000);
  l_salt varchar2(4000) := '1MCUOL1KDPFZO7R1WQNCRI55FPHRP3';
begin
```

```
-- This function should be wrapped, as the hash algorhythm is exposed here.
-- You can change the value of l_salt or the method of which to call the
-- DBMS_OBFUSCATOIN toolkit, but you much reset all of your passwords
-- if you choose to do this.

l_password := utl_raw.cast_to_raw(dbms_obfuscation_toolkit.md5
  (input_string => p_password || substr(l_salt,10,13) || p_username ||
    substr(l_salt, 4,10)));
return l_password;
end;

/
--------------------------------------------------------
--  DDL for Function ORDER_COST
--------------------------------------------------------

  CREATE OR REPLACE FUNCTION "VELERI"."ORDER_COST" (ID_OR IN NUMBER)
   RETURN FLOAT
   IS
    TOTAL_COST FLOAT;
   BEGIN
   SELECT SUM(c.AMOUNT*p.PRICE)
   INTO TOTAL_COST
   FROM CONTENR_ORDER c, PROPERTIES p, ITEM i
   WHERE i.ID_PROPERTIES=p.ID_BOOK
   AND ID_OR = c.ID_ORDER
   AND c.ID_BOOK=i.ID_ITEM;

   RETURN TOTAL_COST;
   END;


/
--------------------------------------------------------
--  DDL for Function ORDER_COST_DISCOUNT
--------------------------------------------------------

  CREATE OR REPLACE FUNCTION "VELERI"."ORDER_COST_DISCOUNT" (ID_OR IN NUMBER)
   RETURN FLOAT
   IS
   TOTAL_COST FLOAT;
   DISCOUNT FLOAT;
   BEGIN
   SELECT SUM(c.AMOUNT*p.PRICE)
   INTO TOTAL_COST
   FROM CONTENR_ORDER c, PROPERTIES p, ITEM i
   WHERE i.ID_PROPERTIES=p.ID_BOOK
   AND ID_OR = c.ID_ORDER
   AND c.ID_BOOK=i.ID_ITEM;
  IF(TOTAL_COST>200) THEN
    DISCOUNT:=TOTAL_COST*0.2;
    TOTAL_COST:=TOTAL_COST-DISCOUNT;
    END IF;
   RETURN TOTAL_COST;
   END;


/
--------------------------------------------------------
--  Constraints for Table PROPERTIES
--------------------------------------------------------
```

```
  ALTER TABLE "VELERI"."PROPERTIES" MODIFY ("ID_BOOK" NOT NULL ENABLE);
  ALTER TABLE "VELERI"."PROPERTIES" ADD CONSTRAINT "PROPERTIES_PK" PRIMARY KEY
("ID_BOOK")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
--------------------------------------------------------
--  Constraints for Table CONTENR_ORDER
--------------------------------------------------------

  ALTER TABLE "VELERI"."CONTENR_ORDER" ADD CONSTRAINT "CONTENR_ORDER_PK" PRIMARY
KEY ("ID_CONTENT", "ID_BOOK")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "VELERI"."CONTENR_ORDER" MODIFY ("ID_ORDER" NOT NULL ENABLE);
  ALTER TABLE "VELERI"."CONTENR_ORDER" MODIFY ("ID_CONTENT" NOT NULL ENABLE);
--------------------------------------------------------
--  Constraints for Table ORDERS
--------------------------------------------------------

  ALTER TABLE "VELERI"."ORDERS" ADD CONSTRAINT "ORDERS_PK" PRIMARY KEY
("ID_ORDER", "ID_CUSTOMER")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "VELERI"."ORDERS" ADD CONSTRAINT "ORDERS_UK_ID" UNIQUE
("ID_ORDER")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
  ALTER TABLE "VELERI"."ORDERS" MODIFY ("ID_CUSTOMER" NOT NULL ENABLE);
  ALTER TABLE "VELERI"."ORDERS" MODIFY ("ID_ORDER" NOT NULL ENABLE);
--------------------------------------------------------
--  Constraints for Table ITEM
--------------------------------------------------------

  ALTER TABLE "VELERI"."ITEM" MODIFY ("TYPE" NOT NULL ENABLE);
  ALTER TABLE "VELERI"."ITEM" ADD CONSTRAINT "ITEM_PK" PRIMARY KEY ("ID_ITEM")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
--------------------------------------------------------
--  Constraints for Table CUSTOMER
--------------------------------------------------------

  ALTER TABLE "VELERI"."CUSTOMER" MODIFY ("ROLE" NOT NULL ENABLE);
  ALTER TABLE "VELERI"."CUSTOMER" MODIFY ("ID_CUSTOMER" NOT NULL ENABLE);
  ALTER TABLE "VELERI"."CUSTOMER" ADD CONSTRAINT "ID_CUSTOMER" PRIMARY KEY
("ID_CUSTOMER")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
```

```
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
--------------------------------------------------------
--  Constraints for Table AUTHOR
--------------------------------------------------------

  ALTER TABLE "VELERI"."AUTHOR" MODIFY ("ID_AUTHOR" NOT NULL ENABLE);
  ALTER TABLE "VELERI"."AUTHOR" ADD CONSTRAINT "AUTHOR_PK" PRIMARY KEY
("ID_AUTHOR")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT FLASH_CACHE
DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table CONTENR_ORDER
--------------------------------------------------------

  ALTER TABLE "VELERI"."CONTENR_ORDER" ADD CONSTRAINT "CONTENR_ORDER_FK2"
FOREIGN KEY ("ID_BOOK")
        REFERENCES "VELERI"."ITEM" ("ID_ITEM") ON DELETE CASCADE ENABLE;
  ALTER TABLE "VELERI"."CONTENR_ORDER" ADD CONSTRAINT "CONTENR_ORDER_ORDER_FK"
FOREIGN KEY ("ID_ORDER")
        REFERENCES "VELERI"."ORDERS" ("ID_ORDER") ON DELETE CASCADE ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table ITEM
--------------------------------------------------------

  ALTER TABLE "VELERI"."ITEM" ADD CONSTRAINT "ITEM_FK1" FOREIGN KEY
("ID_PROPERTIES")
        REFERENCES "VELERI"."PROPERTIES" ("ID_BOOK") ENABLE;
  ALTER TABLE "VELERI"."ITEM" ADD CONSTRAINT "PARENT_FK" FOREIGN KEY
("PARENT_ID")
        REFERENCES "VELERI"."ITEM" ("ID_ITEM") ON DELETE CASCADE ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table ORDERS
--------------------------------------------------------

  ALTER TABLE "VELERI"."ORDERS" ADD CONSTRAINT "ORDERS_CUSTOMER_FK" FOREIGN KEY
("ID_CUSTOMER")
        REFERENCES "VELERI"."CUSTOMER" ("ID_CUSTOMER") ON DELETE CASCADE ENABLE;
--------------------------------------------------------
--  Ref Constraints for Table PROPERTIES
--------------------------------------------------------

  ALTER TABLE "VELERI"."PROPERTIES" ADD CONSTRAINT "PROPERTIES_FK1" FOREIGN KEY
("ID_AUTHOR")
        REFERENCES "VELERI"."AUTHOR" ("ID_AUTHOR") ON DELETE CASCADE ENABLE;
```

Додаток В

Лістинг В.1. – Зміст файлу pom.xml

```
<goals><project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>BookShopGroupID</groupId>
  <artifactId>BookShopArtifactID</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>BookShopArtifactID Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
    </dependency>

    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
    </dependency>
    <dependency>
      <groupId>javax.mail</groupId>
      <artifactId>mail</artifactId>
      <version>1.4.2</version>
    </dependency>


  </dependencies>

  <build>
    <finalName>BookShopArtifactID</finalName>

    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.6</version>
        <configuration>
          <packagingExcludes>WEB-INF/web.xml</packagingExcludes>
        </configuration>
      </plugin>

      <plugin>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.8.2</version>
```

```
        <configuration>
          <skip>true</skip>
        </configuration>
      </plugin>

      <plugin>
        <groupId>com.oracle.weblogic</groupId>
        <artifactId>weblogic-maven-plugin</artifactId>
        <version>12.2.1-0-0</version>
        <configuration>
          <adminurl>t3://localhost:7001</adminurl>
          <user>weblogic</user>
          <password>220162Liar</password>
          <name>WLSLocale</name>
          <remote>true</remote>
          <upload>true</upload>
          <targets>AdminServer</targets>
        </configuration>

        <executions>
          <execution>
            <id>deploy</id>
            <phase>pre-integration-test</phase>

              <goal>deploy</goal>
            </goals>
            <configuration>
              <source>target/BookShopArtifactID.war</source>
            </configuration>
          </execution>
        </executions>

      </plugin>

      <plugin>
        <artifactId>maven-ear-plugin</artifactId>
        <version>2.10.1</version>
        <configuration>
          <!--<packagingIncludes>META-INF/**,**/Book*.war</packagingIncludes>-->
          <generatedDescriptorLocation>${basedir}/src/main/application/META-
INF</generatedDescriptorLocation>
          <modules>
            <webModule>
              <groupId>BookShopGroupID</groupId>
              <artifactId>BookShopArtifactID</artifactId>
              <!--<bundleFileName>.war</bundleFileName>-->
              <contextRoot>/BookShopArtifactID</contextRoot>
            </webModule>
          </modules>
          <displayName>BookShop</displayName>
          <earSourceDirectory>${basedir}</earSourceDirectory>
          <earSourceIncludes>**/META-INF/web.xml</earSourceIncludes>
          <generateApplicationXml>true</generateApplicationXml>
        </configuration>
      </plugin>

      <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <version>2.3</version>
        <configuration>
          <encoding>UTF-8</encoding>
        </configuration>
```

```
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>

    </plugins>

  </build>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  </properties>

  <distributionManagement>
    <snapshotRepository>
      <id>BookShopArtifactID</id>
      <name>Diplom_Valerie</name>
      <url>file://C:/Users/vare0516/.m2/diplomM2</url>
    </snapshotRepository>
  </distributionManagement>

</project>
```

Лістинг В.2. – Зміст jsp-сторінки «AboutUs»

```
<%--
 Document   : bookDetail
 Author     : Veleri Vovk
--%>

<HTML>

<HEAD>

    <TITLE>login</TITLE>

    <style type="text/css">

        <!--
        @import url("css/style.css");
        -->
    </style>

</HEAD>
    <BODY>

    <jsp:include page="Head.jsp" />
    <jsp:include page="Menu.jsp" />
    <jsp:include page="Sidebar.jsp" />
    <jsp:include page="Footer.jsp" />

        <p class="textSideBar2" align="center">BookSop is online bookstore
number one! </P>
        <p class="textSideBar2" align="center">We have only quality books from
reliable publishers.</p>

        <p class="textWhite"> white text </p>
        <p class="textSideBar2Small" align="center"> Our contact </p>
        <p class="textSideBar2Small" align="center">Ukraine, Sumy,
Petropavlovskaya street 7</p>

        <p class="textWhite"> white text </p>
        <p class="textSideBar2Small" align="center"> Call</p>
        <p class="textSideBar2Small" align="center"> +38095 467 98 17</p>

        <p class="textWhite"> white text </p>
        <p class="textSideBar2Small" align="center"> Work schedule: 9:00 -
19:00</p>

        <jsp:include page="Footer.jsp" />

    </BODY>

</HTML>
```

Лістинг В.3. – Зміст jsp-сторінки «AddItem»

```
<%@ page import="model.Customer" %>
<%@ page import="model.Book" %>
<%@ page import="java.util.List" %>
<%@ page import="model.Item" %>
<%@ page import="controller.processors.*" %>
<%@ page errorPage="errorPage.jsp"%>
<div id="modal_form">
    <div>
        <h3>Type the fields!</h3>
    </div>
    <div >
        <form name = "add_form" method="post"
action="MainServlet?action=addRubric">
            <p class="prob">
            <div><label for="<%=AddItem.ITEM_NAME+"ID"%>">Name</label></div>
            <input type="text" id ="<%=AddItem.ITEM_NAME+"ID"%>" name
="<%=AddItem.ITEM_NAME%>"
                    placeholder="Rubric name" required/>
            </label>
            </p>
            <p class="prob">
            <div><label for="<%=AddItem.ITEM_DESCRIPTION + "ID"%>">Description
</label></div>
            <input id ="<%=AddItem.ITEM_DESCRIPTION+"ID"%>" type="text" name
="<%=AddItem.ITEM_DESCRIPTION%>"
                    placeholder="description"required/>
            </p>
            <p class="prob">
            <div><label for="<%=AddItem.ITEM_SECTION_NAME +
"ID"%>">Section</label></div>
            <p>
            <div>
                <select id = "<%=AddItem.ITEM_SECTION_NAME + "ID"%>"
name="<%=AddItem.ITEM_SECTION_NAME%>" size="1" required>
                    <option disabled selected></option>
                    <optgroup label="Add new section">
                        <option>Create section</option>
                    </optgroup>
                    <% List<Item> sections = (List<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_SECTION);%>
                        <optgroup label="Sections:">
                        <%for(Item section:sections){%>

                    <option ><%=section.getName()%></option>
                    <%}%>
                        </optgroup>
                </select>
            </div>
            </p>
            </p>
            <div class="prob">  <input class="btn" type="submit"  value="Create
rubric" required /></div>
        </form>
    </div>
                    <span id="modal_close">
                        <a class="close"  title="Close" >X</a>
                    </span>

</div>
<div id="overlay"></div>
```

Лістинг В.4. – Зміст jsp-сторінки «BookDetail»

```
<%@ page import="model.Book" %>
<%@ page import="model.Customer" %>
<%@ page errorPage="errorPage.jsp"%>
<%--
 Document   : bookDetail
 Author     : Veleri
--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%   Book book = (Book)session.getAttribute("DetailBook");
    Boolean isAdmin=false;
    String buttonName ="";
    String modalForm ;
    Boolean isEdit;
    try {
        Customer cus = (Customer) session.getAttribute("customer");
        buttonName ="Buy";
        isEdit=false;
        if(cus.getRole()==10){
            isAdmin=true;
            buttonName ="Edit";
            isEdit=true;
        }
    }
    catch (NullPointerException e1){
        isAdmin= false;
        buttonName ="Buy";
        isEdit=false;
    }
%>
<html>
<head>
    <title>Books information</title>
    <style type="text/css">
        <!--
        @import url("css/style.css");
        -->
    </style>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
    <script src="js/main.js"></script>
    <script src="js/modal.js"></script>
</head>
<body>
 <jsp:include page="Head.jsp" />
 <jsp:include page="Menu.jsp" />
 <jsp:include page="Sidebar.jsp" />
 <jsp:include page="Footer.jsp" />
<br>

<div class="bookInf">
    <table class="table">
        <tr>
            <td>Book's name:</td>

            <td> <%=book.getName()%> </td>
        </tr>
        <tr>
            <td>Rubric:</td>
            <td> <%=book.getParent().getName()%> </td>
```

```
        </tr>
        <tr>
            <td>Author:</td>
            <td><%=book.getAuthor()%></td>
        </tr>
    </table>
</div>
<br>

<br>
<div class="sideBook2">

            <table>
                <tr align ="center">
                    <td colspan="2" align="center"><img
src="images/books/<%=book.getPicture()%>" width="130" height="200"
align="center"/></td>
                     <td>Description:</td>  <td><%=book.getDescription()%> </td>
                </tr >
                <tr align ="center">
                    <td>Price:</td> <td>  <%=book.getPrice()%> </td>
<td>Pages:</td>  <td><%=book.getPages()%> </td>
                <tr/>
                <tr align ="center">
                     <td>Amount:</td>  <td> <%=book.getAmount()%> </td>    <th
colspan="2" align ="center">

                    <p>
                        <p align="right">
                            <a href="#" id="<%=isAdmin?"edit":"go"%>">
                                <input type="submit" value="<%=buttonName %>"
class="book2">
                            </a>

                            <%if( isAdmin) {

request.getSession().setAttribute("AuthorID",book.getAuthor().getId());

request.getSession().setAttribute(UpdateBook.BOOK_RUBRIC,book.getParent());
                            %>

                                <%@include file="EditModalForm.jsp"%>
                            <%}else {
                            boolean isMain =false;%>

                                <%@include file="BuyModalForm.jsp"%>
                            <%}%>
                        </p>
                    </p>
            </th>
                </tr>
            </table>

</div>

</br>
</body>
</html>
```

Лістинг В.5. – Зміст jsp-сторінки «BuyModalForm»

```
<%@ page import="controller.processors.AddOrder" %>
<%@ page import="controller.processors.AddCustomer" %>

<div id="<%=isMain?"modal_form" +book.getId():"modal_form"%>">

    <div >
<h3>
        Do you want to buy a book?<br>Please, fill in the form below.<br>
</h3>
    </div>

    <div >
        <%Customer cus = (Customer)
request.getSession().getAttribute("customer");%>

        <form name = "buy_form" method="post"
action="<%="MainServlet?action=addOrder&IdDetail=" + book.getId() %>">
            <% if (cus ==null){%>

            <p class="prob"><input type="tel" name ="<%=AddCustomer.CUS_PHONE
%>" <%=cus!=null?"value=" + cus.getPhone():"placeholder=phone" %> required/>
</p>
            <p class="prob"><input type="email" name="<%=AddCustomer.CUS_E_MAIL
%>" <%=cus!=null?"value=" + cus.getMail():"placeholder=email"%> required/>
</p>
            <%}%>
            <p class="prob"><input type="number" name="<%=AddOrder.Book_Amount
%>" placeholder="amount" required/>
</p>
          <div>  <input class="btn" type="submit" value="Buy book" />
</div>
        </form>

    </div>

                    <span id="modal_close">
                        <a class="close"  title="Close" >X</a>
                    </span>

</div>

      <div id="overlay">
</div>
```

Лістинг В.6. – Зміст jsp-сторінки «DeleteBook»

```
<%@ page import="controller.processors.UpdateBook" %>

<div id="<%="modal_form" + book.getId()%>" >

    <div >
        <h3>Are you sure?</h3>
    </div>

    <div >

        <form name = "buy_form" method="post"
action="<%="MainServlet?action=deleteBook&IdDetail=" + book.getId()%>">

            <p class="prob">
<div>
        <label for="<%=UpdateBook.BOOK_NAME + "ID"%>">Name</label>
</div>
            <input type="text" id ="<%=UpdateBook.BOOK_NAME+"ID"%>"  name
="<%=UpdateBook.BOOK_NAME%>" value= "<%=book.getName() %>" />
            </label>
            </p>

            <p class="prob">
            <div>
        <label for="<%=UpdateBook.BOOK_AUTHOR_ID %>">Author</label>
</div>
            <input type="text" id ="<%=UpdateBook.BOOK_AUTHOR_ID%>"
name="<%=UpdateBook.BOOK_AUTHOR_ID%>" value= "<%=book.getAuthor() %>" />
            </p>

            <p class="prob">
             <div>
                <label  for="<%=UpdateBook.BOOK_PRICE + "ID"%>">Price</label>
          </div>
            <input id ="<%=UpdateBook.BOOK_PRICE+"ID"%>" type="number"
name="<%=UpdateBook.BOOK_PRICE%>" value= "<%=book.getPrice() %>" />
            </p>
            <div class="prob">
                 <input class="btn" type="submit"  value="Delete book" />
             </div>

        </form>
    </div>
                    <span id="modal_close">
                        <a class="close"  title="Close" >X</a>
                    </span>

</div>
<div id="overlay"></div>
```

Лістинг В.7. – Зміст jsp-сторінки «DeleteItem»

```
<%@ page import="model.Customer" %>
<%@ page import="model.Book" %>
<%@ page import="java.util.List" %>
<%@ page import="model.Item" %>
<%@ page import="controller.processors.*" %>
<%@ page errorPage="errorPage.jsp"%>


<div id="del_modal_form">
    <div>
        <h3>Choose rubric what you need <br/> to delete </h3>
    </div>
    <div >
        <form name = "add_form" method="post"
action="MainServlet?action=deleteItem">
            <p class="prob">
            <div><label for="<%=AddItem.ITEM_NAME + "ID"%>"></label></div>
            <p>
            <div>
                <select id = "<%=AddItem.ITEM_NAME + "ID"%>"
name="<%=AddItem.ITEM_NAME%>" size="1" required>
                    <option disabled selected></option>
                    <% List<Item> rubrics = (List<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_SECTION);%>
                    <optgroup label="Sections">
                        <%for(Item rubric:rubrics){%>
                        <option ><%=rubric.getName()%></option>
                    <%}%>
                    </optgroup>
                    <%  rubrics = (List<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_All_CATEGORY);%>
                    <optgroup label="Rubrics:">
                        <%for(Item rubric:rubrics){%>
                        <option ><%=rubric.getName()%></option>

                    <%}%>
                            </optgroup>
                </select>
            </div>
            </p>
            </p>

            <div class="prob">  <input class="btn" type="submit"  value="Delete"
required /></div>
        </form>
    </div>
                    <span id="del_modal_close">
                        <a class="close"  title="Close" >X</a>
                    </span>

</div>
<div id="overlay"></div>
```

Лістинг В.8. – Зміст jsp-сторінки «DeleteOrder»

```
<%@ page import="controller.processors.UpdateBook" %>
<%@ page import="controller.processors.DeleteOrder" %>
<div id="<%="modal_form" + order.getIdOrder()%>" >
    <div >
        <h3>Are you sure?</h3>
    </div>
    <div >
        <form name = "buy_form" method="post"
action="MainServlet?action=deleteOrder">
            <input type="hidden" value="<%=order.getIdOrder()%>"
name="<%=UpdateOrder.UPDATE_ORDER_ID%>">
            <div class="prob">  <input class="btn" type="submit"  value="Delete
order" /></div>
        </form>
    </div>
                    <span id="modal_close">
                        <a class="close"  title="Close" >X</a>
                    </span>

</div>
<div id="overlay"></div>
```

Лістинг В.9. – Зміст jsp-сторінки «Delivery»

```
<HTML>
<HEAD>
    <TITLE>login</TITLE>
    <style type="text/css">
        <!--
        @import url("css/style.css");
        -->
    </style>
</HEAD>
<BODY>
<jsp:include page="Head.jsp" />
<jsp:include page="Menu.jsp" />
<jsp:include page="Sidebar.jsp" />
<jsp:include page="Footer.jsp" />
<div class="content">
<p class="textSideBar2" align="center">DELIVERY AND PAYMENT</p>

<p class="textSideBar2Small"> Be sure to leave your current phone number,
because without prior communication with you the order will not be shipped !</p>

<p class="textSideBar2Small">Delivery in Ukraine in the amount of up to 100 UAH
is carried out only with prepayment to the card.</p>
<p class="textWhite"> white text </p>
<p class="textSideBar2">Types of delivery:</p>

<p class="textSideBar2Small">1. Delivery in Ukraine is carried out exclusively
by the service "New Mail". Shipping cost: about 70 UAH when paying by cash on
delivery and from 35 UAH when paying by Privatbank card. The payment option
convenient for you should be indicated on the order form. Delivery time: 1-3
days depending on the distance of the region. Please indicate in your order the
number of the "New Mail" branch in your settlement, where it will be most
convenient for you to pick up the order.</p>

<p class="textSideBar2Small">2. Delivery outside Ukraine is carried out by
```

```
airmail only after 100% prepayment. Shipping cost depends on the weight of the
package. Prepayment is made by bank transfer Western Union or through the Golden
Crown payment system. Delivery time: 7 - 45 days.</p>

<p class="textSideBar2Small">3. Courier delivery. Delivery by courier in Kharkiv
within metro stations - 50 UAH. The cost of shipping orders directly to the
customer - 75 UAH. In the place and time specified by the operator - for
free.</p>
<p class="textWhite"> white text </p>
<p class="textSideBar2Small">IMPORTANT! If within 5 business days you do not
pick up the parcel, on the 6th day the parcel automatically returns to us. Re-
sending is possible only after compensation for shipment in both directions or
full prepayment for subsequent shipments.</p>
<p class="textWhite"> white text </p>
<p class="textSideBar2">Return or exchange conditions:</p>

<p class="textSideBar2Small">1) Return or exchange of goods is carried out
through the transport company "New Mail". You pay return service (about 35
UAH.)</p>

<p class="textSideBar2Small">2) After sending a refund or exchange, it is
necessary to tell us the parcel invoice number.</p>

<p class="textSideBar2Small">3) The goods must be returned in their original
form, without damage, pollution and intact packaging.</p>

<p class="textSideBar2Small">4) We receive a return package, check the quantity
and quality of the goods. If there are no damages and claims to the product and
packaging, we will refund or exchange.</p>
<p class="textWhite"> white text </p>
<p class="textSideBar2Small">5) Refund is made on the card Privatbank.</p>

<p class="textSideBar2Small">You can also use the service "easy return" from the
company "New Mail", if you need to quickly and conveniently return the goods in
the period up to 14 days from the receipt of the order. For this, it is enough
to contact any branch of “New Mail” and call the EH number (express waybill),
you do not need to enter the rest of the information, this will help save your
time when sending.</p>

<p class="textSideBar2Small">You receive the invoice and details for payment of
the order by e-mail or by sms-message after confirming the order made and the
amount of payment by the manager of the store. You can pay at any branch of
Privat Bank (or another convenient financial institution), as well as through an
online payment system. After paying the invoice, the manager forms and sends the
order to your city. If the payment is made before 13-00, the order is sent on
the same day (except Saturday and Sunday). </p>

</div>
<jsp:include page="Footer.jsp" />
</BODY>
</HTML>
```

Лістинг В.10. – Зміст jsp-сторінки «EditModalForm»

```
<%@ page import="model.Customer" %>
<%@ page import="controller.processors.AddCustomer" %>
<%@ page import="controller.processors.AddOrder" %>
<%@ page import="model.Book" %>
<%@ page import="controller.processors.UpdateBook" %>
<%@ page import="java.util.List" %>
<%@ page import="model.Item" %>
<%@ page import="controller.processors.Welcome" %>
<%@ page errorPage="errorPage.jsp"%>


<div id="edit_modal_form">
    <div>
        <h3><%=isEdit?"Change the fields that you need":"Type the
fields!"%></h3>
    </div>
    <div >
        <form name = "edit_form" method="post"
action="<%=isEdit?"MainServlet?action=updateBook&IdDetail=" +
book.getId():"MainServlet?action=addBook"%>">

            <p class="prob">
            <div><label for="<%=UpdateBook.BOOK_NAME +
"ID"%>">Name</label></div>
            <input type="text" id ="<%=UpdateBook.BOOK_NAME+"ID"%>"  name
="<%=UpdateBook.BOOK_NAME%>"
                    <%=isEdit?"value=" + "\"" +  book.getName()
+"\"":"placeholder=\"Book name\"" %>required/>
            </label>
            </p>

            <p class="prob">
            <div> <label for="<%=UpdateBook.BOOK_AMOUNT +
"ID"%>">Amount</label></div>
            <input type="number" id ="<%=UpdateBook.BOOK_AMOUNT+"ID"%>"
name="<%=UpdateBook.BOOK_AMOUNT%>"
                    <%=isEdit?"value=" + "\""+ book.getAmount()+
"\"":"placeholder=\"amount\"" %> required/>
            </p>

            <p class="prob">
            <div><label  for="<%=UpdateBook.BOOK_PAGES +
"ID"%>">Pages</label></div>
            <input id ="<%=UpdateBook.BOOK_PAGES+"ID"%>" type="number"
name="<%=UpdateBook.BOOK_PAGES%>"
                    <%=isEdit?"value="+ "\"" + book.getPages()+
"\"":"placeholder=\"pages\"" %>required/>
            </p>

            <p class="prob">
            <div><label for="<%=UpdateBook.BOOK_AUTHOR_NAME + "ID"%>">Author
name </label></div>
            <input id ="<%=UpdateBook.BOOK_AUTHOR_NAME+"ID"%>" type="text" name
="<%=UpdateBook.BOOK_AUTHOR_NAME%>"
                    <%=isEdit?"value=" + "\""+ book.getAuthor().getName()+
"\"":"placeholder=\"Author name\"" %>required/>
            </p>

            <p class="prob">
            <div><label for="<%=UpdateBook.BOOK_AUTHOR_SURNAME + "ID"%>">Author
```

```
surname </label></div>
            <input id ="<%=UpdateBook.BOOK_AUTHOR_SURNAME+"ID"%>" type="text"
name ="<%=UpdateBook.BOOK_AUTHOR_SURNAME%>"
                    <%=isEdit?"value=" + "\""+ book.getAuthor().getSurname()+
"\"":"placeholder=\"Author surname\"" %>required/>
            </p>

            <p class="prob">
            <div><label for="<%=UpdateBook.BOOK_PRICE + "ID"%>">Price
</label></div>
            <input id ="<%=UpdateBook.BOOK_PRICE+"ID"%>" type="number" name
="<%=UpdateBook.BOOK_PRICE%>"
                    <%=isEdit?"value="+ "\"" + book.getPrice()+
"\"":"placeholder=\"price\"" %>required/>
            </p>

            <p class="prob">
            <div><label for="<%=UpdateBook.BOOK_DESCRIPTION +
"ID"%>">Description </label></div>
            <input id ="<%=UpdateBook.BOOK_DESCRIPTION+"ID"%>" type="text" name
="<%=UpdateBook.BOOK_DESCRIPTION%>"
                    <%=isEdit?"value=" + "\""+ book.getDescription()+
"\"":"placeholder=\"description\"" %>required/>
            </p>
            <p class="prob">
            <div><label for="<%=UpdateBook.BOOK_RUBRIC_NAME +
"ID"%>">Rubric</label></div>
            <p>
            <div>
                <select id = "<%=UpdateBook.BOOK_RUBRIC_NAME + "ID"%>"
name="<%=UpdateBook.BOOK_RUBRIC_NAME%>" size="1" required>
                    <option selected> <%=isEdit?book.getParent().getName():" "
%></option>
                    <% List<Item> rubrics = (List<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_All_CATEGORY);
                        for(Item rubric:rubrics){
                            if(!rubric.equals(book.getParent())){%>
                            <option  ><%=rubric.getName()%></option>
                            <%}
                    }%>
                </select>
            </div>
            </p>
            </p>

            <div class="prob">  <input class="btn" type="submit"
value="<%=isEdit?"Edit book":"Create book"%>"required /></div>
        </form>
    </div>
                    <span id="edit_modal_close">
                        <a class="close"  title="Close" >X</a>
                    </span>

</div>
<div id="overlay"></div>
```

Лістинг В.11. – Зміст jsp-сторінки «ErrorPage»

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page isErrorPage="true"%>

<html>
<head>
    <title>Error</title>
</head>
    <body bgcolor="white">
        <p><% exception.getCause(); %>
        </p>
        <p><center><h1>Error</h1></center></p>
        <p><h4>Details:</h4>
            <h3>with message: <%=exception.getMessage()%></h3>
            </br>
            </br> <% exception.printStackTrace(response.getWriter()); %>
        </p>
    </body>
</html>
```

Лістинг В.12. – Зміст jsp-сторінки «FeedBack»

```
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="styles.css" >

</HEAD>
<BODY>

    <div id="feedback">
        <!-- Five color spans, floated to the left of each other -->

        <span class="color color-1"></span>
        <span class="color color-2"></span>
        <span class="color color-3"></span>
        <span class="color color-4"></span>
        <span class="color color-5"></span>
        <div class="section">

            <!-- The arrow span is floated to the right -->
            <h6><span class="arrow up"></span>Feedback</h6>

            <p class="message">Please include your contact information if you'd
like to receive a reply.</p>

            <textarea name = "textMessage"></textarea>
            <a class="submit" href="index.jsp"+>Submit</a>
           <!-- MainServlet?action=sendMail&IdDetail=" + book.getId() -->
            <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
            <script src="script.js"></script>
        </div>

</div>

</BODY>
</HTML>
```

Лістинг В.13. – Зміст jsp-сторінки «Footer»

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
    <div class="footer">
        <p>
          Copyright © 2018. All rights reserved.
        </p>
    </div>
```

Лістинг В.14. – Зміст jsp-сторінки «Head»

```
<%@ page contentType="text/html;charset=UTF-8" language="java"%>
<div class="head">
    <p class="logoTest" align="center">BOOK SHOP
     </p>
</div>
```

Лістинг В.15. – Зміст jsp-сторінки «Index»

```
<%@ page errorPage="errorPage.jsp"%>
<html>
<head>
    <title>Main</title>
    <style type="text/css">
    </style>
</head>
<body>
    <jsp:include page="Head.jsp" />
    <jsp:include page="Menu.jsp" />
    <jsp:include page="Sidebar.jsp" />
    <jsp:include page="ListBooks.jsp" />
    <jsp:include page="Footer.jsp" />

</body>
</html>
```

Лістинг В.16. – Зміст jsp-сторінки «ListOfBook»

```
<%@ page import="model.Book" %>
<%@ page import="java.util.List" %>
<%@ page import="model.Customer" %>
<%@ page import="Servlet.Commands" %>
<%@ page import="controller.processors.*" %>
<%@ page errorPage="errorPage.jsp"%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%  Boolean isAdmin = false;
    String buttonName;
    try {
        Customer cus = (Customer) session.getAttribute("customer");
        buttonName = "Buy";
        if(cus.getRole() == 10){
            isAdmin = true;
            buttonName = "Delete";
        }
    } catch (NullPointerException e1){
        isAdmin = false;
        buttonName = "Buy";
    }
%>
<head>
    <meta charset="UTF-8">
    <title>Modal</title>
    <link rel="stylesheet" href="css/style.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
</head>
<body>
<div class="content">
    <%  List<Book> listOfAllBooks;
        try {
            listOfAllBooks = (List<Book>)
request.getSession().getAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS);
        } catch (Exception e) {
            listOfAllBooks = null;
        }
        String mess = (String)
request.getSession().getAttribute(Welcome.ATTRIBUTE_MESSAGE);
        if(mess == null) {
            if (listOfAllBooks != null || listOfAllBooks.size() == 0) {
                for (Book book : listOfAllBooks) {%>
                    <div class="book">
                         <p>
                             <b><img src="images/books/<%=book.getPicture()%>"
width="130" height="170" align="center"/></b>
                            <b>
                                <a href="<%="MainServlet?action=" +
Commands.ACTION_DETAIL +
                                        "&" + DetailBook.ID_DETAIL + "=" +
book.getId() %>"> <%= book.getName() %></a>
                                </a>
                            </b>
                            <p align="right">


                            <a href="#" id="<%="go" + book.getId()%>" >

                            <input class="btn" type="submit"
```

```
value="<%=buttonName%>">
                            </a>
                    </form>
                        <style>
                            #modal_form<%=book.getId()%> #modal_close {
                                width: 21px;
                                height: 21px;
                                position: absolute;
                                top: 10px;
                                right: 10px;
                                cursor: pointer;
                                display: block;
                            }
                            #modal_form<%=book.getId()%> {
                                text-align: center;
                                width: 300px;
                                height: 220px;
                                border-radius: 5px;
                                border: 3px #000 solid;
                                background: #fff;
                                position: fixed;
                                top: 45%;
                                left: 50%;
                                margin-top: -150px;
                                margin-left: -150px;
                                display: none;
                                opacity: 0;
                                z-index: 5;
                                padding: 20px 10px;
                            }

                        </style>

                        <script> $(document).ready(function() { // вся магия
после загрузки страницы
                            $('a#go' + '<%=book.getId()%>').click(
function(event){ // ловим клик по ссылки с id="go"
                                event.preventDefault(); // выключаем стандартную
роль элемента
                                $('#overlay').fadeIn(400, // сначала плавно
показываем темную подложку
                                        function(){ // после выполнения
предъидущей анимации
                                            $('#modal_form'+
'<%=book.getId()%>')
                                                    .css('display', 'block') //
убираем у модального окна display: none;
                                                    .animate({opacity: 1, top:
'50%'}, 200); // плавно прибавляем прозрачность одновременно со съезжанием вниз
                                        });
                            });
                            /* Закрытие модального окна, тут делаем то же самое
но в обратном порядке */
                            $('#modal_close, #overlay').click( function(){ //
ловим клик по крестику или подложке
                                $('#modal_form'+ '<%=book.getId()%>')
                                        .animate({opacity: 0, top: '45%'}, 200,
// плавно меняем прозрачность на 0 и одновременно двигаем окно вверх
                                                function(){ // после анимации
                                                    $(this).css('display',
'none'); // делаем ему display: none;
                                                    $('#overlay').fadeOut(400);
```

```
// скрываем подложку
                                                    }
                                                );
                                });
                            });</script>
                            <%if( isAdmin) {

//request.getSession().setAttribute("AuthorID",book.getAuthor().getId());

request.getSession().setAttribute(UpdateBook.BOOK_RUBRIC,book.getParent());
                            %>

                            <%@include file="DeleteBook.jsp"%>
                            <%}else {
                            boolean isMain =true;%>

                            <%@include file="BuyModalForm.jsp"%>

                            <%}%>
                                </p>
                                </p>
                        </div>
                        <br>
                    <%}%>
                    <center>
                        <% if( !(Commands.START_OR_PLUS_BOOKS_TO_LIST >
listOfAllBooks.size() ||

request.getSession().getAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS)
                                        .equals(ViewListBooks.NAME_ACTION_FOR_ALL))
) {%>
                            <a href="MainServlet?action=viewListBooks">view more
books</a>
                        <%}%>
                    </center>
                <%} else { %>
                        List of books is empty.
                    <br>
                    <br>
                <%}%>
            <%} else {
                request.getSession().setAttribute(Welcome.ATTRIBUTE_MESSAGE, null);
%>
                <%= mess %>
            <% }%>
        <br>
        <br>

</div>
</body>
```

Лістинг В.17. – Зміст jsp-сторінки «ListOfOrder»

```
<%@ page import="java.util.List" %>
<%@ page import="controller.processors.AddCustomer" %>
<%@ page import="controller.processors.LoginUser" %>
<%@ page import="java.util.ArrayList" %>
<%@ page import="model.*" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <link rel="stylesheet" href="css/style.css">
    <link rel="stylesheet" href="css/order.css">
    <title>Title</title>
</head>
<body>
<div class="admin_order_content">
    <%List<Order> listOrders = null;
        Customer cus = null;
        try {
            cus =(Customer)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_CUSTOMER);
            listOrders = (List<Order>)
request.getSession().getAttribute(UpdateOrder.LIST_ORDERS);
        } catch (Exception e) {
            listOrders = null;
        }
        if (listOrders != null && listOrders.size() > 0) {%>
    <table >
        <tr>
            <td>Book name</td>
            <td>Amount</td>
            <td>Price</td>
            <%if(cus.getRole()==10){ %>
            <td>Phone</td>
            <td>Email</td>
            <%}else {%>
            <td>Author</td>
            <%}%></tr>
            <%for (Order order : listOrders) {%>
            <%for(Order.ContentOrder con:order.getContents()){
            if(con.getBooks()!=null){%><tr>
            <td><%=con.getBooks().getName()%></td>
            <td><%=con.getAmount()%></td>
            <td><%=con.getBooks().getPrice()%></td>
            <%if(cus.getRole()==10){ %>
            <td><%=order.getCustomer().getPhone()%></td>
            <td><%=order.getCustomer().getMail()%></td>
            <%}else {%>
            <td><%=con.getBooks().getAuthor()%></td>
            <%}%>
            <th><p class="leftstr" >
                <a href="#" id="<%="go" + order.getIdOrder()%>" >
                    <input class="btn"  type="submit" value="Delete">
                </a>
                <style>
                    #modal_form<%=order.getIdOrder()%> #modal_close {
                        width: 21px;
                        height: 21px;
                        position: absolute;
                        top: 10px;
                        right: 10px;
                        cursor: pointer;
```

```
                    display: block;
                }
                #modal_form<%=order.getIdOrder()%> {
                    text-align: center;
                    width: 300px;
                    height: 80px;
                    border-radius: 5px;
                    border: 3px #000 solid;
                    background: #fff;
                    position: fixed;
                    top: 45%;
                    left: 50%;
                    margin-top: -150px;
                    margin-left: -150px;
                    display: none;
                    opacity: 0;
                    z-index: 5;
                    padding: 20px 10px;
                } </style>
            <script> $(document).ready(function() { // вся магия после
загрузки страницы
                $('a#go' + '<%=order.getIdOrder()%>').click(
function(event){ // ловим клик по ссылки с id="go"
                    event.preventDefault(); // выключаем стандартную роль
элемента
                    $('#overlay').fadeIn(400, // сначала плавно показываем
темную подложку
                            function(){ // после выполнения предъидущей
анимации
                                $('#modal_form'+ '<%=order.getIdOrder()%>')
                                        .css('display', 'block') // убираем
у модального окна display: none;
                                        .animate({opacity: 1, top: '50%'},
200); // плавно прибавляем прозрачность одновременно со съезжанием вниз
                            });
                });
                /* Закрытие модального окна, тут делаем то же самое но в
обратном порядке */
                $('#modal_close, #overlay').click( function(){ // ловим клик
по крестику или подложке
                    $('#modal_form'+ '<%=order.getIdOrder()%>')
                            .animate({opacity: 0, top: '45%'}, 200,  //
плавно меняем прозрачность на 0 и одновременно двигаем окно вверх
                                    function(){ // после анимации
                                        $(this).css('display', 'none'); //
делаем ему display: none;
                                        $('#overlay').fadeOut(400); //
скрываем подложку
                                    }
                            );
                });
            });</script>
            <%@include file="DeleteOrder.jsp"%>
</p>
</th>
<th>  <p class="leftstr">
        <a href="#" id="<%="g" + order.getIdOrder()%>" >
            <input class="btn"  type="submit" value="Update"> </a>
        <style>
            #modal_form_<%=order.getIdOrder()%> #modal_close {
                width: 21px;
                height: 21px;
```

```
                position: absolute;
                top: 10px;
                right: 10px;
                cursor: pointer;
                display: block;
            }
            #modal_form_<%=order.getIdOrder()%> {
                text-align: center;
                width: 300px;
                height: 120px;
                border-radius: 5px;
                border: 3px #000 solid;
                background: #fff;
                position: fixed;
                top: 45%;
                left: 50%;
                margin-top: -150px;
                margin-left: -150px;
                display: none;
                opacity: 0;
                z-index: 5;
                padding: 20px 10px;
            }
        </style>
        <script> $(document).ready(function() { // вся магия после загрузки
страницы
            $('a#g' + '<%=order.getIdOrder()%>').click( function(event){ //
ловим клик по ссылки с id="go"
                event.preventDefault(); // выключаем стандартную роль элемента
                $('#overlay').fadeIn(400, // сначала плавно показываем темную
подложку
                        function(){ // после выполнения предъидущей анимации
                            $('#modal_form_'+ '<%=order.getIdOrder()%>')
                                    .css('display', 'block') // убираем у
модального окна display: none;
                                    .animate({opacity: 1, top: '50%'}, 200); //
плавно прибавляем прозрачность одновременно со съезжанием вниз
                        });
            });
            /* Закрытие модального окна, тут делаем то же самое но в обратном
порядке */
            $('#modal_close, #overlay').click( function(){ // ловим клик по
крестику или подложке
                $('#modal_form_'+ '<%=order.getIdOrder()%>')
                        .animate({opacity: 0, top: '45%'}, 200,  // плавно
меняем прозрачность на 0 и одновременно двигаем окно вверх
                                function(){ // после анимации
                                    $(this).css('display', 'none'); // делаем
ему display: none;
                                    $('#overlay').fadeOut(400); // скрываем
подложку
                                }
                        );
            });
        });</script>
        <%@include file="UpdateOrder.jsp"%> </p> </th> </tr> <%}}}%>
</table>
</div>
<%} else { %>
List of orders is empty.<br><br><%}%><br><br>
</div> </body>
</html>
```

Лістинг В.18. – Зміст jsp-сторінки «Login»

```
<%@ page import="controller.processors.LoginUser" %>
<%@ page errorPage="errorPage.jsp"%>


<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<HTML>
<HEAD>
    <TITLE>login</TITLE>
    <style type="text/css">
        <!--
        @import url("css/style.css");
        -->
    </style>
</HEAD>
<BODY>
<jsp:include page="Head.jsp" />
<jsp:include page="Menu.jsp" />

<table border="2" cellspacing="1" cellpadding="15" align="center" class="login">
    <tr>
        <td>
            <FORM NAME="login_form" ACTION="MainServlet?action=loginUser"
method="POST">
                <%-- <INPUT TYPE="hidden" NAME="action" value="loginUser"> --%>
                <hr size="3">
                Name:
                <BR>
                <INPUT TYPE="text" NAME="<%=LoginUser.NAME_LOGIN_INPUT%>"
placeholder="login" VALUE="" SIZE="17" MAXLENGTH="60">
                <BR>
                <hr size="3">
                <BR>
                Password:
                <BR>
                <INPUT TYPE="password" name="<%=LoginUser.NAME_PASSWORD_INPUT%>"
placeholder="password" SIZE="17" NAME="password">
                <BR>
                <hr size="3">
                    <INPUT TYPE="submit" VALUE=" Enter ">
                    <INPUT TYPE="reset" name="" value="  Clean  ">
            </FORM>
        </td>
    </tr>

</table>

<jsp:include page="Footer.jsp" />

</BODY>
</HTML>
```

Лістинг В.19. – Зміст jsp-сторінки «Menu»

```
<%@ page import="controller.processors.LoginUser" %>
<%@ page import="model.Customer" %>
<%@ page import="model.Item" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<head>
    <meta charset="UTF-8">
    <title>Modal</title>
    <link rel="stylesheet" href="css/style.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
    <script src="js/modal.js"></script>
    <script src="js/main.js"></script>
    <script src="js/delete_item.js"></script>
</head>
<body>
<%String login = (String)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_LOGIN);%>
<div class="menu">
    <p class="leftstr">
        <a href="index.jsp">Shop</a> |
        <a href="AboutUs.jsp">Contacts</a>
        <a href="Delivery.jsp">Delivery</a>
        <a href="Feedback.jsp">Feedback</a>
        <%if(login!=null&& login.equals("Admin")){
        Book book = new Book();
        boolean isEdit=false;%>
        <a  href="#" id="edit">AddBook</a>
        <%@include file="EditModalForm.jsp"%>
        <%}%>
    </p>
    <p class="rightstr">
        <a href="index.jsp"><%
                if (login!=null) {%>

                <a href="showProfile.jsp"><%=login%></a>
                <a href="MainServlet?action=unLogin">Exit</a>
            <%} else {%>
                <a href="Login.jsp">Entry</a>
                <a href="showProfile.jsp">Registration</a>
            <%}%>
        </a>
    </p>
</div>
</body>
```

Лістинг В.20. – Зміст jsp-сторінки «ShowProfile»

```
<%@ page import="controller.processors.AddCustomer" %>
<%@ page import="java.util.List" %>
<%@ page import="java.util.ArrayList" %>
<%@ page import="model.Customer" %>
<%@ page errorPage="errorPage.jsp"%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <TITLE>login</TITLE>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/reg.css" media="screen" type="text/css" />
    <link rel="stylesheet" href="css/style.css" media="screen" type="text/css"/>
</head>
<body>
<jsp:include page="Head.jsp" />
<jsp:include page="Menu.jsp" />
<%List<String> list = null; int i = 0;%>
<% Customer cus  = (Customer) request.getSession().getAttribute("customer");
    i =0;
    list = new ArrayList<> ();
if(cus==null){
    list.add(" ");
    list.add("Registration");
    list.add("MainServlet?action=addCustomer");
    list.add("Login");
    list.add("Password");
    list.add("Email");
    list.add("Phone");
    list.add("Signup");
}else{
    list.add("class=\"leftstr\"");
    list.add("Profile");
    list.add("MainServlet?action=updateCustomer");
    list.add(cus.getLogin());
    list.add(cus.getPassword());
    list.add(cus.getMail());
    list.add(cus.getPhone());
    list.add("Edit");}%>
<div id="login" <%=list.get(i++)%>>
    <div >
        <div class="form-signup">
            <h1><%=list.get(i++)%></h1>
            <fieldset>
                <form method="post" action="<%=list.get(i++)%>">
                    <input type="text" name = "<%=AddCustomer.CUS_LOGIN%>" <%=
cus==null?"placeholder=":"value ="%> <%= list.get(i++)%> required />
                    <input type="password" name =
"<%=AddCustomer.CUS_PASSWORD%>" <%= cus==null?"placeholder=":"value ="%>
<%=list.get(i++)%> required />
                    <input type="email" name = "<%=AddCustomer.CUS_E_MAIL%>" <%=
cus==null?"placeholder=":"value ="%> <%=list.get(i++)%> required />
                    <input type="text" name = "<%=AddCustomer.CUS_PHONE%>" <%=
cus==null?"placeholder=":"value ="%> <%=list.get(i++)%> required />
                    <input type="submit" value=<%=list.get(i)%> />
                </form></fieldset>
        </div></div></div>
<%if(cus!=null){%>
<jsp:include page="ListOrders.jsp"/><%}%>
</body>
</html>
```

Лістинг В.21. – Зміст jsp-сторінки «SideBar»

```
<%@ page import="model.Item" %>
<%@ page import="Servlet.Commands" %>
<%@ page import="java.util.*" %>
<%@ page import="controller.processors.ViewListBooks" %>
<%@ page import="controller.processors.Welcome" %>
<jsp:include page="Feedback.jsp" />


<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<div class="sidebar">
    <p>
    <form class="find_input" method="POST" action="<%= "MainServlet?action=" +
Commands.ACTION_VIEW_LIST_BOOKS +
                            "&" + ViewListBooks.ACTION_VIEW_LIST_ATTRIBUTE + "="
+ ViewListBooks.NAME_ACTION_FOR_SEARCH%>">
        <input class = "find_input" type = "text" name =
"<%=ViewListBooks.PARAMETER_ACTION_FOR_LIST_ATTRIBUTE%>" placeholder="search
book" required />
    </form>

    <!--a href="<%= "MainServlet?action=" + Commands.ACTION_VIEW_LIST_BOOKS +
            "&" + ViewListBooks.ACTION_VIEW_LIST_ATTRIBUTE + "=" +
ViewListBooks.NAME_ACTION_FOR_ALL%>">
        <p class="textSideBar2Small"> All Category: </p></a-->
    <p class="textSideBar2" align="center"> Categories: </p>
        <div >
        <%String login = (String)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_LOGIN);%>
        <%if(login!=null && login.equals("Admin")){%>
        <a href="#" id="go">  &nbspADD   </a>
        <%@include file="AddItem.jsp"%>
        <a href="#" id="del">REMOVE</a>
        <%@include file="DeleteItem.jsp"%>
        <%}%>
        </div>
<%
        ArrayList<Item> section = (ArrayList<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_SECTION);
        HashMap<Item, ArrayList<Item>> listRubric = (HashMap)
request.getSession().getAttribute(Welcome.ATTRIBUTE_CATEGORY);

        for (Item it : section) { %>
              <p class="textSideBar2Small"> <%= it.getName()  %> </p>
                <ul>
                    <% for (int i = 0; i <= listRubric.get(it).size() - 1 ; i
++)  { %>
                    <li><a href="<%= "MainServlet?action=" +
Commands.ACTION_VIEW_LIST_BOOKS +
                            "&" + ViewListBooks.ACTION_VIEW_LIST_ATTRIBUTE + "="
+ ViewListBooks.NAME_ACTION_FOR_RUBRIC +
                            "&" +
ViewListBooks.PARAMETER_ACTION_FOR_LIST_ATTRIBUTE + "="
+listRubric.get(it).get(i).getId()%>">
                        <%= listRubric.get(it).get(i).getName()%></a>
                    </li>
                    <%}%>
                </ul>
        <%}%>
    </p>
```

```
    <p class="textWhite"> white text </p>
    <a href="<%= "MainServlet?action=" + Commands.ACTION_VIEW_LIST_BOOKS +
            "&" + ViewListBooks.ACTION_VIEW_LIST_ATTRIBUTE + "=" +
ViewListBooks.NAME_ACTION_FOR_ALL%>">
        <p class="textSideBar2Small"> View all categories </p></a>

</div>

<div class="sidebar2">
    <p class="textSideBar2" align="center" >Welcome!</p>

    <p class="textWhite"> white text </p>
    <p class="textSideBar2Small" align="center">BookSop is online bookstore
number one! We have only quality books from reliable publishers.</p>
    <p class="textWhite"> white text </p>
    <p class="textSideBar2Small" align="center"> Call</p>
    <p class="textSideBar2Small" align="center"> +38095 467 98 17</p>
    <p class="textWhite"> white text </p>
    <p class="textSideBar2Small" align="center"> Work schedule: 9:00 - 19:00</p>
</div>
```

Лістинг В.22. – Зміст jsp-сторінки «UpdateOrder»

```
<%@ page import="controller.processors.DeleteOrder" %>
<%@ page import="controller.processors.UpdateBook" %>
<%@ page import="controller.processors.UpdateOrder" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<div id="<%="modal_form_" + order.getIdOrder()%>" >
    <div >
        <h3>Are you sure?</h3>
    </div>
    <div >
        <form name = "buy_form" method="post"
action="MainServlet?action=updateOrder">
            <p class="prob">
            <div> <label for="<%=UpdateBook.BOOK_AMOUNT +
"ID"%>">Amount</label></div>
            <input type="hidden" value="<%=order.getIdOrder()%>"
name="<%=UpdateOrder.UPDATE_ORDER_ID%>" required>
            <input type="hidden" value="<%=con.getBooks().getId()%>"
name="<%=UpdateOrder.UPDATE_BOOK_ID%>" required>
            <input type="number" id ="<%=UpdateBook.BOOK_AMOUNT+"ID"%>"
name="<%=UpdateBook.BOOK_AMOUNT%>"
                    value="<%=con.getAmount()%>" />
            </p>
            <div class="prob">  <input class="btn" type="submit"  value="Edit
order" /></div>
        </form>
    </div>
                    <span id="modal_close">
                        <a class="close"  title="Close" >X</a>
                    </span>

</div>
<div id="overlay"></div>
```

Додаток Г

Лістинг Г.1. – Код класу «ActionBook»

```
package controller.processors;

import exception.DataBaseException;
import model.Author;
import model.Book;
import model.Item;
import model.OracleDataAccess;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.List;

/**
 * Abstract class for work with book.
 *
 * @author Vovk Veleri
 * @version %I%, %G%
 */

abstract class ActionBook implements GeneralProcess{
    public final static String BOOK_NAME = "bookName";
    public final static String BOOK_PAGES = "bookPages";
    public final static String BOOK_AUTHOR_ID ="AuthorID";
    public final static String BOOK_AUTHOR_NAME ="bookAuthorName";
    public final static String BOOK_AUTHOR_SURNAME = "bookAuthorSurname";
    public final static String BOOK_DESCRIPTION = "bookDescription";
    public final static String BOOK_AMOUNT = "bookAmount";
    public final static String BOOK_PRICE = "bookPrice";
    public final static String BOOK_RUBRIC = "bookRubric";
    public final static String BOOK_RUBRIC_NAME = "nameRubric";
    public final static String BOOK_PUCTURE= "bookPicture";
    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        String authorSurname =request.getParameter(BOOK_AUTHOR_SURNAME);
        String authorName = request.getParameter(BOOK AUTHOR NAME);
        List<Author> listAuthor=  OracleDataAccess.getInstance().getAllAuthor();

        Author author = null;

        for(Author auth:listAuthor){

            if(auth.getName() != null &&
auth.getSurname().equals(authorSurname)&&auth.getName().equals(authorName)){

  author = auth;
            }
        }

        if(author == null) {
            author = new Author();
            author.setName(authorName);
            author.setSurname(authorSurname);
            OracleDataAccess.getInstance().createAuthor(author);
            listAuthor = OracleDataAccess.getInstance().getAllAuthor();
            for(Author auth:listAuthor){
                if(auth.getName() != null&&
```

```
auth.getSurname().equals(authorSurname)&&auth.getName().equals(authorName)){
                    author = auth;
                }
            }
        }

        String bookName = request.getParameter(BOOK_NAME);
        String description = request.getParameter(BOOK_DESCRIPTION);
        String rubricName =request.getParameter(BOOK_RUBRIC_NAME);
        int idBook =0;

        if(request.getParameter("IdDetail")!=null) {

            idBook = Integer.parseInt(request.getParameter("IdDetail"));
        }

        int amount = Integer.parseInt(request.getParameter(BOOK_AMOUNT));
        int price = Integer.parseInt(request.getParameter(BOOK_PRICE));
        int pages = Integer.parseInt(request.getParameter(BOOK_PAGES));
        String picture =request.getParameter(BOOK_PUCTURE);
        List<Item> listItem = OracleDataAccess.getInstance().getAllRubric();

        Item rubric = null;

        for(Item item:listItem){

            if(item.getName().equals(rubricName)){
                rubric = item;
            }
        }
        Book book = new

Book(idBook,bookName,description,rubric,author,pages,price,amount, picture);

        forwardBook(book,request,response);
    }

    abstract void forwardBook(Book book,HttpServletRequest request,
HttpServletResponse response) throws DataBaseException;
}
```

Лістинг Г.2. – Код класу «AddBook»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Book;
import model.OracleDataAccess;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;

/**
 * Class for add book.
 *
 * @author Vovk Veleri
 * @version %I%, %G%
 */

public class AddBook extends ActionBook {

    @Override
    void forwardBook(Book book, HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {

        OracleDataAccess.getInstance().createBook(book);

        ArrayList books = (ArrayList)

OracleDataAccess.getInstance().getAmountOfBooks(Commands.AMOUNT_OF_BOOKS_ON_LIST
);

request.getSession().setAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS,
null);

request.getSession().setAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS,
books);

        Commands.forward("/index.jsp", request, response);
    }

}
```

Лістинг Г.3. – Код класу «AddCustomer»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Customer;
import model.OracleDataAccess;
import model.Order;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Class for add customer.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class AddCustomer implements GeneralProcess {

    public final static String CUS_ID      = "ID_CUSTOMER";
    public final static String CUS_LOGIN   = "LOGIN";
    public final static String CUS_PASSWORD= "PASSWORD";
    public final static String CUS_E_MAIL  = "E_MAIL";
    public final static String CUS_PHONE   = "PHONE_NUMBER";
    public final static String CUS_ROLE    = "ROLE";
    public final static String CUS_IS_REG  = "isRegistration";

    public void process(HttpServletRequest request,HttpServletResponse response)
throws DataBaseException  {
       // int id = Integer.valueOf(request.getParameter(CUS_ID));
        String login = request.getParameter(CUS_LOGIN);
        String password = request.getParameter(CUS_PASSWORD);
        String phone = request.getParameter(CUS_PHONE);
        Customer cusWithId  =
OracleDataAccess.getInstance().getCustomer(login,password);
        if(phone.length()<13 && cusWithId==null) {
            String eMail = request.getParameter(CUS_E_MAIL);
            int role = 1;
            Customer cus = new Customer(login, password, eMail, phone, role);
            OracleDataAccess.getInstance().createCustomer(cus);
            request.getSession().setAttribute(LoginUser.ATTRIBUTE_LOGIN, login);
            cusWithId = OracleDataAccess.getInstance().getCustomer(login,
password);
            request.getSession().setAttribute(LoginUser.ATTRIBUTE_CUSTOMER,
cusWithId);
            List<Order> listOrders = new ArrayList<Order>();
            request.getSession().setAttribute("listOfAllOrders", listOrders);
            Commands.forward("/index.jsp", request, response);
        }else{
            Commands.forward("/showProfile.jsp", request, response);
        }
    }
}
```

Лістинг Г.4. – Код класу «AddItem»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Item;
import model.OracleDataAccess;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.List;

/**
 * Class for add item.
 *
 * @author Vovk Veleri
 * @version %I%, %G%
 */
public class AddItem implements GeneralProcess {
    public final static String ITEM_NAME = "itemName";
    public final static String ITEM_DESCRIPTION = "itemDescription";
    public final static String ITEM_SECTION_NAME = "itemSection";
    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        String name = request.getParameter(ITEM_NAME);
        String desc = request.getParameter(ITEM_DESCRIPTION);
        String sectionName = request.getParameter(ITEM_SECTION_NAME);
        Item section = null;
        if(sectionName.equals("Create section")){
            section = new Item(0,name,desc,null, Item.ItemType.Section);
            OracleDataAccess.getInstance().createSection(section);
        }else {
            List<Item> listSection = (List<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_SECTION);
            for (Item item : listSection) {
                if (item.getName().equals(sectionName)) {
                    section = item;
                    break;
                }
            }
            Item rubric = new Item(0, name, desc, section,
Item.ItemType.Rubric);

            OracleDataAccess.getInstance().createRubric(rubric);
        }

        Commands.forward("/MainServlet?action=" + Commands.ACTION_WELCOME,
request, response);
    }
}
```

Лістинг Г.5. – Код класу «AddOrder»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Book;
import model.Customer;
import model.OracleDataAccess;
import model.Order;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Class for add order.
 *
 * @author Vovk Veleri
 * @version %I%, %G%
 */
public class AddOrder implements GeneralProcess {

    public final static String user ="Common user";
    public final static String Book_Amount ="amount";


    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        Customer cus = (Customer)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_CUSTOMER);
        int pass_user = 1111;
        if(cus == null){
            do{
                cus =
OracleDataAccess.getInstance().getCustomer(user,String.valueOf(pass_user++));
            }while (cus!=null);


            String phone =  request.getParameter(AddCustomer.CUS_PHONE);
            String mail  = request.getParameter(AddCustomer.CUS_E_MAIL);

            cus = new Customer(user,String.valueOf(pass_user),mail,phone, 1);
            OracleDataAccess.getInstance().createCustomer(cus);
            cus =
OracleDataAccess.getInstance().getCustomer(user,String.valueOf(pass_user));
        }

        int IdDetail =
Integer.valueOf(request.getParameter(DetailBook.ID_DETAIL));
        int amount = Integer.valueOf(request.getParameter(Book_Amount));

        Book book = OracleDataAccess.getInstance().getBookById(IdDetail);
        ArrayList<Order.ContentOrder> contents = new
ArrayList<Order.ContentOrder>();
        Order order = new Order(cus,new java.sql.Date(new Date().getTime()));
        Order.ContentOrder cont = order.new ContentOrder();

        cont.setBook(book,amount);
        contents.add(cont);
```

```
        order.setContents(contents);
        OracleDataAccess.getInstance().createOrder(order);

        List<Order> listOrders;
        if(cus.getRole()==10) {
            listOrders = OracleDataAccess.getInstance().getAllOrder();
        }else {
            listOrders =
OracleDataAccess.getInstance().getOrderByIdCustomer(cus.getId());
        }
        request.getSession().setAttribute(UpdateOrder.LIST_ORDERS, null);
        request.getSession().setAttribute(UpdateOrder.LIST_ORDERS, listOrders);
        Commands.forward("/index.jsp", request, response);
    }

}
```

Лістинг Г.6. – Код класу «DeleteBook»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Customer;
import model.OracleDataAccess;
import model.Order;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
import java.util.List;
/**
 * Class for delete book.
 * @author Vovk Veleri
 * @version %I%, %G%
 */
public class DeleteBook implements GeneralProcess {
    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        int idBook =
Integer.parseInt(request.getParameter(DetailBook.ID_DETAIL));

        OracleDataAccess.getInstance().removeBook(idBook);
        ArrayList books = (ArrayList)
OracleDataAccess.getInstance().getAmountOfBooks(Commands.AMOUNT_OF_BOOKS_ON_LIST
);
request.getSession().setAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS,
books);
        List<Order> listOrders;
        Customer customer  = (Customer)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_CUSTOMER);
        if(customer.getRole()==10) {
            listOrders = OracleDataAccess.getInstance().getAllOrder();
        }else {
            listOrders =
OracleDataAccess.getInstance().getOrderByIdCustomer(customer.getId());
        }
        request.getSession().setAttribute(UpdateOrder.LIST_ORDERS, null);
        request.getSession().setAttribute(UpdateOrder.LIST_ORDERS, listOrders);
        Commands.forward("/index.jsp", request, response);
    }

}
```

Лістинг Г.7. – Код класу «DeleteItem»

```
package controller.processors;
import Servlet.Commands;
import exception.DataBaseException;
import model.Book;
import model.Item;
import model.OracleDataAccess;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
import java.util.List;
/**
 * Class for delete item.
 * @author Vovk Veleri
 * @version %I%, %G%
 */
public class DeleteItem implements GeneralProcess {
    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        String nameItem = request.getParameter(AddItem.ITEM_NAME);
        List<Item> listSection = (List<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_SECTION);
        List<Item> listRubrics = (List<Item>)
request.getSession().getAttribute(Welcome.ATTRIBUTE_All_CATEGORY);
        Item item = null;
        for (Item index : listSection) {
            if (index.getName().equals(nameItem)) {
                item = index;
                break;
            }
        }
        if(item!=null) {
            ArrayList<Item> rubric = (ArrayList<Item>)
OracleDataAccess.getInstance().getRubricBySection(item.getId());
            if(rubric.size()==0) {
                OracleDataAccess.getInstance().removeSection(item.getId());
            }else {

request.getSession().setAttribute(Welcome.ATTRIBUTE_MESSAGE,"Section is don't
empty. Please delete item in section.");
            }
        }else {
            for (Item index : listRubrics) {
                if (index.getName().equals(nameItem)) {
                    item = index;
                    break;
                }
            }
            ArrayList<Book> books = (ArrayList<Book>)
OracleDataAccess.getInstance().getAllBooksByRubric(item.getId());
            if(books.size()==0) {
                OracleDataAccess.getInstance().removeRubric(item.getId());
            }else {
                request.getSession().setAttribute(Welcome.ATTRIBUTE_MESSAGE,
"Rubric is don't empty. Please delete item in rubric.");
            }
        }
        Commands.forward("/MainServlet?action=" + Commands.ACTION_WELCOME,
request, response);
    }
}
```

Лістинг Г.8. – Код класу «DeleteOrder»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Customer;
import model.OracleDataAccess;
import model.Order;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.List;

/**
 * Class for delete order.
 *
 * @author Vovk Veleri
 * @version %I%, %G%
 */
public class DeleteOrder implements GeneralProcess {

    public final static String DELETE_ORDER = "deleteOrder";

    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {

        int orderId
=Integer.valueOf(request.getParameter(UpdateOrder.UPDATE_ORDER_ID));
        Order order = OracleDataAccess.getInstance().getOrderById(orderId);

       Customer customer = (Customer)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_CUSTOMER);
        OracleDataAccess.getInstance().removeOrder(order);

        List<Order> orders;
        if(customer.getRole()==10) {

            orders = OracleDataAccess.getInstance().getAllOrder();
        }else {
            orders =
OracleDataAccess.getInstance().getOrderByIdCustomer(customer.getId());

        }
        request.getSession().setAttribute(UpdateOrder.LIST_ORDERS, null);

        request.getSession().setAttribute(UpdateOrder.LIST_ORDERS, orders);

        Commands.forward("/showProfile.jsp", request, response);
    }

}
```

Лістинг Г.9. – Код класу «DetailBook»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Book;
import model.OracleDataAccess;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Class that handling command DetailBook.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class DetailBook implements GeneralProcess {

    public final static String ID_DETAIL                = "IdDetail";

    public final static String ATTRIBUTE_BOOK_FOR_DETAIL = "DetailBook";

    public void process(HttpServletRequest request, HttpServletResponse
response) throws  DataBaseException{

        int IdDetail = Integer.valueOf(request.getParameter(ID_DETAIL));

        Book book = OracleDataAccess.getInstance().getBookById(IdDetail);

        request.getSession().setAttribute(ATTRIBUTE_BOOK_FOR_DETAIL, book);

        Commands.forward("/bookDetail.jsp",request,response);
    }

}
```

Лістинг Г.10. – Код інтерфейсу «GeneralProcess»

```
package controller.processors;

import exception.DataBaseException;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Interface that describes processors.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */
public interface GeneralProcess {

    void process(HttpServletRequest request, HttpServletResponse response)
throws DataBaseException;

}
```

Лістинг Г.11. – Код класу «LoginUser»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Customer;
import model.OracleDataAccess;
import model.Order;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.List;

/**
 * Class that specified precess for login.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
 public class LoginUser implements GeneralProcess {

     public final static String ATTRIBUTE_CUSTOMER       = "customer";
     public final static String ATTRIBUTE_LOGIN          = "login";
     public final static String NAME_LOGIN_INPUT         = "nameLogin";
     public final static String NAME_PASSWORD_INPUT      = "namePassword";

     public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {

         String login    = request.getParameter(NAME_LOGIN_INPUT);
         String password = request.getParameter(NAME_PASSWORD_INPUT);
         Customer customer = OracleDataAccess.getInstance().getCustomer(login,
password);

         if (customer != null) {
             request.getSession().setAttribute(ATTRIBUTE_CUSTOMER, customer);
             request.getSession().setAttribute(ATTRIBUTE_LOGIN, login);

             List<Order> listOrders;
             if(customer.getRole()==10) {
                 listOrders = OracleDataAccess.getInstance().getAllOrder();
             }else {
                 listOrders =
OracleDataAccess.getInstance().getOrderByIdCustomer(customer.getId());
             }
             request.getSession().setAttribute(UpdateOrder.LIST_ORDERS, null);
             request.getSession().setAttribute(UpdateOrder.LIST_ORDERS,
listOrders);
             Commands.forward("/index.jsp", request, response);
         } else {
             Commands.forward("/Login.jsp", request, response);
         }
     }

 }
```

Лістинг Г.12. – Код класу «UnLogin»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Class for log out of user.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class UnLogin implements GeneralProcess {
    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        /*HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate();
        }*/
        request.getSession().setAttribute(LoginUser.ATTRIBUTE_CUSTOMER,null);
        request.getSession().setAttribute(LoginUser.ATTRIBUTE_LOGIN, null);
        Commands.forward("/index.jsp", request, response);
    }
}
```

Лістинг Г.13. – Код класу «UpdateBook»

```
package controller.processors;
import Servlet.Commands;
import exception.DataBaseException;
import model.Book;
import model.OracleDataAccess;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
/**
 * Class for handling update of book.
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class UpdateBook extends ActionBook {
    @Override
    void forwardBook(Book book, HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        OracleDataAccess.getInstance().updateBook(book);

        ArrayList books = (ArrayList)
OracleDataAccess.getInstance().getAmountOfBooks(Commands.AMOUNT_OF_BOOKS_ON_LIST
);
request.getSession().setAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS,
null);
request.getSession().setAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS,
books);
        Commands.forward("/MainServlet?action=" + Commands.ACTION_DETAIL +
                "&"+ DetailBook.ID_DETAIL + "=" + book.getId(), request,
response);
    }
}
```

Лістинг Г.14. – Код класу «UpdateCustomer»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Customer;
import model.OracleDataAccess;
import model.Order;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Class for update customer.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class UpdateCustomer implements GeneralProcess {

    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {

        Customer cus = (Customer)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_CUSTOMER);

        cus.setLogin(request.getParameter(AddCustomer.CUS_LOGIN));

        cus.setPassword(request.getParameter(AddCustomer.CUS_PASSWORD));

        cus.seteMail(request.getParameter(AddCustomer.CUS_E_MAIL));

        cus.setPhone(request.getParameter(AddCustomer.CUS_PHONE));
        Customer cusCheck  =
OracleDataAccess.getInstance().getCustomer(cus.getLogin(),cus.getPassword());

        if(cusCheck==null) {

            OracleDataAccess.getInstance().updateCustomer(cus);

            request.getSession().setAttribute(AddCustomer.CUS_LOGIN,
cus.getLogin());

            request.getSession().setAttribute(LoginUser.ATTRIBUTE_CUSTOMER,
cus);
            request.getSession().setAttribute(LoginUser.ATTRIBUTE_LOGIN,
cus.getLogin());

        }

        Commands.forward("/showProfile.jsp", request, response);
    }

}
```

Лістинг Г.15. – Код класу «UpdateOrder»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Customer;
import model.OracleDataAccess;
import model.Order;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Class for update order.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class UpdateOrder implements GeneralProcess {

    public final static String LIST_ORDERS = "testMagicOrder";

    public final static String UPDATE_ORDER_ID = "updateOrderID";

    public final static String UPDATE_BOOK_ID  = "updateBookID";

    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {
        int amount =
Integer.valueOf(request.getParameter(UpdateBook.BOOK_AMOUNT));
        int order =Integer.valueOf(request.getParameter(UPDATE_ORDER_ID));

        int book = Integer.valueOf(request.getParameter(UPDATE_BOOK_ID));

        Customer customer = (Customer)
request.getSession().getAttribute(LoginUser.ATTRIBUTE_CUSTOMER);

        OracleDataAccess.getInstance().updateBookOfOrder(order,book,amount);
        List<Order> orders ;

        if(customer.getRole()==10) {

            orders = OracleDataAccess.getInstance().getAllOrder();

        } else {
            orders =
OracleDataAccess.getInstance().getOrderByIdCustomer(customer.getId());

        }
        request.getSession().setAttribute(LIST_ORDERS, null);

        request.getSession().setAttribute(LIST_ORDERS, orders);

        Commands.forward("/showProfile.jsp", request, response);

    }
}
```

Лістинг Г.16. – Код класу «ViewListBooks»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.OracleDataAccess;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;

/**
 * Class that handling command ViewListBooks.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class ViewListBooks implements GeneralProcess {

    public static final String ATTRIBUTE_LIST_OF_ALL_BOOKS = "listOfAllBooks";
//list with books
    public final static String ACTION_VIEW_LIST_ATTRIBUTE =
"attribute_list_books";//action for list: all, search, RubId
    public final static String PARAMETER_ACTION_FOR_LIST_ATTRIBUTE =
"parameterForList_action";

    public final static String NAME_ACTION_FOR_ALL    = "actionAll";
    public final static String NAME_ACTION_FOR_RUBRIC = "actionRubric";
    public final static String NAME_ACTION_FOR_SEARCH = "actionSearch";

    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException {

        String requestAction = request.getParameter(ACTION_VIEW_LIST_ATTRIBUTE);

        String nameForSearch;
        ArrayList books = null;

        if (requestAction != null) {
            switch (requestAction) {
                case NAME_ACTION_FOR_ALL:
                    books = getListOfAlLBooks(request);
                    break;
                case NAME_ACTION_FOR_RUBRIC:
                    books = getListOfRubric(request, true);
                    break;
                case NAME_ACTION_FOR_SEARCH:

request.getSession().setAttribute(ACTION_VIEW_LIST_ATTRIBUTE,
NAME_ACTION_FOR_SEARCH);
                    nameForSearch =
request.getParameter(PARAMETER_ACTION_FOR_LIST_ATTRIBUTE);

request.getSession().setAttribute(PARAMETER_ACTION_FOR_LIST_ATTRIBUTE,
nameForSearch);
                    books = (ArrayList)
OracleDataAccess.getInstance().getBooksByName(nameForSearch);
                    break;
                default:
                    books = getListOfAlLBooks(request);
                    break;
```

```
            }
            Commands.AMOUNT_OF_BOOKS_ON_LIST =
Commands.START_OR_PLUS_BOOKS_TO_LIST;
        } else {
            requestAction = (String)
request.getSession().getAttribute(ACTION_VIEW_LIST_ATTRIBUTE);
            switch (requestAction) {
                case NAME_ACTION_FOR_ALL:
                    Commands.AMOUNT_OF_BOOKS_ON_LIST +=
Commands.START_OR_PLUS_BOOKS_TO_LIST;
                    books = getListOfAlLBooks(request);
                    break;
                case NAME_ACTION_FOR_RUBRIC:
                    books = getListOfRubric(request, false);
                    break;
                case NAME_ACTION_FOR_SEARCH:
                    nameForSearch = (String)
request.getSession().getAttribute(PARAMETER_ACTION_FOR_LIST_ATTRIBUTE);
                    books = (ArrayList)
OracleDataAccess.getInstance().getBooksByName(nameForSearch);
                    break;
                default:
                    books = getListOfAlLBooks(request);
                    break;
            }

        }
        request.getSession().setAttribute(ATTRIBUTE_LIST_OF_ALL_BOOKS, books);
        Commands.forward("/index.jsp", request, response);
    }
    private ArrayList getListOfAlLBooks(HttpServletRequest request) throws
DataBaseException {
        request.getSession().setAttribute(ACTION_VIEW_LIST_ATTRIBUTE,
NAME_ACTION_FOR_ALL);
        return  (ArrayList)
OracleDataAccess.getInstance().getAmountOfBooks(Commands.AMOUNT_OF_BOOKS_ON_LIST
);
    }
    private ArrayList getListOfRubric(HttpServletRequest request, boolean
flagRequest) throws DataBaseException {
        Integer idRubric;
        ArrayList books;

        if (flagRequest) {
            try {
                idRubric =
Integer.valueOf(request.getParameter(PARAMETER_ACTION_FOR_LIST_ATTRIBUTE));

request.getSession().setAttribute(PARAMETER_ACTION_FOR_LIST_ATTRIBUTE,
idRubric);
            } catch (Exception e) {
                return getListOfAlLBooks(request);
            }
        } else {
            idRubric = (Integer)
request.getSession().getAttribute(PARAMETER_ACTION_FOR_LIST_ATTRIBUTE);
        }
        books = (ArrayList)
OracleDataAccess.getInstance().getAllBooksByRubric(idRubric);
        return books;
    }
}
```

Лістинг Г.17. – Код класу «Welcome»

```
package controller.processors;

import Servlet.Commands;
import exception.DataBaseException;
import model.Item;
import model.OracleDataAccess;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

/**
 * Class that handling command Welcome.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class Welcome implements GeneralProcess{

    public static final String ATTRIBUTE_SECTION   = "Section";
    public static final String ATTRIBUTE_CATEGORY  = "Category";
    public static final String ATTRIBUTE_All_CATEGORY  = "AllCategory";
    public static final String ATTRIBUTE_MESSAGE   = "Message";

    public void process(HttpServletRequest request, HttpServletResponse
response) throws DataBaseException{

        ArrayList books = (ArrayList)
OracleDataAccess.getInstance().getAmountOfBooks(Commands.AMOUNT_OF_BOOKS_ON_LIST
);

request.getSession().setAttribute(ViewListBooks.ATTRIBUTE_LIST_OF_ALL_BOOKS,
books);

        ArrayList<Item> sectionAll = (ArrayList<Item>)
OracleDataAccess.getInstance().getAllSection();
        HashMap<Item, ArrayList<Item>> listRubric = new HashMap<Item,
ArrayList<Item>>();

        for (int i = 0; i <= sectionAll.size() - 1; i++){
            ArrayList<Item> rubric = (ArrayList<Item>)
OracleDataAccess.getInstance().getRubricBySection(sectionAll.get(i).getId());
            listRubric.put(sectionAll.get(i), rubric);
        }

        request.getSession().setAttribute(ATTRIBUTE_CATEGORY, listRubric);
        request.getSession().setAttribute(ATTRIBUTE_SECTION,  sectionAll);

        List<Item> allRubric = OracleDataAccess.getInstance().getAllRubric();
        request.getSession().setAttribute(ATTRIBUTE_All_CATEGORY, allRubric);


request.getSession().setAttribute(ViewListBooks.ACTION_VIEW_LIST_ATTRIBUTE,
ViewListBooks.NAME_ACTION_FOR_ALL);
       Commands.forward("/index.jsp", request, response);
    }

}
```

Лістинг Г.18. – Код класу «DataBaseException»

```
package exception;

import org.apache.log4j.Logger;

import java.util.Arrays;


/**
 * Class that describes the exception for database.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */


public class DataBaseException  extends Exception {

    private static final Logger LOG = Logger.getLogger(DataBaseException.class);

    private String message;

    public DataBaseException(String message, Exception e){

        super(message,e);


        this.message = message;


        LOG.error(Arrays.toString(e.getStackTrace()).replaceAll(" ", "\t\n"));

    }

    @Override
    public String getMessage() {


        return super.getMessage() + "is: " + message;


    }

}
```

Лістинг Г.19. – Код класу «Author»

```
package model;

/**
 * Class that describes the author's.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */

public class Author {
    private int id;
    private String surname;
    private String name;

    public Author(int id, String surname, String name) {

        this.id = id;
        this.surname = surname;
        this.name = name;
    }

    public Author(String surname, String name) {

        this.surname = surname;
        this.name = name;
    }

    public Author() {
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return getSurname() + " " + getName();
    }
```

```
    @Override
    public boolean equals(Object obj) {

        if ((obj == null) || (getClass() != obj.getClass()))

            return false;

        Author other = (Author) obj;

        if ((id != other.id) && (!surname.equals(other.surname)) &&
(!name.equals(other.name)))

            return false;

        if (this == obj)

            return true;

        return true;
    }

    @Override
    public int hashCode() {

        int result = 1;

        int one = 17;

        result = one * result + (this.getName() == null ? 0 :
this.getName().hashCode());

        return result;
    }

}
```

Лістинг Г.20. – Код класу «Book»

```
package model;


/**
 * Class that describes the book's.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */
public class Book extends Item {

    private Author author;
    private int pages;
    private int price;
    private int amount;
    private String picture;

    public Book() {

    }

    public Book(int id, String name, String des, Item rubric, Author author, int
pages, int price, int amount, String picture) {
        super(id, name, des, rubric, ItemType.Book);
        if (rubric.getType() != ItemType.Rubric) {
            throw new IllegalArgumentException("This is not a rubric!");
        }
        this.author = author;
        this.pages = pages;
        this.price = price;
        this.amount = amount;
        this.picture = picture;


    }
    public void setPicture(String picture){
        this.picture = picture;
    }
    public String getPicture() {
        return this.picture;
    }
    public Author getAuthor() {
        return author;
    }

    public void setAuthor(Author author) {
        this.author = author;
    }

    public int getPages() {
        return pages;
    }

    public void setPages(int pages) {
        this.pages = pages;
    }

    public int getPrice() {
        return price;
    }
```

```
    public void setPrice(int price) {
        this.price = price;
    }

    public int getAmount() {
        return amount;
    }

    public void setAmount(int amount) {

        this.amount = amount;
    }

    @Override
    public boolean equals(Object obj) {

        if ((obj == null) || (getClass() != obj.getClass()))
            return false;
        Book other = (Book) obj;
        if ((author != other.author) && (pages != other.pages) && (price !=
other.price) && (amount != other.amount))
            return false;
        if (this == obj)
            return true;
        return true;

    }

    @Override
    public int hashCode() {

        int result = 1;
        int one = 17;
        result = one * result + (this.getName() == null ? 0 :
this.getName().hashCode());
        return result;

    }

    @Override
    public String toString() {

        return getName() + " Author: " + getAuthor().toString() + " Pages: " +
getPages() + " Price: " + getPrice() + " Rubric: " + getParent().getName() + "
Amount: " + getAmount() + " Description: " + getDescription();

    }
}
```

Лістинг Г.21. – Код класу «Customer»

```
package model;


import org.apache.log4j.Logger;

/**
 * Class that describes the customer's.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */
public class Customer {
    private int id;
    private String login;
    private String password;
    private String eMail;
    private String phone;
    private int role;
    private static final Logger LOG = Logger.getLogger(Customer.class);

    public Customer(int id, String login, String password, String eMail, String
phone, int role) throws IllegalArgumentException {
        this.id = id;
        this.login = login;
        this.password = password;
        this.eMail = eMail;
        this.phone = phone;
        setRole(role);

    }

    public Customer(String login, String password, String eMail, String phone,
int role) throws IllegalArgumentException {
        this.login = login;
        this.password = password;
        this.eMail = eMail;
        this.phone = phone;
        setRole(role);

    }

    public Customer() {
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }
```

```
    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getMail() {
        return eMail;
    }

    public void seteMail(String eMail) {
        this.eMail = eMail;
    }
    public int getRole() {
        return role;
    }
    public void setRole(int role) throws IllegalArgumentException {
        if ((role != 0) && (role != 10) && (role != 1)) {
            LOG.error("Wrong number of role. Operation rejected!");
            throw new IllegalArgumentException("Wrong number of role. Operation
rejected!");
        } else {
            this.role = role;
        }
    }
    @Override
    public boolean equals(Object obj) {
        if ((obj == null) || (getClass() != obj.getClass()))
            return false;
        Customer other = (Customer) obj;
        if ((id != other.id) && (!login.equals(other.login)) &&
(!password.equals( other.password)) && (!eMail.equals(other.eMail)) &&
(!phone.equals( other.phone)) && (role != other.role))
            return false;
        if (this == obj)
            return true;
        return true;
    }
    @Override
    public int hashCode() {
        int result = 1;
        int one = 17;
        result = one * result + (this.getLogin() == null ? 0 :
this.getLogin().hashCode());
        return result;
    }
    @Override
    public String toString() {
        return getLogin() + " e-mail: " + getMail() + " phone: " + getPhone();
    }
}
```

Лістинг Г.22. – Код класу «Item»

```
package model;

/**
 * Class that describes the item's.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */
public class Item {
    private int idItem;
    private String name;
    private String description;
    private ItemType type;
    private Item parent;

    public Item() {

    }

    public Item(int id) {
        this.idItem = id;
    }

    public Item(int id, String name, String des, Item par, ItemType itemType) {
        this.idItem = id;
        this.name = name;
        this.description = des;
        this.parent = par;
        this.type = itemType;
    }

    public Item(String name, String des, Item par, ItemType itemType) {
        this.name = name;
        this.description = des;
        this.parent = par;
        this.type = itemType;
    }

    public enum ItemType {
        Rubric,
        Book,
        Section;
    }

    public void setId(int id) {
        this.idItem = id;
    }

    public int getId() {
        return idItem;
    }

    public ItemType getType() {
        return type;
    }

    public void setType(ItemType type) {
        this.type = type;
    }
```

```
    public Item getParent() {
        return parent;
    }

    public void setParent(Item parent) {
        this.parent = parent;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public boolean equals(Object obj) {
        if ((obj == null) || (getClass() != obj.getClass()))
            return false;
        Item other = (Item) obj;
        if ((idItem != other.idItem) && (!name.equals(other.name)) &&
(!description.equals(other.description)) && (type != other.type) && (parent !=
other.parent))
            return false;
        if (this == obj)
            return true;
        return true;
    }

    @Override
    public int hashCode() {

        int result = 1;
        int one = 17;
        result = one * result + (this.getName() == null ? 0 :
this.getName().hashCode());
        return result;
    }

    @Override
    public String toString() {

        return getName() + " Type: " + getType() + " Description:" +
getDescription() +
                (this.getName() == null ? "not" : " Parent: " +
getParent().getName());

    }
}
```

Лістинг Г.23. – Код класу «Order»

```
package model;


import java.util.*;

/**
 * Class that describes the order's.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */
public class Order {

    private int idOrder;
    private Customer customer;
    private Date dateOfOrder;
    private ArrayList<ContentOrder> content;

    public Order(int id, Customer customer, Date dateOfOrder,
ArrayList<ContentOrder> con) {
        content = new ArrayList<ContentOrder>();
        this.idOrder = id;
        this.customer = customer;
        this.dateOfOrder = dateOfOrder;
        this.content = con;

    }

    public Order(int id, Customer customer, Date dateOfOrder) {
        content = new ArrayList<ContentOrder>();
        this.idOrder = id;
        this.customer = customer;
        this.dateOfOrder = dateOfOrder;


    }

    public Order(Customer customer, Date dateOfOrder) {
        content = new ArrayList<ContentOrder>();
        this.customer = customer;
        this.dateOfOrder = dateOfOrder;


    }

    public Order() {
        content = new ArrayList<ContentOrder>();
    }

    public int getIdOrder() {
        return idOrder;
    }

    public void setIdOrder(int id) {
        this.idOrder = id;
    }

    public Customer getCustomer() {
        return customer;
    }
```

```
public void setCustomer(Customer customer) {
    this.customer = customer;
}

public Date getDateOfOrder() {
    return dateOfOrder;
}

public void setDateOfOrder(Date dateOfOrder) {
    this.dateOfOrder = dateOfOrder;
}

public ArrayList<ContentOrder> getContents() {
    return content;
}

public void setContents(ArrayList<ContentOrder> contents) {
    this.content = contents;
}

public void addCon(ContentOrder con) {
    content.add(con);
}


public class ContentOrder {

    private Book book;
    private int amount;

    public ContentOrder() {

    }

    public Book getBooks() {
        return book;
    }

    public int getAmount() {
        return amount;
    }

    public void setBook(Book book, int count) {
        this.book = book;
        this.amount = count;
    }

    public void removeBook() {
        this.book = null;
        this.amount = 0;
    }

    @Override
    public boolean equals(Object obj) {
        if ((obj == null) || (getClass() != obj.getClass()))
            return false;
        ContentOrder other = (ContentOrder) obj;
        if ((book != other.book) && (amount != other.amount))
            return false;
        if (this == obj)
            return true;
```

```
            return true;
        }

        @Override
        public int hashCode() {
            int result = 1;
            int one = 17;
            result = one * result + (this.getBooks() == null ? 0 :
this.getBooks().hashCode()) + (Integer.valueOf(this.getAmount()) == null ? 0 :
getAmount());
            return result;
        }

        @Override
        public String toString() {
            return "Book: " + getBooks() + " Count: " + getAmount();
        }
    }

    @Override
    public boolean equals(Object obj) {
        if ((obj == null) || (getClass() != obj.getClass()))
            return false;
        Order other = (Order) obj;
        if ((idOrder != other.idOrder) && (customer != other.customer) &&
(dateOfOrder != other.dateOfOrder) && (content != other.content))
            return false;
        if (this == obj)
            return true;
        return true;
    }

    @Override
    public int hashCode() {

        int result = 1;
        int one = 17;
        result = one * result + (this.getDateOfOrder() == null ? 0 :
this.getDateOfOrder().hashCode());
        return result;

    }

    @Override
    public String toString() {

        return "Customer: " + getCustomer().getLogin() + " Date: " +
getDateOfOrder();

    }

}
```

Лістинг Г.24. – Код інтерфейсу «ModelDataBase»

```
package model;

import exception.DataBaseException;

import java.sql.Connection;
import java.util.List;

/**
 * Interface that describes the communication with the database.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */
public interface ModelDataBase {

    Connection getConnection();

    void updateBook(Book book)throws DataBaseException;
    void updateAuthor(Author author)throws DataBaseException;
    void updateCustomer(Customer customer)throws DataBaseException;
    void updateItem(Item item) throws DataBaseException;
    void updateBookOfOrder(int idOrder,int idBook, int count)throws
DataBaseException;

    void createCustomer(Customer customer)throws DataBaseException;
    void createOrder(Order order)throws DataBaseException;
    void createBook(Book book)throws DataBaseException;
    void createAuthor(Author author)throws DataBaseException;
    void createRubric(Item rubric) throws DataBaseException;
    void createSection(Item section) throws DataBaseException;
    void addBookToOrder(Order order,Book book, int count)throws
DataBaseException;

    void removeBook(int bookId)throws DataBaseException;
    void removeAuthor(int authorId)throws DataBaseException;
    void removeOrder(Order order)throws DataBaseException;
    void removeCustomer(int customerId)throws DataBaseException;
    void removeRubric(int rubricId) throws DataBaseException;
    void removeSection(int sectionId) throws DataBaseException;
    void removeBookFromOrder(int idOrder,int idBook)throws  DataBaseException;

    List<Customer> getAllCustomer() throws DataBaseException;
    Customer getCustomer(String login, String Password) throws
DataBaseException;
    List<Author> getAllAuthor() throws DataBaseException;
    List<Order> getAllOrder() throws DataBaseException;
    List<Item> getAllRubric() throws DataBaseException;
    List<Item> getAllSection() throws DataBaseException;
    List<Book> getAllBooks() throws DataBaseException;
    List<Book> getAllBooksByRubric(int idRubric)throws DataBaseException;
    List<Book> getAmountOfBooks(int amount) throws DataBaseException;

    Book getBookById(int bookId) throws DataBaseException;
    Customer getCustomerById(int customerId) throws DataBaseException;
    Order getOrderById(int orderId) throws DataBaseException;
    Author getAuthorById(int authorId) throws DataBaseException;
    Item getRubricById(int rubricId) throws DataBaseException;
    Item getSectionById(int sectionId) throws DataBaseException;

}
```

Лістинг Г.25. – Код класу «OracleDataBaseAccess»

```
package model;

import exception.DataBaseException;
import org.apache.log4j.Logger;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
import java.sql.*;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.List;

/**
 * Class for connected to Data Base.
 *
 * @author Veleri Rechembei
 * @version %I%, %G%
 */

public class OracleDataAccess implements ModelDataBase{
    private static final Logger LOG = Logger.getLogger(OracleDataAccess.class);

    protected static class Singleton {
        public static final OracleDataAccess _INSTANCE = new OracleDataAccess();
    }

    public static OracleDataAccess getInstance() {
        return Singleton._INSTANCE;
    }

    private DataSource ds;
    private Context ctx;
    private Hashtable ht = new Hashtable();

    private OracleDataAccess(){
        ht.put(Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");
        ht.put(Context.PROVIDER_URL, "t3://localhost:7001");
        try {
            ctx = new InitialContext(ht);
            ds = (javax.sql.DataSource) ctx.lookup("myJNDIDBName"); // change
your JNDI_name
        } catch (NamingException e) {
            LOG.error("InitialContext or DataSource error", e);
        }finally {
            try {
                if (ctx != null) {ctx.close();}
            } catch (NamingException e) {
                LOG.error("error of close connection", e);
            }
        }
    }

    /**
     * Method for get connection with database.
     * @return connection with database.
     */
    public Connection getConnection(){
        Connection connection = null;
```

```
        try {
            connection = ds.getConnection();
        } catch (SQLException e) {
            LOG.error(e);
        }
        return connection;
    }

    /**
     * Method for disconnected of database.
     * @param connection connection of database.
     * @param result result of query.
     * @param statement statement for query.
     */
    private void disconnect(Connection connection, ResultSet result, Statement
statement) {
        try {
            if(statement != null)
                statement.close();
            if(connection != null)
                connection.close();
            if(result != null)
                result.close();
        } catch (SQLException e) {
            LOG.error(e);
        }
    }


    private int propertiesId(Book book)throws  DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        int id=-1;
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_PROPERTIES_BY_ID);
            statement.setInt(1, book.getId());
            result= statement.executeQuery();
            while (result.next()) {
                id =result.getInt("ID_PROPERTIES");
            }
        }
        catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return id;
    }

    /**
     * Method for update book.
     * @param book book,that needed to update.
     * @throws DataBaseException exception with data.
     */
    public void updateBook(Book book) throws  DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.UPDATE_ITEM);
```

```
            statement.setInt(1, book.getParent().getId());
            statement.setString(2, book.getName());
            statement.setString(3, book.getDescription());
            statement.setInt(4, book.getId());
            statement.executeUpdate();
            statement=null;
            int idP=propertiesId(book);
            if(idP!=-1) {
                statement =
connection.prepareStatement(SqlScripts.UPDATE_BOOK_PROPERTIES);
                statement.setInt(1, book.getAuthor().getId());
                statement.setInt(2, book.getPages());
                statement.setInt(3, book.getPrice());
                statement.setInt(4, book.getAmount());
                statement.setInt(5, idP);
                statement.executeUpdate();
            }
            else {
                throw new SQLException();
            }
        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }

    }

    /**
     * Method for update author.
     * @param author author,that needed to update.
     * throws DataBaseException exception with data.
     */
    public void updateAuthor(Author author) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.UPDATE_AUTHOR);
            statement.setString(1, author.getSurname());
            statement.setString(2, author.getName());
            statement.setInt(3, author.getId());
            statement.executeUpdate();

        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for update customer.
     * @param customer customer, that needed to update.
     * @throws DataBaseException exception with data.
     */
    public void updateCustomer(Customer customer) throws DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.UPDATE_CUSTOMER);
```

```
            statement.setString(1, customer.getLogin());
            statement.setString(2, customer.getPassword());
            statement.setString(3, customer.getMail());
            statement.setString(4, customer.getPhone());
            statement.setInt(5, customer.getRole());
            statement.setInt(6, customer.getId());
            statement.executeUpdate();

        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for create customer.
     * @param customer customer, that needed to create.
     * @return created customer.
     * @throws DataBaseException exception with data.
     */
    public void createCustomer(Customer customer) throws DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.CREATE CUSTOMER);
            statement.setString(1, customer.getLogin());
            statement.setString(2, customer.getPassword());
            statement.setString(3, customer.getMail());
            statement.setString(4, customer.getPhone());
            statement.setInt(5, customer.getRole());
            statement.execute();
        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }

    }

    /**
     * Method for create order.
     * @param order order, that needed to create.
     * @return created order.
     * @throws DataBaseException exception with data.
     */
    public void createOrder(Order order)throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.CREATE_ORDER);
            statement.setInt(1,order.getCustomer().getId());
            statement.setInt(2, order.getContents().get(0).getBooks().getId());
            statement.setInt(3,order.getContents().get(0).getAmount());
            statement.setDate(4, (java.sql.Date) order.getDateOfOrder());
            statement.execute();
            statement = null;
            result = null;

            statement =
```

```
connection.prepareStatement(SqlScripts.SELECT_LAST_ID_ORDER);
            result= statement.executeQuery();
            while (result.next()) {
                int idOr = result.getInt("MAX(ID_ORDER)");
                order.setIdOrder(idOr);
            }
            statement = null;
            for(int i=1;i<=order.getContents().size()-1;i++){
                statement =
connection.prepareStatement(SqlScripts.CREATE_NEW_CON);
                statement.setInt(1, order.getIdOrder());
                statement.setInt(2,
order.getContents().get(i).getBooks().getId());
                statement.setInt(3, order.getContents().get(i).getAmount());
                statement.execute();
                statement = null;
            }


        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }
    /**
     * Method for remove book from order.
     * @param idOrder id of order, that needed to update.
     * @param idBook id of book, that needed to remove.
     * @throws DataBaseException  exception with data.
     */
    public void removeBookFromOrder(int idOrder,int idBook)throws
DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.DELETE_ONE_CON_FROM_ORDER);
            statement.setInt(1, idOrder);
            statement.setInt(2, idBook);
            statement.execute();
        }
        catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }

    }
    /**
     * Method that add book to order.
     * @param order order, that needed to update.
     * @param book book,that needed to add.
     * @param count amount of book.
     * @throws DataBaseException exception with data.
     */
    public void addBookToOrder(Order order,Book book, int count)throws
DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
```

```
        Order.ContentOrder newCon = order.new ContentOrder();
        newCon.setBook(book,count);
        int id = order.getContents().size()+1;
        order.addCon(newCon);
        try {
            statement = connection.prepareStatement(SqlScripts.CREATE_NEW_CON);
            statement.setInt(1, order.getIdOrder());
            statement.setInt(2, order.getContents().get(id).getBooks().getId());
            statement.setInt(3, order.getContents().get(id).getAmount());
            statement.execute();
        }
        catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }

    }

    /**
     * Method for update book of order.
     * @param idOrder id of order, that needed to update.
     * @param idBook id of book,that needed to update.
     * @param count amount of book.
     * @throws DataBaseException  exception with data.
     */
    public void updateBookOfOrder(int idOrder,int idBook, int count)throws
DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.UPDATE_ORDER_CON);
            statement.setInt(1, count);
            statement.setInt(2, idOrder);
            statement.setInt(3, idBook);
            statement.execute();
        }
        catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }

    }

    /**
     * Method for create book.
     * @param book book, that needed to create.
     * @return created book.
     * @throws DataBaseException  exception with data.
     */
    public void createBook(Book book) throws DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.CREATE_BOOK);
            statement.setString(1, book.getName());
            statement.setString(2, book.getDescription());
            statement.setInt(3, book.getParent().getId());
```

```
            statement.setInt(4, book.getAuthor().getId());
            statement.setInt(5, book.getPages());
            statement.setInt(6, book.getPrice());
            statement.setInt(7, book.getAmount());
            statement.execute();
        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for create author.
     * @param author author, that needed to create.
     * @return created author.
     * @throws DataBaseException  exception with data.
     */
    public void createAuthor(Author author) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.CREATE_AUTHOR);
            statement.setString(1, author.getSurname());
            statement.setString(2, author.getName());
            statement.execute();
        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for remove book.
     * @param bookId id of book, that needed to remove.
     * @throws DataBaseException  exception with data.
     */
    public void removeBook(int bookId) throws DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.DELETE_BOOK);
            statement.setInt(1, bookId);
            statement.execute();
        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for remove author.
     * @param authorId id of author, that needed to remove.
     * @throws DataBaseException  exception with data.
     */
    public void removeAuthor(int authorId) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
```

```
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.DELETE_AUTHOR);
            statement.setInt(1, authorId);
            statement.execute();
        } catch (SQLException e) {
            throw new DataBaseException("Exception for remove", e);
        }
        finally {
            disconnect(connection, result, statement);
        }
    }


    /**
     * Method for remove order.
     * @param order is order, that needed to remove.
     * @throws DataBaseException  exception with data.
     */
    public void removeOrder(Order order) throws  DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.DELETE_ORDER);
            statement.setInt(1,order.getIdOrder());
            statement.execute();
            statement = null;
            result = null;

            for(int i=1;i<=order.getContents().size()-1;i++){
                statement =
connection.prepareStatement(SqlScripts.DELETE_CON_FOR_ORDERS);
                statement.setInt(1,order.getIdOrder());
                statement.execute();
                statement = null;
            }

        } catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for remove customer.
     * @param customerId id of customer, that needed to remove.
     * @throws DataBaseException  exception with data.
     */
    public void removeCustomer(int customerId)  throws DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.DELETE_CUSTOMER);
            statement.setInt(1, customerId);
            statement.execute();
        } catch (SQLException e) {
            throw new DataBaseException("Exception for remove", e);
        }
        finally {
```

```
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for return list of all customers.
     * @return list of all customers.
     * @throws DataBaseException if method have exception.
     */
    public List<Customer> getAllCustomer() throws DataBaseException {

        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Customer> listCustomer = new ArrayList<Customer>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ALL_CUSTOMER);
            result = statement.executeQuery();
            while (result.next()) {
                listCustomer.add(getCustomer(result));
            }

        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
       } finally {
            disconnect(connection, result, statement);
        }

        return listCustomer;
    }

    /**
     * Method that return customer if he exists.
     * @param login    is String.
     * @param password is String.
     * @return customers
     * @throws DataBaseException if was error.
     */
    public Customer getCustomer(String login, String password) throws
DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;

        Customer customer = null;
        try {
            statement = connection.prepareStatement(SqlScripts.SELECT_CUSTOMER);
            statement.setString(1, login);
            statement.setString(2, password);
            result = statement.executeQuery();
            while (result.next()) {
                customer = getCustomer(result);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }

        return customer;
    }
```

```
    private Customer getCustomer(ResultSet result) throws DataBaseException {
        Customer customer;
        try {
            int id = result.getInt("ID_CUSTOMER");
            String login = result.getString("LOGIN");
            String password = result.getString("PASSWORD");
            String eMail = result.getString("E_MAIL");
            String phone = result.getString("PHOME_NUBMER");
            int role = result.getInt("ROLE");
            customer = new Customer(id, login, password, eMail, phone, role);

        } catch (SQLException e) {
            throw new DataBaseException("Exception with data from result set",
e);
        }
        return customer;
    }

    /**
     * Method for return list of all authors.
     * @return list of all authors.
     * @throws DataBaseException if method have exception.
     */
    public List<Author> getAllAuthor() throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Author> listAuthor = new ArrayList<Author>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ALL_AUTHOR);
            result = statement.executeQuery();
            while (result.next()) {
                listAuthor.add(getAuthor(result));
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listAuthor;
    }

    private Author getAuthor(ResultSet result) throws DataBaseException {
        Author author;
        try {
            int id = result.getInt("ID_AUTHOR");
            String surname = result.getString("SURNAME");
            String name = result.getString("NAME");
            author = new Author(id, surname, name);
        } catch (SQLException e) {
            throw new DataBaseException("Exception with data from result set",
e);
        }
        return author;
    }

    /**
     * Method for return list of all orders.
     * @return list of all orders.
     * @throws DataBaseException if method have exception.
```

```
     */
    public List<Order> getAllOrder() throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Order> listOrder = new ArrayList<Order>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ALL_ORDER);
            result = statement.executeQuery();
            while (result.next()) {
               /* listOrder.add(getOrder(result));
                getConOfOrder(getOrder(result));
                */
                int idOr = result.getInt("ID_ORDER");
                Order or = getOrderById(idOr);
                listOrder.add(or);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listOrder;
    }
    public List<Order> getOrderByIdCustomer(int idCustomer) throws
DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Order> listOrder = new ArrayList<Order>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ORDER_BY_ID_CUSTOMER);
            statement.setInt(1,idCustomer);
            result = statement.executeQuery();
            while (result.next()) {
                int idOr = result.getInt("ID_ORDER");
                Order or = getOrderById(idOr);
                listOrder.add(or);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listOrder;
    }

    private void getConOfOrder(Order order) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_CON_OF_ORDER);
            statement.setInt(1, order.getIdOrder());
            result = statement.executeQuery();
            while (result.next()) {
                int idBook = result.getInt("ID_BOOK");
                Book book = getBookById(idBook);
                int amount = result.getInt("AMOUNT");
```

```
                Order.ContentOrder con = order.new ContentOrder();
                con.setBook(book,amount);
                order.getContents().add(con);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }

    }

    private Order getOrder(ResultSet result) throws DataBaseException {
        Order order=new Order();
        try {
            int idOr = result.getInt("ID_ORDER");
            int idCus = result.getInt("ID_CUSTOMER");
            Customer cus =
OracleDataAccess.getInstance().getCustomerById(idCus);
            Date data = result.getDate("DATA");
            Order.ContentOrder con = order.new ContentOrder();
            order = new Order(idOr, cus, data);
            order.addCon(con);
        } catch (SQLException e) {
            throw new DataBaseException("Exception with data from result set",
e);
        }
        return order;
    }

    /**
     * Method for get book by id.
     * @param bookId id of book.
     * @return book.
     * @throws DataBaseException exception with data.
     */
    public Book getBookById(int bookId) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        Book book = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ALL_BOOK_BY_ID);
            statement.setInt(1, bookId);
            result = statement.executeQuery();
            while (result.next()) {
                book = getBook(result);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return book;
    }

    /**
     * Method for get customer by id.
     * @param customerId id of customer.
     * @return customer.
     * @throws DataBaseException  exception with data.
```

```
     */
    public Customer getCustomerById(int customerId) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        Customer cus = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_CUSTOMER_BY_ID);
            statement.setInt(1, customerId);
            result = statement.executeQuery();
            while (result.next()) {
                cus = getCustomer(result);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return cus;
    }

    /**
     * Method for get order by id.
     * @param orderId if of order.
     * @return order.
     * @throws DataBaseException exception with data.
     */
    public Order getOrderById(int orderId) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        Order order = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ODER_BY_ID);
            statement.setInt(1, orderId);
            result = statement.executeQuery();
            while (result.next()) {
                order = getOrder(result);
                getConOfOrder(order);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return order;
    }

    /**
     * Method for get author by id.
     * @param authorId if of author.
     * @return author.
     * @throws DataBaseException exception with data.
     */
    public Author getAuthorById(int authorId) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        Author author = null;
        try {
```

```
            statement =
connection.prepareStatement(SqlScripts.SELECT_AUTHOR_BY_ID);
            statement.setInt(1, authorId);
            result = statement.executeQuery();
            while (result.next()) {
                author = getAuthor(result);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return author;
    }

    /**
     * Method for get all rubrics.
     * @return list of all rubrics.
     * @throws DataBaseException exception with data.
     */
    public List<Item> getAllRubric() throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Item> listRubric = new ArrayList<Item>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ALL_RUBRIC);
            result = statement.executeQuery();
            while (result.next()) {
                listRubric.add(getItem(result, Item.ItemType.Rubric));
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listRubric;
    }

    private Item getItem(ResultSet result, Item.ItemType type) throws
DataBaseException,IllegalArgumentException {
        Item item;
        try {
            int id = result.getInt("ID_ITEM");
            String name = result.getString("NAME");
            String description = result.getString("DESCRIPTION");
            int par = result.getInt("PARENT_ID");
            Item newItem=null;
            if(type==Item.ItemType.Rubric){
                newItem = getSectionById(par);
            }

            item = new Item(id, name, description, newItem, type);
        } catch (SQLException e) {
            throw new DataBaseException("Exception with data from result set",
e);
        }
        return item;
    }

    /**
```

```
     * Method for get all sections.
     * @return list of all sections.
     * @throws DataBaseException exception with data.
     */
    public List<Item> getAllSection() throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Item> listSection = new ArrayList<Item>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_ALL_SECTION);
            result = statement.executeQuery();
            while (result.next()) {
                listSection.add(getItem(result, Item.ItemType.Section));
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listSection;
    }

    /**
     * Method return amount of books that you need.
     * @param amount of books.
     * @return List<Book>.
     * @throws DataBaseException Exception with data.
     */
    public List<Book> getAmountOfBooks(int amount) throws DataBaseException {
        int page = 1;
        Connection connection       = getConnection();
        ResultSet result            = null;
        PreparedStatement statement = null;

        List<Book> listBooks = new ArrayList<Book>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_PAGE_OF_LIST_BOOKS);
            statement.setInt(1, amount);
            statement.setInt(2, page);
            result    = statement.executeQuery();

            while (result.next()) {
                listBooks.add(getBook(result));
            }

        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }

        return listBooks;
    }

    /**
     * Method for get rubrics by section.
     * @param id id of rubric.
     * @return list of all rubrics by section.
     * @throws DataBaseException  exception with data.
```

```
     */
    public List<Item> getRubricBySection(int id)throws  DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Item> listRubric = new ArrayList<Item>();
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_RUBRIC_BY_SECTION);
            statement.setInt(1, id);
            result = statement.executeQuery();
            while (result.next()) {
                listRubric.add(getItem(result, Item.ItemType.Rubric));
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listRubric;
    }

    /**
     * Method for get all books.
     * @return list of books.
     * @throws DataBaseException  exception with data.
     */
    public List<Book> getAllBooks() throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Book> listBooks = new ArrayList<Book>();

        try {
            statement = connection.prepareStatement(SqlScripts.SELECT_ALL_BOOK);
            result = statement.executeQuery();
            while (result.next()) {
                listBooks.add(getBook(result));
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listBooks;
    }

    /**
     * Method for get all books by rubric.
     * @param idRubric id of rubric.
     * @return list of all books by rubric.
     * @throws @throws DataBaseException  exception with data.
     */
    public List<Book> getAllBooksByRubric(int idRubric) throws DataBaseException
{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Book> listBooks = new ArrayList<Book>();

        try {
            statement =
```

```
connection.prepareStatement(SqlScripts.SELECT_BOOK_BY_RUBRIC);
            statement.setInt(1, idRubric);
            result = statement.executeQuery();
            while (result.next()) {
                listBooks.add(getBook(result));
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listBooks;
    }
    private Book getBook(ResultSet result) throws DataBaseException {
        Book book;
        try {
            int id = result.getInt("ID_ITEM");
            String name = result.getString("NAME");
            int rubricId = result.getInt("RUBRIC");
            int authorId = result.getInt("AUTHOR");
            int pages = result.getInt("PAGES");
            int price = result.getInt("PRICE");
            int amount = result.getInt("AMOUNT");
            String description = result.getString("DESCRIPTION");
            String picture =  result.getString("PICTURE");
            Author findAuthor = getAuthorById(authorId);
            Item findRubric = getRubricById(rubricId);
            book = new Book(id, name, description,
findRubric,findAuthor,pages,price,amount,picture);

        } catch (SQLException e) {
            throw new DataBaseException("Exception with data from result set",
e);
        }
        return book;
    }
    /**
     * Method for remove rubric.
     * @param rubricId if of rubric, that needed to remove.
     * @throws @throws DataBaseException  exception with data.
     */
    public void removeRubric(int rubricId) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.DELETE_RUBRIC);
            statement.setInt(1, rubricId);
            statement.execute();
        } catch (SQLException e) {
            throw new DataBaseException("Exception for remove", e);
        }
        finally {
            disconnect(connection, result, statement);
        }
    }

    /**
     * Method for remove section.
     * @param sectionId id of section,that needed to remove.
     * @throws @throws DataBaseException  exception with data.
     */
```

```
public void removeSection(int sectionId) throws DataBaseException {
    Connection connection = getConnection();
    ResultSet result = null;
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(SqlScripts.DELETE_SECTION);
        statement.setInt(1, sectionId);
        statement.execute();
    } catch (SQLException e) {
        throw new DataBaseException("Exception for remove", e);
    }
    finally {
        disconnect(connection, result, statement);
    }
}

/**
 * Method for create rubric.
 * @param rubric rubric, that needed to create.
 * @throws @throws DataBaseException  exception with data.
 */
public void createRubric(Item rubric) throws DataBaseException {
    Connection connection = getConnection();
    ResultSet result = null;
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(SqlScripts.CREATE RUBRIC);
        statement.setString(1, rubric.getName());
        statement.setInt(2, rubric.getParent().getId());
        statement.setString(3, rubric.getDescription());
        statement.execute();
    } catch (SQLException e) {
        throw new DataBaseException("Exception for create", e);
    } finally {
        disconnect(connection, result, statement);
    }
}

/**
 * Method for create section.
 * @param section section, that needed to create.
 * @throws @throws DataBaseException  exception with data.
 */
public void createSection(Item section) throws DataBaseException {
    Connection connection = getConnection();
    ResultSet result = null;
    PreparedStatement statement = null;
    try {
        statement = connection.prepareStatement(SqlScripts.CREATE_SECTION);
        statement.setString(1, section.getName());
        statement.setString(2, section.getDescription());
        statement.execute();
    } catch (SQLException e) {
        throw new DataBaseException("Exception for create", e);
    } finally {
        disconnect(connection, result, statement);
    }
}

/**
 * Method for update item(rubric or section).
 * @param item item,that needed to update.
```

```
     * @throws @throws DataBaseException  exception with data.
     */
    public void updateItem(Item item) throws DataBaseException {
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        try {
            statement = connection.prepareStatement(SqlScripts.UPDATE_ITEM);
            if(item.getType()== Item.ItemType.Rubric) {
                statement.setInt(1, item.getParent().getId());
            }
            else{
                statement.setNull(1,java.sql.Types.NULL );
            }

            statement.setString(2, item.getName());
            statement.setString(3, item.getDescription());
            statement.setInt(4, item.getId());
            statement.executeUpdate();
        }
        catch (SQLException e) {
            throw new DataBaseException("Exception for create", e);
        } finally {
            disconnect(connection, result, statement);
        }
    }


    /**
     * Method, that get rubric by id.
     * @param rubricId id of rubric.
     * @return rubric.
     * @throws @throws DataBaseException  exception with data.
     */
    public Item getRubricById(int rubricId)throws DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        Item rubric = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_RUBRIC_BY_ID);
            statement.setInt(1, rubricId);
            result = statement.executeQuery();
            while (result.next()) {
                rubric = getItem(result,Item.ItemType.Rubric);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return rubric;
    }

    /**
     * Method, that get section by id.
     * @param sectionId id of section.
     * @return section.
     * @throws @throws DataBaseException  exception with data.
     */
    public  Item getSectionById(int sectionId)throws DataBaseException{
```

```
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        Item section = null;
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_SECTION_BY_ID);
            statement.setInt(1, sectionId);
            result = statement.executeQuery();
            while (result.next()) {
                section = getItem(result,Item.ItemType.Section);
            }
        } catch (Exception e) {
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return section;
    }
    public List<Book> getBooksByName(String name)throws DataBaseException{
        Connection connection = getConnection();
        ResultSet result = null;
        PreparedStatement statement = null;
        List<Book> listBooks = new ArrayList<Book>();
        name = name.replace("!", "!!")
                .replace("%", "!%")
                .replace(" ", "! ")
                .replace("[", "![");
        try {
            statement =
connection.prepareStatement(SqlScripts.SELECT_BOOK_BY_NAME);
            statement.setString(1, "%" + name + "%");
            result = statement.executeQuery();
            while (result.next()) {
                Book book = getBook(result);
                System.out.println(book.toString());
                listBooks.add(book);
            }

        } catch (Exception e) {
            e.printStackTrace();
            throw new DataBaseException("Exception with data from database", e);
        } finally {
            disconnect(connection, result, statement);
        }
        return listBooks;
    }
}
```

Лістинг Г.26. – Код класу «SqlScripts»

```
package model;
/**
 * Class that describes the scripts to database.
 * @author Veleri Rechembei
 * @version %I%, %G%
 */
public class SqlScripts {
    public static final String SELECT_ALL_BOOK = "SELECT
i.ID_ITEM,i.NAME,rub.ID_ITEM AS \"RUBRIC\",a.ID_AUTHOR AS\"AUTHOR\",\n" +
            "           p.PAGES,p.PRICE,p.AMOUNT,i.DESCRIPTION, i.PICTURE FROM
ITEM i,PROPERTIES p,AUTHOR a, \n" +
            "           ITEM rub WHERE i.TYPE =0 AND
i.ID_PROPERTIES=p.ID_BOOK\n" + "            AND p.ID_AUTHOR=a.ID_AUTHOR AND
i.PARENT_ID=rub.ID_ITEM AND rub.TYPE=1";
    public static final String SELECT_ALL_AUTHOR = "SELECT * FROM AUTHOR";
    public static final String SELECT_ALL_RUBRIC = "SELECT * FROM ITEM WHERE
TYPE =1";
    public static final String SELECT_ALL_SECTION = "SELECT * FROM ITEM WHERE
TYPE =2";
    public static final String SELECT_ALL_CUSTOMER = "SELECT * FROM CUSTOMER";
    public static final String SELECT_ALL_ORDER = "SELECT ID_ORDER FROM ORDERS";
    public static final String SELECT_CON_OF_ORDER = "SELECT ID_BOOK,AMOUNT FROM
CONTENR_ORDER WHERE ID_ORDER=?";
    public static final String SELECT_ID_ORDER = "SELECT
ORDERS.ID_ORDER,c.ID_CONTENT FROM ORDERS,CONTENR_ORDER c  WHERE ID_CUSTOMER=?
AND DATA=? AND c.ID_BOOK=? AND c.ID_ORDER=ORDERS.ID_ORDER";
    public static final String SELECT_AUTHOR_BY_ID = "SELECT * FROM AUTHOR WHERE
ID_AUTHOR =?";
    public static final String SELECT_ODER_BY_ID = "SELECT * FROM ORDERS WHERE
ID_ORDER =?";
    public static final String SELECT_ORDER_BY_ID_CUSTOMER = "SELECT * FROM
ORDERS WHERE ID_CUSTOMER = ?";
    public static final String SELECT_CUSTOMER_BY_ID = "SELECT * FROM CUSTOMER
WHERE ID_CUSTOMER=?";
    public static final String SELECT_RUBRIC_BY_ID = "SELECT * FROM ITEM WHERE
TYPE=1 AND ID_ITEM=?";
    public static final String SELECT_SECTION_BY_ID = "SELECT * FROM ITEM WHERE
TYPE=2 AND ID_ITEM=?";
    public static final String SELECT_ALL_BOOK_BY_ID = "SELECT
i.ID_ITEM,i.NAME,rub.ID_ITEM AS \"RUBRIC\",a.ID_AUTHOR AS\"AUTHOR\",\n" +
            "p.PAGES,p.PRICE,p.AMOUNT,i.DESCRIPTION, i.PICTURE FROM ITEM
i,PROPERTIES p,AUTHOR a," +
            "ITEM rub WHERE i.TYPE =0 AND i.ID_PROPERTIES=p.ID_BOOK " +
            "AND p.ID_AUTHOR=a.ID_AUTHOR AND i.PARENT_ID=rub.ID_ITEM AND
rub.TYPE=1 AND i.ID_ITEM=?";
    public static final String SELECT_BOOK_BY_RUBRIC = "SELECT
i.ID_ITEM,i.NAME,rub.ID_ITEM AS \"RUBRIC\",a.ID_AUTHOR AS\"AUTHOR\",\n" +
            "p.PAGES,p.PRICE,p.AMOUNT,i.DESCRIPTION, i.PICTURE FROM ITEM
i,PROPERTIES p,AUTHOR a," +
            "ITEM rub WHERE i.TYPE =0 AND i.ID_PROPERTIES=p.ID_BOOK" +
            " AND p.ID_AUTHOR=a.ID_AUTHOR AND i.PARENT_ID=rub.ID_ITEM AND
rub.TYPE=1 AND rub.ID_ITEM=?";
    public static final String SELECT_PROPERTIES_BY_ID = "SELECT ID_PROPERTIES
FROM ITEM WHERE ID_ITEM=? AND TYPE=0";
    public static final String SELECT_RUBRIC_BY_SECTION = "SELECT * FROM ITEM
WHERE TYPE =1 AND ITEM.PARENT_ID=?";
    public static final String CREATE_CUSTOMER = "INSERT INTO
CUSTOMER(LOGIN,PASSWORD,E_MAIL,PHOME_NUBMER, ROLE) values(?,?,?,?,?)";
    public static final String CREATE_AUTHOR = "INSERT INTO AUTHOR(SURNAME,NAME)
values(?,?)";
```

```
    public static final String CREATE_BOOK = "{call ADDBOOK(?,?,?,?,?,?,?)}";
    public static final String CREATE_RUBRIC = "INSERT INTO
ITEM(NAME,PARENT_ID,DESCRIPTION,TYPE) values(?,?,?,1)";
    public static final String CREATE_SECTION = "INSERT INTO
ITEM(NAME,PARENT_ID,DESCRIPTION,TYPE) values(?,null,?,2)";
    public static final String CREATE_ORDER = "{call  ADDORDER(?,?,?,?)}";
    public static final String CREATE_NEW_CON = "INSERT INTO
CONTENR_ORDER(ID_ORDER,ID_BOOK,AMOUNT) values(?,?,?)";
    public static final String DELETE_AUTHOR = "DELETE FROM AUTHOR WHERE
ID_AUTHOR = ?";
    public static final String DELETE_CUSTOMER = "DELETE FROM CUSTOMER WHERE
ID_CUSTOMER = ?";
    public static final String DELETE_RUBRIC = "DELETE FROM ITEM WHERE ID_ITEM =
? AND TYPE=1";
    public static final String DELETE_SECTION = "DELETE FROM ITEM WHERE ID_ITEM
= ? AND TYPE=2";
    //public static final String DELETE_BOOK = "DELETE ITEM WHERE ID_ITEM = ?
AND TYPE =0";
    public static final String DELETE_BOOK =  "{call  DELETEBOOK(?)}";
    public static final String UPDATE_ITEM = "UPDATE ITEM SET
PARENT_ID=?,NAME=?,DESCRIPTION=? WHERE ID_ITEM = ?";
    public static final String UPDATE BOOK PROPERTIES = "UPDATE PROPERTIES SET
ID_AUTHOR=?,PAGES=?,PRICE=?,AMOUNT=? WHERE ID_BOOK=?";
    public static final String UPDATE_AUTHOR = "UPDATE AUTHOR SET
SURNAME=?,NAME=? WHERE ID_AUTHOR = ?";
    public static final String UPDATE_CUSTOMER = "UPDATE CUSTOMER SET
LOGIN=?,PASSWORD=?,E MAIL=?,PHOME NUBMER=?,ROLE=? WHERE ID CUSTOMER = ?";
    public static final String SELECT_LAST_ID_ORDER = "SELECT MAX(ID_ORDER) FROM
ORDERS";
    public static final String DELETE_ORDER = "DELETE ORDERS WHERE ID_ORDER =
?";
    public static final String DELETE_CON_FOR_ORDERS = "DELETE CONTENR_ORDER
WHERE ID_ORDER = ?";
    public static final String UPDATE_ORDER_CON = "UPDATE CONTENR_ORDER SET
AMOUNT=? WHERE ID_ORDER = ? AND ID_BOOK=?";
    public static final String DELETE_ONE_CON_FROM_ORDER = "DELETE CONTENR_ORDER
WHERE ID_ORDER = ? AND ID_BOOK=?";
    public static final String SELECT_CUSTOMER = "SELECT * FROM CUSTOMER WHERE
LOGIN=? and PASSWORD=?";
    public static final String SELECT_PAGE_OF_LIST_BOOKS = "select * from (
select a.*, ROWNUM rnum\n" +
            "  from (SELECT i.ID_ITEM, i.NAME, rub.ID_ITEM AS \"RUBRIC\",
a.ID_AUTHOR AS \"AUTHOR\", p.PAGES, p.PRICE, p.AMOUNT, i.DESCRIPTION,
i.PICTURE\n" +
            "        FROM ITEM i, PROPERTIES p, AUTHOR a, ITEM rub\n" +
            "        WHERE i.TYPE = 0 AND i.ID_PROPERTIES = p.ID_BOOK AND
p.ID_AUTHOR = a.ID_AUTHOR AND i.PARENT_ID = rub.ID_ITEM AND rub.TYPE = 1)a\n" +
            "  where ROWNUM <=  ?)\n" +
            "where rnum  >= ?";
    public static final String SELECT_BOOK_BY_NAME =
            "SELECT i.ID_ITEM, i.NAME, rub.ID_ITEM AS \"RUBRIC\", a.ID_AUTHOR
AS\"AUTHOR\",\n" +
            "           p.PAGES, p.PRICE, p.AMOUNT, i.DESCRIPTION, i.PICTURE\n"
+ "FROM ITEM i, PROPERTIES p, AUTHOR a, ITEM rub\n" +
            "WHERE i.TYPE = 0\n" +
            "      AND i.ID_PROPERTIES = p.ID_BOOK\n" +
            "      AND p.ID_AUTHOR = a.ID_AUTHOR\n" +
            "      AND i.PARENT_ID = rub.ID_ITEM\n" +
            "      AND rub.TYPE = 1\n" +
            "      AND lower(i.name || i.DESCRIPTION || a.NAME || a.SURNAME)
like lower(?) ESCAPE '!'";
}
```

Лістинг Г.27. – Код класу «Commands»

```
package Servlet;

import controller.processors.*;
import org.apache.log4j.Logger;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

/**
 * Class that generate commands for servlet.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class Commands {
    private static final Logger LOG = Logger.getLogger(Commands.class);
    private Map<String, Object> map;

    protected static class Singleton {
        public static final Commands _INSTANCE = new Commands();
    }

    private Commands() {
        this.initMap();
    }

    public static Commands getInstance() {
        return Singleton._INSTANCE;
    }

    public Map<String, Object> getCommandsMap(){
        return map;
    }

    private void initMap() {
        map = new HashMap<String, Object>();

        map.put(ACTION_WELCOME, new Welcome());
        map.put(ACTION_ADD_CUSTOMER, new AddCustomer());
        map.put(ACTION_DETAIL, new DetailBook());
        map.put(ACTION_VIEW_LIST_BOOKS, new ViewListBooks());
        map.put(ACTION_LOGIN_USER, new LoginUser());
        map.put(ACTION_ADD_ORDER, new AddOrder());
        map.put(ACTION_UPDATE_CUSTOMER, new UpdateCustomer());
        map.put(ACTION_UN_LOGIN, new UnLogin());
        map.put(ACTION_UPDATE_BOOK,new UpdateBook());
        map.put(ACTION_DELETE_BOOK,new DeleteBook());
        map.put(ACTION_ADD_BOOK,new AddBook());
        map.put(ACTION_DELETE_ORDER,new DeleteOrder());
        map.put(ACTION_UPDATE_ORDER,new UpdateOrder());
        map.put(ACTION_ADD_RUBRIC,new AddItem());
        map.put(ACTION_DELETE_ITEM,new DeleteItem());
        map.put(ACTION_SEND_MAIL,new SendMail());
    }
```

```
    public static void forward(String url, HttpServletRequest request,
HttpServletResponse response) {
        RequestDispatcher rd = request.getRequestDispatcher(url);
        try {

            rd.forward(request, response);

        } catch (ServletException e) {

            LOG.error(e);

        } catch (IOException e) {

            LOG.error(e);
        }

    }

    public static final String ACTION_ADD_CUSTOMER     = "addCustomer";

    public static final String ACTION_WELCOME          = "welcome";
    public static final String ACTION DETAIL           = "viewDetailBooks";

    public static final String ACTION_UPDATE_CUSTOMER = "updateCustomer";

    public static final String ACTION_UPDATE_BOOK     = "updateBook";

    public static final String ACTION_ADD_ORDER       = "addOrder";

    public static final String ACTION_DELETE_BOOK     = "deleteBook";

    public static final String ACTION_ADD_BOOK        = "addBook";

    public static final String ACTION_DELETE_ORDER    = "deleteOrder";

    public static final String ACTION_UPDATE_ORDER    = "updateOrder";

    public static final String ACTION_DELETE_ITEM       = "deleteItem";

    public static final String ACTION_UN_LOGIN        = "unLogin";

    public static final String ACTION_VIEW_LIST_BOOKS  = "viewListBooks";

    public static       int    AMOUNT_OF_BOOKS_ON_LIST     = 6;             //
when session starts it equals 6 and after each
    public static final int    START_OR_PLUS_BOOKS_TO_LIST = 6;             //
request + 6 in ViewListBooks.java
    public static final String ACTION_LOGIN_USER       = "loginUser";

    public static final String ACTION_ADD_RUBRIC      =  "addRubric";

    public static final String ACTION_SEND_MAIL      =  "sendMail";
}
```

Лістинг Г.28. – Код класу «MainServlets»

```
package Servlet;

import controller.processors.GeneralProcess;
import exception.DataBaseException;
import org.apache.log4j.Logger;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.Map;

/**
 * Class for handling requests and responses.
 *
 * @author Veleri Vovk
 * @version %I%, %G%
 */
public class MainServlet extends HttpServlet {
    private static final Logger LOG = Logger.getLogger(MainServlet.class);
    private Map<String, Object> map = Commands.getInstance().getCommandsMap();

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");

        String action = request.getParameter("action");
        if (action == null || action.isEmpty()) {
            action = Commands.ACTION_WELCOME;
        }

        GeneralProcess process;
        process = (GeneralProcess) map.get(action);

        if (process == null) {
            process = (GeneralProcess) map.get(Commands.ACTION_WELCOME);
        }
        if (process != null) {
            try {
                process.process(request, response);
            } catch (DataBaseException e) {
                LOG.error(e);
            }
        }
    }
}
```