Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Кручка Дмитра Вікторовича*
(ПІБ)

академічної групи *122-20ск-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка Web сайту з продажу книжок з використанням фреймворку NodeJS*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В.*			
розділів:				
спеціальний	*доц. Кабак Л.В.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20ск-1* (група) *Кручка Дмитра Вікторовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка Web сайту з продажу книжок з використанням фреймворку NodeJS*

затверджена наказом ректора НТУ «ДП» від *16.05.2023 р.* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПО й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ (підпис) *доц. Кабак Л.В.* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Кручок Д.В.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *3.07.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 92 с., 17 рис., 6 табл., 3 дод., 22 джерела.

Об'єкт розробки: інтерактивний веб-сайт з продажу книжок.

Мета дипломного проекту: забезпечити зручний інструментарій для автоматизації та поліпшення бізнес-процесів книжкового магазину.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної області, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування системи, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленого веб-сайту, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Актуальність даного програмного забезпечення визначається необхідністю оптимізації роботи книжкового магазина, зниження ризиків помилок та підвищення рівня задоволеності клієнтів.

Список ключових слів: ВЕБ-САЙТ, ЗАМОВЛЕННЯ, ІНТЕРНЕТ-МАГАЗИН, САЙТ, БАЗА ДАНИХ, МОДЕЛЬ ДАНИХ, PHP, HTML, NODEJS.

ABSTRACT

Explanatory note: 92 pages, 17 pics, 6 tables, 3 apps, 22 sources.

Object of development: an interactive website for the sale of books.

The purpose of the diploma project: to provide convenient tools for automating and improving bookstore business processes.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the system, determines the input and output data, provides the characteristics of the parameters of hardware, describes the call and application load, describes the program.

In the economic section, the complexity of the developed website is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The relevance of this software is determined by the need to optimize the work of the bookstore, reduce the risk of errors and increase the level of customer satisfaction.

List of keywords: WEBSITE, ORDER, ONLINE STORE, SITE, DATABASE, DATA MODEL, PHP, HTML, NODEJS.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс програмування застосунків;

CSS – Cascading Styles Sheets;

DOM – об’єктна модель документа;

HTML – HyperText Markup Language;

PHP – Hypertext Preprocessor.

ВСТУП

У сучасному світі книжкові магазини зберігають свою актуальність та привабливість завдяки впровадженню інформаційних систем, які допомагають автоматизувати та ефективно керувати різними аспектами їхньої діяльності. Веб-орієнтовані застосунки сприяють оптимізації управління товарами, квитанціями, постачальниками та клієнтами для книжкових магазинів різного масштабу.

Книжковий магазин пропонує широкий асортимент книг для всіх категорій клієнтів, та в процесі своєї роботи він стикається з різноманітними завданнями, такими як прийом та продаж книг, замовлення нових книг від постачальників, ведення обліку товарів, створення фінансових звітів та багато інших. Застосування веб-орієнтованого застосунку надасть магазину зручні та ефективні інструменти для вирішення цих завдань.

Основною метою впровадження веб-сайту є автоматизація та поліпшення бізнес-процесів книжкового магазину. Застосунок дозволить магазину ефективно керувати своїм асортиментом книг, відстежувати наявність товарів на складі, здійснювати продажі та обробку повернень, здійснювати замовлення нових книг від постачальників та створювати звіти для аналізу фінансової продуктивності магазину. Окрім того, інформаційна система може включати довідники постачальників та постійних клієнтів магазину. Це дозволить зберігати та керувати інформацією про постачальників, їхню контактну інформацію та умови постачання, а також про клієнтів, які роблять попередні замовлення книжок. Веб-сайт буде надійним помічником у повсякденних завданнях магазину та дозволить зосередитися на розвитку його бізнесу.

Метою кваліфікаційної роботи є створення веб-сайту, який давав би змогу звичайному користувачеві здійснювати купівлю книг різного формату у зручній візуалізованій формі. У якості основи веб-сайту запропоновано використати фреймоврк Node.js.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

На рис.1.1. графічно представлена спрощена організаційна структура книжкового магазину:

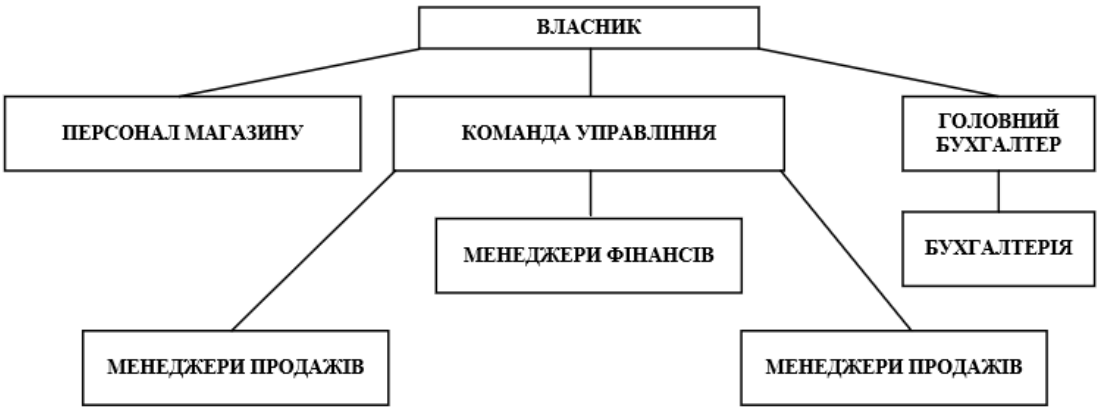

Рис. 1.1. Організаційна структура магазину

Бізнес-правила книжкового магазину:

Клієнтоорієнтованість: фірма ставить клієнтів на перше місце і забезпечує їм високу якість обслуговування. Всі працівники магазину повинні бути доброзичливими, ввічливими та готовими надати необхідну допомогу клієнтам.

Якість товарів: магазин прагне пропонувати високоякісні книги, які задовольняють потреби та очікування клієнтів. Забезпечується відповідний відбір книг від постачальників, перевірка наявності ліцензій та якісних сертифікатів.

Різноманітність асортименту: магазин пропонує широкий вибір книг у різних жанрах та категоріях, щоб задовольнити різноманітні інтереси та потреби клієнтів.

Конкурентоспроможні ціни: фірма встановлює конкурентоспроможні ціни на свої товари, забезпечуючи баланс між якістю та доступністю для клієнтів.

Ефективне управління запасами: магазин використовує інформаційну систему та базу даних для ефективного управління запасами книг. Забезпечується постійне оновлення запасів, уникнення дублювання або нестачі товарів.

Найбільш важливими компонентами для бізнесу книжкового магазину є:

- асортимент книг;
- продаж та обслуговування клієнтів;
- замовлення від постачальників;
- облік товарів та складського управління;
- маркетинг та реклама;
- фінансове управління на аналітика.

Основними процесами, що відбуваються у книжковому магазині, є:

- закупівля книг;
- приймання та перевірка товару;
- складське управління;
- продаж та обслуговування клієнтів;
- облік та фінансове управління.

Дані процеси відбуваються в магазині та складському приміщенні.

Також створюються щомісячні звіти про кількість проданих книг, та звіти щодо місячного та щорічного доходу. Вони виконуються з метою забезпечення ефективної та успішної діяльності магазину.

Мета роботи книжкового магазину:

- продаж книг;
- закупівля книг;
- складське управління;
- обслуговування клієнтів.

Формалізований опис постановки задачі виглядає наступним чином:

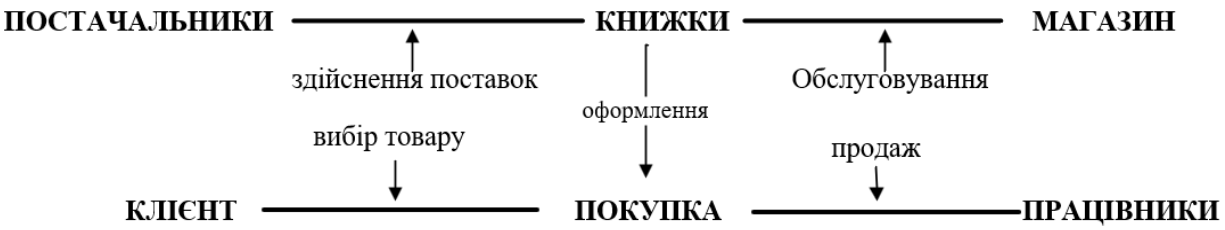

Рис. 1.2. Концептуальна схема діяльності книжкового магазину

Необхідно створити інформаційну систему, яка буде здійснювати ведення протоколу дій невеликого книжкового магазину. Застосунок повинен забезпечувати відстеження наявності товарів на складі, обробку продажів та повернень, здійснювати замовлення нових книг від постачальників та створювати звіти для аналізу фінансової продуктивності магазину. Окрім того, база даних повинна включати довідники постачальників та постійних клієнтів магазину, що дозволить зберігати та керувати інформацією про постачальників, а також про клієнтів, які роблять попередні замовлення книжок.

Перелічені функції та можливості застосунку допоможуть книжковому магазину оптимізувати свою роботу, знизити ризики помилок та покращити задоволеність клієнтів.

1.2 Призначення розробки та галузь застосування

Темою кваліфікаційної роботи виступає розробка Web сайту з продажу книжок з використанням фреймворку NodeJS. У якості інструментів для розробки програмного забезпечення обрано Node.js та Visual Studio Code. Метою роботи є створення веб-сайту для інформаційної підтримки купівлі книг різного формату (електронна, аудікнига, фізична з доставкою).

Головними вимогами до інтерфейсу розроблювального веб-застосунку є:

- зручність в використанні;
- максимальна доступність для користувача;

- легкість в опануванні.

Система призначена для:

- реєстрація користувача;
- перегляду книг на сайті;
- купівлі книг.

Система позиціонується як веб-застосунок, який дає можливість робити купівлю книг різного формату.

1.3 Підстава для розробки

Відповідно до освітньо-професійної програми спеціальності, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- освітньо-професійна програма за спеціальністю 122 «Комп'ютерні науки»;
- графік навчального процесу та навчальний план;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
- завдання на кваліфікаційну роботу на тему «Розробка Web сайту з продажу книжок з використанням фреймворку NodeJS».

1.4 Постановка завдання

Метою кваліфікаційної роботи є розробка Web сайту з продажу книжок з використанням фреймворку NodeJS. Сайт призначений для швидкого та зручного перегляду галереї товарів та купівлі книг.

Веб-додаток повинен реалізувати наступні дії:

- збереження даних локально;

- реєстрація різних категорій користувачів (адміністраторів, звичайних клієнтів);
- виконання зручної фільтрації для перегляду галереї книг.

Для виконання проекту необхідно:

- проаналізувати існуючі рішення для книжкових інтернет-магазинів;
- спроектувати архітектуру сайту;
- створити та підключити базу даних;
- створити код веб-сайту на основі фреймворку Node.js;
- розробити дизайн та зручний інтерфейс застосунку.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей розроблене програмне забезпечення повинно підтримувати виконання таких дій:

- Реагування на дії користувача в онлайн режимі;
- Авторизація;
- Авто-збереження будь-яких змін, здійснених користувачем;
- Здійснення максимально швидкої навігації по сайту;
- Коректна візуалізація елементів веб-застосунку.

Для підтримки вище перераховані функцій у застосунку має бути реалізовано:

- Стандартна конфігурація, яка дає змогу легко ввести застосунок в експлуатацію;
- Сумісність з веб-браузером та реалізація доступу до програми через нього;
- Підтримка програмної та апаратної сумісності.

1.5.2 Вимоги до інформаційної безпеки

Для коректної роботи програми потрібно реалізувати:

- Можливість редагування даних;
- Можливість тривалої роботи протягом 24 годин (1 доба);
- Контроль та обробку вхідних даних;
- Збереження цілісності даних у випадку збою системи;
- Локальне збереження даних для більшої захищеності.

Також веб-сайт книжкового магазину повинен мати наступні характеристики:

- Захист від несанкціонованого доступу;
- Відновлення після збою протягом 10 хвилин;
- Захист даних користувача при повторному підключені.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для забезпечення надійного функціонування програмного забезпечення необхідно, щоб обчислювальна машина, на якій буде експлуатуватися веб-застосунок, мала такі характеристики:

- Процесор класу Intel Pentium CPU N3710 з тактовою частотою не нижче 1.6 ГГц;
- Не менше ніж 4 Гб оперативної пам’яті;
- 1 Гб вільного місця на жорсткому диску;
- Маніпулятор “миша”;
- Клавіатура;
- Доступ до мережі Інтернет;
- Рідкокристалічний монітор.

Вище наведені характеристики являють собою рекомендовані. Це означає, що при наявності характеристик не нижче зазначених, розроблений

застосунок буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

1.5.4 Вимоги інформаційної та програмної сумісності

Для коректного функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде експлуатуватися веб-застосунок, відповідало наступним вимогам:

- Операційна система Windows 7/8/10, Linux;
- Веб браузер Google Chrome / Firefox / Opera / Microsoft Edge.

Веб-орієнтована підсистема має бути реалізована на мові програмування HTML та PHP з використанням бази даних MySQL (під управлінням PhpMyAdmin). Середовищем для написаня коду виступатиме Visual Studio Code. Створення та відображення сайту у браузері здійснюється за домогою набору програм Denwer. Для візуалізації було використано CSS, який описує зовнішній вигляд сторінки.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Під час виконання кваліфікаційної роботи було розроблено веб-сайт з продажу книжок з використанням фреймворку NodeJS, головна мета якого забезпечити зручний та якісний сайт для замовлення книг різного формату.

Призначення розробленого застосунку:

- Реалізація онлайн середовища для перегляду галереї книг із зручними способами фільтрування та сортування;
- Надання можливості робити замовлення книг;
- Здійснення реєстрації різних категорій користувачів (клієнтів та адміністраторів);
- Надання інструментів редагування галереї книг адміністратором;
- Надання можливості візуального простеження за створеними елементами.

Для досягнення вище перерахованих функціональних можливостей розроблена програма повинна:

- Мати сумісність з стандартною апаратною конфігурацією обчислювальної машини;
- Забезпечувати підтримку операційних систем Windows 10/8/7, Linux;
- Мати підтримку сучасних браузерів Firefox / Google Chrome / Opera / Microsoft Edge.

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці веб-сайту для книжкового магазину математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Веб-застосунок має бути реалізован на мові програмування HTML та PHP з використанням фреймворку Node.js для рендеру веб-компонентів та зберігання даних. У якості середовища програмування було обрано Visual Studio Code. Для візуалізації було використано CSS який описує зовнішній вигляд сторінки. Також використовується phpmyadmin та Denwer.

Опис мови програмування HTML

HTML — стандартизована мова розмітки документів для перегляду вебсторінок у браузері. Браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою конструкцій HTML, зображення та інші об'єкти, такі як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML окреслені тегами, написаними з використанням кутових дужок. Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

В HTML можна вбудовувати програми, написані на скриптових мовах, наприклад JavaScript, які впливають на поведінку та вміст вебсторінок. Включення CSS визначає вигляд і компонування вмісту. World Wide Web

Consortium (W3C), який супроводжує стандарти HTML та CSS, заохочує використання CSS над явним презентаційним HTML з 1997 року.

HTML впроваджує засоби для:

- створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
- отримання інформації зі Всесвітньої мережі через гіперпосилання;
- створення інтерактивних форм;
- включення зображень, звуку, відео, та інших об'єктів до тексту.

Опис мови програмування PHP

PHP — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні вебсервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веброзробок (разом із Java, .NET, JavaScript, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проєкт відкритого програмного забезпечення.

PHP інтерпретується вебсервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, тому що браузер отримує готовий html-код. Це є перевагою з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

Опис фремворку Node.js

Платформа Node.js призначена для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript. Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback). Як способи мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select. Для мультиплексування з'єднань

використовується бібліотека libuv, для створення пулу нитей (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що спричиняють блокування, виконуються всередині пулу потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe).

Node.js має наступні властивості:

- асинхронна одно-нитева модель виконання запитів;
- неблокуючий ввід/вивід;
- система модулів CommonJS;
- рушій JavaScript Google V8;

Для керування модулями використовується пакетний менеджер npm (node package manager).

За своєю суттю Node.js схожий на фреймворки Perl AnyEvent, Ruby Event Machine і Python Twisted, але цикл обробки подій (event loop) у Node.js прихований від розробника і нагадує обробку подій у веб-застосунку, що працює в браузері. При написанні програм для Node.js необхідно враховувати специфіку подієво-орієнтованого програмування.

Опис CSS

Каскадні таблиці стилів (CSS) - це таблиця стилів, яка використовується для опису подання документа, написаного мовою розмітки, як HTML. CSS - це основна технологія всесвітньої павутини, поряд із HTML та JavaScript.

CSS розроблений, щоб забезпечити розділення презентації та контенту, включаючи макет, кольори та шрифти. Цей поділ може покращити доступність контенту, забезпечити більшу гнучкість та контроль у конкретизації характеристик презентації, дасть змогу декільком веб-сторінкам обмінюватися форматуванням, вказавши відповідний CSS в окремому файлі .css, а також зменшити складність та повторність структурного вмісту.

Поділ форматування та контенту також робить можливим подання однієї і тієї ж сторінки розмітки в різних стилях для різних методів візуалізації, таких

як екран, друк, голос (через мовленнєвий браузер чи зчитувач екрана). CSS також має правила альтернативного форматування.

Специфікації CSS підтримує Всесвітній консорціум із веб-сторінок (W3C). Текст CSS типу Інтернет-медіа (тип MIME) зареєстрований для використання в CSS RFC 2318 (березень 1998 р.).

Опис середовища Visual Studio Code

Visual Studio Code, який також зазвичай називають VS Code — це редактор вихідного коду, створений Microsoft із Electron Framework для Windows, Linux і macOS. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш, параметри та встановлювати розширення, які додають функціональність.

Опис веб-застосунку phpMyAdmin

phpMyAdmin — веб-застосунок з відкритим кодом, написаний мовою PHP із графічним вебінтерфейсом для адміністрування бази даних MySQL або MariaDB. phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Ця програма користується великою популярністю у веброзробників, оскільки дозволяє керувати базу даних MySQL без вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера під'єднаного до інтернету без необхідності встановлення додаткового програмного забезпечення.

На сьогоднішній день phpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, з огляду на всі нововведення СУБД MySQL. Переважна більшість українських провайдерів використовують цей застосунок як панель керування для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних.

Програма розповсюджується під ліцензією GNU General Public License і тому деякі інші розробники інтегрують його у свої розробки, наприклад XAMPP, Denwer.

2.4. Опис структури системи та алгоритмів її функціонування

Якщо формалізувати процес покупки товарів в книжковому інтернет-магазині, то можна отримати послідовністю наступних кроків:

• вибір категорії товару;

• вибір певної кількості певної позиції товару;

• додавання товару в корзину;

• перегляд кошика і оформлення замовлення;

• реєстрація замовлення в системі і відправка його менеджеру магазину в відділ продажів або доставки.

Узагальнена схема алгоритму поведінки клієнта інтернет-магазину зображена на рис. 2.1.

На рис.2.2 представлена схема елементів архітектури веб-сайту книжкового магазину. Вона має клієнт-серверну архітектуру, в якій клієнтом виступає браузер, а сервер представлений Web-сервером і MySQL-сервером, під керуванням якого працює віддалена база даних інтернет-магазину.

Така організація дає наступні переваги: web-браузер вбудований в більшість операційних систем, таким чином, функції по розробці, установці, відновленні і підтримці клієнтської частини не виконуються розробником підсистеми. Так само, клієнти не залежать від конкретної операційної системи користувача, і розроблена підсистема, таким чином, є міжплатформеною. При цьому функції підсистеми реалізуються один раз, замість того, щоб розробляти різні версії для Microsoft Windows, Mac OS X, GNU / Linux та інших операційних систем.

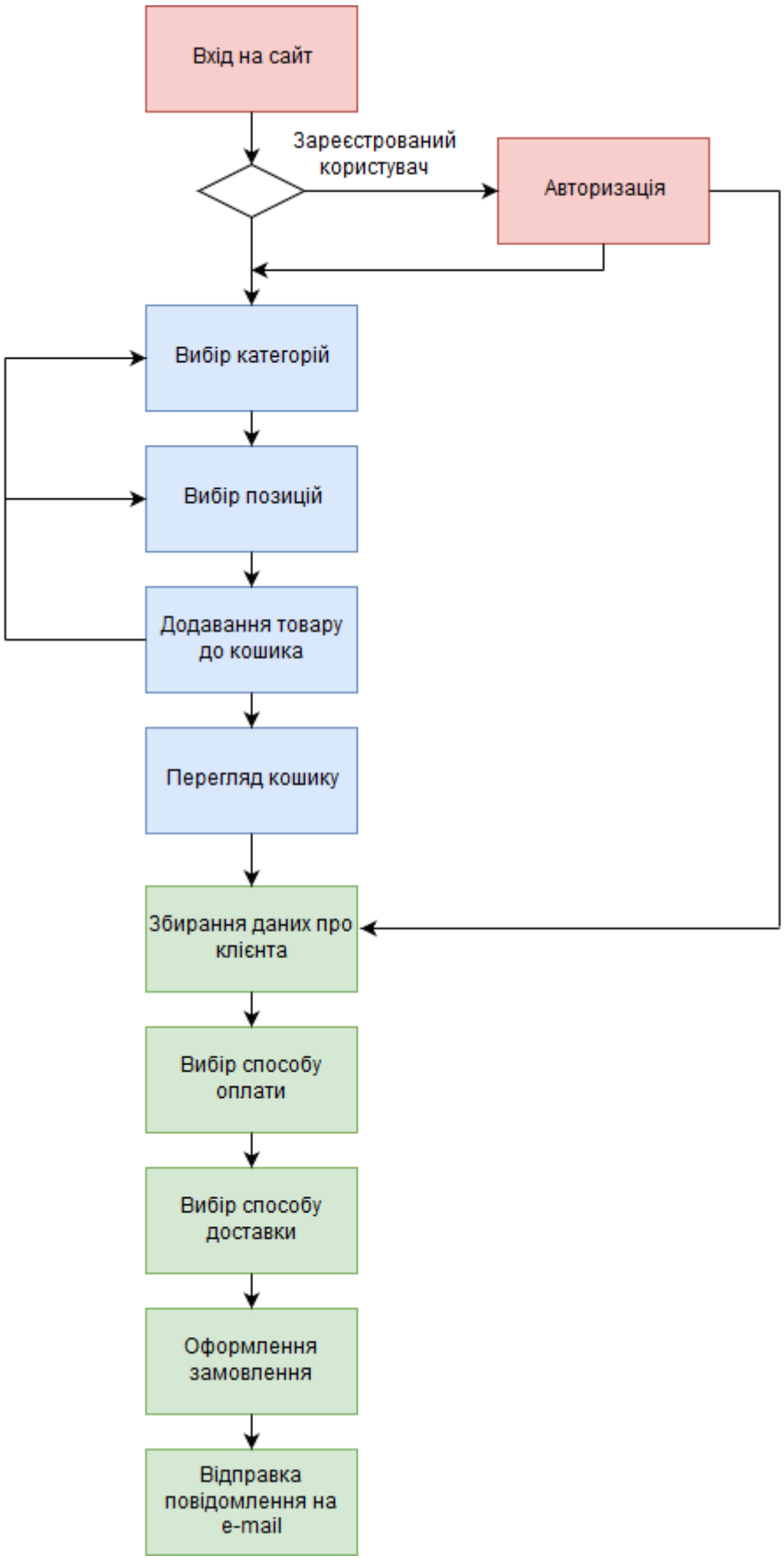

Рис. 2.1. Узагальнена схема формування замовлення

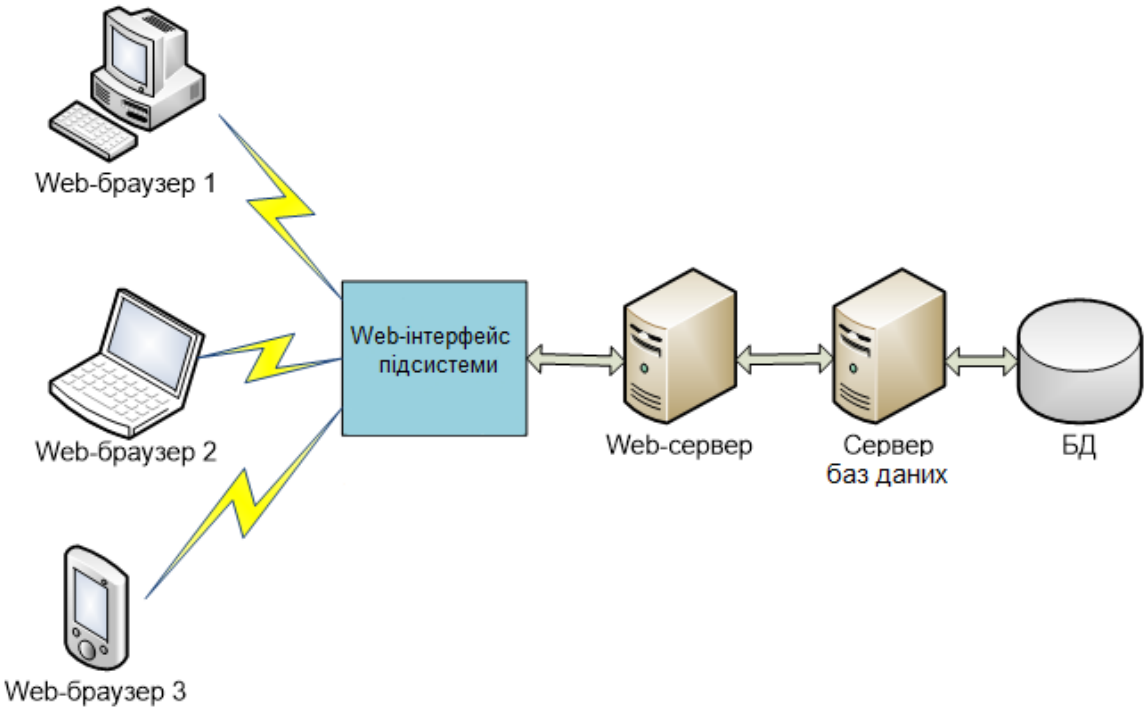

Рис. 2.2. Архітектура веб-сайту книжкового магазину

Логіка роботи застосунку зосереджена на сервері. Для обміну даними між Web-сервером і терміналами користувачів використовується протокол https. Вікно Web-браузера служить для введення даних від користувача, їх валідації, відправки на сервер і відображення текстових даних, які прийшли у відповідь.

Для зберігання даних про клієнтів інтернет-магазину використовується реляційна база даних, що дозволяє економити місце на жорсткому диску, знизити витрати на запис даних, полегшити маніпулювання окремими блоками даними і логічно розбити на частини все безліч інформації.

На рис.2.3 представлена фізична модель бази даних книжкового магазину, яка складається з 5 таблиць, наведених до 3-й нормальної форми:

- Book: дана таблиця представляє книгу, яка є в книжковому магазині. Вона зберігає інформацію про назву книги, автора, видавця, жанр, ціну, кількість екземплярів, що доступна у магазині.
- Customer: відображає клієнта, який здійснює покупку у книжковому магазині. Вона зберігає інформацію про ім'я клієнта, прізвище, електронну пошту, телефон та адресу.

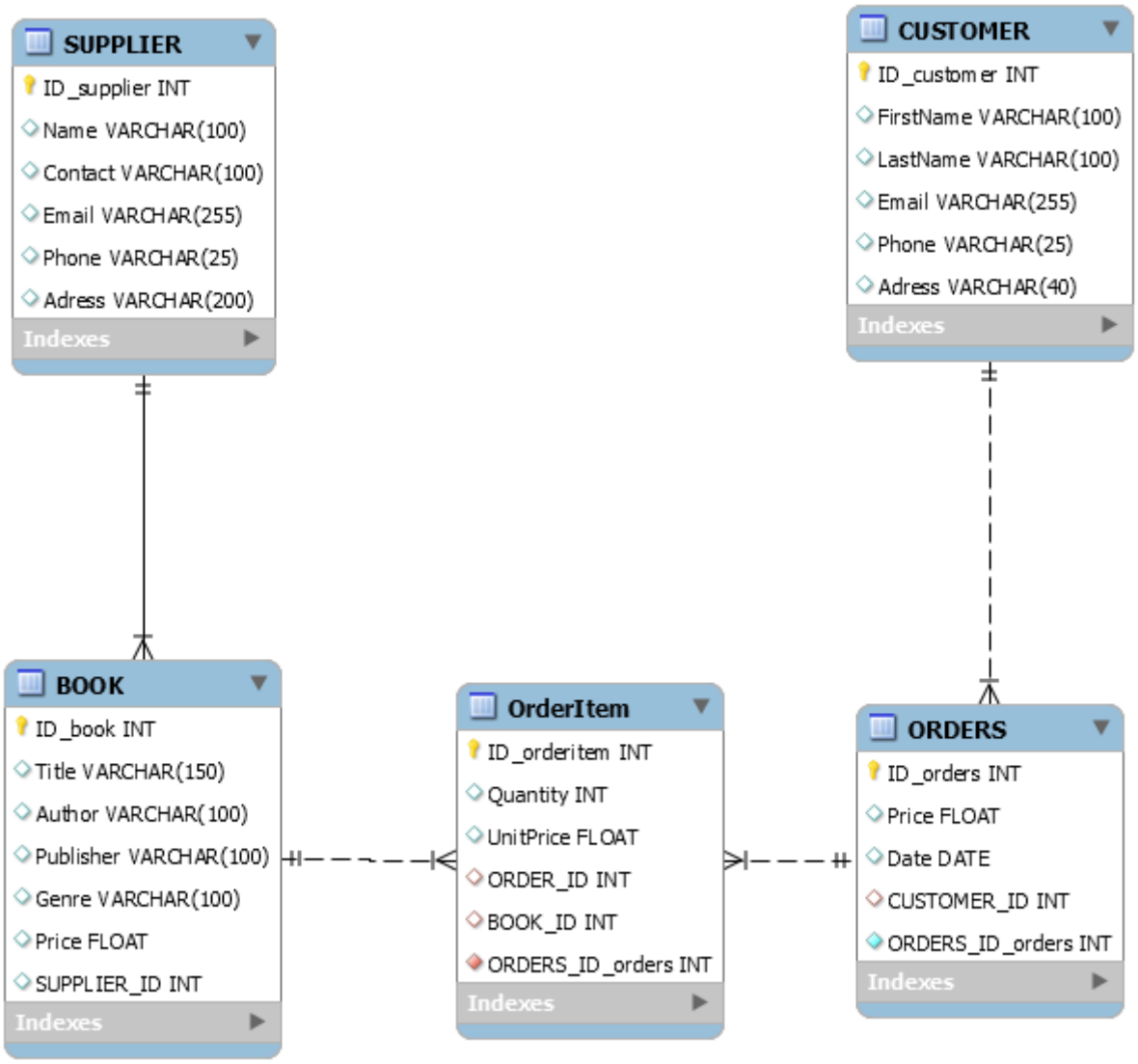

Рис. 2.3. Фізична ER-діаграма бази даних веб-сайту з продажу книжок

- Supplier: представляє постачальника, який постачає книги в книжковий магазин. Вона зберігає інформацію про назву постачальника, контактну особу, електронну пошту, телефон та адресу.
- Order: дана таблиця відображає замовлення, зроблене клієнтом у магазині. Вона зберігає інформацію про номер замовлення, ідентифікатор клієнта, дату замовлення та загальну суму.
- OrderItem: зберігає окремий елемент замовлення, який містить конкретну книгу, що була обрана. Вона зберігає інформацію про номер елемента замовлення, ідентифікатор замовлення, ідентифікатор книги, кількість та ціну за одиницю.

Між таблицями бази дани встановлені наступні звязки:

- зв'язок «один до багатьох» між Book і OrderItem (одна книга може мати кілька OrderItem);
- зв'язок «один до багатьох» між клієнтом і замовленням (один клієнт може розміщувати декілька замовлень);
- зв'язок «один до багатьох» між постачальником і книгою (один постачальник може постачати кілька книг);
- зв'язок «багато до одного» між Замовленням та Клієнтом (багато Замовлень можуть належати одному Клієнту);
- зв'язок «багато до одного» між OrderItem та Order (багато OrderItem можуть належати одному Order).

Таблиця 2.1

Таблиця атрибутів сутностей

Сутність	Первинний ключ	Атрибути
BOOK	Унікальний ключ номер книги	Унікальний ключ номер книги Назва книги Автор книги Видавництво книги Жанр книги Ціна Постачальник
SUPPLIER	Унікальний ключ номер поставки	Унікальний ключ номер поставки Назва постачальника Контактна інформація постачальника Емейл постачальника Телефон постачальника Адреса постачальника
CUSTOMER	Унікальний ключ номер клієнта	Унікальний ключ номер клієнта Ім’я клієнта Прізвище клієнта Емейл клієнта Номер телефону клієнта Адреса клієнта

Сутність	Первинний ключ	Атрибути
ORDER	Унікальний ключ номер замовлення	Унікальний ключ номер замовлення Ціна замовлення Дата замовлення Унікальний ключ номер клієнта
OrderItem	Унікальний ключ номер одиниці замовлення	Унікальний ключ номер одиниці замовлення Номер замовлення Номер книги Кількість товару Ціна за одиницю Дата додавання одиниці замовлення

Таблиця 2.2

Таблиця Book (Книжки)

№ з/п	Найменування стовпців	Примітка	Тип	Розмір
1	ID_book	Унікальний ключ номер книги	INTEGER	
2	Title	Назва книги	VARCHAR	150
3	Author	Автор книги	VARCHAR	100
4	Publisher	Видавництво книги	VARCHAR	100
5	Genre	Жанр книги	VARCHAR	200
6	Price	Ціна книги	FLOAT	
7	SUPPLIED_ID	Унікальний ключ номер постачальника	INTEGER	

Таблиця 2.3

Таблиця Supplier (Постачальники)

№ з/п	Найменування стовпців	Примітка	Тип	Розмір
1	ID_supplier	Унікальний ключ номер поставки	INTEGER	

№ з/п	Найменування стовпців	Примітка	Тип	Розмір
2	Name	Назва постачальника	VARCHAR	100
3	Contact	Контактні дані постачальника	VARCHAR	100
4	Email	Емейл адреса постачальника	VARCHAR	255
5	Phone	Номер телефону постачальника	VARCHAR	25
6	Adress	Адреса постачальника	VARCHAR	200

Таблиця 2.4

Таблиця Customer (Замовник)

№ з/п	Найменування стовпців	Примітка	Тип	Розмір
1	ID_customer	Унікальний ключ номер замовника	INTEGER	
2	FirstName	Ім’я замовника	VARCHAR	100
3	LastName	Прізвище замовника	VARCHAR	100
4	Email	Емейл замовника	VARCHAR	255
5	Phone	Номер телефону замовника	VARCHAR	25
6	Adress	Адреса замовника	VARCHAR	40

Таблиця 2.5

Таблиця Order (Замовлення)

№ з/п	Найменування стовпців	Примітка	Тип	Розмір
1	ID_orders	Унікальний ключ номер замовлення	INTEGER	
2	Price	Ціна	FLOAT	
3	Date	Дата замовлення	DATE	
4	CUSTOMER_ID	Унікальний ключ номер замовника	INTEGER	

Таблиця 2.6

Таблиця OrderItem (Одиниця замовлення)

№ з/п	Найменування стовпців	Примітка	Тип	Розмір
1	ID_orderitem	Унікальний ключ номер одиниці замовлення	INTEGER	
2	Quantity	Кількість замовлених одиниць	INTEGER	
3	UnitPrice	Ціна за одиницю	FLOAT	
4	Date	Дата додавання одиниці замовлення	DATE	
5	ORDER_ID	Унікальний ключ номер замовлення	INTEGER	
6	BOOK_ID	Унікальний ключ номер книги	INTEGER	

Код веб-сайту розділений на окремі класи, виходячи з виконуваних ним функцій. Можна виділити наступні основні класи застосунку:

- site.php - клас, який регулює роботу головної сторінки магазину;
- login.php – виконує авторизацію користувачів сайту;
- product.php - клас, який відповідає за каталог товарів;
- add_item.php - клас, який записує обраний товар в таблицю OrderItem бази даних сайту;
- basket.php - реалізує сторінку «Кошик», де будуть відображатися вибрані користувачем товари;
- order.php - відправляє на електронну пошту клієнта повідомлення про оформлення замовлення.

Для реалізації окремого примірника Кошику для кожного відвідувача інтернет-магазину використовується механізм сесій. Коли покупець заходить на сайт інтернет-магазину, сервер посилає йому копію оригінальної сторінки сайту. Після того, як копія завантажиться на комп'ютер клієнта, зв'язок з сервером переривається до наступного запиту, і завдяки цьому кілька

відвідувачів можуть відкрити одночасно свій кошик, при цьому їх вміст буде відрізнятися один від одного.

Всі вибрані покупцями товари записуються в таблицю OrderItem. Для того, щоб визначити, кому належить який товар, кожному покупцеві надається унікальний номер, id, і цей же номер буде присвоюватися кожному обраному даними покупцем товару. Далі, коли клієнт відкриває свій кошик, то відкривається сторінка (шаблон) кошику, на якій знаходитися скрипт, що виконує запит до таблиці OrderItem і видобуває з неї усі товари, які мають номер id_orders, ідентичний номеру id покупця.

Файл: «site.php»

На початку файлу необхідно розмістити код лічильника відвідувачів, який буде вважати відвідувачів, і одночасно присвоювати їм унікальний номер. Для цього потрібно створити глобальну змінну, так звану сесію, з ім'ям «talon», в якій буде зберігатися значення показання лічильника.

Якщо відвідувач вперше заходить на головну сторінку, то сесія з ім'ям «talon» ($ _SESSION ['talon']) буде порожній. В цьому випадку спрацьовує запит до таблиці customer_entity, витягуючи з неї число відвідувачів, додає до нього «1» і вийшла результат записується в $ _SESSION ['talon'], після чого змінюються дані самого лічильника. Ця сесія буде прив'язана до даного покупцеві весь час, поки він знаходиться на сайті, зберігаючись на деякий час і після його виходу.

Якщо користувач перезавантажить сторінку, то код лічильника перевірить наявність значення $ _SESSION ['talon'] - якщо воно не порожнє, він не спрацює і не запише даного користувача як нового користувача.

Файл: «product.php»

У цьому файлі також встановлено код лічильника, тому що відвідувач може зайти на цю сторінку через пошукову систему, минаючи головну сторінку, тому бажано встановлювати цей код на початку кожної сторінки, яка доступна для індексації пошуковими системами.

Далі представлений код, який відправляє обраний товар на обробку файлу «add_product.php»:

<form action="add_product.php?idproduct=19" method="POST">

<input type="submit" value="До кошику" onClick="alert ('Товар додано до кошику!')">

</form>

Дані передаються з одній сторінці на іншу двома способами: method = "POST" і method = "GET". Відрізняються вони тим, що method = "GET" передає дані через адресний рядок, а method = "POST" через форму «form», іншими словами GET передає дані прив'язані до посилання, а POST передає дані, які знаходиться в формі (де треба заповнювати поля) і тільки після натискання кнопки.

<Form action = "add_product.php? idproduct = 19" method = "POST"> - вказує, на яку сторінку будуть відправлені дані при натисканні кнопки саме цієї форми.

<Input type = "submit" value = "До кошику" onClick = "alert ('Товар доданий до кошику!')"> - це код самої кнопки, а onClick = "alert ('рядок')" викликає спливаючі вікно з підказкою (в даному випадку « 'Товар доданий до кошику!'»), вона спрацює кожен раз, коли буде натиснута кнопка.

Файл: «add_product.php»

У цьому файлі всі необхідні дані записуються в таблицю OrderIitem.

$ talon = $ _SESSION ['talon']; - значення глобальної змінної «$ _SESSION ['talon']» записується в змінну «$ talon»

$ data = date ('y / n / d'); - створюється змінна «$ data» і заповнюється значенням поточної дати у форматі ('рік / місяць / день')

$ idproduct = $ _GET ['idproduct']; - в змінну «$ idproduct» заноситься значення, отримане методом «GET» з ім'ям «idproduct»

$ conn2 = mysql_connect ("localhost", "Ім'я користувача", "Пароль користувача") or die ("Could not connect:". Mysql_error ());

mysql_select_db ("cortini", $ conn2); - рядок підключення до бази даних.

if (! $ update) - якщо змінна «$ update» не створена, то:

$ sqlInsert = mysql_query ("INSERT INTO OrderItem (order_id, book_id, quantity, created_at) values ('$ talon', '$ idproduct', 1, '$ data')", $ conn2); - вставка рядка в таблицю OrderItem

else

$sqlUpdate = mysql_query("UPDATE OrderItem SET quantity = quantity+ 1 WHERE order_id ='$talon' AND book_id = '$idproduct'", $conn2); - створюється змінна, яка змінить дані в таблиці OrderItem

if (! $ sqlUpdate) {echo "Запит не пройшов! Спробуйте ще раз.";} -якщо змінна «$ sqlUpdate» не створена, то виводиться текст помилки.

mysql_close ($ conn2); - закриття з'єднання з базою даних.

header ("Location: product.php"); - якщо все пройшло успішно, то відбувається автоматичний перехід на сторінку «product.php».

Все це відбувається миттєво, користувач помітить тільки те, що перезавантажиться сторінка «product.php». Про те, що був перехід на сторінку «add_product.php» і назад, свідчить спливаюче вікно з підказкою «Товар додано до кошику!»

Файл: «basket.php»

Ця сторінка містить інформацію товари, які вибрав покупець. Також на цій сторінці є скрипт, який буде змінювати кількість товару або видаляти товар за бажанням покупця.

Тут також знаходиться форма, до якої покупець внесе свої дані, і при натисканні кнопки «Відправити» ці дані передаються на сторінку «order.php».

Файл: «order.php»

На цю сторінку покупець потрапить тільки в тому випадку, якщо його кошик не порожній, і він заповнив форму замовлення. Тут же відбувається відправка листа на електронну пошту з даними, які відвідувач ввів в форму замовлення.

session_start (); - запуск сесії.

if ($ _SESSION ['talon'] == "" or $ _POST ["ПІБ"] == "") - перевірка: якщо сесія з ім'ям «talon» порожня, або поле з ім'ям «ПІБ» порожнє, то виконується код, записаний в фігурних дужках.

exit ("<body> <h1> Форма заповнена некоректно! </ h1> </ div> </ body>"); - на екран висвітиться напис «Форма заповнена некоректно!» І код читатися далі не буде.

$mail=$_POST["mail"];

$pib=$_POST["ПІБ"];

$tel=$_POST["телефон"];

$adres=$_POST["adres"];

$summa=$_GET["suma"];

$talon=$_SESSION['talon']; - в змінні заносяться значення, отримані методом «POST» від форми і методом «GET» через адресний рядок.

echo "ПІБ: ". $pib."
";

echo "Адреса: ".$adres."
";

echo "Телефон: ".$tel."
";

echo "e-mail: ".$mail."
";

echo "Вартість замовлення: ".$suma ." p
";

echo "Номер замовлення: ".$talon."
";

echo "Дата: ".date('y/n/d').""; - виведення даних на екран.

$to = 'kruchok_dm_vik@gmail.com'; - створюється змінна «$ to», в яку записується адреса електронної пошти.

$subject = $pib;

$message = "Номер сесії: ".$talon."
".

"ПІБ: ".$pib."
".

"Вартість замовлення: ".$suma." p
".

"Адреса: ".$adres."
".

"Телефон: ".$tel."
".

"e-mail: ".$mail."
".

"Дата: ".date('d/n/y')."
".

"Коментарі:". $com; - створюється змінна «$ message», в яку записуються ті змінні з даними, які формують тексту листа.

$headers = "content-Type: text / html; charset = utf-8 \ r \ n"; - вказівка кодування листи.

mail ($ to, $ subject, $ message, $ headers); - код, який відправляє лист на вказану адресу від зазначеного суб'єкта з текстом у зазначеній кодуванні.

session_destroy (); - знищення сесії. Виконується для очищення кошика покупця.

2.5. Обгрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані шляхом введення користувачем інформації з клавіатури, та через передачу значень через глобальні змінні застосунку інтернет-магазина.

Вхідні дані:

- унікальний ідентифікатор користувача та пароль (для зареєстрованих користувачів);
- ім'я та прізвище клієнта;
- контактні дані клієнта (адреса електронної пошти, телефон);
- найменування способу оплати замовлення;
- найменування способу доставки замовлення;
- текстові повідомлення користувача;
- обліковий номер товару;
- текстовий опис книги;
- зображення обкладинки книги.

Вихідні дані:

- найменування, ціна і кількість товарів, доданих до кошика;
- загальна вартість замовлення;

- заповнена форма замовлення;
- номер і статус поточного замовлення;
- текстові повідомлення служби клієнтської підтримки магазину;
- поточний баланс коштів на бонусному рахунку клієнта;
- історія замовлень.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Для серверних технічних засобів рекомендована конфігурація, що забезпечує цілодобову роботу програми з резервуванням даних:

- ✓ процесор класу Intel ® Xeon з тактовою частотою 2.4GHz;
- ✓ шина даних - 1066MHz,
- ✓ кеш другого рівня - 2048 КБ;
- ✓ оперативна пам’ять 2 x DIMM DDR2-800 1024 Мб;
- ✓ жорсткі диски 3x 250 Гб SATA 2 16 Мб буфер, 7200 RPM;
- ✓ рідкокристалічний монітор з діагоналлю не менше 17 ";
- ✓ доступ до мережі Internet;
- ✓ клавіатура;
- ✓ маніпулятор "миша".

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки, висунутими замовником.

2.6.2. Використані програмні засоби

Проект реалізований на мові програмування PHP 5, в якості СУБД була обрана MySQL - реляційна СУБД з вільною ліцензією. Також для коректної

роботи розробленого додатка необхідний довільний веб-браузер з підтримкою технології JavaScript.

Необхідні програмні засоби:

Для клієнтського комп'ютера:

- ✓ ОС сімейства Windows (7,8,10), Linux;
- ✓ веб-браузер Google Chrome / Firefox / Opera / Microsoft Edge.

Для сервера:

- ✓ серверна ОС сімейства Microsoft Windows, Linux, FreeBSD;
- ✓ СУБД MySQL;
- ✓ мова програмування PHP 5.

2.6.3. Виклик та завантаження програми

Розроблений веб-сайт використовується для інтернет-магазину Книгосклад, його завантаження виконується за допомогою веб-браузера з підтримкою JavaScript. Для виклику застосунку в рядку запиту веб-браузера необхідно ввести адресу інтернет-магазину https://knigosklad.com.ua/.

2.6.4. Опис інтерфейсу користувача

Головна сторінка сайту інтернет-магазину Книгосклад починається із «шапки» сайту, що містить (зліва направо) пошуковий рядок, логотип, назву, контактні дані, піктограми кошику та особистого кабінету клієнта (рис. 2.4). Центральну частину головної сторінки займає слайдер зображень, на яких демонструються оголошення щодо акційних пропозицій, актуальних товарів та новинок, ліворуч від нього розмістилася панель книжкового каталогу.

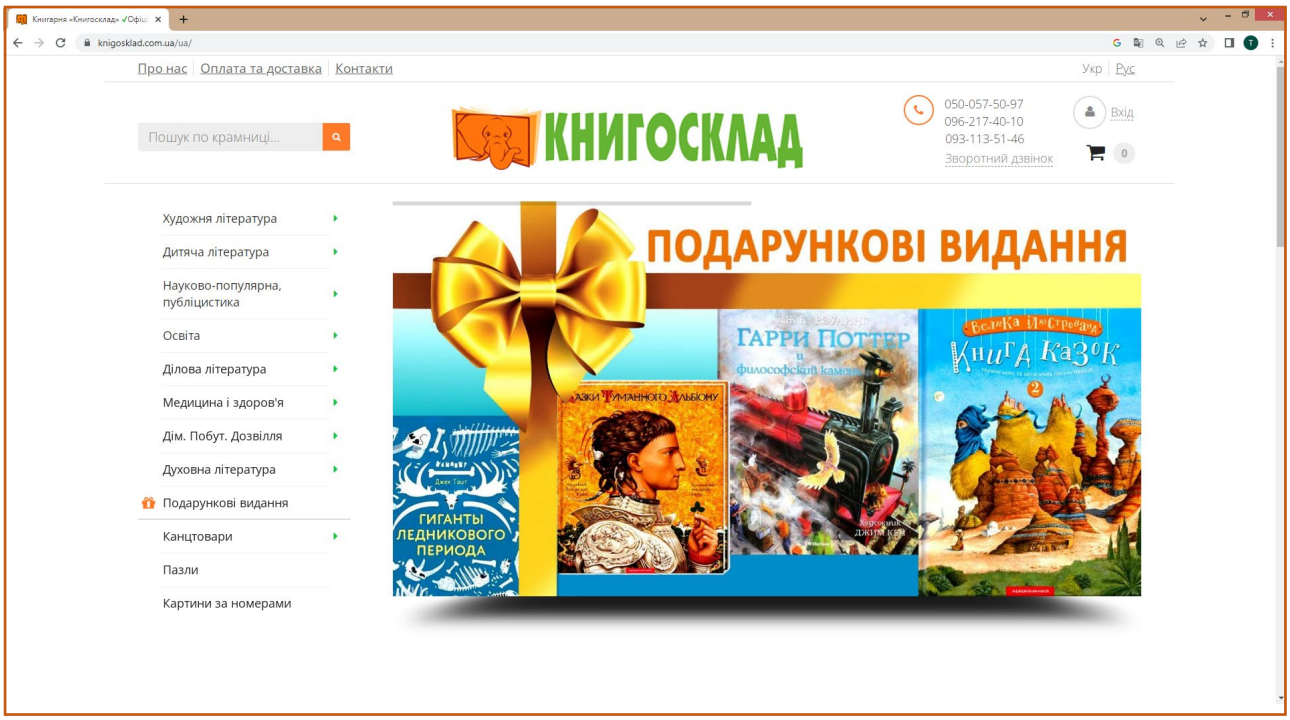

Рис. 2.4. Головна сторінка інтернет-магазину Книгосклад

Нижче знаходяться плиткові каталоги бестселлерів (рис. 2.5) та популярних категорій (рис 2.6). Завершує головну сторінку розділ новин та футер, що містить контактну інформацію про магазин, відомості про способи доставки та оплати.

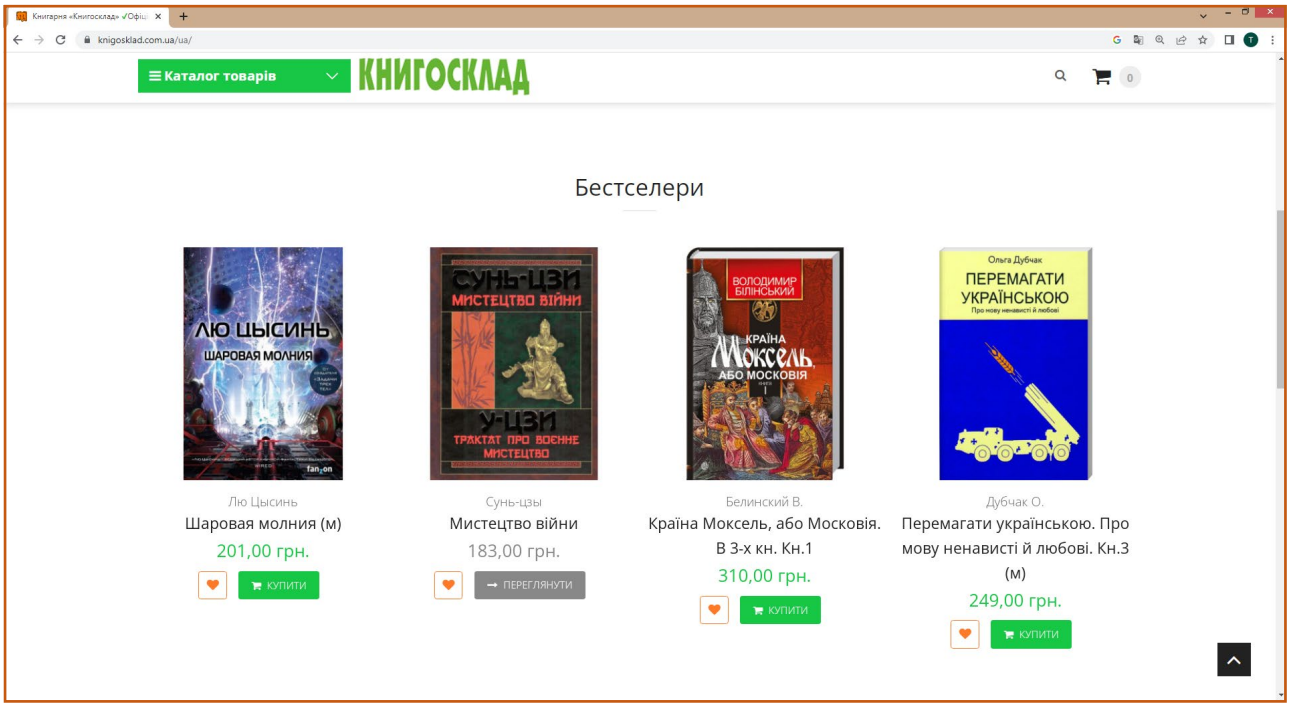

Рис. 2.5. Каталог бестселерів

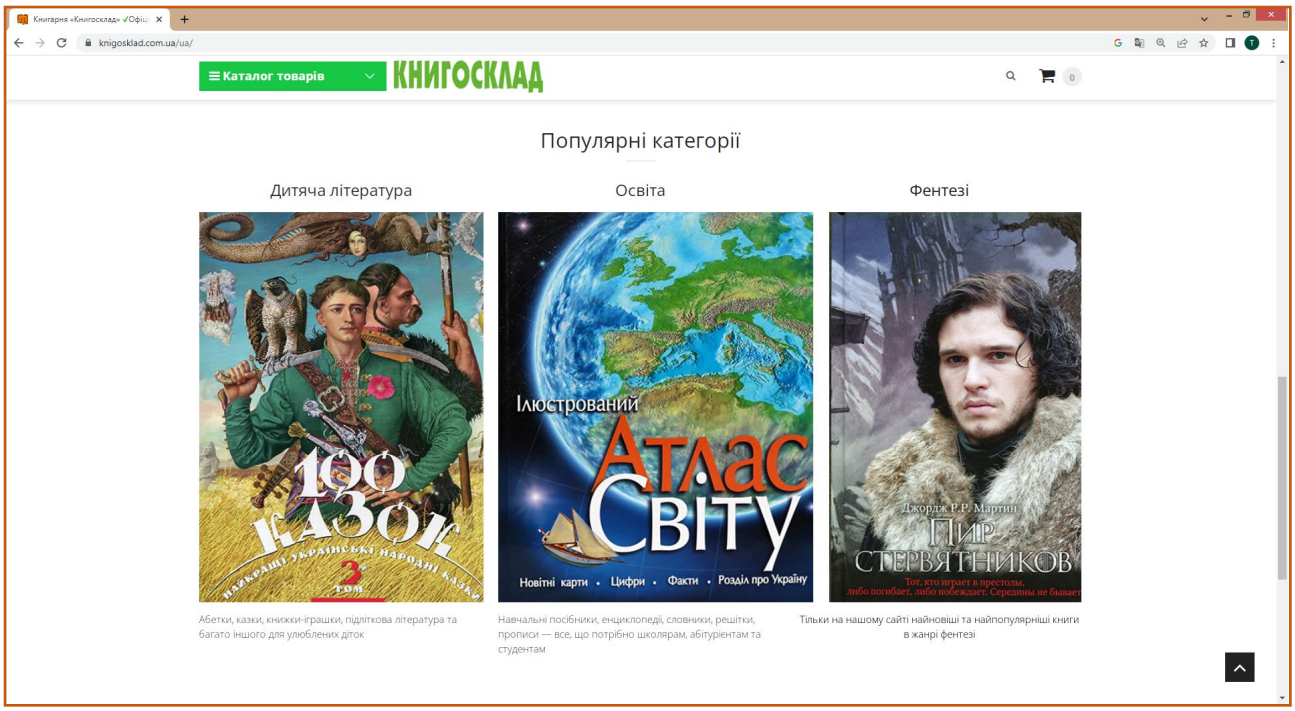

Рис. 2.6. Популярні категорії

Для здійснення навігації по каталогу товарів можна скористатися панеллю фільтрації, яка розміщена у лівій частині сторінки каталогу (рис. 2.7). Доступні такі критерії фільтрації, як категорія книг, ціна, та мова книги.

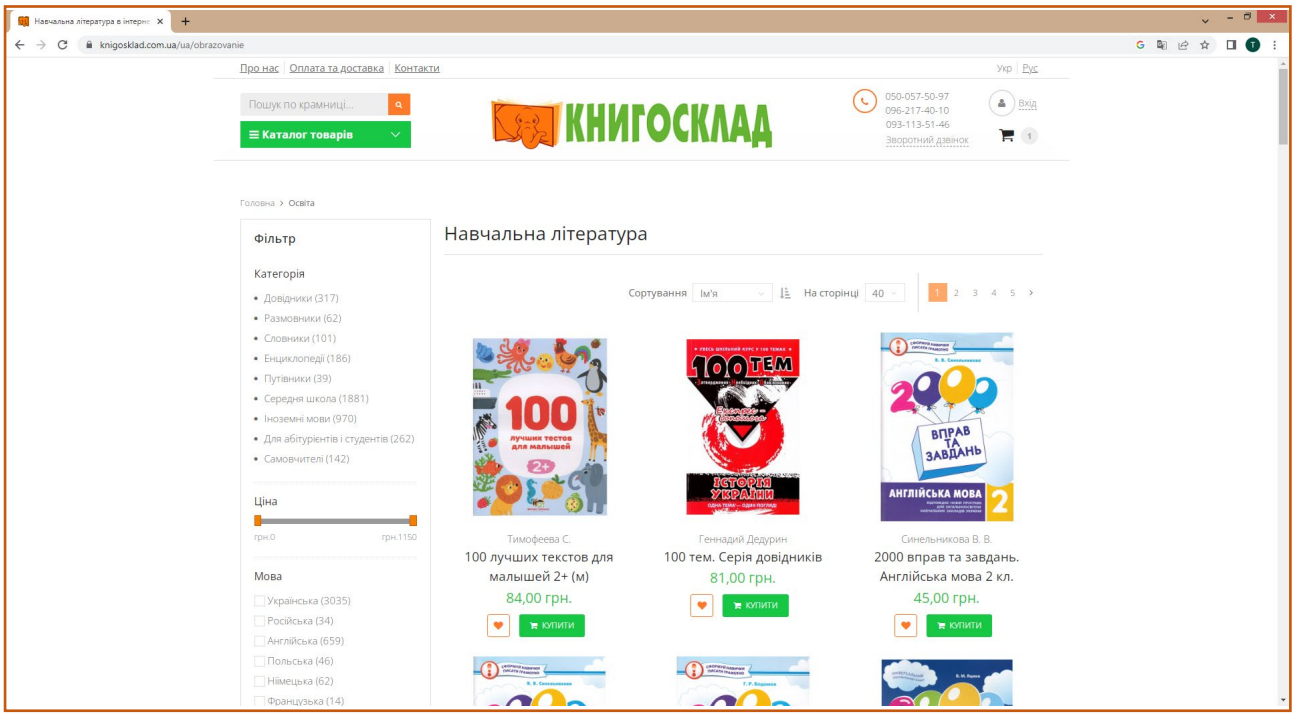

Рис. 2.7. Фільтрація товарів

Для налаштування способу відображення товарних позицій на сторінках каталогу клієнт може використати інструменти сортування. Панель сортування знаходиться у верхньому правому куті плиточного каталогу (рис. 2.8), і складається з випадаючого меню для вибору критерію сортування, піктограми встановлення порядку сортування, та позначення числа товарних позицій, що одночасно відображаються на сторінці.

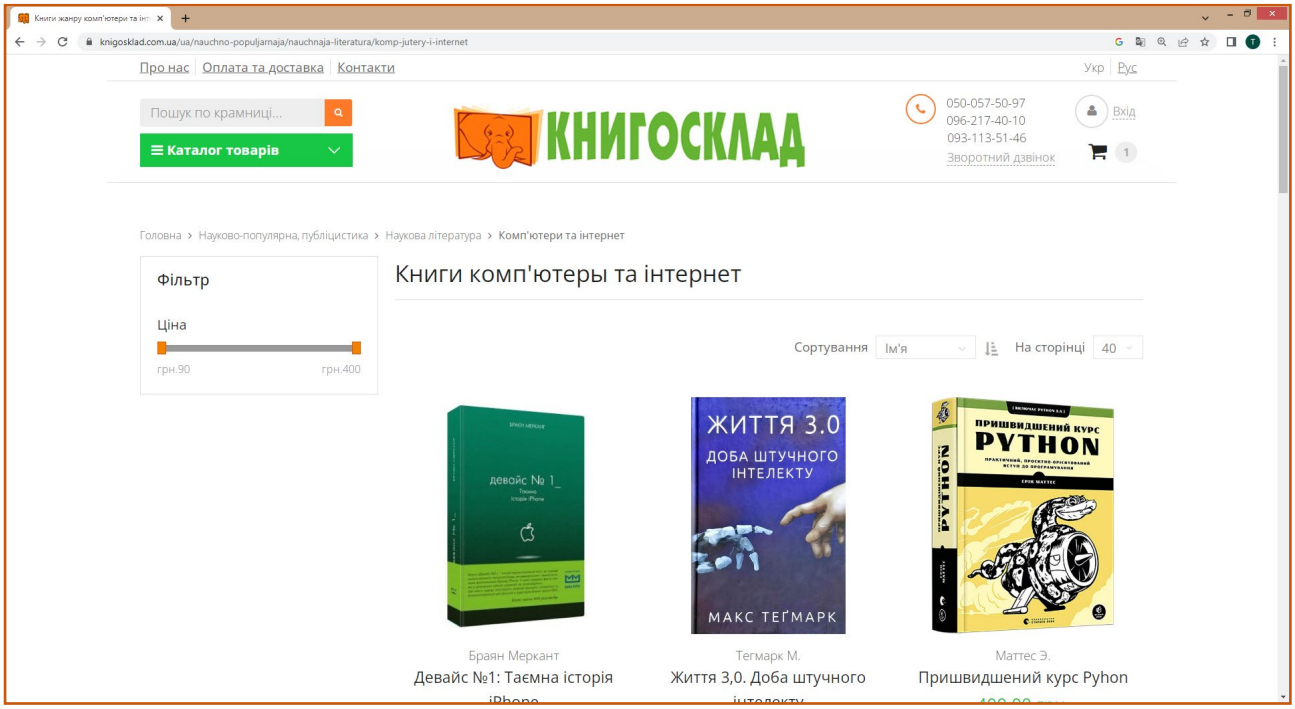

Рис. 2.8. Інструменти сортування

Зарєєстровані користувачі інтернет-магазину можуть здійснити вхід у свій акаунт, натиснувши на піктограму для входу, яка знаходиться у правому верхньому кутку хедера сайту. Після цього на екрані з’явиться панель авторизації, де необхідно ввести облікові дані (адресу електроної пошти та пароль), і підтвердити вхід натисканням на кнопку «Вхід» (рис. 2.9). Також відвідувачі сайту можуть авторизуватися за допомогою облікового запису Google (за бажанням).

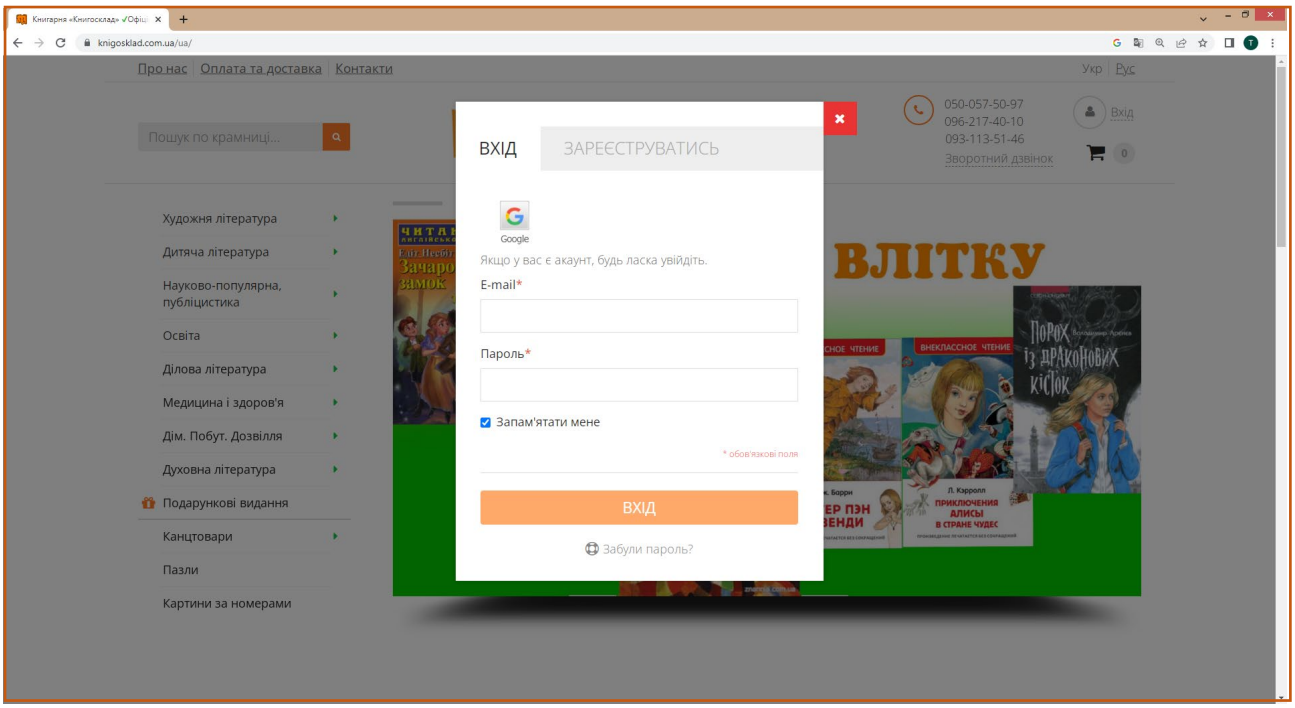

Рис. 2.9. Панель авторизації

Якщо користувач ще не має облікового запису, він може пройти процедуру реєстрації, натиснувши на вкладку «Зареєструватися» на панелі авторизації (рис. 2.10). У відповідні поля на формі необхідно ввести ім'я, прізвище, адресу електроної пошти та пароль (двічі). Також можливо підписатися на емейл-розсилку від магазину, та зберігти введені облікові дані, встановивши відмітки у відповідні чек-бокси на формі реєстрації. Для відправки введеної інформації потрібно натиснути на кнопку «Зареєструватися» у нижній частині форми.

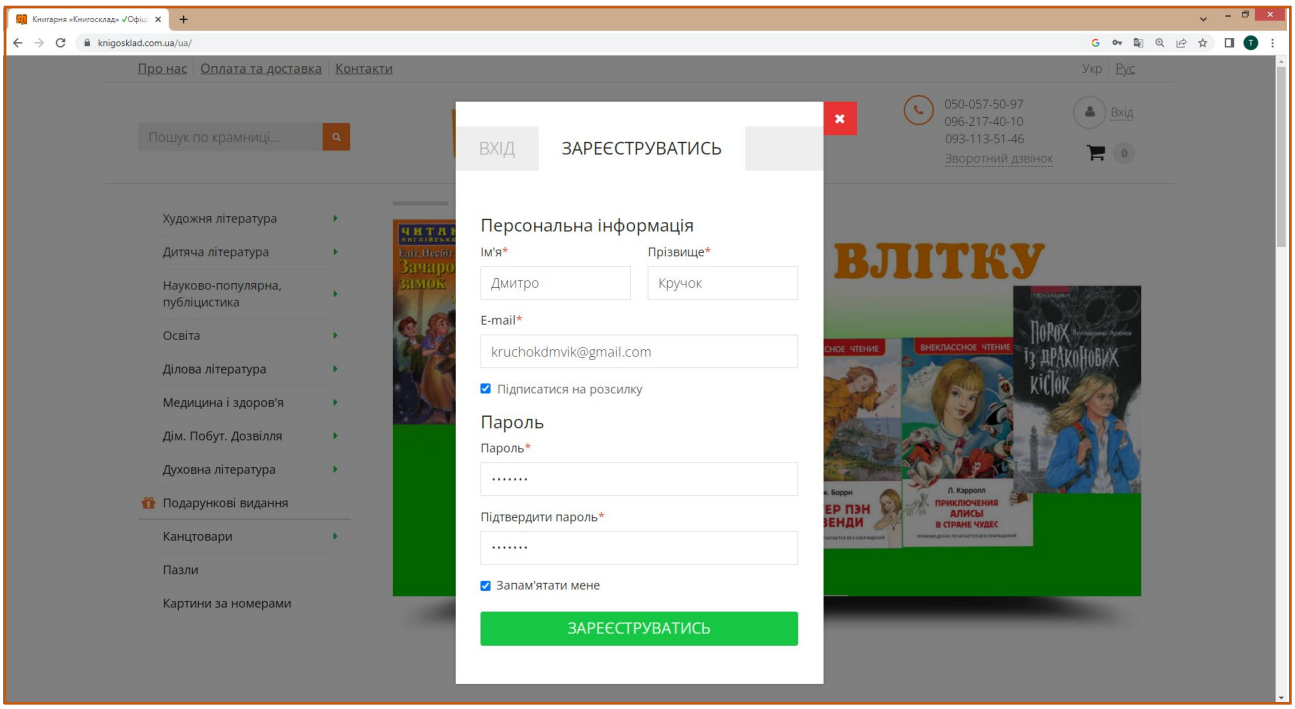

Рис. 2.10. Реєстрація нового користувача

Для здійснення замовлення необхідно увійти до кошику, клікнувши на відповідне символічне зображення, яке знаходиться в правому верхньому кутку шапки сайту інтернет-магазину. Поряд із зображення кошику знаходиться лічильник, який відображає кількість внесених то кошик товарів. Якщо клієнт за час сесії ще не обрав щодного товару, поточне значення лічильника встановлено «0» (рис. 2.11).

Щоб додати товар до кошика, необхідно перейти на сторінку з описом даного товару, і натиснути на кнопку «Купити», розташовану під полем із зазначенням ціни. У разі успішного виконання цієї дії на екрані з’явится модальне вікно з повідомленням, що підтверджує додавання товарної позиції до замовлення клієнта (рис. 2.12).

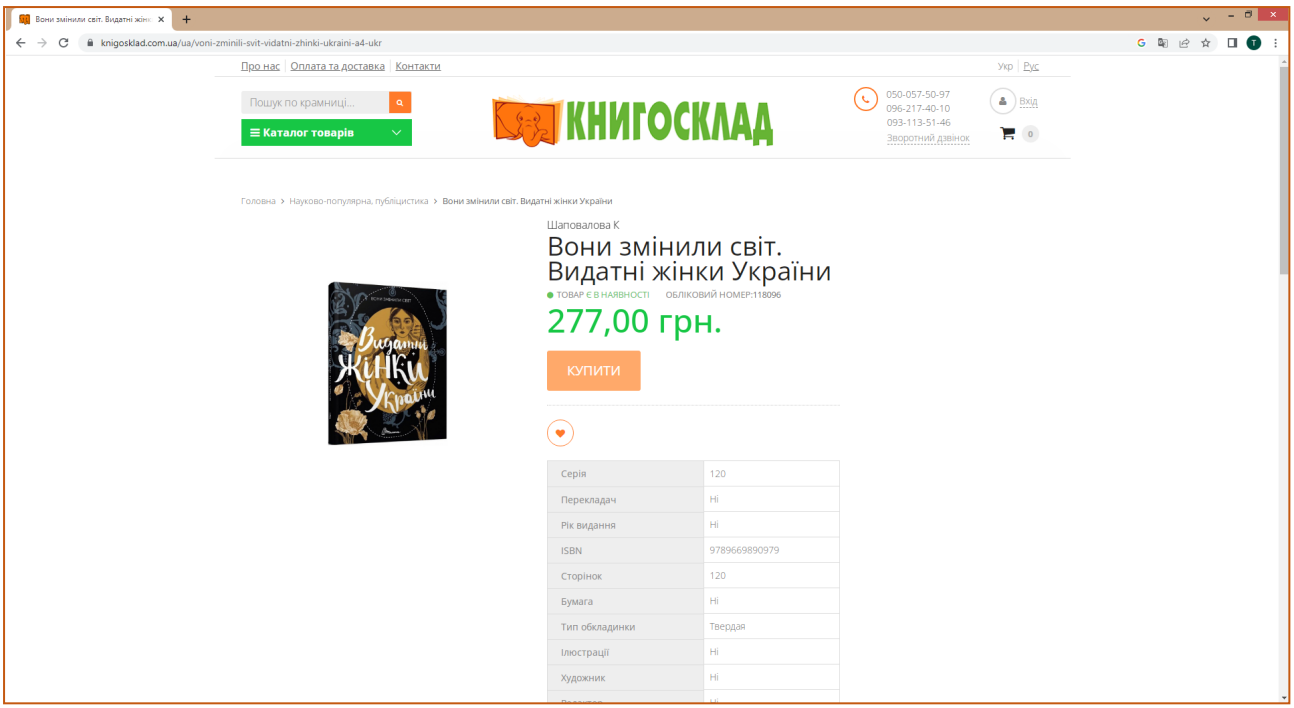

Рис. 2.11. Сторінка товару

Після того, як до кошика буде внесено хоча б один товар, значення лічильника вмісту збільшується на відповідне значення, і у користувача підсистеми з’являється доступ до функції швидкого перегляду вмісту кошику.

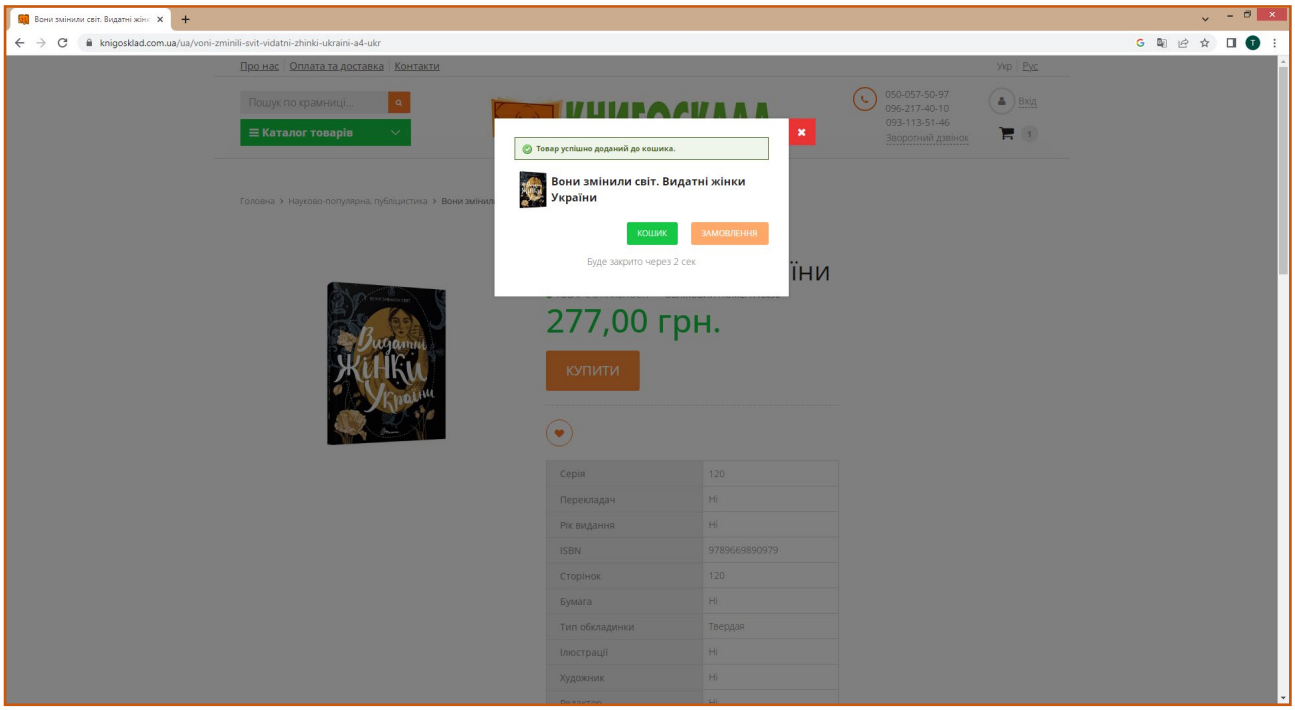

Рис. 2.12. Повідомлення про успішне додавання до кошику

При наведені курсору миші на символ кошику на екран виводиться вікно зі стислим переліком обраних товарів, їх зображеннями, кількістю та цінами, та загальною сумою замовлення (рис. 2.13). Також клієнт може вилучити товари, натиснувши на кнопку «х», або перейти безпосередньо на сторінку підсистеми формування замовлення, через кнопки «Переглянути кошик», або «Оформити замовлення».

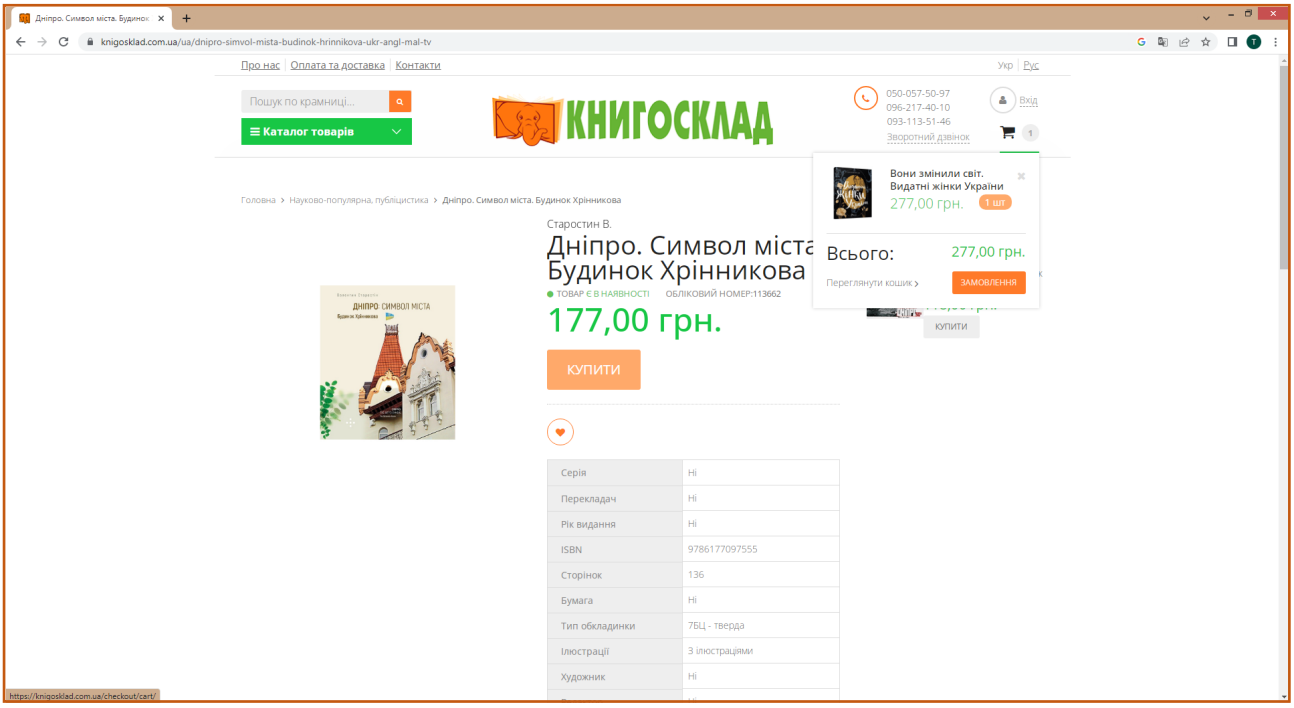

Рис. 2.13. Швидкий перегляд вмісту кошика

На сторінці «Кошик» знаходиться перелік обраних покупцем товарів, включаючи їх найменування, зображення, ціну, кількість одиниць кожної позиції і загальна вартість покупки (рис.2.14).

На цій сторінці користувач може виконати наступні дії: повернутися до вибору товарів (натиснувши на відповідне посилання у лівому нижньому кутку сторінки), вилучити окремий товар із кошика, змінити кількість обраних товарних позицій, видалити весь вміст кошика (кнопка у правому нижньому кутку), та перейти до оформлення замовлення, натиснувши на кнопку «Оформити замовлення», розташовану під сумою покупки. Після цього клієнта буде перенаправлено на сторінку оформлення замовлення (рис. 2.15).

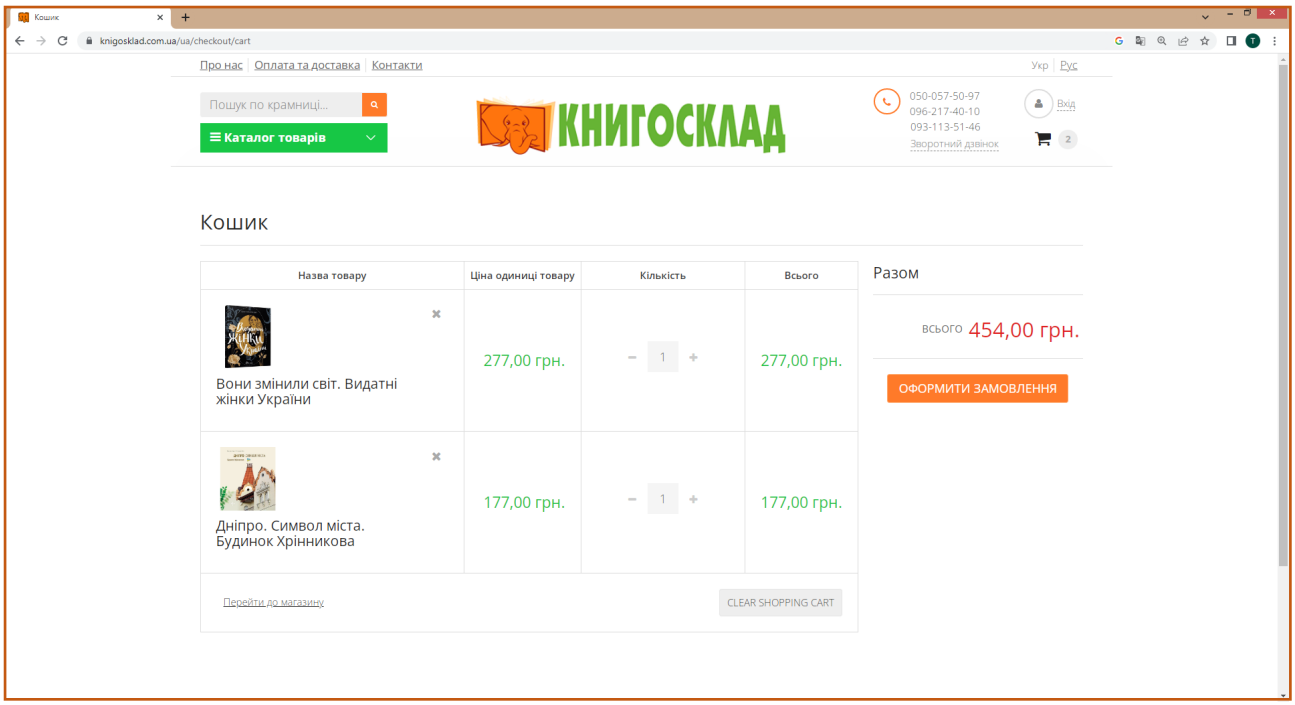

Рис. 2.14. Перегляд вмісту кошика

Процедура оформлення замовлення включає, в першу чергу, внесення особистих даних, контактного телефону та адреси електронної пошти покупця. Слід зазначити, що дані дії є обов'язковими для незареєстрованих користувачів; для постійних клієнтів форма «Ім’я та адреса» заповнюється автоматично відповідно до особистими даними клієнта, введеними їм при реєстрації на сайті.

Потім користувачеві необхідно вказати спосіб доставки і оплати замовлення, скориставшись запропонованим переліком варіантів надання даних послуг, ввести промо-код (за наявності) та додати коментар до замовлення.

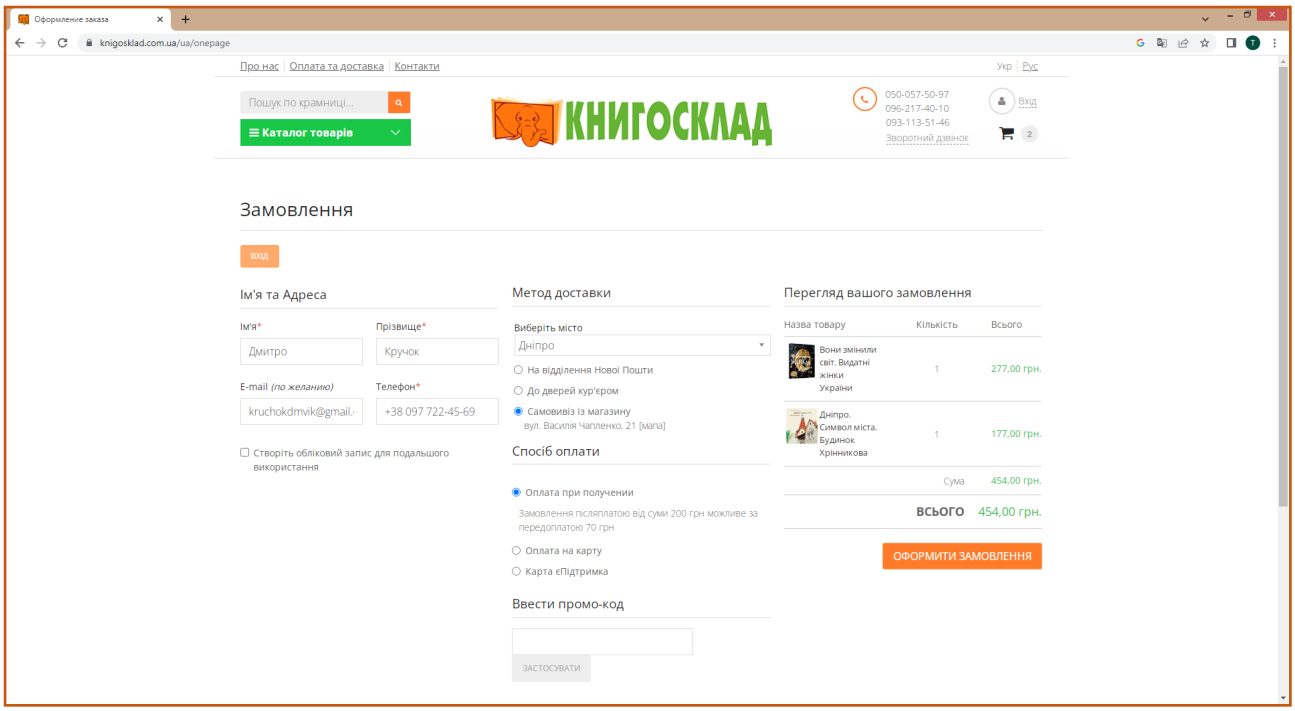

Рис. 2.15. Внесення даних до форми замовлення

У разі, якщо клієнт не додав до кошику жодного товару, видалив всі товари із кошика за допомогою інструменту видалення товару (кнопка «х»), або скористався функцією очищення кошику, сторінка «Кошик» буде містити лише повідомлення «Ваш кошик порожній», та посилання на перехід до каталогу інтернет-магазину.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1200;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,05;
4. годинна заробітна плата програміста – 80 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,2;
7. вартість машино-години ЕОМ – 20 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1200);

C - коефіцієнт складності програми (1,6);

p - коефіцієнт корекції програми в ході її розробки (0,05).

Звідси умовне число операторів в програмі:

$$Q = 1{,}6 \cdot 1200 \cdot (1 + 0{,}05) = 2016$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.2). З урахуванням коефіцієнта кваліфікації k = 1,2, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (2016 \cdot 1{,}2) / (75 \cdot 1{,}2) = 26{,}88 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 2016 / (20 \cdot 1{,}2) = 84 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 2016 / (25 \cdot 1{,}2) = 67{,}2 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 2016 / (5 \cdot 1{,}2) = 336 \text{ чел.-ч.}$$

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^k_{отл} = 1{,}5 \cdot 336 = 504 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}\ ,\ \text{людино-годин},$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k}\ ,\ \text{людино-годин},$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}\ ,\ \text{людино-годин}.$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 2016 / (18 · 1,2) = 93,33 людино-годин.

$t_{до}$ = 0,75 · 93,33 = 70 людино-годин.

$t_д$ = 93,33 + 70 = 163,33 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 26,88 + 84 + 67,2 + 336 + 163,33 = 727,41 людино-годин.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 80 грн / год, отримуємо:

$$З_{ЗП} = 727{,}41 \cdot 80 = 58\ 192{,}8 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (20 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 336 \cdot 20 = 6720 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 58192{,}8 + 6720 = 64\,912{,}8 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 727{,}41 / 1 \cdot 176 \approx 4 \text{ міс.}$$

Висновок

Програмне забезпечення призначене для забезпечення функціонування інтерактивного веб-сайту з продажу книг. Вартість даного програмного забезпечення становить 64,9 тис. грн. і не вимагає додаткових витрат при розробці та супроводженні програми. Очікуваний час розробки становить 4 місяці. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

В даній кваліфікаційній роботі був розроблений інтерактивний веб-сайт з продажу книг.

Це програмне забезпечення призначене для автоматизації обслуговування покупців книжкового інтернет-магазину. Практичне призначення даного застосунку полягає в забезпеченні зручного перегляду галереї товарів, та простого і інтуїтивно зрозумілого механізму формування замовлення. Інформаційна система забезпечить магазину більш точне та актуальне управління запасами книг. Завдяки цьому буде зменшено втрати через невірне розміщення товарів або недооцінку попиту. Також застосунок дозволить ефективно відстежувати рух товарів від постачальників до кінцевих клієнтів, що сприятиме покращенню логістики та скороченню часу доставки. Все це зробить процес покупки швидшим і зручнішим, і підвищить конкурентоспроможність та ефективність роботи книжкового магазину.

Під час виконання даної кваліфікаційної були виконані наступні задачі:

- вивчено предметну область розв'язуваної задачі;

- створено алгоритм для реалізації поставленого завдання;

- створено базу даних і веб-орієнтований клієнтський застосунок, що працює з нею.

Розроблене програмне забезпечення дозволяє:

- зручне візуальне представлення асортименту книжкового магазину;
- легку та швидку навігацію по каталогу товарів;
- реєстрацію користувачів;
- додавання товарів до кошика;
- перевірку поточного стану кошика;
- оформлення замовлення (включаючи розрахунок загальної суми покупки, вибір способу оплати і доставки);
- перевірка статусу замовлення;
- інформування співробітників магазину про надходження замовлення.

Програма реалізована на базі фреймворка Node.js з використанням мови програмування PHP і СУБД MySQL.

Також у кваліфікаційній роботі було визначено трудомісткість розробленого застосунку (727 люд-год), проведений підрахунок вартості роботи по створенню програми (64 913 грн) та розраховано час її створення (4 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Балик Н.Р., Мандзюк В.І. Бази даних MySQL: Навчальний посібник. – Тернопіль: Навчальна книга – Богдан, 2010.
2. Бернерс-Лі Заснування павутини: З чого починалася і до чого прийде Всесвітня мережа / Тім Бернерс-Лі разом з Марком Фічетті; пер. з англ. А. Іщенка. – Київ: Києво-Могилянська академія, 2007. – 208с.
3. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К.: Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України).
4. Бородкіна І. Інформаційний пошук в мережі Інтернет / І. Бородкіна - К.: Центр учбової літератури, 2011. – 342 с.
5. Гайна Г.А. Основи проектування баз даних: Навчальний посібник / Г.А. Гайна. − К.: КНУБА, 2005.
6. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.
7. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. — Київ: Держстандарт України, 1994. – 88 с.
8. Завадський І.О. Основи баз даних: [Навч. посіб.] / І.О. Завадський. – К.: Видавець І.О. Завадський, 2011. – 192 с.
9. Зарицька О.Л. Бази даних та інформаційні системи: [Метод. пос.] /О.Л. Зарицька. – Житомир : Вид-во ЖДУ ім. І. Франка, 2009. – 132 с., ил.
10. Кривий С. Вступ до методів творення програмних продуктів. – Чернівці: Букрек, 2012. – 424 с.
11. Куленко М.Я. Основи графічного дизайну: підручник для студентів вищих навч. закладів / Михайло Куленко; МОНУ; Київський нац. ун-т

будівництва і архітектури. – 2-ге вид., виправл. та доп. – Київ : Кондор, 2007. – 492с.

12. Матвієнко О., Бородкіна І. Internet-технології: проектування Web-сторінки: навч. Пос. / О. Матвієнко, І. Бородкіна. – К.: Альтерпрес, 2003. – 132 с.

13. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп'ютерні системи» / О.Г. Вагонова, О.Б. Нікітіна, Н.Н. Романюк; М-во освіти і науки України, ДВНЗ «Нац. гірн. ун-т». – Д.: НГУ, 2013. – 11 с.

14. Методичні рекомендації до виконання кваліфікаційних робіт здобувачів першого рівня вищої освіти спеціальності 122 Комп'ютерні науки/ В.В. Спірінцев, П.О. Іщук, О.С. Шевцова; Д : НТУ «Дніпровська політехніка», 2021. – 59 с.

15. Пасічник В. В. Організація баз даних та знань / В.В. Пасічник, В.А. Резніченко. – К.: Видавнича група BHV, 2006. – 384 с.

16. Перевозчикова О. Інформаційні системи і структури даних. – К.: Києво-Могилянська академія, 2007. – 288 с.

17. Романюк О.Н. Організація баз даних та знань / О.Н. Романюк, Т.О. Савчук. – Вінниця: ВДТУ, 2001.

18. Оператори SQL [Електронний ресурс]. URL: https://www.tutorialspoint.com/sql/sql–opera–tors.htm (дата звернення: 18.05.2023).

19. Страхарчук А.Я. Реляційна модель даних, Нормалізація даних, Інформаційні системи і технології в банках / А.Я. Страхарчук, В.П. Страхарчук // Бібліотека українських підручників. [Електронний ресурс]. URL: http://libfree.com/134926096_bankivska_spravarelyatsiyna_model_danih.html. (дата звернення: 18.05.2023).

20. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги: ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).

21. Цеслів О.В. WEB-програмування : навч. посібнник / О.В. Цеслів ; М-во освіти і науки, молоді та спорту України, Нац. техн. ун-т України “Київ. політехн. ін-т”. – Київ : НТУУ “КПІ”, 2011. – 296, с. .

22. Шмідт Я. Нова мережа: ознаки, практики і наслідки веб 2.0: посібник для вузів / Ян Шмідт; [пер. з нім. В. Климченко; за заг. ред. В. Іванова]. – Київ : Академія Української Преси, Центр Вільної Преси, 2013. – 283

ДОДАТОК А

КОД ПРОГРАМИ

Customer.php //Клас для роботи з обліковими даними користувачів підсистеми

```
<?php
class Mage_Customer_Model_Customer extends Mage_Core_Model_Abstract
{
   const XML_PATH_REGISTER_EMAIL_TEMPLATE = 'customer/create_account/email_template';
   const XML_PATH_REGISTER_EMAIL_IDENTITY = 'customer/create_account/email_identity';
   const XML_PATH_REMIND_EMAIL_TEMPLATE = 'customer/password/remind_email_template';
   const XML_PATH_FORGOT_EMAIL_TEMPLATE = 'customer/password/forgot_email_template';
   const XML_PATH_FORGOT_EMAIL_IDENTITY = 'customer/password/forgot_email_identity';
   const XML_PATH_DEFAULT_EMAIL_DOMAIN        = 'customer/create_account/email_domain';
   const XML_PATH_IS_CONFIRM                  = 'customer/create_account/confirm';
   const XML_PATH_CONFIRM_EMAIL_TEMPLATE      = 'customer/create_account/email_confirmation_template';
   const XML_PATH_CONFIRMED_EMAIL_TEMPLATE    = 'customer/create_account/email_confirmed_template';
   const XML_PATH_GENERATE_HUMAN_FRIENDLY_ID  = 'customer/create_account/generate_human_friendly_id';
   const XML_PATH_CHANGED_PASSWORD_OR_EMAIL_TEMPLATE = 'customer/changed_account/password_or_email_template';
   const XML_PATH_CHANGED_PASSWORD_OR_EMAIL_IDENTITY = 'customer/changed_account/password_or_email_identity';
   const EXCEPTION_EMAIL_NOT_CONFIRMED      = 1;
   const EXCEPTION_INVALID_EMAIL_OR_PASSWORD = 2;
   const EXCEPTION_EMAIL_EXISTS             = 3;
   const EXCEPTION_INVALID_RESET_PASSWORD_LINK_TOKEN = 4;
   const EXCEPTION_INVALID_RESET_PASSWORD_LINK_CUSTOMER_ID = 5;
   const SUBSCRIBED_YES = 'yes';
   const SUBSCRIBED_NO  = 'no';
   const CACHE_TAG = 'customer';
   const MINIMUM_PASSWORD_LENGTH = 6;
   const MAXIMUM_PASSWORD_LENGTH = 256;
   protected $_eventPrefix = 'customer';
   protected $_eventObject = 'customer';
   protected $_errors = array();
   protected $_attributes;
   protected $_addresses = null;
   protected $_addressesCollection;
   protected $_isDeleteable = true;
   protected $_isReadonly = false;
   protected $_cacheTag = self::CACHE_TAG;
   private static $_isConfirmationRequired;

   function _construct()
   {
      $this->_init('customer/customer');
   }

   public function getSharingConfig()
   {
      return Mage::getSingleton('customer/config_share');
   }

   public function authenticate($login, $password)
   {
      $this->loadByEmail($login);
      if ($this->getConfirmation() && $this->isConfirmationRequired()) {
         throw Mage::exception('Mage_Core', Mage::helper('customer')->__('This account is not confirmed.'),
            self::EXCEPTION_EMAIL_NOT_CONFIRMED
         );
      }
      if (!$this->validatePassword($password)) {
         throw Mage::exception('Mage_Core', Mage::helper('customer')->__('Invalid login or password.'),
            self::EXCEPTION_INVALID_EMAIL_OR_PASSWORD
         );
      }
      Mage::dispatchEvent('customer_customer_authenticated', array(
        'model'    => $this,
       'password' => $password,
      ));

      return true;
   }

   public function loadByEmail($customerEmail)
   {
      $this->_getResource()->loadByEmail($this, $customerEmail);
      return $this;
```

```
}


protected function _beforeSave()
{
   parent::_beforeSave();

   $storeId = $this->getStoreId();
   if ($storeId === null) {
      $this->setStoreId(Mage::app()->getStore()->getId());
   }

   $this->getGroupId();
   return $this;
}

public function changePassword($newPassword)
{
   $this->_getResource()->changePassword($this, $newPassword);
   return $this;
}

public function getName()
{
   $name = '';
   $config = Mage::getSingleton('eav/config');
   if ($config->getAttribute('customer', 'prefix')->getIsVisible() && $this->getPrefix()) {
      $name .= $this->getPrefix() . ' ';
   }
   $name .= $this->getFirstname();
   if ($config->getAttribute('customer', 'middlename')->getIsVisible() && $this->getMiddlename()) {
      $name .= ' ' . $this->getMiddlename();
   }
   $name .=  ' ' . $this->getLastname();
   if ($config->getAttribute('customer', 'suffix')->getIsVisible() && $this->getSuffix()) {
      $name .= ' ' . $this->getSuffix();
   }
   return $name;
}

public function addAddress(Mage_Customer_Model_Address $address)
{
   $this->getAddressesCollection()->addItem($address);
   $this->getAddresses();
   $this->_addresses[] = $address;
   return $this;
}

public function getAddressById($addressId)
{
   $address = Mage::getModel('customer/address')->load($addressId);
   if ($this->getId() == $address->getParentId()) {
      return $address;
   }
   return Mage::getModel('customer/address');
}

public function getAddressItemById($addressId)
{
   return $this->getAddressesCollection()->getItemById($addressId);
}

public function getAddressCollection()
{
   return Mage::getResourceModel('customer/address_collection');
}

public function getAddressesCollection()
{
   if ($this->_addressesCollection === null) {
      $this->_addressesCollection = $this->getAddressCollection()
         ->setCustomerFilter($this)
         ->addAttributeToSelect('*');
      foreach ($this->_addressesCollection as $address) {
         $address->setCustomer($this);
      }
   }

   return $this->_addressesCollection;
```

```
}

public function getAddresses()
{
   $this->_addresses = $this->getAddressesCollection()->getItems();
   return $this->_addresses;
}

public function getAttributes()
{
   if ($this->_attributes === null) {
      $this->_attributes = $this->_getResource()
      ->loadAllAttributes($this)
      ->getSortedAttributes();
   }
   return $this->_attributes;
}


public function getAttribute($attributeCode)
{
   $this->getAttributes();
   if (isset($this->_attributes[$attributeCode])) {
      return $this->_attributes[$attributeCode];
   }
   return null;
}

public function setPassword($password)
{
   $this->setData('password', $password);
   $this->setPasswordHash($this->hashPassword($password));
   $this->setPasswordConfirmation(null);
   return $this;
}

public function hashPassword($password, $salt = null)
{
   return $this->_getHelper('core')
      ->getHash(trim($password), !is_null($salt) ? $salt : Mage_Admin_Model_User::HASH_SALT_LENGTH);
}

protected function _getHelper($helperName)
{
   return Mage::helper($helperName);
}

public function generatePassword($length = 8)
{
   $chars = Mage_Core_Helper_Data::CHARS_PASSWORD_LOWERS
      . Mage_Core_Helper_Data::CHARS_PASSWORD_UPPERS
      . Mage_Core_Helper_Data::CHARS_PASSWORD_DIGITS
      . Mage_Core_Helper_Data::CHARS_PASSWORD_SPECIALS;
   return Mage::helper('core')->getRandomString($length, $chars);
}

public function validatePassword($password)
{
   $hash = $this->getPasswordHash();
   if (!$hash) {
      return false;
   }
   return Mage::helper('core')->validateHash($password, $hash);
}

public function encryptPassword($password)
{
   return Mage::helper('core')->encrypt($password);
}

public function decryptPassword($password)
{
   return Mage::helper('core')->decrypt($password);
}

public function getPrimaryAddress($attributeCode)
{
   $primaryAddress = $this->getAddressesCollection()->getItemById($this->getData($attributeCode));
```

```
    return $primaryAddress ? $primaryAddress : false;
}

public function getPrimaryBillingAddress()
{
    return $this->getPrimaryAddress('default_billing');
}

public function getDefaultBillingAddress()
{
    return $this->getPrimaryBillingAddress();
}
public function getPrimaryShippingAddress()
{
    return $this->getPrimaryAddress('default_shipping');
}

public function getDefaultShippingAddress()
{
    return $this->getPrimaryShippingAddress();
}

public function getPrimaryAddressIds()
{
    $ids = array();
    if ($this->getDefaultBilling()) {
        $ids[] = $this->getDefaultBilling();
    }
    if ($this->getDefaultShipping()) {
        $ids[] = $this->getDefaultShipping();
    }
    return $ids;
}

public function getPrimaryAddresses()
{
    $addresses = array();
    $primaryBilling = $this->getPrimaryBillingAddress();
    if ($primaryBilling) {
        $addresses[] = $primaryBilling;
        $primaryBilling->setIsPrimaryBilling(true);
    }

    $primaryShipping = $this->getPrimaryShippingAddress();
    if ($primaryShipping) {
        if ($primaryBilling->getId() == $primaryShipping->getId()) {
            $primaryBilling->setIsPrimaryShipping(true);
        } else {
            $primaryShipping->setIsPrimaryShipping(true);
            $addresses[] = $primaryShipping;
        }
    }
    return $addresses;
}

public function getAdditionalAddresses()
{
    $addresses = array();
    $primatyIds = $this->getPrimaryAddressIds();
    foreach ($this->getAddressesCollection() as $address) {
        if (!in_array($address->getId(), $primatyIds)) {
            $addresses[] = $address;
        }
    }
    return $addresses;
}

public function isAddressPrimary(Mage_Customer_Model_Address $address)
{
    if (!$address->getId()) {
        return false;
    }
    return ($address->getId() == $this->getDefaultBilling()) || ($address->getId() == $this->getDefaultShipping());
}

public function sendNewAccountEmail($type = 'registered', $backUrl = '', $storeId = '0', $password = null)
{
    $types = array(
        'registered'   => self::XML_PATH_REGISTER_EMAIL_TEMPLATE, // welcome email, when confirmation is disabled
```

```
        'confirmed'    => self::XML_PATH_CONFIRMED_EMAIL_TEMPLATE, // welcome email, when confirmation is enabled
        'confirmation' => self::XML_PATH_CONFIRM_EMAIL_TEMPLATE, // email with confirmation link
    );
    if (!isset($types[$type])) {
        Mage::throwException(Mage::helper('customer')->__('Wrong transactional account email type'));
    }

    if (!$storeId) {
        $storeId = $this->_getWebsiteStoreId($this->getSendemailStoreId());
    }

    if (!is_null($password)) {
        $this->setPassword($password);
    }

    $this->_sendEmailTemplate($types[$type], self::XML_PATH_REGISTER_EMAIL_IDENTITY,
        array('customer' => $this, 'back_url' => $backUrl), $storeId);
    $this->cleanPasswordsValidationData();

    return $this;
}

public function isConfirmationRequired()
{
    if ($this->canSkipConfirmation()) {
        return false;
    }
    if (self::$_isConfirmationRequired === null) {
        $storeId = $this->getStoreId() ? $this->getStoreId() : null;
        self::$_isConfirmationRequired = (bool)Mage::getStoreConfig(self::XML_PATH_IS_CONFIRM, $storeId);
    }

    return self::$_isConfirmationRequired;
}

public function getRandomConfirmationKey()
{
    return md5(uniqid());
}

public function sendPasswordReminderEmail()
{
    $storeId = $this->getStoreId();
    if (!$storeId) {
        $storeId = $this->_getWebsiteStoreId();
    }

    $this->_sendEmailTemplate(self::XML_PATH_REMIND_EMAIL_TEMPLATE, self::XML_PATH_FORGOT_EMAIL_IDENTITY,
        array('customer' => $this), $storeId);

    return $this;
}

public function sendChangedPasswordOrEmail()
{
    $storeId = $this->getStoreId();
    if (!$storeId) {
        $storeId = $this->_getWebsiteStoreId();
    }

    $this->_sendEmailTemplate(self::XML_PATH_CHANGED_PASSWORD_OR_EMAIL_TEMPLATE,
        self::XML_PATH_CHANGED_PASSWORD_OR_EMAIL_IDENTITY,
        array('customer' => $this), $storeId, $this->getOldEmail());

    return $this;
}

protected function _sendEmailTemplate($template, $sender, $templateParams = array(), $storeId = null, $customerEmail = null)
{
    $customerEmail = ($customerEmail) ? $customerEmail : $this->getEmail();
    $mailer = Mage::getModel('core/email_template_mailer');
    $emailInfo = Mage::getModel('core/email_info');
    $emailInfo->addTo($customerEmail, $this->getName());
    $mailer->addEmailInfo($emailInfo);

    // Set all required params and send emails
    $mailer->setSender(Mage::getStoreConfig($sender, $storeId));
    $mailer->setStoreId($storeId);
    $mailer->setTemplateId(Mage::getStoreConfig($template, $storeId));
```

```
        $mailer->setTemplateParams($templateParams);
        $mailer->send();
        return $this;
    }

    public function sendPasswordResetConfirmationEmail()
    {
        $storeId = Mage::app()->getStore()->getId();
        if (!$storeId) {
            $storeId = $this->_getWebsiteStoreId();
        }

        $this->_sendEmailTemplate(self::XML_PATH_FORGOT_EMAIL_TEMPLATE, self::XML_PATH_FORGOT_EMAIL_IDENTITY,
            array('customer' => $this), $storeId);

        return $this;
    }

    public function getGroupId()
    {
        if (!$this->hasData('group_id')) {
            $storeId = $this->getStoreId() ? $this->getStoreId() : Mage::app()->getStore()->getId();
            $groupId = Mage::getStoreConfig(Mage_Customer_Model_Group::XML_PATH_DEFAULT_ID, $storeId);
            $this->setData('group_id', $groupId);
        }
        return $this->getData('group_id');
    }

    public function getTaxClassId()
    {
        if (!$this->getData('tax_class_id')) {
            $this->setTaxClassId(Mage::getModel('customer/group')->getTaxClassId($this->getGroupId()));
        }
        return $this->getData('tax_class_id');
    }

    public function isInStore($store)
    {
        if ($store instanceof Mage_Core_Model_Store) {
            $storeId = $store->getId();
        } else {
            $storeId = $store;
        }

        $availableStores = $this->getSharedStoreIds();
        return in_array($storeId, $availableStores);
    }

    public function getStore()
    {
        return Mage::app()->getStore($this->getStoreId());
    }

    public function getSharedStoreIds()
    {
        $ids = $this->_getData('shared_store_ids');
        if ($ids === null) {
            $ids = array();
            if ((bool)$this->getSharingConfig()->isWebsiteScope()) {
                $ids = Mage::app()->getWebsite($this->getWebsiteId())->getStoreIds();
            } else {
                foreach (Mage::app()->getStores() as $store) {
                    $ids[] = $store->getId();
                }
            }
            $this->setData('shared_store_ids', $ids);
        }

        return $ids;
    }

    public function getSharedWebsiteIds()
    {
        $ids = $this->_getData('shared_website_ids');
        if ($ids === null) {
            $ids = array();
            if ((bool)$this->getSharingConfig()->isWebsiteScope()) {
                $ids[] = $this->getWebsiteId();
            } else {
```

```
            foreach (Mage::app()->getWebsites() as $website) {
                $ids[] = $website->getId();
            }
        }
        $this->setData('shared_website_ids', $ids);
    }
    return $ids;
}

public function setStore(Mage_Core_Model_Store $store)
{
    $this->setStoreId($store->getId());
    $this->setWebsiteId($store->getWebsite()->getId());
    return $this;
}

public function validate()
{
    $errors = array();
    if (!Zend_Validate::is( trim($this->getFirstname()) , 'NotEmpty')) {
        $errors[] = Mage::helper('customer')->__('The first name cannot be empty.');
    }

    if (!Zend_Validate::is( trim($this->getLastname()) , 'NotEmpty')) {
        $errors[] = Mage::helper('customer')->__('The last name cannot be empty.');
    }

    if (!Zend_Validate::is($this->getEmail(), 'EmailAddress')) {
        $errors[] = Mage::helper('customer')->__('Invalid email address "%s".', $this->getEmail());
    }

    $password = $this->getPassword();
    if (!$this->getId() && !Zend_Validate::is($password , 'NotEmpty')) {
        $errors[] = Mage::helper('customer')->__('The password cannot be empty.');
    }
    if (strlen($password) && !Zend_Validate::is($password, 'StringLength', array(self::MINIMUM_PASSWORD_LENGTH))) {
        $errors[] = Mage::helper('customer')
            ->__('The minimum password length is %s', self::MINIMUM_PASSWORD_LENGTH);
    }
    if (strlen($password) && !Zend_Validate::is($password, 'StringLength', array('max' => self::MAXIMUM_PASSWORD_LENGTH))) {
        $errors[] = Mage::helper('customer')
            ->__('Please enter a password with at most %s characters.', self::MAXIMUM_PASSWORD_LENGTH);
    }
    $confirmation = $this->getPasswordConfirmation();
    if ($password != $confirmation) {
        $errors[] = Mage::helper('customer')->__('Please make sure your passwords match.');
    }

    $entityType = Mage::getSingleton('eav/config')->getEntityType('customer');
    $attribute = Mage::getModel('customer/attribute')->loadByCode($entityType, 'dob');
    if ($attribute->getIsRequired() && '' == trim($this->getDob())) {
        $errors[] = Mage::helper('customer')->__('The Date of Birth is required.');
    }
    $attribute = Mage::getModel('customer/attribute')->loadByCode($entityType, 'taxvat');
    if ($attribute->getIsRequired() && '' == trim($this->getTaxvat())) {
        $errors[] = Mage::helper('customer')->__('The TAX/VAT number is required.');
    }
    $attribute = Mage::getModel('customer/attribute')->loadByCode($entityType, 'gender');
    if ($attribute->getIsRequired() && '' == trim($this->getGender())) {
        $errors[] = Mage::helper('customer')->__('Gender is required.');
    }

    if (empty($errors)) {
        return true;
    }
    return $errors;
}

public function validateResetPassword()
{
    $errors   = array();
    $password = $this->getPassword();
    if (!Zend_Validate::is($password, 'NotEmpty')) {
        $errors[] = Mage::helper('customer')->__('The password cannot be empty.');
    }
    if (!Zend_Validate::is($password, 'StringLength', array(self::MINIMUM_PASSWORD_LENGTH))) {
        $errors[] = Mage::helper('customer')
            ->__('The minimum password length is %s', self::MINIMUM_PASSWORD_LENGTH);
    }
```

```
        if (!Zend_Validate::is($password, 'StringLength', array('max' => self::MAXIMUM_PASSWORD_LENGTH))) {
            $errors[] = Mage::helper('customer')
                ->__('Please enter a password with at most %s characters.', self::MAXIMUM_PASSWORD_LENGTH);
        }
        $confirmation = $this->getPasswordConfirmation();
        if ($password != $confirmation) {
            $errors[] = Mage::helper('customer')->__('Please make sure your passwords match.');
        }

        if (empty($errors)) {
            return true;
        }
        return $errors;
    }

    public function importFromTextArray(array $row)
    {
        $this->resetErrors();
        $line = $row['i'];
        $row = $row['row'];

        $regions = Mage::getResourceModel('directory/region_collection');

        $website = Mage::getModel('core/website')->load($row['website_code'], 'code');

        if (!$website->getId()) {
            $this->addError(Mage::helper('customer')->__('Invalid website, skipping the record, line: %s', $line));

        } else {
            $row['website_id'] = $website->getWebsiteId();
            $this->setWebsiteId($row['website_id']);
        }

        // Validate Email
        if (empty($row['email'])) {
            $this->addError(Mage::helper('customer')->__('Missing email, skipping the record, line: %s', $line));
        } else {
            $this->loadByEmail($row['email']);
        }

        if (empty($row['entity_id'])) {
            if ($this->getData('entity_id')) {
                $this->addError(Mage::helper('customer')->__('The customer email (%s) already exists, skipping the record, line: %s', $row['email'],
$line));
            }
        } else {
            if ($row['entity_id'] != $this->getData('entity_id')) {
                $this->addError(Mage::helper('customer')->__('The customer ID and email did not match, skipping the record, line: %s', $line));
            } else {
                $this->unsetData();
                $this->load($row['entity_id']);
                if (isset($row['store_view'])) {
                    $storeId = Mage::app()->getStore($row['store_view'])->getId();
                    if ($storeId) $this->setStoreId($storeId);
                }
            }
        }

        if (empty($row['website_code'])) {
            $this->addError(Mage::helper('customer')->__('Missing website, skipping the record, line: %s', $line));
        }

        if (empty($row['group'])) {
            $row['group'] = 'General';
        }

        if (empty($row['firstname'])) {
            $this->addError(Mage::helper('customer')->__('Missing first name, skipping the record, line: %s', $line));
        }
        if (empty($row['lastname'])) {
            $this->addError(Mage::helper('customer')->__('Missing last name, skipping the record, line: %s', $line));
        }

        if (!empty($row['password_new'])) {
            $this->setPassword($row['password_new']);
            unset($row['password_new']);
            if (!empty($row['password_hash'])) unset($row['password_hash']);
        }
```

```
$errors = $this->getErrors();
if ($errors) {
    $this->unsetData();
    $this->printError(implode('<br />', $errors));
    return;
}

foreach ($row as $field => $value) {
    $this->setData($field, $value);
}

if (!$this->validateAddress($row, 'billing')) {
    $this->printError(Mage::helper('customer')->__('Invalid billing address for (%s)', $row['email']), $line);
} else {
    // Handling billing address
    $billingAddress = $this->getPrimaryBillingAddress();
    if (!$billingAddress  instanceof Mage_Customer_Model_Address) {
        $billingAddress = Mage::getModel('customer/address');
    }

    $regions->addRegionNameFilter($row['billing_region'])->load();
    if ($regions) foreach($regions as $region) {
        $regionId = intval($region->getId());
    }

    $billingAddress->setFirstname($row['firstname']);
    $billingAddress->setLastname($row['lastname']);
    $billingAddress->setCity($row['billing_city']);
    $billingAddress->setRegion($row['billing_region']);
    if (isset($regionId)) {
        $billingAddress->setRegionId($regionId);
    }
    $billingAddress->setCountryId($row['billing_country']);
    $billingAddress->setPostcode($row['billing_postcode']);
    if (isset($row['billing_street2'])) {
        $billingAddress->setStreet(array($row['billing_street1'], $row['billing_street2']));
    } else {
        $billingAddress->setStreet(array($row['billing_street1']));
    }
    if (isset($row['billing_telephone'])) {
        $billingAddress->setTelephone($row['billing_telephone']);
    }

    if (!$billingAddress->getId()) {
        $billingAddress->setIsDefaultBilling(true);
        if ($this->getDefaultBilling()) {
            $this->setData('default_billing', '');
        }
        $this->addAddress($billingAddress);
    } // End handling billing address
}

if (!$this->validateAddress($row, 'shipping')) {
    $this->printError(Mage::helper('customer')->__('Invalid shipping address for (%s)', $row['email']), $line);
} else {
    // Handling shipping address
    $shippingAddress = $this->getPrimaryShippingAddress();
    if (!$shippingAddress instanceof Mage_Customer_Model_Address) {
        $shippingAddress = Mage::getModel('customer/address');
    }

    $regions->addRegionNameFilter($row['shipping_region'])->load();

    if ($regions) foreach($regions as $region) {
      $regionId = intval($region->getId());
    }

    $shippingAddress->setFirstname($row['firstname']);
    $shippingAddress->setLastname($row['lastname']);
    $shippingAddress->setCity($row['shipping_city']);
    $shippingAddress->setRegion($row['shipping_region']);
    if (isset($regionId)) {
        $shippingAddress->setRegionId($regionId);
    }
    $shippingAddress->setCountryId($row['shipping_country']);
    $shippingAddress->setPostcode($row['shipping_postcode']);
    if (isset($row['shipping_street2'])) {
        $shippingAddress->setStreet(array($row['shipping_street1'], $row['shipping_street2']));
    } else {
```

```
            $shippingAddress->setStreet(array($row['shipping_street1']));
        }
        if (!empty($row['shipping_telephone'])) {
            $shippingAddress->setTelephone($row['shipping_telephone']);
        }

        if (!$shippingAddress->getId()) {
          $shippingAddress->setIsDefaultShipping(true);
          $this->addAddress($shippingAddress);
        }
        // End handling shipping address
    }
    if (!empty($row['is_subscribed'])) {
        $isSubscribed = (bool)strtolower($row['is_subscribed']) == self::SUBSCRIBED_YES;
        $this->setIsSubscribed($isSubscribed);
    }
    unset($row);
    return $this;
}

function unsetSubscription()
{
    if (isset($this->_isSubscribed)) {
        unset($this->_isSubscribed);
    }
    return $this;
}

function cleanAllAddresses() {
    $this->_addressesCollection = null;
    $this->_addresses           = null;
}

function addError($error)
{
    $this->_errors[] = $error;
    return $this;
}

function getErrors()
{
    return $this->_errors;
}

function resetErrors()
{
    $this->_errors = array();
    return $this;
}

function printError($error, $line = null)
{
    if ($error == null) {
        return false;
    }

    $liStyle = 'background-color: #FDD; ';
    echo '<li style="' . $liStyle . '">';
    echo '<img src="' . Mage::getDesign()->getSkinUrl('images/error_msg_icon.gif') . '" class="v-middle"/>';
    echo $error;
    if ($line) {
        echo '<small>, Line: <b>' . $line . '</b></small>';
    }
    echo '</li>';
}

function validateAddress(array $data, $type = 'billing')
{
    $fields = array('city', 'country', 'postcode', 'telephone', 'street1');
    $usca   = array('US', 'CA');
    $prefix = $type ? $type . '_' : '';

    if ($data) {
        foreach ($fields as $field) {
            if (!isset($data[$prefix . $field])) {
                return false;
            }
            if ($field == 'country'
                && in_array(strtolower($data[$prefix . $field]), array('US', 'CA'))) {
```

```
            if (!isset($data[$prefix . 'region'])) {
                return false;
            }

            $region = Mage::getModel('directory/region')
                ->loadByName($data[$prefix . 'region']);
            if (!$region->getId()) {
                return false;
            }
            unset($region);
        }
    }
    unset($data);
    return true;
  }
  return false;
}
protected function _beforeDelete()
{
  $this->_protectFromNonAdmin();
  return parent::_beforeDelete();
}

public function getCreatedAtTimestamp()
{
  $date = $this->getCreatedAt();
  if ($date) {
    return Varien_Date::toTimestamp($date);
  }
  return null;
}


public function reset()
{
  $this->setData(array());
  $this->setOrigData();
  $this->_attributes = null;

  return $this;
}

public function isDeleteable()
{
  return $this->_isDeleteable;
}

public function setIsDeleteable($value)
{
  $this->_isDeleteable = (bool)$value;
  return $this;
}

public function isReadonly()
{
  return $this->_isReadonly;
}

public function setIsReadonly($value)
{
  $this->_isReadonly = (bool)$value;
  return $this;
}

public function canSkipConfirmation()
{
  return $this->getId() && $this->hasSkipConfirmationIfEmail()
    && strtolower($this->getSkipConfirmationIfEmail()) === strtolower($this->getEmail());
}

public function __clone()
{
  $newAddressCollection = $this->getPrimaryAddresses();
  $newAddressCollection = array_merge($newAddressCollection, $this->getAdditionalAddresses());
  $this->setId(null);
  $this->cleanAllAddresses();
  foreach ($newAddressCollection as $address) {
    $this->addAddress(clone $address);
```

```
    }
}
public function getEntityType()
{
    return $this->_getResource()->getEntityType();
}

public function getEntityTypeId()
{
    $entityTypeId = $this->getData('entity_type_id');
    if (!$entityTypeId) {
        $entityTypeId = $this->getEntityType()->getId();
        $this->setData('entity_type_id', $entityTypeId);
    }
    return $entityTypeId;
}

protected function _getWebsiteStoreId($defaultStoreId = null)
{
    if ($this->getWebsiteId() != 0 && empty($defaultStoreId)) {
        $storeIds = Mage::app()->getWebsite($this->getWebsiteId())->getStoreIds();
        reset($storeIds);
        $defaultStoreId = current($storeIds);
    }
    return $defaultStoreId;
}

public function changeResetPasswordLinkToken($newResetPasswordLinkToken) {
    if (!is_string($newResetPasswordLinkToken) || empty($newResetPasswordLinkToken)) {
        throw Mage::exception('Mage_Core', Mage::helper('customer')->__('Invalid password reset token.'),
            self::EXCEPTION_INVALID_RESET_PASSWORD_LINK_TOKEN);
    }
    $this->_getResource()->changeResetPasswordLinkToken($this, $newResetPasswordLinkToken);
    return $this;
}

public function changeResetPasswordLinkCustomerId($newResetPasswordLinkCustomerId)
{
    if (!is_string($newResetPasswordLinkCustomerId) || empty($newResetPasswordLinkCustomerId)) {
        throw Mage::exception(
            'Mage_Core',
            Mage::helper('customer')->__('Invalid password reset customer Id.'),
            self::EXCEPTION_INVALID_RESET_PASSWORD_LINK_CUSTOMER_ID
        );
    }
    $this->_getResource()->changeResetPasswordLinkCustomerId($this, $newResetPasswordLinkCustomerId);
    return $this;
}

public function isResetPasswordLinkTokenExpired()
{
    $resetPasswordLinkToken = $this->getRpToken();
    $resetPasswordLinkTokenCreatedAt = $this->getRpTokenCreatedAt();

    if (empty($resetPasswordLinkToken) || empty($resetPasswordLinkTokenCreatedAt)) {
        return true;
    }

    $tokenExpirationPeriod = Mage::helper('customer')->getResetPasswordLinkExpirationPeriod();

    $currentDate = Varien_Date::now();
    $currentTimestamp = Varien_Date::toTimestamp($currentDate);
    $tokenTimestamp = Varien_Date::toTimestamp($resetPasswordLinkTokenCreatedAt);
    if ($tokenTimestamp > $currentTimestamp) {
        return true;
    }

    $hoursDifference = floor(($currentTimestamp - $tokenTimestamp) / (60 * 60));
    if ($hoursDifference >= $tokenExpirationPeriod) {
        return true;
    }

    return false;
}

public function cleanPasswordsValidationData()
{
    $this->setData('password', null);
    $this->setData('password_confirmation', null);
```

```
        return $this;
    }
}
```

Session.php //Клас для реалізації механізму сесії

```
<?php
class Mage_Customer_Model_Session extends Mage_Core_Model_Session_Abstract
{
    protected $_customer;
    protected $_isCustomerIdChecked = null;
    protected $_persistentCustomerGroupId = null;
    public function getCustomerConfigShare()
    {
        return Mage::getSingleton('customer/config_share');
    }

    public function __construct()
    {
        $namespace = 'customer';
        if ($this->getCustomerConfigShare()->isWebsiteScope()) {
            $namespace .= '_' . (Mage::app()->getStore()->getWebsite()->getCode());
        }

        $this->init($namespace);
        Mage::dispatchEvent('customer_session_init', array('customer_session'=>$this));
    }

    public function setCustomer(Mage_Customer_Model_Customer $customer)
    {
        // check if customer is not confirmed
        if ($customer->isConfirmationRequired()) {
            if ($customer->getConfirmation()) {
                return $this->_logout();
            }
        }
        $this->_customer = $customer;
        $this->setId($customer->getId());
        // save customer as confirmed, if it is not
        if ((!$customer->isConfirmationRequired()) && $customer->getConfirmation()) {
            $customer->setConfirmation(null)->save();
            $customer->setIsJustConfirmed(true);
        }
        return $this;
    }

    public function getCustomer()
    {
        if ($this->_customer instanceof Mage_Customer_Model_Customer) {
            return $this->_customer;
        }

        $customer = Mage::getModel('customer/customer')
            ->setWebsiteId(Mage::app()->getStore()->getWebsiteId());
        if ($this->getId()) {
            $customer->load($this->getId());
        }

        $this->setCustomer($customer);
        return $this->_customer;
    }

    public function setCustomerId($id)
    {
        $this->setData('customer_id', $id);
        return $this;
    }

    public function getCustomerId()
    {
        if ($this->getData('customer_id')) {
            return $this->getData('customer_id');
        }
        return ($this->isLoggedIn()) ? $this->getId() : null;
    }

    public function setCustomerGroupId($id)
    {
```

```
    $this->setData('customer_group_id', $id);
    return $this;
}

public function getCustomerGroupId()
{
    if ($this->getData('customer_group_id')) {
        return $this->getData('customer_group_id');
    }
    if ($this->isLoggedIn() && $this->getCustomer()) {
        return $this->getCustomer()->getGroupId();
    }
    return Mage_Customer_Model_Group::NOT_LOGGED_IN_ID;
}

public function isLoggedIn()
{
    return (bool)$this->getId() && (bool)$this->checkCustomerId($this->getId());
}

public function checkCustomerId($customerId)
{
    if ($this->_isCustomerIdChecked === null) {
        $this->_isCustomerIdChecked = Mage::getResourceSingleton('customer/customer')->checkCustomerId($customerId);
    }
    return $this->_isCustomerIdChecked;
}

public function login($username, $password)
{
    $customer = Mage::getModel('customer/customer')
        ->setWebsiteId(Mage::app()->getStore()->getWebsiteId());

    if ($customer->authenticate($username, $password)) {
        $this->setCustomerAsLoggedIn($customer);
        return true;
    }
    return false;
}

public function setCustomerAsLoggedIn($customer)
{
    $this->setCustomer($customer);
    $this->renewSession();
    Mage::getSingleton('core/session')->renewFormKey();
    Mage::dispatchEvent('customer_login', array('customer'=>$customer));
    return $this;
}

public function loginById($customerId)
{
    $customer = Mage::getModel('customer/customer')->load($customerId);
    if ($customer->getId()) {
        $this->setCustomerAsLoggedIn($customer);
        return true;
    }
    return false;
}

public function logout()
{
    if ($this->isLoggedIn()) {
        Mage::dispatchEvent('customer_logout', array('customer' => $this->getCustomer()) );
        $this->_logout();
    }
    return $this;
}

public function authenticate(Mage_Core_Controller_Varien_Action $action, $loginUrl = null)
{
    if ($this->isLoggedIn()) {
        return true;
    }

    $this->setBeforeAuthUrl(Mage::getUrl('*/*/*', array('_current' => true)));
    if (isset($loginUrl)) {
        $action->getResponse()->setRedirect($loginUrl);
    } else {
        $action->setRedirectWithCookieCheck(Mage_Customer_Helper_Data::ROUTE_ACCOUNT_LOGIN,
```

```
            Mage::helper('customer')->getLoginUrlParams()
        );
    }

    return false;
}

protected function _setAuthUrl($key, $url)
{
    $url = Mage::helper('core/url')
        ->removeRequestParam($url, Mage::getSingleton('core/session')->getSessionIdQueryParam());
    // Add correct session ID to URL if needed
    $url = Mage::getModel('core/url')->getRebuiltUrl($url);
    return $this->setData($key, $url);
}

protected function _logout()
{
    $this->setId(null);
    $this->setCustomerGroupId(Mage_Customer_Model_Group::NOT_LOGGED_IN_ID);
    $this->getCookie()->delete($this->getSessionName());
    Mage::getSingleton('core/session')->renewFormKey();
    return $this;
}

public function setBeforeAuthUrl($url)
{
    return $this->_setAuthUrl('before_auth_url', $url);
}

public function setAfterAuthUrl($url)
{
    return $this->_setAuthUrl('after_auth_url', $url);
}

public function renewSession()
{
    parent::renewSession();
    Mage::getSingleton('core/session')->unsSessionHosts();

    return $this;
}
}
```

OrderController.php //Клас, що раелізує контроллер замовлення

```
<?php

class Mage_Sales_OrderController extends Mage_Sales_Controller_Abstract
{

public function preDispatch()
{
    parent::preDispatch();
    $action = $this->getRequest()->getActionName();
    $loginUrl = Mage::helper('customer')->getLoginUrl();

    if (!Mage::getSingleton('customer/session')->authenticate($this, $loginUrl)) {
        $this->setFlag('', self::FLAG_NO_DISPATCH, true);
    }
}

public function historyAction()
{
    $this->loadLayout();
    $this->_initLayoutMessages('catalog/session');

    $this->getLayout()->getBlock('head')->setTitle($this->__('My Orders'));

    if ($block = $this->getLayout()->getBlock('customer.account.link.back')) {
        $block->setRefererUrl($this->_getRefererUrl());
    }
    $this->renderLayout();
}

protected function _canViewOscommerceOrder($order)
{
    return false;
```

```
    }

    public function viewOldAction()
    {
        $this->_forward('noRoute');
        return;
    }
}
```

Order.php //Клас для реалізації функціоналу кошику

```
<?php
class Mage_Sales_Model_Order extends Mage_Sales_Model_Abstract
{
    const ENTITY                            = 'order';
    const EMAIL_EVENT_NAME_NEW_ORDER    = 'new_order';
    const EMAIL_EVENT_NAME_UPDATE_ORDER = 'update_order';
    const XML_PATH_EMAIL_TEMPLATE               = 'sales_email/order/template';
    const XML_PATH_EMAIL_GUEST_TEMPLATE         = 'sales_email/order/guest_template';
    const XML_PATH_EMAIL_IDENTITY               = 'sales_email/order/identity';
    const XML_PATH_EMAIL_COPY_TO                = 'sales_email/order/copy_to';
    const XML_PATH_EMAIL_COPY_METHOD            = 'sales_email/order/copy_method';
    const XML_PATH_EMAIL_ENABLED                = 'sales_email/order/enabled';
    const XML_PATH_UPDATE_EMAIL_TEMPLATE        = 'sales_email/order_comment/template';
    const XML_PATH_UPDATE_EMAIL_GUEST_TEMPLATE  = 'sales_email/order_comment/guest_template';
    const XML_PATH_UPDATE_EMAIL_IDENTITY        = 'sales_email/order_comment/identity';
    const XML_PATH_UPDATE_EMAIL_COPY_TO         = 'sales_email/order_comment/copy_to';
    const XML_PATH_UPDATE_EMAIL_COPY_METHOD     = 'sales_email/order_comment/copy_method';
    const XML_PATH_UPDATE_EMAIL_ENABLED         = 'sales_email/order_comment/enabled';

    /**
     * Order states
     */
    const STATE_NEW             = 'new';
    const STATE_PENDING_PAYMENT = 'pending_payment';
    const STATE_PROCESSING      = 'processing';
    const STATE_COMPLETE        = 'complete';
    const STATE_CLOSED          = 'closed';
    const STATE_CANCELED        = 'canceled';
    const STATE_HOLDED          = 'holded';
    const STATE_PAYMENT_REVIEW  = 'payment_review';

    /**
     * Order statuses
     */
    const STATUS_FRAUD  = 'fraud';

    /**
     * Order flags
     */
    const ACTION_FLAG_CANCEL                = 'cancel';
    const ACTION_FLAG_HOLD                  = 'hold';
    const ACTION_FLAG_UNHOLD                = 'unhold';
    const ACTION_FLAG_EDIT                  = 'edit';
    const ACTION_FLAG_CREDITMEMO            = 'creditmemo';
    const ACTION_FLAG_INVOICE               = 'invoice';
    const ACTION_FLAG_REORDER               = 'reorder';
    const ACTION_FLAG_SHIP                  = 'ship';
    const ACTION_FLAG_COMMENT               = 'comment';
    const ACTION_FLAG_PRODUCTS_PERMISSION_DENIED= 'product_permission_denied';
    const REPORT_DATE_TYPE_CREATED = 'created';
    const REPORT_DATE_TYPE_UPDATED = 'updated';
    const HISTORY_ENTITY_NAME = 'order';

    protected $_eventPrefix = 'sales_order';
    protected $_eventObject = 'order';

    protected $_addresses       = null;
    protected $_items           = null;
    protected $_payments        = null;
    protected $_statusHistory   = null;
    protected $_invoices;
    protected $_tracks;
    protected $_shipments;
    protected $_creditmemos;

    protected $_relatedObjects  = array();
    protected $_orderCurrency   = null;
```

```
protected $_baseCurrency    = null;

protected $_actionFlag = array();

protected $_canSendNewEmailFlag = true;

protected $_historyEntityName = self::HISTORY_ENTITY_NAME;

protected function _construct()
{
    $this->_init('sales/order');
}

protected function _initOldFieldsMap()
{
    $this->_oldFieldsMap = Mage::helper('sales')->getOldFieldMap('order');
    return $this;
}

public function unsetData($key=null)
{
    parent::unsetData($key);
    if (is_null($key)) {
        $this->_items = null;
    }
    return $this;
}

public function getActionFlag($action)
{
    if (isset($this->_actionFlag[$action])) {
        return $this->_actionFlag[$action];
    }
    return null;
}

public function setActionFlag($action, $flag)
{
    $this->_actionFlag[$action] = (boolean) $flag;
    return $this;
}

public function getCanSendNewEmailFlag()
{
    return $this->_canSendNewEmailFlag;
}

public function setCanSendNewEmailFlag($flag)
{
    $this->_canSendNewEmailFlag = (boolean) $flag;
    return $this;
}

public function loadByIncrementId($incrementId)
{
    return $this->loadByAttribute('increment_id', $incrementId);
}

public function loadByAttribute($attribute, $value)
{
    $this->load($value, $attribute);
    return $this;
}

public function getStore()
{
    $storeId = $this->getStoreId();
    if ($storeId) {
        return Mage::app()->getStore($storeId);
    }
    return Mage::app()->getStore();
}

public function canCancel()
{
    if (!$this->_canVoidOrder()) {
        return false;
    }
    if ($this->canUnhold()) {  // $this->isPaymentReview()
```

```
        return false;
    }

    $allInvoiced = true;
    foreach ($this->getAllItems() as $item) {
        if ($item->getQtyToInvoice()) {
            $allInvoiced = false;
            break;
        }
    }
    if ($allInvoiced) {
        return false;
    }

    $state = $this->getState();
    if ($this->isCanceled() || $state === self::STATE_COMPLETE || $state === self::STATE_CLOSED) {
        return false;
    }

    if ($this->getActionFlag(self::ACTION_FLAG_CANCEL) === false) {
        return false;
    }
        if ($item->getQtyToCancel()>0) {
            return true;
        }
    }
    return false;*/
    return true;
}

public function canVoidPayment()
{
    return $this->_canVoidOrder() ? $this->getPayment()->canVoid($this->getPayment()) : false;
}

protected function _canVoidOrder()
{
    if ($this->canUnhold() || $this->isPaymentReview()) {
        return false;
    }
    return true;
}

public function canInvoice()
{
    if ($this->canUnhold() || $this->isPaymentReview()) {
        return false;
    }
    $state = $this->getState();
    if ($this->isCanceled() || $state === self::STATE_COMPLETE || $state === self::STATE_CLOSED) {
        return false;
    }

    if ($this->getActionFlag(self::ACTION_FLAG_INVOICE) === false) {
        return false;
    }

    foreach ($this->getAllItems() as $item) {
        if ($item->getQtyToInvoice()>0 && !$item->getLockedDoInvoice()) {
            return true;
        }
    }
    return false;
}

public function canCreditmemo()
{
    if ($this->hasForcedCanCreditmemo()) {
        return $this->getForcedCanCreditmemo();
    }

    if ($this->canUnhold() || $this->isPaymentReview()) {
        return false;
    }

    if ($this->isCanceled() || $this->getState() === self::STATE_CLOSED) {
        return false;
    }
```

```
    if (abs($this->getStore()->roundPrice($this->getTotalPaid()) - $this->getTotalRefunded()) < .0001) {
        return false;
    }

    if ($this->getActionFlag(self::ACTION_FLAG_EDIT) === false) {
        return false;
    }
    return true;
}

public function canHold()
{
    $state = $this->getState();
    if ($this->isCanceled() || $this->isPaymentReview()
        || $state === self::STATE_COMPLETE || $state === self::STATE_CLOSED || $state === self::STATE_HOLDED) {
        return false;
    }

    if ($this->getActionFlag(self::ACTION_FLAG_HOLD) === false) {
        return false;
    }
    return true;
}

public function canUnhold()
{
    if ($this->getActionFlag(self::ACTION_FLAG_UNHOLD) === false || $this->isPaymentReview()) {
        return false;
    }
    return $this->getState() === self::STATE_HOLDED;
}

public function canComment()
{
    if ($this->getActionFlag(self::ACTION_FLAG_COMMENT) === false) {
        return false;
    }
    return true;
}

public function canShip()
{
    if ($this->canUnhold() || $this->isPaymentReview()) {
        return false;
    }

    if ($this->getIsVirtual() || $this->isCanceled()) {
        return false;
    }

    if ($this->getActionFlag(self::ACTION_FLAG_SHIP) === false) {
        return false;
    }

    foreach ($this->getAllItems() as $item) {
        if ($item->getQtyToShip()>0 && !$item->getIsVirtual()
            && !$item->getLockedDoShip())
        {
            return true;
        }
    }
    return false;
}

public function canEdit()
{
    if ($this->canUnhold()) {
        return false;
    }

    $state = $this->getState();
    if ($this->isCanceled() || $this->isPaymentReview()
        || $state === self::STATE_COMPLETE || $state === self::STATE_CLOSED) {
        return false;
    }

    if (!$this->getPayment()->getMethodInstance()->canEdit()) {
        return false;
    }
```

```
    if ($this->getActionFlag(self::ACTION_FLAG_EDIT) === false) {
        return false;
    }

    return true;
}

public function canReorder()
{
    return $this->_canReorder(false);
}

public function canReorderIgnoreSalable()
{
    return $this->_canReorder(true);
}

protected function _canReorder($ignoreSalable = false)
{
    if ($this->canUnhold() || $this->isPaymentReview() || !$this->getCustomerId()) {
        return false;
    }

    if ($this->getActionFlag(self::ACTION_FLAG_REORDER) === false) {
        return false;
    }

    $products = array();
    foreach ($this->getItemsCollection() as $item) {
        $products[] = $item->getProductId();
    }

    if (!empty($products)) {
        $productsCollection = Mage::getModel('catalog/product')->getCollection()
            ->setStoreId($this->getStoreId())
            ->addIdFilter($products)
            ->addAttributeToSelect('status')
            ->load();

        foreach ($productsCollection as $product) {
            if (!$product->isSalable()) {
                return false;
            }
        }
        */

        foreach ($products as $productId) {
            $product = Mage::getModel('catalog/product')
                ->setStoreId($this->getStoreId())
                ->load($productId);
            }
            if (!$product->getId() || (!$ignoreSalable && !$product->isSalable())) {
                return false;
            }
        }

    return true;
}

public function isPaymentReview()
{
    return $this->getState() === self::STATE_PAYMENT_REVIEW;
}

public function canReviewPayment()
{
    return $this->isPaymentReview() && $this->getPayment()->canReviewPayment();
}

public function canFetchPaymentReviewUpdate()
{
    return $this->isPaymentReview() && $this->getPayment()->canFetchTransactionInfo();
}

public function getConfig()
{
    return Mage::getSingleton('sales/order_config');
}
```

```
protected function _placePayment()
{
    $this->getPayment()->place();
    return $this;
}

public function getPayment()
{
    foreach ($this->getPaymentsCollection() as $payment) {
        if (!$payment->isDeleted()) {
            return $payment;
        }
    }
    return false;
}

public function setBillingAddress(Mage_Sales_Model_Order_Address $address)
{
    $old = $this->getBillingAddress();
    if (!empty($old)) {
        $address->setId($old->getId());
    }
    $this->addAddress($address->setAddressType('billing'));
    return $this;
}

public function setShippingAddress(Mage_Sales_Model_Order_Address $address)
{
    $old = $this->getShippingAddress();
    if (!empty($old)) {
        $address->setId($old->getId());
    }
    $this->addAddress($address->setAddressType('shipping'));
    return $this;
}

public function getBillingAddress()
{
    foreach ($this->getAddressesCollection() as $address) {
        if ($address->getAddressType()=='billing' && !$address->isDeleted()) {
            return $address;
        }
    }
    return false;
}

public function getShippingAddress()
{
    foreach ($this->getAddressesCollection() as $address) {
        if ($address->getAddressType()=='shipping' && !$address->isDeleted()) {
            return $address;
        }
    }
    return false;
}

public function setState($state, $status = false, $comment = '', $isCustomerNotified = null)
{
    return $this->_setState($state, $status, $comment, $isCustomerNotified, true);
}

protected function _setState($state, $status = false, $comment = '',
    $isCustomerNotified = null, $shouldProtectState = false)
{
    // attempt to set the specified state
    if ($shouldProtectState) {
        if ($this->isStateProtected($state)) {
            Mage::throwException(
                Mage::helper('sales')->__('The Order State "%s" must not be set manually.', $state)
            );
        }
    }
    $this->setData('state', $state);

    // add status history
    if ($status) {
        if ($status === true) {
            $status = $this->getConfig()->getStateDefaultStatus($state);
```

```
        }
        $this->setStatus($status);
        $history = $this->addStatusHistoryComment($comment, false); // no sense to set $status again
        $history->setIsCustomerNotified($isCustomerNotified); // for backwards compatibility
    }
    return $this;
}

public function isStateProtected($state)
{
    if (empty($state)) {
        return false;
    }
    return self::STATE_COMPLETE == $state || self::STATE_CLOSED == $state;
}

public function getStatusLabel()
{
    return $this->getConfig()->getStatusLabel($this->getStatus());
}

public function addStatusToHistory($status, $comment = '', $isCustomerNotified = false)
{
    $history = $this->addStatusHistoryComment($comment, $status)
        ->setIsCustomerNotified($isCustomerNotified);
    return $this;
}

public function addStatusHistoryComment($comment, $status = false)
{
    if (false === $status) {
        $status = $this->getStatus();
    } elseif (true === $status) {
        $status = $this->getConfig()->getStateDefaultStatus($this->getState());
    } else {
        $this->setStatus($status);
    }
    $history = Mage::getModel('sales/order_status_history')
        ->setStatus($status)
        ->setComment($comment)
        ->setEntityName($this->_historyEntityName);
    $this->addStatusHistory($history);
    return $history;
}

public function setHistoryEntityName( $entityName )
{
    $this->_historyEntityName = $entityName;
    return $this;
}

public function place()
{
    Mage::dispatchEvent('sales_order_place_before', array('order'=>$this));
    $this->_placePayment();
    Mage::dispatchEvent('sales_order_place_after', array('order'=>$this));
    return $this;
}

public function hold()
{
    if (!$this->canHold()) {
        Mage::throwException(Mage::helper('sales')->__('Hold action is not available.'));
    }
    $this->setHoldBeforeState($this->getState());
    $this->setHoldBeforeStatus($this->getStatus());
    $this->setState(self::STATE_HOLDED, true);
    return $this;
}

public function unhold()
{
    if (!$this->canUnhold()) {
        Mage::throwException(Mage::helper('sales')->__('Unhold action is not available.'));
    }
    $this->setState($this->getHoldBeforeState(), $this->getHoldBeforeStatus());
    $this->setHoldBeforeState(null);
    $this->setHoldBeforeStatus(null);
    return $this;
```

```
}

public function cancel()
{
    if ($this->canCancel()) {
        $this->getPayment()->cancel();
        $this->registerCancellation();

        Mage::dispatchEvent('order_cancel_after', array('order' => $this));
    }

    return $this;
}

public function registerCancellation($comment = '', $graceful = true)
{
    if ($this->canCancel() || $this->isPaymentReview()) {
        $cancelState = self::STATE_CANCELED;
        foreach ($this->getAllItems() as $item) {
            if ($cancelState != self::STATE_PROCESSING && $item->getQtyToRefund()) {
                if ($item->getQtyToShip() > $item->getQtyToCancel()) {
                    $cancelState = self::STATE_PROCESSING;
                } else {
                    $cancelState = self::STATE_COMPLETE;
                }
            }
            $item->cancel();
        }

        $this->setSubtotalCanceled($this->getSubtotal() - $this->getSubtotalInvoiced());
        $this->setBaseSubtotalCanceled($this->getBaseSubtotal() - $this->getBaseSubtotalInvoiced());

        $this->setTaxCanceled($this->getTaxAmount() - $this->getTaxInvoiced());
        $this->setBaseTaxCanceled($this->getBaseTaxAmount() - $this->getBaseTaxInvoiced());

        $this->setShippingCanceled($this->getShippingAmount() - $this->getShippingInvoiced());
        $this->setBaseShippingCanceled($this->getBaseShippingAmount() - $this->getBaseShippingInvoiced());

        $this->setDiscountCanceled(abs($this->getDiscountAmount()) - $this->getDiscountInvoiced());
        $this->setBaseDiscountCanceled(abs($this->getBaseDiscountAmount()) - $this->getBaseDiscountInvoiced());

        $this->setTotalCanceled($this->getGrandTotal() - $this->getTotalPaid());
        $this->setBaseTotalCanceled($this->getBaseGrandTotal() - $this->getBaseTotalPaid());

        $this->_setState($cancelState, true, $comment);
    } elseif (!$graceful) {
        Mage::throwException(Mage::helper('sales')->__('Order does not allow to be canceled.'));
    }
    return $this;
}

public function getTrackingNumbers()
{
    if ($this->getData('tracking_numbers')) {
        return explode(',', $this->getData('tracking_numbers'));
    }
    return array();
}

public function getShippingCarrier()
{
    $carrierModel = $this->getData('shipping_carrier');
    if (is_null($carrierModel)) {
        $carrierModel = false;
        $method = $this->getShippingMethod(true);
        if ($method instanceof Varien_Object) {
            $className = Mage::getStoreConfig('carriers/' . $method->getCarrierCode() . '/model');
            if ($className) {
                $carrierModel = Mage::getModel($className);
            }
        }
        $this->setData('shipping_carrier', $carrierModel);
    }
    return $carrierModel;
}

public function getShippingMethod($asObject = false)
{
    $shippingMethod = parent::getShippingMethod();
```

```
    if (!$asObject) {
        return $shippingMethod;
    } else {
        $segments = explode('_', $shippingMethod, 2);
        if (!isset($segments[1])) {
            $segments[1] = $segments[0];
        }
        list($carrierCode, $method) = $segments;
        return new Varien_Object(array(
            'carrier_code' => $carrierCode,
            'method'       => $method
        ));
    }
}

public function queueNewOrderEmail($forceMode = false)
{
    $storeId = $this->getStore()->getId();

    if (!Mage::helper('sales')->canSendNewOrderEmail($storeId)) {
        return $this;
    }

    // Get the destination email addresses to send copies to
    $copyTo = $this->_getEmails(self::XML_PATH_EMAIL_COPY_TO);
    $copyMethod = Mage::getStoreConfig(self::XML_PATH_EMAIL_COPY_METHOD, $storeId);

    // Start store emulation process
    $appEmulation = Mage::getSingleton('core/app_emulation');
    $initialEnvironmentInfo = $appEmulation->startEnvironmentEmulation($storeId);

    try {
        // Retrieve specified view block from appropriate design package (depends on emulated store)
        $paymentBlock = Mage::helper('payment')->getInfoBlock($this->getPayment())
            ->setIsSecureMode(true);
        $paymentBlock->getMethod()->setStore($storeId);
        $paymentBlockHtml = $paymentBlock->toHtml();
    } catch (Exception $exception) {
        // Stop store emulation process
        $appEmulation->stopEnvironmentEmulation($initialEnvironmentInfo);
        throw $exception;
    }

    // Stop store emulation process
    $appEmulation->stopEnvironmentEmulation($initialEnvironmentInfo);

    // Retrieve corresponding email template id and customer name
    if ($this->getCustomerIsGuest()) {
        $templateId = Mage::getStoreConfig(self::XML_PATH_EMAIL_GUEST_TEMPLATE, $storeId);
        $customerName = $this->getBillingAddress()->getName();
    } else {
        $templateId = Mage::getStoreConfig(self::XML_PATH_EMAIL_TEMPLATE, $storeId);
        $customerName = $this->getCustomerName();
    }

    $mailer = Mage::getModel('core/email_template_mailer');
    $emailInfo = Mage::getModel('core/email_info');
    $emailInfo->addTo($this->getCustomerEmail(), $customerName);
    if ($copyTo && $copyMethod == 'bcc') {
        // Add bcc to customer email
        foreach ($copyTo as $email) {
            $emailInfo->addBcc($email);
        }
    }
    $mailer->addEmailInfo($emailInfo);

    // Email copies are sent as separated emails if their copy method is 'copy'
    if ($copyTo && $copyMethod == 'copy') {
        foreach ($copyTo as $email) {
            $emailInfo = Mage::getModel('core/email_info');
            $emailInfo->addTo($email);
            $mailer->addEmailInfo($emailInfo);
        }
    }

    // Set all required params and send emails
    $mailer->setSender(Mage::getStoreConfig(self::XML_PATH_EMAIL_IDENTITY, $storeId));
    $mailer->setStoreId($storeId);
    $mailer->setTemplateId($templateId);
```

```
    $mailer->setTemplateParams(array(
        'order'        => $this,
        'billing'      => $this->getBillingAddress(),
        'payment_html' => $paymentBlockHtml
    ));

    $emailQueue = Mage::getModel('core/email_queue');
    $emailQueue->setEntityId($this->getId())
        ->setEntityType(self::ENTITY)
        ->setEventType(self::EMAIL_EVENT_NAME_NEW_ORDER)
        ->setIsForceCheck(!$forceMode);

    $mailer->setQueue($emailQueue)->send();

    $this->setEmailSent(true);
    $this->_getResource()->saveAttribute($this, 'email_sent');

    return $this;
}

public function sendNewOrderEmail()
{
    $this->queueNewOrderEmail(true);
    return $this;
}

public function queueOrderUpdateEmail($notifyCustomer = true, $comment = '', $forceMode = false)
{
    $storeId = $this->getStore()->getId();

    if (!Mage::helper('sales')->canSendOrderCommentEmail($storeId)) {
        return $this;
    }
    // Get the destination email addresses to send copies to
    $copyTo = $this->_getEmails(self::XML_PATH_UPDATE_EMAIL_COPY_TO);
    $copyMethod = Mage::getStoreConfig(self::XML_PATH_UPDATE_EMAIL_COPY_METHOD, $storeId);
    // Check if at least one recipient is found
    if (!$notifyCustomer && !$copyTo) {
        return $this;
    }

    // Retrieve corresponding email template id and customer name
    if ($this->getCustomerIsGuest()) {
        $templateId = Mage::getStoreConfig(self::XML_PATH_UPDATE_EMAIL_GUEST_TEMPLATE, $storeId);
        $customerName = $this->getBillingAddress()->getName();
    } else {
        $templateId = Mage::getStoreConfig(self::XML_PATH_UPDATE_EMAIL_TEMPLATE, $storeId);
        $customerName = $this->getCustomerName();
    }

    $mailer = Mage::getModel('core/email_template_mailer');
    if ($notifyCustomer) {
        $emailInfo = Mage::getModel('core/email_info');
        $emailInfo->addTo($this->getCustomerEmail(), $customerName);
        if ($copyTo && $copyMethod == 'bcc') {
            // Add bcc to customer email
            foreach ($copyTo as $email) {
                $emailInfo->addBcc($email);
            }
        }
        $mailer->addEmailInfo($emailInfo);
    }

    // Email copies are sent as separated emails if their copy method is
    // 'copy' or a customer should not be notified
    if ($copyTo && ($copyMethod == 'copy' || !$notifyCustomer)) {
        foreach ($copyTo as $email) {
            $emailInfo = Mage::getModel('core/email_info');
            $emailInfo->addTo($email);
            $mailer->addEmailInfo($emailInfo);
        }
    }

    // Set all required params and send emails
    $mailer->setSender(Mage::getStoreConfig(self::XML_PATH_UPDATE_EMAIL_IDENTITY, $storeId));
    $mailer->setStoreId($storeId);
    $mailer->setTemplateId($templateId);
    $mailer->setTemplateParams(array(
            'order'   => $this,
```

```
            'comment' => $comment,
            'billing' => $this->getBillingAddress()
        )
    );

    $emailQueue = Mage::getModel('core/email_queue');
    $emailQueue->setEntityId($this->getId())
        ->setEntityType(self::ENTITY)
        ->setEventType(self::EMAIL_EVENT_NAME_UPDATE_ORDER)
        ->setIsForceCheck(!$forceMode);
    $mailer->setQueue($emailQueue)->send();

    return $this;
}

public function sendOrderUpdateEmail($notifyCustomer = true, $comment = '')
{
    $this->queueOrderUpdateEmail($notifyCustomer, $comment, true);
     return $this;
 }

protected function _getEmails($configPath)
{
    $data = Mage::getStoreConfig($configPath, $this->getStoreId());
    if (!empty($data)) {
        return explode(',', $data);
    }
    return false;
}

/********************** ADDRESSES ***************************/

public function getAddressesCollection()
{
    if (is_null($this->_addresses)) {
        $this->_addresses = Mage::getResourceModel('sales/order_address_collection')
            ->setOrderFilter($this);

        if ($this->getId()) {
            foreach ($this->_addresses as $address) {
                $address->setOrder($this);
            }
        }
    }

    return $this->_addresses;
}

public function getAddressById($addressId)
{
    foreach ($this->getAddressesCollection() as $address) {
        if ($address->getId()==$addressId) {
            return $address;
        }
    }
    return false;
}

public function addAddress(Mage_Sales_Model_Order_Address $address)
{
    $address->setOrder($this)->setParentId($this->getId());
    if (!$address->getId()) {
        $this->getAddressesCollection()->addItem($address);
    }
    return $this;
}

public function getItemsCollection($filterByTypes = array(), $nonChildrenOnly = false)
{
    if (is_null($this->_items)) {
        $this->_items = Mage::getResourceModel('sales/order_item_collection')
            ->setOrderFilter($this);

        if ($filterByTypes) {
            $this->_items->filterByTypes($filterByTypes);
        }
        if ($nonChildrenOnly) {
            $this->_items->filterByParent();
        }
```

```
        if ($this->getId()) {
            foreach ($this->_items as $item) {
                $item->setOrder($this);
            }
        }
    }
    return $this->_items;
}

public function getItemsRandomCollection($limit = 1)
{
    return $this->_getItemsRandomCollection($limit);
}

public function getParentItemsRandomCollection($limit = 1)
{
    return $this->_getItemsRandomCollection($limit, true);
}

protected function _getItemsRandomCollection($limit, $nonChildrenOnly = false)
{
    $collection = Mage::getModel('sales/order_item')->getCollection()
        ->setOrderFilter($this)
        ->setRandomOrder();

    if ($nonChildrenOnly) {
        $collection->filterByParent();
    }
    $products = array();
    foreach ($collection as $item) {
        $products[] = $item->getProductId();
    }

    $productsCollection = Mage::getModel('catalog/product')
        ->getCollection()
        ->addIdFilter($products)
        ->setVisibility(Mage::getSingleton('catalog/product_visibility')->getVisibleInSiteIds())
        /* Price data is added to consider item stock status using price index */
        ->addPriceData()
        ->setPageSize($limit)
        ->load();

    foreach ($collection as $item) {
        $product = $productsCollection->getItemById($item->getProductId());
        if ($product) {
            $item->setProduct($product);
        }
    }

    return $collection;
}

public function getAllItems()
{
    $items = array();
    foreach ($this->getItemsCollection() as $item) {
        if (!$item->isDeleted()) {
            $items[] =  $item;
        }
    }
    return $items;
}

public function getAllVisibleItems()
{
    $items = array();
    foreach ($this->getItemsCollection() as $item) {
        if (!$item->isDeleted() && !$item->getParentItemId()) {
            $items[] =  $item;
        }
    }
    return $items;
}

public function getItemById($itemId)
{
    return $this->getItemsCollection()->getItemById($itemId);
}
```

```
    public function getItemByQuoteItemId($quoteItemId)
    {
        foreach ($this->getItemsCollection() as $item) {
            if ($item->getQuoteItemId()==$quoteItemId) {
                return $item;
            }
        }
        return null;
    }

    public function addItem(Mage_Sales_Model_Order_Item $item)
    {
        $item->setOrder($this);
        if (!$item->getId()) {
            $this->getItemsCollection()->addItem($item);
        }
        return $this;
    }

    public function isNominal()
    {
        foreach ($this->getAllVisibleItems() as $item) {
            if ('0' == $item->getIsNominal()) {
                return false;
            }
        }
        return true;
    }

/********************** PAYMENTS ***************************/

    public function getPaymentsCollection()
    {
        if (is_null($this->_payments)) {
            $this->_payments = Mage::getResourceModel('sales/order_payment_collection')
                ->setOrderFilter($this);

            if ($this->getId()) {
                foreach ($this->_payments as $payment) {
                    $payment->setOrder($this);
                }
            }
        }
        return $this->_payments;
    }

    public function getAllPayments()
    {
        $payments = array();
        foreach ($this->getPaymentsCollection() as $payment) {
            if (!$payment->isDeleted()) {
                $payments[] =  $payment;
            }
        }
        return $payments;
    }


    public function getPaymentById($paymentId)
    {
        foreach ($this->getPaymentsCollection() as $payment) {
            if ($payment->getId()==$paymentId) {
                return $payment;
            }
        }
        return false;
    }

    public function addPayment(Mage_Sales_Model_Order_Payment $payment)
    {
        $payment->setOrder($this)
            ->setParentId($this->getId());
        if (!$payment->getId()) {
            $this->getPaymentsCollection()->addItem($payment);
        }
        return $this;
    }
```

```
    public function setPayment(Mage_Sales_Model_Order_Payment $payment)
    {
        if (!$this->getIsMultiPayment() && ($old = $this->getPayment())) {
            $payment->setId($old->getId());
        }
        $this->addPayment($payment);
        return $payment;
    }

/********************** STATUSES ***************************/

    public function getStatusHistoryCollection($reload=false)
    {
        if (is_null($this->_statusHistory) || $reload) {
            $this->_statusHistory = Mage::getResourceModel('sales/order_status_history_collection')
                ->setOrderFilter($this)
                ->setOrder('created_at', 'desc')
                ->setOrder('entity_id', 'desc');

            if ($this->getId()) {
                foreach ($this->_statusHistory as $status) {
                    $status->setOrder($this);
                }
            }
        }
        return $this->_statusHistory;
    }

    public function getAllStatusHistory()
    {
        $history = array();
        foreach ($this->getStatusHistoryCollection() as $status) {
            if (!$status->isDeleted()) {
                $history[] =  $status;
            }
        }
        return $history;
    }

    public function getVisibleStatusHistory()
    {
        $history = array();
        foreach ($this->getStatusHistoryCollection() as $status) {
            if (!$status->isDeleted() && $status->getComment() && $status->getIsVisibleOnFront()) {
                $history[] =  $status;
            }
        }
        return $history;
    }

    public function getStatusHistoryById($statusId)
    {
        foreach ($this->getStatusHistoryCollection() as $status) {
            if ($status->getId()==$statusId) {
                return $status;
            }
        }
        return false;
    }

    public function addStatusHistory(Mage_Sales_Model_Order_Status_History $history)
    {
        $history->setOrder($this);
        $this->setStatus($history->getStatus());
        if (!$history->getId()) {
            $this->getStatusHistoryCollection()->addItem($history);
        }
        return $this;
    }

    public function getRealOrderId()
    {
        $id = $this->getData('real_order_id');
        if (is_null($id)) {
            $id = $this->getIncrementId();
        }
        return $id;
    }
```

```
public function getOrderCurrency()
{
    if (is_null($this->_orderCurrency)) {
        $this->_orderCurrency = Mage::getModel('directory/currency')->load($this->getOrderCurrencyCode());
    }
    return $this->_orderCurrency;
}

public function formatPrice($price, $addBrackets = false)
{
    return $this->formatPricePrecision($price, 2, $addBrackets);
}

public function formatPricePrecision($price, $precision, $addBrackets = false)
{
    return $this->getOrderCurrency()->formatPrecision($price, $precision, array(), true, $addBrackets);
}

public function formatPriceTxt($price)
{
    return $this->getOrderCurrency()->formatTxt($price);
}

public function getBaseCurrency()
{
    if (is_null($this->_baseCurrency)) {
        $this->_baseCurrency = Mage::getModel('directory/currency')->load($this->getBaseCurrencyCode());
    }
    return $this->_baseCurrency;
}

public function getStoreCurrency()
{
    return $this->getData('store_currency');
}

public function formatBasePrice($price)
{
    return $this->formatBasePricePrecision($price, 2);
}

public function formatBasePricePrecision($price, $precision)
{
    return $this->getBaseCurrency()->formatPrecision($price, $precision);
}

public function isCurrencyDifferent()
{
    return $this->getOrderCurrencyCode() != $this->getBaseCurrencyCode();
}

public function getTotalDue()
{
    $total = $this->getGrandTotal()-$this->getTotalPaid();
    $total = Mage::app()->getStore($this->getStoreId())->roundPrice($total);
    return max($total, 0);
}

public function getBaseTotalDue()
{
    $total = $this->getBaseGrandTotal()-$this->getBaseTotalPaid();
    $total = Mage::app()->getStore($this->getStoreId())->roundPrice($total);
    return max($total, 0);
}

public function getData($key='', $index=null)
{
    if ($key == 'total_due') {
        return $this->getTotalDue();
    }
    if ($key == 'base_total_due') {
        return $this->getBaseTotalDue();
    }
    return parent::getData($key, $index);
}

public function getInvoiceCollection()
{
    if (is_null($this->_invoices)) {
```

```
        $this->_invoices = Mage::getResourceModel('sales/order_invoice_collection')
            ->setOrderFilter($this);

        if ($this->getId()) {
            foreach ($this->_invoices as $invoice) {
                $invoice->setOrder($this);
            }
        }
    }
    return $this->_invoices;
}

public function getShipmentsCollection()
{
    if (empty($this->_shipments)) {
        if ($this->getId()) {
            $this->_shipments = Mage::getResourceModel('sales/order_shipment_collection')
                ->setOrderFilter($this)
                ->load();
        } else {
            return false;
        }
    }
    return $this->_shipments;
}

public function getCreditmemosCollection()
{
    if (empty($this->_creditmemos)) {
        if ($this->getId()) {
            $this->_creditmemos = Mage::getResourceModel('sales/order_creditmemo_collection')
                ->setOrderFilter($this)
                ->load();
        } else {
            return false;
        }
    }
    return $this->_creditmemos;
}

public function getTracksCollection()
{
    if (empty($this->_tracks)) {
        $this->_tracks = Mage::getResourceModel('sales/order_shipment_track_collection')
            ->setOrderFilter($this);

        if ($this->getId()) {
            $this->_tracks->load();
        }
    }
    return $this->_tracks;
}

public function hasInvoices()
{
    return $this->getInvoiceCollection()->count();
}

public function hasShipments()
{
    $result = false;
    $shipmentsCollection = $this->getShipmentsCollection();
    if ($shipmentsCollection) {
        $result = (bool)$shipmentsCollection->count();
    }
    return $result;
}

public function hasCreditmemos()
{
    $result = false;
    $creditmemosCollection = $this->getCreditmemosCollection();
    if ($creditmemosCollection) {
        $result = (bool)$creditmemosCollection->count();
    }
    return $result;
}
```

```
public function getRelatedObjects()
{
    return $this->_relatedObjects;
}

public function getCustomerName()
{
    if ($this->getCustomerFirstname()) {
        $customerName = Mage::helper('customer')->getFullCustomerName($this);
    } else {
        $customerName = Mage::helper('sales')->__('Guest');
    }
    return $customerName;
}

public function addRelatedObject(Mage_Core_Model_Abstract $object)
{
    $this->_relatedObjects[] = $object;
    return $this;
}

public function getCreatedAtFormated($format)
{
    return Mage::helper('core')->formatDate($this->getCreatedAtStoreDate(), $format, true);
}

public function getEmailCustomerNote()
{
    if ($this->getCustomerNoteNotify()) {
        return $this->getCustomerNote();
    }
    return '';
}

protected function _beforeSave()
{
    parent::_beforeSave();
    $this->_checkState();
    if (!$this->getId()) {
        $store = $this->getStore();
        $name = array($store->getWebsite()->getName(),$store->getGroup()->getName(),$store->getName());
        $this->setStoreName(implode("\n", $name));
    }

    if (!$this->getIncrementId()) {
        $incrementId = Mage::getSingleton('eav/config')
            ->getEntityType('order')
            ->fetchNewIncrementId($this->getStoreId());
        $this->setIncrementId($incrementId);
    }

    if (!$this->getId()) {
        $itemsCount = 0;
        foreach ($this->getAllItems() as $item) {
            $parent = $item->getQuoteParentItemId();
            if ($parent && !$item->getParentItem()) {
                $item->setParentItem($this->getItemByQuoteItemId($parent));
            } elseif (!$parent) {
                $itemsCount++;
            }
        }
        // Set items count
        $this->setTotalItemCount($itemsCount);
    }
    if ($this->getCustomer()) {
        $this->setCustomerId($this->getCustomer()->getId());
    }

    if ($this->hasBillingAddressId() && $this->getBillingAddressId() === null) {
        $this->unsBillingAddressId();
    }

    if ($this->hasShippingAddressId() && $this->getShippingAddressId() === null) {
        $this->unsShippingAddressId();
    }

    $this->setData('protect_code', substr(md5(uniqid(mt_rand(), true) . ':' . microtime(true)), 5, 6));
    return $this;
}
```

```
protected function _checkState()
{
    if (!$this->getId()) {
        return $this;
    }

    $userNotification = $this->hasCustomerNoteNotify() ? $this->getCustomerNoteNotify() : null;

    if (!$this->isCanceled()
        && !$this->canUnhold()
        && !$this->canInvoice()
        && !$this->canShip()) {
        if (0 == $this->getBaseGrandTotal() || $this->canCreditmemo()) {
            if ($this->getState() !== self::STATE_COMPLETE) {
                $this->_setState(self::STATE_COMPLETE, true, '', $userNotification);
            }
        }
        elseif (floatval($this->getTotalRefunded()) || (!$this->getTotalRefunded()
            && $this->hasForcedCanCreditmemo())
        ) {
            if ($this->getState() !== self::STATE_CLOSED) {
                $this->_setState(self::STATE_CLOSED, true, '', $userNotification);
            }
        }
    }

    if ($this->getState() == self::STATE_NEW && $this->getIsInProcess()) {
        $this->setState(self::STATE_PROCESSING, true, '', $userNotification);
    }
    return $this;
}

protected function _afterSave()
{
    if (null !== $this->_addresses) {
        $this->_addresses->save();
        $billingAddress = $this->getBillingAddress();
        $attributesForSave = array();
        if ($billingAddress && $this->getBillingAddressId() != $billingAddress->getId()) {
            $this->setBillingAddressId($billingAddress->getId());
            $attributesForSave[] = 'billing_address_id';
        }

        $shippingAddress = $this->getShippingAddress();
        if ($shippingAddress && $this->getShippigAddressId() != $shippingAddress->getId()) {
            $this->setShippingAddressId($shippingAddress->getId());
            $attributesForSave[] = 'shipping_address_id';
        }

        if (!empty($attributesForSave)) {
            $this->_getResource()->saveAttribute($this, $attributesForSave);
        }

    }
    if (null !== $this->_items) {
        $this->_items->save();
    }
    if (null !== $this->_payments) {
        $this->_payments->save();
    }
    if (null !== $this->_statusHistory) {
        $this->_statusHistory->save();
    }
    foreach ($this->getRelatedObjects() as $object) {
        $object->save();
    }
    return parent::_afterSave();
}

public function getStoreGroupName()
{
    $storeId = $this->getStoreId();
    if (is_null($storeId)) {
        return $this->getStoreName(1); // 0 - website name, 1 - store group name, 2 - store name
    }
    return $this->getStore()->getGroup()->getName();
}
```

```
  public function reset()
  {
    $this->unsetData();
    $this->_actionFlag = array();
    $this->_addresses = null;
    $this->_items = null;
    $this->_payments = null;
    $this->_statusHistory = null;
    $this->_invoices = null;
    $this->_tracks = null;
    $this->_shipments = null;
    $this->_creditmemos = null;
    $this->_relatedObjects = array();
    $this->_orderCurrency = null;
    $this->_baseCurrency = null;

    return $this;
  }

  public function getIsNotVirtual()
  {
    return !$this->getIsVirtual();
  }

  public function getFullTaxInfo()
  {
    $rates = Mage::getModel('tax/sales_order_tax')->getCollection()->loadByOrder($this)->toArray();
    return Mage::getSingleton('tax/calculation')->reproduceProcess($rates['items']);
  }

  public function prepareInvoice($qtys = array())
  {
    $invoice = Mage::getModel('sales/service_order', $this)->prepareInvoice($qtys);
    return $invoice;
  }

  public function prepareShipment($qtys = array())
  {
    $shipment = Mage::getModel('sales/service_order', $this)->prepareShipment($qtys);
    return $shipment;
  }

  public function isCanceled()
  {
    return ($this->getState() === self::STATE_CANCELED);
  }

  protected function _beforeDelete()
  {
    $this->_protectFromNonAdmin();
    return parent::_beforeDelete();
  }
}
```

Скрипт створення таблиць бази даних

```
CREATE DATABASE IF NOT EXISTS `books`
  DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;
USE books;

CREATE TABLE IF NOT EXISTS `BOOK` (
  `ID_book`    INT NOT NULL,
  `Title`      VARCHAR(150),
  `Author`     VARCHAR(100),
  `Publisher`   VARCHAR(100),
  `Genre`      VARCHAR(100),
  `Price`       FLOAT,
  `SUPPLIER_ID` INT,
PRIMARY KEY (`ID_book`))

ENGINE=MyISAM COMMENT='Books in stock'
  DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;

CREATE TABLE IF NOT EXISTS `CUSTOMER` (
  `ID_customer` INT NOT NULL,
  `FirstName`  VARCHAR(100),
  `LastName`   VARCHAR(100),
  `Email`      VARCHAR(255),
```

```
  `Phone`      VARCHAR(25),
  `Adress`     VARCHAR(40),
PRIMARY KEY (`ID_customer`))

ENGINE=MyISAM COMMENT='Clients info'
  DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;

CREATE TABLE IF NOT EXISTS `SUPPLIER` (
  `ID_supplier` INT NULL,
  `Name`    VARCHAR(100),
  `Contact` VARCHAR(100),
  `Email`   VARCHAR(255),
  `Phone`   VARCHAR(25),
  `Adress`  VARCHAR(200),
PRIMARY KEY (`ID_supplier`))

ENGINE=MyISAM COMMENT='Supplier info'
  DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;

CREATE TABLE IF NOT EXISTS `ORDERS` (
  `ID_orders`  INT NOT NULL,
  `Price`          FLOAT,
  `Date`           DATE,
  `CUSTOMER_ID`    INT,
PRIMARY KEY(`ID_orders`))

ENGINE=MyISAM COMMENT='Orders'
  DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;

CREATE TABLE IF NOT EXISTS `OrderItem` (
  `ID_orderitem`    INT NOT NULL,
  `Quantity`  INT,
  `UnitPrice` FLOAT,
  `ORDER_ID`  INT,
  `BOOK_ID` INT,
PRIMARY KEY (`ID_orderitem`))

ENGINE=MyISAM COMMENT='Items for each order'
  DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;

ALTER TABLE `BOOK`
       ADD CONSTRAINT `FK_REFERENCE_2`
              FOREIGN KEY(`SUPPLIED_ID`)
                     REFERENCES `SUPPLIER` (`ID_supplier`);

ALTER TABLE `ORDERS`
       ADD CONSTRAINT `FK_REFERENCE_1`
              FOREIGN KEY(`CUSTOMER_ID`)
                     REFERENCES `CUSTOMER` (`ID_customer`);

ALTER TABLE `OrderItem`
       ADD CONSTRAINT `FK_REFERENCE_3`
              FOREIGN KEY(`ORDER_ID`)
                     REFERENCES `ORDER` (`ID_orders`);

ALTER TABLE `OrderItem`
       ADD CONSTRAINT `FK_REFERENCE_4`
              FOREIGN KEY(`BOOK_ID`)
                     REFERENCES `BOOK` (`ID_book`);
```

ДОДАТОК Б

Відгук керівника економічного розділу

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім'я файла	**Опис**
Пояснювальні документи	
Кручок Д.В Кваліфікаційна робота.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Кручок Д.В Кваліфікаційна робота.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Knigosklad.com.ua.zip	Архів. Містить коди програми і откомпільовану програму
Презентація	
Кручок Д.В Презентація.ppt	Презентація дипломного проекту