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 *Качана Ігоря Сергійовича*
(ПІБ)

академічної групи *121-17-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка веб-орієнтованого програмного забезпечення для автоматизації діяльності підприємства в сфері електронної комерції*

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

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-17-1* (група) *Качана Ігоря Сергійовича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-орієнтованого програмного забезпечення для автоматизації діяльності підприємства в сфері електронної комерції*

затверджена наказом ректора НТУ «ДП» від *07.06.2021* № *317-С*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

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

Завдання прийняв до виконання _____ *Качан І.С*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка: 102 с., 35 рис., 3 дод., 39 джерел.

Об'єкт розробки: веб-орієнтована інформаційна система для автоматизації діяльності підприємства в сфері електронної комерції.

Мета кваліфікаційної роботи: розробка програмного забезпечення веб-орієнтованої інформаційної системи для автоматизації діяльності компанії в сфері електронної комерції.

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 проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформу для розробки, виконано проектування і розробку веб-орієнтованої інформаційної системи, описана робота системи, алгоритм і структура його функціонування, а також виклик та завантаження додатку, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню додатку та розраховано час на його створення.

Практичне значення полягає у створенні веб-орієнтованої інформаційної системи, що забезпечує: об’єднання елементів прямого маркетингу та традиційної торгівлі; надання універсального інструменту управління візуальним контентом та адміністративної частини проекту; забезпечення відвідувачам сайту максимальної зручності роботи із системою, простоти і комфортності доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту; підвищення ефективності роботи компанії в цілому, за рахунок автоматизації управління бізнес-процесами.

Актуальність розробки інформаційної системи для автоматизації діяльності компанії в сфері електронної комерції не викликає сумніву та визначається великим попитом на подібні системи, оскільки електронна комерція робить торгівлю більш гнучкою, зрозумілою, відкритою й стандартизованою, а також надає значні переваги (що особливо відчутні в умовах COVID-19): мобільність, безконтактність, ціна, масштабованість, оперативність реагування на зміни ринку товарів і послуг, швидкість і ефективність здійснення комерційних операцій і т.ін.

Список ключових слів: ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРНЕТ-МАГАЗИН, БАЗА ДАНИХ, PHP, HTML, CSS, JQUERY.

ABSTRACT

Explanatory note: 102 pp., 35 fig., 3 extra, 39 sources.

The object of development: web-oriented information system for automating the activities of an enterprise in the field of e-commerce.

The purpose of the diploma project: software development web-based information system to automate the company's activities in the field of e-commerce.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the scope of its application, substantiates the relevance of the topic and clarifies the formulation of the problem.

In the first chapter, the subject area is analyzed, the relevance of the task and the purpose of development are determined, the statement of the problem is formulated, the requirements for software implementation, technologies and software are indicated.

In the second section, the available solutions are analyzed, a platform for development is selected, the design and development of a web-oriented information system is carried out, the operation of the system, the algorithm and structure of its functioning, as well as the call and loading of the application are described, the input and output data are determined, the composition of the parameters of the technical means.

In the economic section, the labor intensity of the developed information system is determined, the cost of work on creating an application is calculated and the time for its creation is calculated.

The practical value lies in the creation of a web-oriented information system that provides: combining elements of direct marketing and traditional trade; providing a universal tool for managing visual content and the administrative part of the project; providing site visitors with the maximum convenience of working with the system, simplicity and comfort of access to the catalog of goods due to the optimal parameters of visualization of its content; increasing the efficiency of the company as a whole, by automating the management of business processes.

The relevance of the development of an information system to automate the company's activities in the field of e-commerce is beyond doubt and is determined by the great demand for such systems, since e-commerce makes trade more flexible, understandable, open and standardized, and also provides significant advantages (especially tangible in the context of COVID-19): mobility, contactlessness, price, scalability, responsiveness to changes in the market for goods and services, speed and efficiency of commercial operations, etc.

List of keywords: INFORMATION SYSTEM, ONLINE STORE, DATABASE, PHP, HTML, CSS, JQUERY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - Програмне забезпечення

ІС - Інформаційна система

ОС - Операційна система

БД - База даних

СУБД - Система управління базами даних

CMS - Система управління контентом

CSS - Cascading Style Sheets

SQL - Structured Query Language

HTML - Hyper Text Markup Language

ВСТУП

Стрімкий розвиток інформаційних технологій і систем сприяє активному розвитку електронної комерції в багатьох сферах підприємницької діяльності, що призводе до суттєвих змін у взаємодії суб’єктів бізнесу. Для вітчизняних підприємств, в умовах посиленої конкуренції між суб'єктами комерційної діяльності, електронний бізнес виступає ефективним інструментом сучасного інтерактивного бізнесу: для ведення розрахунків з клієнтами, розширення обсягів ринків збуту продукції, товарів, послуг, пошуку нових партнерів, створення позитивного іміджу. Електронна комерція як система включає в себе: суб’єкти електронного бізнесу (виробники, продавці, посередники, покупці, споживачі), процеси (реалізація продукції та послуг, маркетинг, розрахункові операції тощо) та мережі (як локальні, так і глобальні). Всі компоненти електронної комерції перебувають у взаємозв’язку завдяки використанню телекомунікаційних технологій і глобальної мережі Інтернет як інструменту організації єдиного інформаційного простору електронного бізнесу.

Актуальність розробки інформаційної системи для автоматизації діяльності підприємства в сфері електронної комерції не викликає сумніву та визначається великим попитом на подібні системи, оскільки електронна комерція робить торгівлю більш гнучкою, зрозумілою, відкритою й стандартизованою, а також надає значні переваги (що особливо відчутні в умовах COVID-19): мобільність, безконтактність, ціна, масштабованість, оперативність реагування на зміни ринку товарів і послуг, швидкість і ефективність здійснення комерційних операцій і т.ін.

Об'єктом дослідження є веб-орієнтована інформаційна система для автоматизації діяльності підприємства в сфері електронної комерції.

Мета кваліфікаційної роботи: розробка програмного забезпечення інформаційної системи маркетингової діяльності підприємства для підвищення ефективності його роботи за рахунок автоматизації діяльності в сфері електронної комерції.

Для вирішення поставленої мети необхідно вирішити наступні задачі:

- розглянути особливості електронної комерції в контексті розвитку національної економіки;
- сформувати основні вимоги щодо розробленого програмного продукту (вимоги до функціональних характеристик, інформаційної безпеки, складу та параметрів технічних засобів, інформаційної та програмної сумісності, змісту інформаційного наповнення системи);
- здійснити опис та обрати архітектуру, технології, мови програмування для створення програмного продукту;
- розглянути функціональне призначення програми;
- здійснити опис структури програми та алгоритмів її функціонування;
- здійснити опис розробленого програмного продукту;
- здійснити розрахунок трудомісткості та витрат на розробку програмного продукту.

Практичне значення полягає у створенні веб-орієнтованої інформаційної системи, що забезпечує: об’єднання елементів прямого маркетингу та традиційної торгівлі; надання універсального інструменту управління візуальним контентом та адміністративної частини проекту; забезпечення відвідувачам сайту максимальної зручності роботи із системою, простоти і комфортності доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту; підвищення ефективності роботи компанії в цілому, за рахунок автоматизації управління бізнес-процесами.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Особливості електронної комерції

Сучасний період розвитку національної економіки характеризується посиленням конкуренції між суб'єктами комерційної діяльності, що обумовлює необхідність оперативного реагування на зміни ринку товарів і послуг та швидкого і ефективного здійснення комерційних операцій. Перспективним напрямком ведення сучасного інтерактивного бізнесу в умовах жорсткої конкуренції стала електронна комерція, яка охоплює всі види торгівлі через Internet.

Електронна комерція (від англ, e-commerce) – це сфера цифрової економіки, що включає всі фінансові та торгові транзакції, які проводяться за допомогою комп'ютерних мереж, та бізнес-процеси, пов'язані з проведенням цих транзакцій [5,26,30].

Основні види електронної комерції:

Бізнес-до-бізнесу (Business-to-Business - B2B). B2B – це тип електронної комерція між компаніями, що має справу з відносинами між видами комерційної діяльності та характеризує продаж оптових партій товару від виробника дилеру (приблизно 80% електронної комерції належать до цього типу). Дилер поставляє товар дрібним оптом інтернет-магазинам, звідки він надходить кінцевому споживачеві. Більш загальні B2B приклади й кращі практичні моделі: IBM, Hewlett Packard, Cisco and Dell.

Споживач-до-споживача (Consumer-to-Consumer - C2C). C2C - це тип електронної комерції, що характеризує торгівлю між приватними особами або споживачами (приклади: торговий майданчик OLX, eBay, і подібні до них, де

людина, може виставити на продаж будь який товар). C2C входить принаймні в три форми:

- аукціони на кшталт eBay, що дозволяє інтерактивно пропонувати ціну в реальному часу;
- однорангові системи, типу моделі Napster (протокол спільного використання файлів між користувачами в системах типу IRC) і пізніші моделі обміну грошей;
- оголошення тематичних категорій у портальних сайтах типу Тематичних категорій Excite і eWanted (діалоговий, мережний ринок, де покупці і продавці можуть вести переговори).

Мобільна торгівля (m-commerce). Мобільна торгівля характеризує закупівлю й продаж товарів і послуг через бездротову технологію (кишенькові пристрої типу мобільних телефонів і особистих цифрових помічників).

Бізнес-до-споживача (Business-to-Consumer - B2C). B2C характеризує торгівлю між компаніями й споживачами, включає збирання інформації клієнтами; купівлю фізичних речей чи інформаційних/електронних товарів; і, для інформаційних товарів, одержування товару (програми, електронної книги) по електронній мережі. Це друга по величині й найперша форма електронної комерції. Приклади B2C моделей – мережні компанії продажу в роздріб типу Drugstore.com, Amazon.com, Beyond.com. B2C електронна комерція зменшує ціну угод, збільшуючи доступ споживачів до інформації й дозволяючи споживачам знайти саму конкурентоспроможну ціну за товар або послугу. B2C електронна комерція також зменшує ринкові бар'єри входу, тому що вартість створення й розкручування сайту набагато менша ніж установка структури фірми. Основу електронної комерції типу B2C складають онлайнові компанії, що здійснюють свій механізм бізнесу повністю через мережу Internet. Найбільш розповсюдженими онлайновими компаніями є інтернет-магазини.

Мережа Internet використовується як для реалізації товарів, тобто продажу деяких видів товарів безпосередньо з веб-сайтів продавців, так і для інформаційно-рекламної підтримки звичайного (оффлайнового) бізнесу. У

другому випадку через мережу Інтернет встановлюються контакти між партнерами, здійснюється обмін, що випереджає угоду, інформацією (запитами, офертами, специфікаціями, чернетками копій контракту й ін.), що істотно прискорює процес здійснення угоди між сторонами.

Інтернет-магазин (електронний магазин, онлайновий магазин) – це програмний комплекс, який дозволяє продавати товари чи послуги через мережу Internet та автоматизувати управління бізнес-процесами. Електронні магазини об’єднують елементи прямого маркетингу та традиційної торгівлі.

Основні функції інтернет-магазину:

- персоналізацію споживачів або відвідувачів;
- надання повної інформації про надані товари та послуги;
- сервісна підтримка;
- приймання та обробку замовлень;
- взаємодія з платіжною системою банку для проведення платежів та розрахунків за придбаний товар;
- порядок та своєчасність постачання;
- збір та аналіз статистичної інформації.

Додаткові можливості інтернет-магазину:

- зробити закладку на сайт для використання в майбутньому;
- здійснити підписку на розсилання інформації про нові надходження;
- залишити замовлення на товари, які відсутні в даний момент;
- запропонувати прийняти участь у форумі щодо діяльності даного магазину;
- заповнити анкету для персоніфікації і т.ін.

Відзначимо основні елементи, на основі яких будують взаємовідносини з покупцями служби електронного магазину:

- каталог товарів (вся необхідна інформація для здійснення покупки);

– віртуальна корзина (список товарів, які вибрав покупець в інтернет-магазині з вказівкою суми покупок) з можливістю управління вмістом до моменту відправлення заявки;
– процедура реєстрації. Деякі електронні магазини пропонують реєструватися до початку роботи з віртуальною корзиною (відвідувачі мають можливість отримувати персоналізовану інформацію та послуги), а деякі після формування заказу.

Переваги інтернет-магазину в порівнянні з реальним магазином:

– зменшення чисельності персоналу за рахунок скорочення обсягу взаємодії із клієнтами;
– персоналізований підхід до кожного відвідувача;
– інтерактивність;
– економія коштів (оренда дискового простору й розміщення «електронної вітрини» дешевше й простіше оренди торгівельних приміщень і розміщення товарів на полках, відсутність потреби в касовому обслуговуванні);
– ефективний спосіб маркетингового дослідження для врахування даних результатів у своїй роботі, тим більше, що сьогодні ця послуга досить дорога в маркетингових агентствах;
– низький бар'єр входу на ринок інтернет-торгівлі;
– економія часу для споживачів (здійснення покупки не прив'язано до місця знаходження покупця і часу здійснення покупки, вибір й оцінка властивостей товару відбувається набагато швидше й зручніше ніж у звичайному магазині, використовуючи фільтри по параметрам);
– відточена логістика та здатність обслуговування великої кількості клієнтів одночасно;
– підтримка різних способів оплати та методів замовлень (як наприклад, через веб-сторінку, електронною поштою, факсом, телефоном тощо);
– здійснення функції електронного контролю виконання замовлень та доставки;

– можливість необмеженого збільшення асортименту продукції й інформативності (купівельний рейтинг, відгуки про товар, статті про товар і т.ін.);

До недоліків інтернет-магазинів можна віднести наступне:

– необхідність гарних каналів зв'язку й апаратно-програмного забезпечення;
– збільшення ціни товару за рахунок додавання частки доставки до собівартості продукції;
– присутній "синдром недовіри", оскільки в інтернет-торгівлі покупець менш захищений від несумлінного продавця, та й постійно присутній в Інтернеті хакерский фактор істотно підвищує ризик угоди;
– товар не може бути оцінений візуально та на дотик, однак цей недолік може компенсуватися великою кількістю інформації з бази даних, яку не завжди зможе надати продавець в традиційному магазині;
– можливі проблеми гарантії й супроводу.

Український ринок інтернет-торгівлі перебуває в стадії становлення: підприємці використають найбільш прості й дешеві способи присутності в мережі, обсяги торгівлі ще невеликі, не вирішені багато організаційних питань. У цій ситуації необхідно активізувати розвиток інформаційних технологій. Останнім часом простежуються тенденції впровадження інноваційних технологій (бізнес активно використовує VR, створюючи 3D-фото і відео-огляди) в онлайн продажу. Збільшується частота використання штучного інтелекту і роботів, що замінюють консультанта і закривають до 90% потреб клієнтів в автоматичному режимі.

Економічна криза підкреслила значення інтернет-магазинів для дистрибуції техніки й деяких товарів – продажу в мережі постраждали менше, ніж у традиційному роздробі, за рахунок чого виросла частка онлайнових торговельних площадок у загальній структурі продажів, що роблять торгівлю зрозумілою, відкритою й стандартизованою.

Переклад традиційної торгівлі в мережу Internet робить її більш гнучкою, тому що електронна торгівля, оперуючи цифровою інформацією в комп'ютерних мережах, полегшує співробітництво людей.

Підбиваючи підсумок, можна зробити висновок, що вітчизняний бізнес став пильніше придивлятися до онлайну і тим можливостям, які відкриває торгівля через Internet. Тригером цього послужив назріваюча економічна криза, яка змушує людей економити і шукати більш вигідні ціни, а введений карантин оголив основні переваги інтернет-торгівлі - мобільність, безконтактність, ціна, масштабованість.

1.1.2. Класифікація веб-сайтів

На даний час не існує єдиної класифікації типів веб-сайтів. Як правило, сайти розрізняють за задачами, що вирішуються. На рис.1.1 наведено один із варіантів класифікації веб-сайтів.

Розглянемо більш докладніше класифікацію сайтів мережі Інтернет [11].

За доступністю сервісів.

Відкриті - всі сервіси повністю доступні для будь-яких відвідувачів і користувачів.

Напіввідкриті - для доступу необхідно зареєструватися.

Закриті - повністю закриті сайти організацій (у тому числі корпоративні сайти), особисті сайти приватних осіб.

За природою змісту.

Статичні - розуміються сайти, які складаються із незмінних HTML-сторінок. Статичні сайти не здатні реагувати на введену користувачем інформацію, тому є пасивними.

Позитивними сторонами статичного сайту є: швидкість завантаження; економність використання; незначне навантаження на сервер; невимогливість до ресурсів хостингу; легкість підтримки та перенесення на інший сервер; простота створення HTML-сторінок

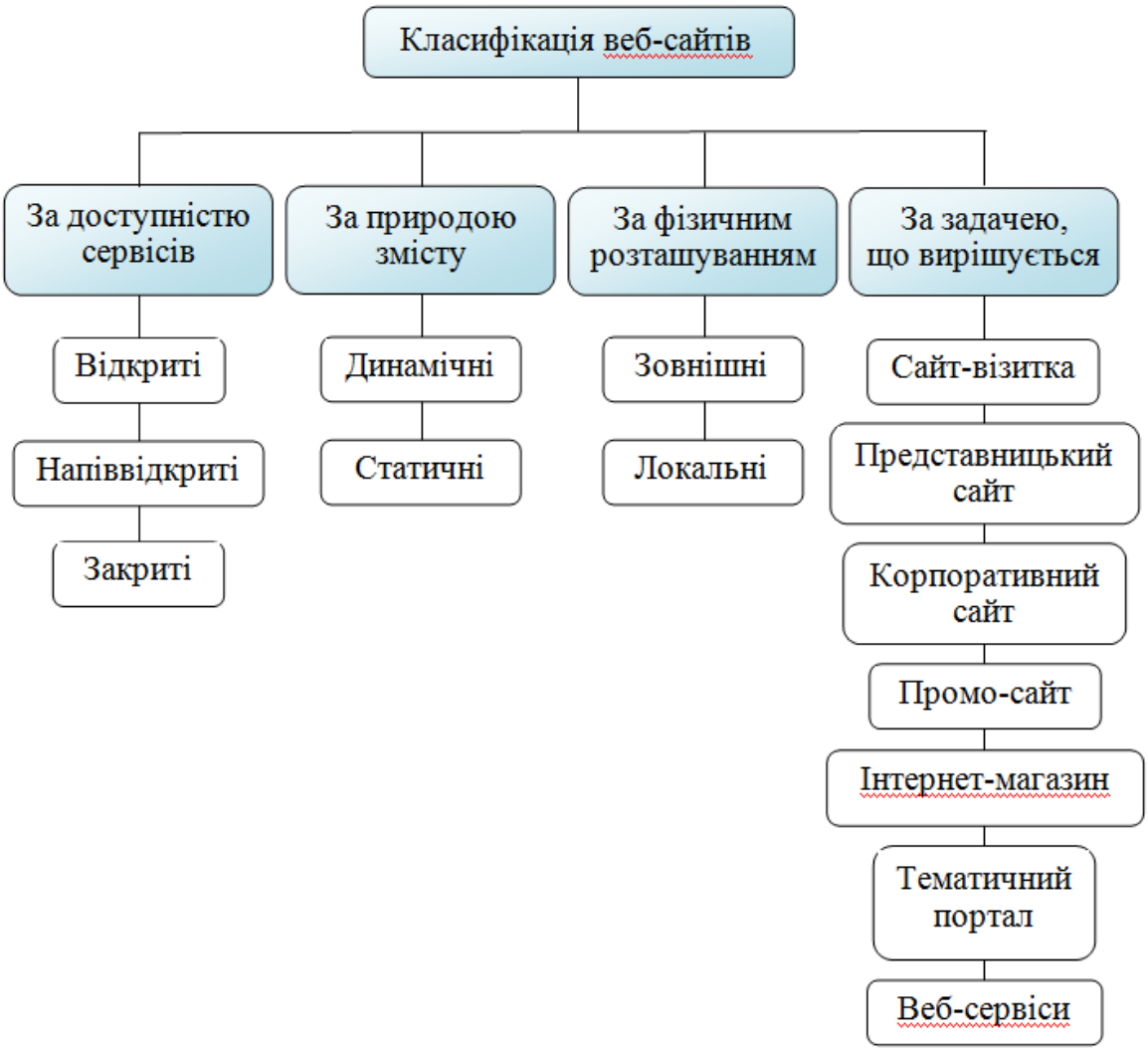

Рис.1.1. Класифікація веб-сайтів

Однак, слід відзначити ряд недоліків: складність оновлення сайту та внесення будь-яких змін (це призведе до додаткових витрат); неможливість повноцінної підтримки відвідувачів; складність у забезпеченні розділення прав доступу до вмісту веб-сайту; неможливо (дуже складно) забезпечувати зворотній зв'язок з користувачем та надавати їм можливість спілкування шляхом підтримки на сайті чатів, форумів, гостьових книг, підписок, новин і т.ін. Таким чином, статичний сайт дешевший в розробці та технічній підтримці, але ці переваги нівелюються значними недоліками в оперативності публікації інформації і витратами на адміністрування сайтом.

Динамічні - складаються із динамічних, тобто, змінних веб-сторінок (вміст яких може змінюватися в залежності від певних умов та/або дій). Такі веб-сторінки генеруються (або формуються) програмно із заздалегідь визначених

даних в процесі виконання запиту користувача [23]. Сторінки динамічного сайту формуються декількома шляхами: поєднанням різних частин в одну сторінку; заповненням сторінки інформацією по шаблону; поєднанням двох вищеперерахованих варіантів.

Перевагами динамічного сайту є: простота в адмініструванні сайту та його верстки; можливість самостійного внесення змін на сайт, без допомоги фахівців; швидке відображення нововведених даних; реалізований поділ контенту й оформлення веб-сторінок, що дозволяє оперативно змінювати інформацію на сайтах без зміни програмних кодів сторінок; широкий спектр функціональних можливостей.

Недоліки динамічного сайту: високе навантаження на сервер; складність при перенесенні на новий хостинг; значні фінансові витрати, оскільки існує необхідність застосування різних технологій та фахівців.

Найбільш популярними технологіями (середовищем розробки) створення динамічних веб-сторінок є: PHP, Ruby, ASP, JSP, Cold Fusion, AJAX, Python, CSS, бази даних SQL, MySQL, PostgreSQL, Oracle, Access і т.ін. Також для розробки використовуються різні види CMS: Joomla, Drupal, WordPress і т.ін.

За фізичним розташуванням.

Зовнішні сайти мережі Internet – забезпечують доступ будь-якому користувачу мережі Internet.

Локальні сайти - доступні тільки в межах локальної мережі. Це можуть бути як корпоративні сайти організацій, так і сайти приватних осіб в локальній мережі провайдера.

За задачею, що вирішується сайтом.

Сайт-візитка - це віртуальна візитна картка (зазвичай, до 4-5 сторінок) містить загальну інформацію про компанію або приватну особу (короткий опис, основні завдання і цілі компанії), вид діяльності, історію, загальні відомості для покупців (прайси, перелік товарів чи послуг), контактні дані (адреси, телефон, електронна пошта, кнопки на соціальні мережі, схему проїзду). Головними цілями сайту-візитки є ознайомлення потенційних клієнтів з компанією,

надання інформації про послуги і товари, а також контактні дані для зв’язку з компанією.

Представницький сайт - так іноді називають сайт-візитку з розширеною функціональністю: докладний опис послуг, портфоліо, відгуки, форма зворотнього зв'язку й т.ін.

Корпоративний сайт - це розширена версія сайту-візитки, містить повну інформацію про компанію-власника, її послуги або продукцію, події в житті компанії. Відрізняється від сайту-візитки повнотою наданої інформації, часто містить різні функціональні інструменти для роботи з контентом (пошук і фільтри, календарі подій, фотогалереї, корпоративні блоги, форуми, он-лайн консультації, обговорення питань у чаті, інтерактивна карта; реєстрація, форми замовлення послуг або розсилання новин). Може бути інтегрований з внутрішніми інформаційними системами компанії-власника (CRM, бухгалтерськими системами). Може містити закриті розділи для тих чи інших груп користувачів - співробітників, дилерів, контрагентів тощо. Для підтримування працездатності сайту потрібна спеціально навчена людина, що має навички у користуванні адміністративною частиною.

Промо-сайт - сайт про конкретну торгівельну марку або продукт, на таких сайтах розміщується вичерпна інформація про бренд, різні рекламні акції (конкурси, вікторини, ігри й т.ін.).

Тематичний портал - це дуже великий веб-ресурс, який надає вичерпну інформацію з певної тематики. Портали містять засоби взаємодії з користувачами і дозволяють користувачам спілкуватися в рамках порталу (форуми, чати).

Веб-сервіс - послуга, що створена для виконання будь-яких завдань в рамках мережі Internet. Прикладами таких сервісів можуть бути наступні: дошка оголошень, каталоги сайтів, пошукові сервіси, поштові сервіси, форуми, блоги, сайти обміну файлами, сайти-квести, сайти соціальних мереж тощо.

Необхідно звернути особливу увагу на такий вид сучасних сайтів, як *блоги*. Блоги є новим модним засобом цифрових комунікацій в теперішній час,

перспективним і простим шляхом підтримки діалогу зі своїми клієнтами і ринком, за допомогою якого інформація розповсюджується з неймовірною швидкістю. Зазвичай блог-маркетинг – це просто канал, який надає ще одну форму спілкування з клієнтами компанії і вашим ринком. Проте динамічні, змістовні та інформативні бізнес-блоги привертають увагу майбутніх, потенційних клієнтів до компанії, що є запорукою її подальшого розвитку і що не можливо зробити використовуючи будь-яки інші засоби комунікацій. Останнім часом набуває популярності сервіс мікроблогів Twitter. Особливістю Twitter є короткий формат повідомлення – всього 140 символів, але цей формат є основою популярності даного сервісу. Компанії все частіше починають використовувати Twitter для просування бренду, товару або ідеї.

Інтернет-магазин - веб-сайт з каталогом продукції, за допомогою якого клієнт може замовити потрібні йому товари.

Інтернет-магазин включає в себе наступні основні компоненти:

- Фронт-офіс - інтернет-вітрина, розташована на веб-сервері і забезпечена віртуальним споживчим кошиком, системою прийому платежів, антифродовою системою;
- Бек-офіс - складські, бухгалтерські, управлінські інформаційні системи, система обліку та контролю виконання замовлень.

Класифікація інтернет-магазинів [13]:

1. За методом роздрібного продажу товарів у мережі: інтернет-магазини; Web-вітрини, торгові системи; торгові ряди; тематичні проекти (споживацькі енциклопедії, системи інтернет-замовлень товарів тощо).
2. За бізнес-моделлю: повністю онлайновий магазин; суміщення офлайнового бізнесу з онлайновим (коли інтернет-магазин створюється на основі вже діючої реальної торгової структури).
3. За взаємовідносинами з постачальниками: магазини, які володіють власним складом (наявність реальних товарних запасів); магазини, що працюють за договорами з постачальниками (відсутність значних товарних запасів);

4. За ступенем автоматизації серед торгових систем електронних магазинів розрізняють: Web-вітрини; власне інтернет-магазини; торгові Internet-системи (TIS).

Web-вітрина - це сукупність товарного каталогу, системи навігації та оформлення замовлення з наступною передачею його менеджеру для оформлення. Той у свою чергу зв'язується зі складом, організовує доставку товару покупцеві, контролює процес оплати за товар. Паралельно ведеться рекламна робота, вивчення попиту, аналітична робота. Web-вітрина є інструментом залучення покупця, інтерфейсом для взаємодії з ним та проведення маркетингових заходів.

Характерною рисою інтернет-магазину є повна автоматизація системи обробки замовлень, завдяки чому можна працювати індивідуально з кожним зареєстрованим клієнтом.

Спільною рисою для Internet-магазину та TIS є можливість здійснювати повний торговий цикл у режимі підключення до мережі. При цьому TIS додатково інтегрована в систему внутрішнього документообігу компанії. Неавтоматизованими для інтернет-магазину та TIS залишаються системи доставки товару.

В даній кваліфікаційній роботі розглянуто створення інформаційної веб-орієнтованої системи (інтернет-магазину) для автоматизації діяльності підприємства в сфері електронної комерції.

1.1.3. Концепція Web 2.0

Зараз більшість інтернет-проектів надають послуги або здійснюють роботу відповідно до сучасної концепції Web 2.0. Вперше поняття «Web 2.0» вжив у своїй публікації американський видавець Тім О'Рейлі 18 жовтня 2005 року [34], де визначив, що Web 2.0 - це методика проектування систем з урахуванням мережевих взаємодій (тобто додаток стає кращим зі зростанням числа його користувачів) та не є якимось чітко сформульованим стандартом, концепцією

або технологією створення сайтів і додатків. Автор підкреслює, що Web 2.0 - це скоріше філософія побудови web-додатків.

Основні принципи концепції Web 2.0 [12]:

- Основа на web. Додаток має повністю базуватися на web, тобто бути повністю доступним з браузера, без необхідності встановлювати додатковий контент або програми для відображення вмісту або отримання додаткового функціоналу та іншої підготовчої роботи з боку користувача.
- Веб-сервіси. Цей підхід дозволяє веб-додатку отримувати дані від іншого веб-додатку без проведення додаткових дій.
- Mash-up. Можливість створювати веб-сервіси за допомогою інших веб-сервісів.
- Соціалізація. Web 2.0 передбачає взаємодію користувачів між собою. Часом сайти Web 2.0 взагалі являють собою тільки майданчики для користувачів, які власне і створюють весь контент (соціальні мережі, вікі-проекти і т.ін.)
- Теги. Використання тегів спрощує пошук такої інформації, не примушує користувача користуватися каталогами.
- Характерні технології та дизайн. Для Web 2.0 характерно використання технологій, що спрощують використання веб-додатків (AJAX, RSS, HTML5 + CSS3 і т.ін., а також сам дизайн сайту).

Особливості проектування веб-додатків відповідно до Web 2.0 [12]:

- Дружній UI. Користувачеві не потрібно витрачати час на вивчення нового інтерфейсу, все повинно бути логічно, максимально спрощено та зрозуміло.
- Орієнтація на веб-серфінг. Користувач не зацікавлений в читанні великих блоків інформації, його потрібно залучати за допомогою виділених слів або банерів, тобто слідувати принципам захоплення уваги.
- Оптимізація сайту для виконання свого основного завдання.

- Легкість сприйняття інформації.
- Однорідність. Більшість рішень (соціальні мережі, форуми, новині стрічки) вже мають чіткий усталений формат, тому необхідно враховувати це при проектуванні та розробці.
- Поділ вмісту, подання та поводження Web-сторінки. Тут вміст - це інформація, що виводиться на Web-сторінці, подання описує формат виводу цієї інформації, а поводження - реакцію Web-сторінки або окремих її елементів на дії відвідувача.
- Підвантажуваний вміст. Замість того, щоб оновлювати всю Web-сторінку у відповідь на щиглик на гіперпосиланні, ми можемо довантажувати тільки ту її частину, що містить необхідну інформацію. Це сильно зменшить обсяг інформації, що передається по мережі, (мережний трафік) і дозволить виконувати будь які дії з даними після їх підвантаження.
- Генерований вміст. Якась частина Web-сторінки може не завантажуватися по мережі, а генеруватися прямо на місці, у Web-оглядачі. Це значно скоротить мережний трафік.
- Семантична розмітка даних. Вона дозволяє зв'язати виведені на Web-сторінку дані відповідно до правил. Наприклад, ми можемо семантично зв'язати сторінки довідника по HTML, і відвідувач, завантаживши будь яку сторінку, зможе відразу ж перейти на пов'язані з нею сторінки, що містять додаткові або родинні відомості.

1.2. Призначення розробки та галузь застосування

Як об'єкт впровадження розроблюваної веб-орієнтованої інформаційної системи для автоматизації діяльності підприємства в сфері електронної комерції розглядається веб-сайт «JackBlack», який реалізує елітні алкогольні напої.

Проведений аналіз існуючих технічних рішень сформував основні вимоги, щодо сучасного інтернет-магазину, які відповідають концепції Web 2.0:

- максимальна зручність роботи із системою для користувача: зрозуміла структуру і категоризація, дизайн сайту відповідає його змісту, сайт працює швидко і безпомилково, легкість сприйняття інформації та відсутність помилок, наявність оптимізації сайту, наповненість якісним та достовірним контентом;
- багатоплатформовість (здатність відображатися та працювати однаково в різних браузерах, на різних апаратних платформах, операційних системах та гаджетах);
- наявність функціональних модулів, що забезпечують взаємодію з користувачем: зручні способи перегляду товару, наявність фільтрації та пошуку за визначеними параметрами; статистика по популярних товарах, інформація щодо нових надходжень, актуальні прайси; підбір аналогів або супутніх товарів; зрозумілий інтерфейс персонального кабінету, історія замовлень, порівняння характеристик вибраних товарів; зручні способи електронної оплати та доставки; опції «Додати до кошику» або «Швидке замовлення»; можливість додавання відгуків і рекомендацій, онлайн-спілкування з консультантами.

Аналізуючи вищезазначені вимоги, було зроблено висновок про те, що обов'язково буде реалізовано в проекті:

- вітрина інтернет-магазину (максимально зручна для користувача), що складатиметься з сукупності товарного каталогу, системи навігації та оформлення замовлення з наступною передачею його менеджеру для оформлення, який організовує доставку товару покупцеві, контролює процес оплати за товар;
- зручні способи перегляду за рахунок розподілу товарів по групах та наявності пошуку за визначеними параметрами (по частині назви та опису);

– реєстрація користувача відбувається після формування замовлення та роботи з віртуальною корзиною. При оформленні замовлення покупець вносить контактну інформацію: ПІП, адреса доставки, телефон і т. ін.;
– наявність адміністративної частини проекту, вхід до якої буде здійснюватися тільки після уведення адміністратором логіну й пароля. Адміністратор буде мати можливість повністю управляти вмістом інтернет-магазину (головна сторінка, розділи, товари, замовлення, статистика).

Головна сторінка першою постає перед користувачем, тому її дизайну та функціональності необхідно приділити чимало уваги, щоб надати змогу забезпечити користувача достатньою кількістю необхідної інформації, співвіднесеною з реальним продавецем-консультантом.

На рис.1.2. зображено прототип головної сторінки за допомогою сервісу «MockupBuilder».

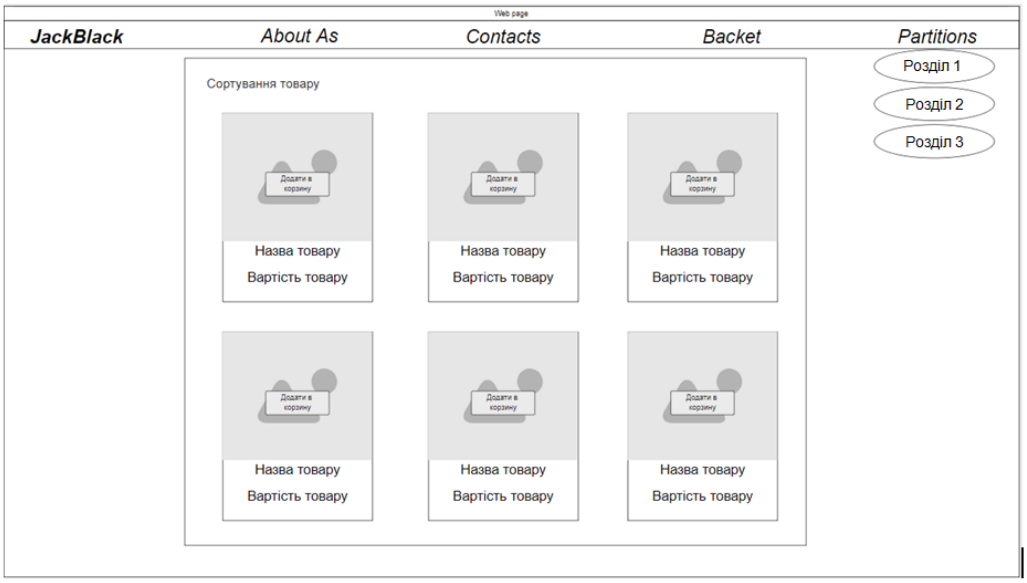

Рис.1.2. Прототип головної сторінки інформаційної системи

Головна сторінка містить:

– логотип, розроблений звичайною стилізацією тексту;

– система навігації, що дозволяє потрапити на головну сторінку системи; сторінку із інформацією про підприємство; сторінку із контактними даними підприємства; блокове меню із розділами продукції;
– блок кошику (містить основну інформацію про поточне замовлення та посилання на сторінку оформлення замовлення у вигляді стилізованого зображення кошика);
– основний блок із інформацією. У випадку із головною сторінкою це список довільних товарів із асортименту підприємства;
– блокове меню із розділами продукції;
– «підвал» або «футер».

Відповідно до прототипу головної сторінки було здійснено її верстку із застосуванням технологій HTML5, CSS3, бібліотеки jQuery. Результатом стала html-сторінка, яка в подальшому була розбита на окремі tpl-файли, а деякі елементи (назва інформаційної системи, опис сторінки, блоки товарів, посилання та інші) замінено рядками php-коду для того, щоб система могла автоматично генерувати цю і інші сторінки.

Розроблена інформаційна система призначена для:

– об’єднання елементів прямого маркетингу та традиційної торгівлі;
– надання універсального інструменту управління візуальним контентом та адміністративної системи проекту;
– забезпечення відвідувачам сайту простоти і комфортності доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту;
– підвищення ефективності роботи підприємства, за рахунок автоматизації управління бізнес-процесами.

1.3. Підстава для розробки

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

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

- ОПП за спеціальністю 121 «Інженерія програмного забезпечення»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-С від 07.06.2021р.;
- завдання на кваліфікаційну роботу на тему «Розробка веб-орієнтованого програмного забезпечення для автоматизації діяльності підприємства в сфері електронної комерції».

1.4. Постановка завдання

В даній кваліфікаційній роботі розглядається створення інформаційної веб-орієнтованої системи для автоматизації діяльності підприємства в сфері електронної комерції.

Інформаційна система створюється для надання інформаційних послуг користувачам мережі Інтернет, а також для отримання та передачі заказів менеджеру на оформлення, який організовує доставку товару покупцеві та контролює процес оплати за товар. Аудиторію інформаційної системи становлять люди у віці від 18 до 65 років.

Тематикою інформаційної системи є реалізація елітних алкогольних напоїв.

Інформаційна система повинна задовольняти наступним основним вимогам: висока швидкість завантаження сторінок; максимальна зручність роботи із системою для користувача; оптимізація сторінок під пошукові

системи; легкість сприйняття інформації; простота й повнота управління змістом.

Створення й розробка інформаційної системи повинно включати наступні етапи:

- визначення цілей створення інформаційної системи та затвердження завдання на її розробку;
- аналіз предметної області і постановка задачі;
- розгляд технічних аспектів проектування інформаційної системи: визначення архітектури системи, файлової та логічної структури сторінок;
- розробка дизайн-макету інформаційної системи;
- розробка програмного коду, модулів, бази даних та інших елементів інформаційної системи, необхідних у проекті;
- заповнення інформаційної системи контентом;
- тестування й публікація інформаційної системи в мережі Інтернет.

1.5. Вимоги до програми або програмного виробу

Згідно з [8] інформаційна система повинна забезпечувати (за умови максимальної віддачі та недопустимості виникнення збоїв та помилок в роботі): здобуття (введення або збір), зберігання, пошук, представлення, обробку (перетворення) і передачу інформації споживачеві.

Вхідні дані, на підставі яких проектуються і створюються автоматизовані інформаційні системи формують основні вимоги до інформаційної системи: програмний продукт має виконувати свою головну функцію (система повинна вміти виконувати обробку запиту клієнта і формувати результуючий масив даних стосовно замовлення, що відповідають параметрам запиту; працювати з базою клієнтів, співробітників та товару) за умови дотримання вимог надійності, продуктивності, швидкості роботи та гнучкості архітектури; програмний продукт має бути орієнтований на користувача і задовольняти його

потреби (з урахуванням вимог максимальної зручності: легкість сприйняття, простота використання, інтуїтивність інтерфейсу, повна зрозумілість), пов'язані зі сферою електронної комерції; програмний продукт має бути економічно ефективним.

Всі інші вимоги, які деталізують сам програмний продукт, можуть змінюватись у рамках уніфікованого підходу та ітеративної розробки.

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставленої в роботі мети в інформаційній системі, що розробляється, повинні бути реалізовані:

- можливість отримати віддалений доступ до веб-орієнтованої інформаційної системи через веб-браузер на комп'ютері користувача;
- швидкість роботи при реалізації більшості основних операцій;
- гнучкість архітектури, що дозволить підтримувати можливість внесення змін у компонентну структуру додатку без негативного впливу на інші частини даної структури;
- відповідність вимогам надійності, що стосуються частоти збоїв системи (має бути зведена до мінімуму) та стійкості бази даних (унеможливити втрату інформації, її блокування або несанкціоноване редагування);
- відповідність вимогам продуктивності, що стосуються часу відгуку (не більше 100 мс), доступності для користувача (має бути доступна для користувача у будь-який час);
- можливість підтримки, що стосується конфігурування та онлайн підтримки;
- програмно-апаратна переносимість;
- зберігання даних системи в реляційній базі даних.

1.5.2. Вимоги до інформаційної безпеки

Під інформаційною безпекою розуміють стан захищеності систем обробки і зберігання даних, при якому забезпечено конфіденційність, доступність і цілісність інформації, використання й розвиток в інтересах громадян або комплекс заходів, спрямованих на забезпечення захищеності інформації особи, суспільства і держави від несанкціонованого доступу, використання, оприлюднення, руйнування, внесення змін, ознайомлення, перевірки запису чи знищення [7].

Загроза інформаційної безпеки - сукупність умов і факторів, що створюють небезпеку порушення інформаційної безпеки [6].

За аспектом інформаційної безпеки, на який спрямовані загрози, виділяють наступні різновиди загроз [6]:

1. *Загрози конфіденційності* (неправомірний доступ до інформації). Загроза порушення конфіденційності полягає в тому, що інформація стає відомою тому, хто не володіє повноваженнями доступу до неї. Вона має місце, коли отримано доступ до деякої інформації обмеженого доступу, що зберігається в комп'ютерній системі або передається від однієї системи до іншої. У зв'язку з загрозою порушення конфіденційності, використовується термін «витік даних».

2. *Загрози цілісності* (неправомірна зміна даних) - це загрози, пов'язані з імовірністю модифікації тієї чи іншої інформації, що зберігається в інформаційній системі.

3. *Загрози доступності* (здійснення дій, які унеможливлюють чи ускладнюють доступ до ресурсів інформаційної системи). Порушення доступності являє собою створення таких умов, при яких доступ до послуги або інформації або заблокований, або можливий за час, який не забезпечить виконання тих чи інших бізнес-цілей.

Класифікацією векторів атак і вразливостей займається спільнота OWASP (Open Web Application Security Project) - міжнародна некомерційна організація, зосереджена на аналізі та поліпшення безпеки програмного забезпечення [35].

OWASP (https://www.owasp.org/) склав список найбільш небезпечних вразливостей інтернет-ресурсів [28]:

- впровадження коду;
- некоректна аутентифікація і управління сесією;
- витік конфіденційних даних;
- впровадження зовнішніх XML - сутностей (це, як правило, DoS-атаки - напрямок численних запитів на сайт магазину, що призводять до зупинки його роботи);
- порушення контролю доступу;
- небезпечна конфігурація;
- міжсайтовий скриптинг;
- небезпечна десеріалізація;
- використання компонентів з відомими уразливими;
- відсутність журналювання і моніторингу.

Більш докладно про найбільш поширені з них **[9]**:

XSS (англ. Cross-Site Scripting - «міжсайтовий скриптинг») - тип атаки на веб-системи, що полягає у впровадженні до сторінки, що видається веб-системою, шкідливого коду (який буде виконаний на комп'ютері користувача при відкритті їм цієї сторінки) і взаємодії цього коду з веб-сервером зловмисника.

XSRF / CSRF (англ. Cross-site request forgery - «міжсайтова підробка запиту», також відома як XSRF) - це вид уразливості, що дозволяє використовувати недоліки HTTP протоколу, при цьому зловмисники працюють за такою схемою: посилання на шкідливий сайт встановлюється на сторінці, що користується довірою у користувача, при переході по шкідливої посиланням виконується скрипт, який зберігає особисті дані користувача (паролі, платіжні

дані і т.д.), або змінює доступ до облікового запису користувача для отримання повного контролю над нею.

Code injections (SQL, PHP, ASP і т.д.) - це вид уразливості, при якому стає можливо здійснити запуск виконуваного коду з метою отримання доступу до системних ресурсів, несанкціонованого доступу до даних, або виведення системи з ладу.

Server-Side Includes (SSI) Injection - це вид уразливості, що використовує вставку серверних команд в HTML код або запуск їх безпосередньо з сервера.

Authorization Bypass - це вид уразливості, при якому можливо отримати несанкціонований доступ до облікового запису або документам іншого користувача.

Основні шляхи перевірки захищеності веб-додатку:

- здійснення ручної та автоматичної перевірки веб-додатку з використанням сканерів вразливостей веб-додатків з використанням спеціалізованого ПЗ;
- побудова векторів атак та проведення тестових спроб експлуатації виявлених вразливостей;
- пошук будь-яких можливостей, які можуть завдати шкоди бізнесу або його клієнтам;
- пошук розкриттів внутрішньої інформації веб-додатку і т. ін.;
- проведення комплексного тесту на проникнення (щорічно, а також після значних змін у веб-додатку);
- впровадження засобів захисту інформації класу Web Application Firewall;
- періодичне оновлення всіх облікових даних сервісів і службових облікових записів.

Для уникнення некоректної роботи програми необхідно реалізувати:

- семантичний та синтаксичний контроль вхідних даних;
- можливість повторного введення даних;
- обробку виняткових ситуацій;

– виведення повідомлень про помилки;
– можливість безперервної роботи протягом не менше 120 годин (5 діб);
– платформну незалежність.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для нормального функціонування веб-орієнтованої інформаційної системи, повинні виконуватися певні вимоги до технічних засобів.

Для клієнтської частини:

– процесор з тактовою частотою не менш 1.2GHz;
– оперативна пам'ять 1Gb;
– жорсткий диск 20Гб;
– монітор – SVGA, з діагоналлю не менше 17';
– операційна система: Microsoft Windows XP/7/8/10;
– клавіатура;
– підключення до мережі Інтернет;
– маніпулятор типу «миша».

Для серверної частини:

– процесор з тактовою частотою не менш 2.4GHz;
– оперативна пам'ять 2Gb;
– жорсткий диск 20Гб;
– монітор;

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.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати веб-орієнтована підсистема, відповідало наступним вимогам:

- операційна система Unix, Linux, Microsoft Windows XP/7/8/10;
- браузер Інтернет (Microsoft Internet Explorer, MozillaFireFox, Opera, Google Chrome);

Веб-орієнтована інформаційна система має бути реалізовано на мові програмування PHP, HTML5 і CSS3 для верстання веб-сторінок, бібліотеки jQuery для написання клієнтських сценаріїв, з використанням веб-серверу Apache HTTP Server та СУБД MySQL.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

В рамках даної кваліфікаційної роботи було розроблено програмне забезпечення, що реалізує веб-орієнтовану інформаційну систему для автоматизації діяльності підприємства в сфері електронної комерції.

Функціональне призначення розробленої системи:

- підвищення ефективності роботи підприємства, за рахунок автоматизації управління бізнес-процесами та об'єднання елементів прямого маркетингу і традиційної торгівлі;
- надання універсального інструменту управління візуальним контентом та адміністративною частиною проекту;
- надання відвідувачам системи вичерпної інформації про надані товари та послуги;
- забезпечення відвідувачам сайту простоти і комфортності доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту;
- приймання та обробку замовлень, персоналізація споживачів або відвідувачів після формування замовлення та роботи з віртуальною корзиною;
- забезпечення взаємодії з платіжною системою банку для проведення платежів та розрахунків за придбаний товар;
- збір та аналіз статистичної інформації.

Для організації управління роботою інформаційної системи і її контентом в роботі було розроблено адміністраторську панель, що надає доступ та повний контроль над функціоналом системи (управління товарами; управління замовленнями; управління розділами; статистика), недоступному звичайним

користувачам. Алгоритм роботи адміністраторської панелі полягає в наступному:

1. Вхід до адмін-панелі, використовуючи логін і пароль адміністратора.
2. Вибір необхідного розділу з меню аккаунта адміністратора ("Головна", "Товари", "Розділи", "Замовлення", "Статистика").
3. Перегляд даних вибраного розділу.
4. Управління даними вибраного розділу (перегляд, редагування, додавання та видалення інформації).
5. Перегляд статистки роботи системи (розділ «Статистика»).

2.2. Опис застосованих математичних методів

Оскільки особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці веб-орієнтованої інформаційної системи для автоматизації діяльності підприємства в сфері електронної комерції математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

При створенні веб-орієнтованої інформаційної системи для автоматизації діяльності підприємства в сфері електронної комерції була обрана трирівнева клієнт-серверна архітектура (рис.2.1).

Трирівнева клієнт-серверна архітектура (англ. three-tier або Multitier architecture) архітектурна модель програмного комплексу, що передбачає наявність наступних компонентів програми: клієнтський додаток (зазвичай говорять «тонкий клієнт» або термінал), підключений до сервера додатків, який в свою чергу, підключений до серверу бази даних [27,39].

Клієнт (перший рівень) - це інтерфейсний (графічний) компонент, який представляє власне додаток для кінцевого користувача. На цьому рівні не реалізуються прямі зв'язки з базою даних (за вимогами безпеки), відсутні

навантаження основною бізнес-логікою (за вимогами масштабованості) і не зберігається стан програми (за вимогами надійності). На першому рівні реалізується найпростіша бізнес-логіка: інтерфейс авторизації; перевірка значень, що вводяться, на допустимість і відповідність формату; алгоритми шифрування; нескладні операції (сортування, групування, підрахунок значень) з даними, вже завантаженими на термінал.

Рис.2.1. Трирівнева архітектура клієнт-сервер

Сервер додатків (другий рівень) зосереджує більшу частину бізнес-логіки. Поза його межами залишаються фрагменти, які експортуються на клієнта, а також елементи логіки, які занурені в базу даних (збережені процедури і тригери). Даний компонент реалізується за допомогою підпрограмного забезпечення. Сервери додатків повинні проектуватись таким чином, щоб при додаванні до них додаткових примірників забезпечувалось горизонтальне масштабування продуктивності програмного комплексу і не вимагало внесення змін до програмного коду програми.

Сервер бази даних (третій рівень) включає в себе механізми збереження даних та рівень доступу до даних. Це стандартна реляційна або об’єктно-орієнтована СУБД [20]. У «правильній» (з точки зору безпеки, надійності,

масштабування) конфігурації сервера баз даних знаходяться на виділеному комп'ютері (або кластері), до якого по мережі підключені один або кілька серверів додатків, до яких, у свою чергу, по мережі підключаються термінали.

До переваг трирівненої архітектури в порівнянні з клієнт-серверною або файл-серверною архітектурою слід віднести наступне:

- масштабованість;
- конфігурованість - логічне розділення між рівнями дозволяє (при правильному розгортанні архітектури) швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів;
- високий рівень безпеки та надійності (оскільки рівень додатків знаходиться між рівнем бази даних і рівнем клієнта, рівень захисту даних буде вищим, оскільки клієнт не матиме безпосереднього доступу до бази даних);
- низькі вимоги до швидкості каналу (мережі) між терміналами і сервером додатків;
- низькі вимоги до продуктивності і технічних характеристик терміналів;
- можливість відображати лише необхідні методи бізнес-рівня на рівні клієнта (інтерфейс авторизації; перевірка значень, що вводяться, на допустимість і відповідність формату; алгоритми шифрування; нескладні операції (сортування, групування, підрахунок значень) з даними, вже завантаженими на термінал);
- легко підтримувати та розуміти великі та складні проекти і т.ін.

Також слід відмітити і недоліки даної архітектури:

- складність у створенні, розгортанні і адмініструванні додатку;
- висока вартість серверного обладнання, за рахунок високих вимог до продуктивності сервера додатків і сервера бази даних;
- високі вимоги до швидкості каналу (мережі) між сервером бази даних і сервером додатків;

– значні витрати часу та наявність значного досвіду в об’єктно-орієнтованій концепції програмування при реалізації.

2.4. Опис використаних технологій та мов програмування

Для реалізації поставленого в роботі завдання виділяють декілька підходів, зокрема:

– використання вже готових систем (систем управління контентом, CMS);
– розробка інформаційної системи в ручному режимі за допомогою скриптових мов програмування та відповідних технологій (створення власної системи управління контентом).

Для того, щоб остаточно визначитися з вибором, розглянемо основні особливості кожного з підходів, їх переваги та недоліки.

Система управління контентом (англ. Content management system, CMS) - це інформаційна система або комп'ютерна програма, яка використовується для забезпечення і організації спільного процесу створення, редагування і управління вмістом (контентом) [19].

Функції CMS можна розділити на декілька основних категорій [17]:

– створення: надання розробникам зручних і звичних інструментів для створення контенту, організація спільної роботи над вмістом;
– управління: зберігання контенту в єдиному репозиторії, контроль версій, дотримання режиму доступу, інтеграція з іншими інформаційними системами та управління потоком документів і т.ін.;
– публікація: автоматичне розміщення контенту на терміналі користувача. Відповідні інструменти автоматично адаптують зовнішній вигляд сторінки до дизайну всього сайту;

– подання: надання додаткових функцій, що дозволяють поліпшити форму подання даних (наприклад, можна будувати навігацію по структурі репозиторію).

Виділяють CMS двох типів: вільно поширювані (безкоштовні) та комерційні. На рис.2.2. наведено рейтинг популярності безкоштовних та комерційних CMS, згідно з [10]:

– найбільш популярні безкоштовні CMS: WordPress, Joomla, Opencart, Drupal, Wix, MODX Revolution, Evolution CMS та інші;
– найбільш популярні комерційні CMS: 1C-Bitrix, DataLife Engine, Shop-Script, Tilda, UMI.CMS, NetCat, HostCMS, InSales та інші.

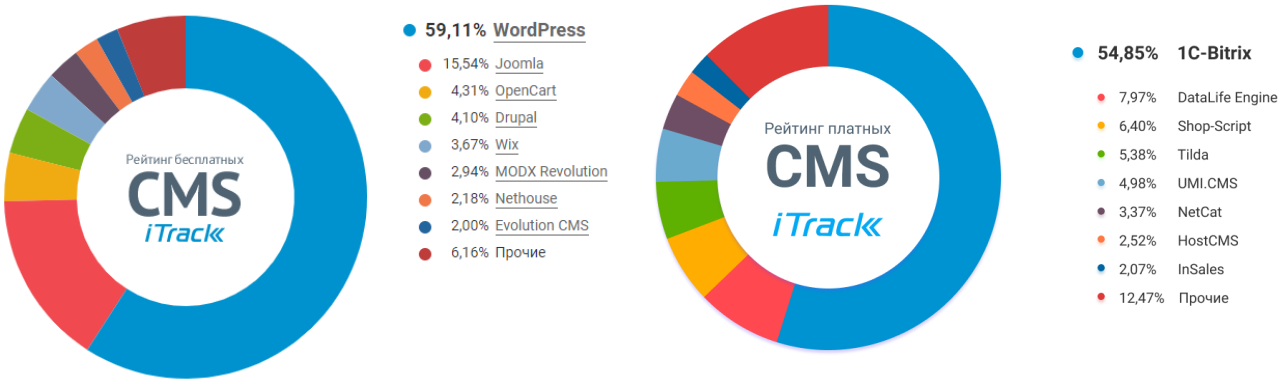

Рис. 2.2. Рейтинг безкоштовних та комерційних CMS

На рис.2.3 наведено загальний рейтинг CMS, згідно з [10].

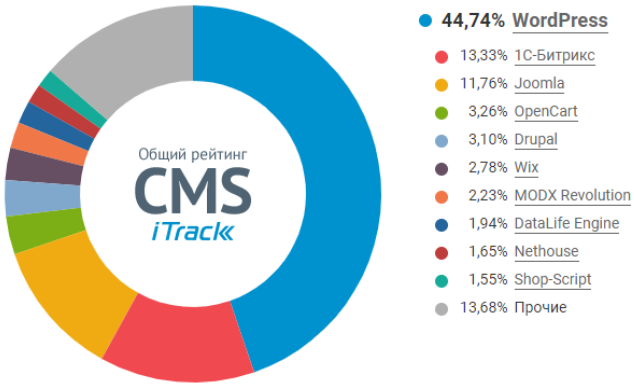

Рис. 2.3. Загальний рейтинг CMS

Переваги використання CMS [17]:

– Зручність. Доступність і зрозумілість інтерфейсу дозволяє швидко реалізувати поставлені задачі.

- Економія. При створенні сайту за допомогою CMS вам не знадобиться вдаватися до послуг фахівців, оскільки даний процес не вимагає глибоких спеціальних знань і навичок.
- Безпека. Системи протистоять атакам хакерів.
- Документація. Модулі мають help-файли, що допомагають розібратися з функціями системи.
- Мультишаблонність. Більшість систем підтримують численні шаблони (розробники постійно оновлюють систему і надають на ваш вибір нові види), що дозволяє автоматично міняти вигляд всього сайту, незалежно від його змісту.
- Функціональність. Більшість CMS мають модульну архітектуру і кожен розділ або підрозділ має свої налаштування і конфігурації, а їх функціональність легко розширюється за допомогою плагинів.
- Контент цілком відокремлено від візуального представлення сайту, що значно спрощує редагування вмісту сайту.
- Комплексність. Можна створювати окремі вкладки і надавати контроль над ними одному або декільком співробітникам (стрічка новин, блоги та ін.).
- Незалежність. Користувач програмного забезпечення не залежить від розробників та може змінювати конфігурації в залежності від свого профілю.
- Можливість розширення. Збереження інформації та функціональності при зміні системи.
- Керованість. Система проста у використанні. За допомогою пару кліків ви можете змінити структуру, поміняти місцями пункти, розділи, кореневу папку, підняти рядок вище або нижче і багато іншого.
- Добра технічна підтримка у вигляді спільноти користувачів.

Однак, не зважаючи на численні переваги, CMS притаманні наступні недоліки:

- Шаблонність. Велика кількість готових шаблонів спричиняє за собою появу однотипних сайтів.
- Обмеженість функціоналу. Простота управління контентом є лише відносною, оскільки якщо виникне бажання реалізувати на сайті якісь нестандартні рішення у функціоналі або дизайні, це буде проблематично.
- Надмірна функціональність CMS. Сайти на CMS працюють повільніше, більше вимогливі до хостингу, ресурсів сервера, в більшій мірі схильні до збоїв.
- Невисока стійкість до навантаження.
- Низький рівень безпеки, в наслідок відкритості коду і безкоштовності систем.
- Необхідність в розширенні доступного функціоналу. Зачасту реалізація простого модуля спричиняє за собою безліч програмної роботи, яка в результаті може себе не виправдати.
- Складнощі вивчення. Внаслідок поганого проектування певних CMS, вивчення конкретних систем вимагає певного часу та зусиль.

Враховуючи вищезазначені недоліки існуючих CMS для вирішення поставленої задачі, в даній кваліфікаційній роботі було прийнято рішення про створення власної системи управління контентом (з нуля), що повинна містити тільки необхідний функціонал, гнучкість і простоту управління, мати високу швидкість дії та високий рівень безпеки.

Для реалізації поставленого завдання (створення самописного движку) виділяють безліч підходів, оскільки, існує багато варіантів для рішення певних задач. Даному питанню і будуть присвячені подальші дослідження.

Перш ніш розглядати основні технології по створенню власної системи управління контентом, слід зазначити, що кожна система зазвичай складається з двох частин:

– front-office - це візуальна частина системи, що забезпечує інтерфейс з користувачем;
– back-office - це програмна частина системи, відповідальна за функціональність і зберігання інформації.

Для реалізації візуальної частини системи, зазвичай застосовують базові технології HTML5 і CSS3 [2,14,18,24].

Мова гіпертекстової розмітки HTML (HyperText Markup Language) є основним будівельним засобом для веб-сторінок, використовується для створення та візуального представлення веб-сторінок. Визначає структуру і описує зміст веб-сторінки в структурованій формі. HTML дозволяє формувати на сторінці сайту текстові блоки, додавати до них зображення, організовувати таблиці, додавати до дизайну сайту звуковий супровід, організовувати гіперпосилання з переходом до інших розділів сайту або ресурсів Інтернету і компонувати всі ці елементи між собою. За допомогою HTML можна створити як статичний так і динамічний сайт.

CSS (Cascading Style Sheets, каскадні таблиці стилів) - це технологія опису зовнішнього вигляду документа, що створено засобами HTML, XML і XHTML. CSS використовується для привласнення певних особливостей для елементів HTML-сторінки: колір, шрифт, розташування на сторінці і інших аспектів представлення документа. Основною метою розробки CSS було принципове розділення вмісту (написаного на HTML або іншій мові розмітки) від представлення (опису) стилю документа, що надало більшу доступність документу, гнучкість, можливість управління його виглядом, а також зменшило складність і повторюваність в структурному вмісті.

Переваги CSS розмітки:

– Застосування кількох варіантів дизайну сторінки для різних пристроїв перегляду, наприклад, для комп’ютера, планшету чи телефону.
– Зменшення часу завантаження сторінок сайту за рахунок перенесення правил опису даних до окремого CSS-файлу. В цьому випадку браузер завантажує лише структуру документа і дані, що містяться на сторінці.

CSS-файл з правилами опису цих даних завантажується браузером лише один раз і зберігається в кеші браузера.

- Простота подальшої зміни дизайну. Не потрібно виправляти кожну сторінку, достатньо лише змінити кілька правил у CSS-файлі.
- Додаткові можливості оформлення. Наприклад, за допомогою CSS-правил можна застосувати обтікання певного блоку текстом або зробити так, щоб меню фіксовано знаходилося в певному місці при перегортанні сторінки.

Для реалізації програмної частини системи зазвичай використовують скриптові мови (англ. scripting language) програмування, розроблені для запису «сценаріїв», послідовностей операцій, які користувач може виконувати на комп'ютері [21]. Характерними особливостями даних мов є, по-перше, їх інтерпретованість (компіляція або неможлива, або небажана), по-друге, простий синтаксис, а по-третє, легка розширюваність.

Найбільш поширеними скриптовими мовами програмування, згідно з [22] є: JavaScript, Python, Java, TypeScript, C #, PHP, C ++, C, Shell, Ruby.

Розглянемо деякі з них.

JavaScript - динамічна, об'єктно-орієнтована скриптова мова програмування, що є діалектом мови ECMAScript. Найчастіше JavaScript використовується для створення сценаріїв веб-сторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки [33]. Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне управління пам'яттю, прототипне програмування, функції як об'єкти першого класу. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну).

Важливою особливістю JavaScript є об'єктна орієнтованість. Програмісту є доступними численні об'єкти, такі, як документи, гіперпосилання, форми,

фрейми тощо. Об'єкти характеризуються описовою інформацією (властивостями) і можливими діями (методами).

Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ, всередині PDF-документів тощо [33].

До основних переваги JavaScript слід віднести [15]: популярність; простота у вивченні; універсальність; сумісність (на відміну від PHP чи інших мов сценаріїв, JavaScript можна вставити на будь-яку веб-сторінку та використовувати в багатьох різних видах програм); навантаження на сервер (JavaScript працює на стороні клієнта, тому загалом не навантажує ресурси сервера); швидкість (JavaScript часто запускається відразу в браузері клієнта та не потребує компіляції коду перед його запуском); розширені інтерфейси (JavaScript можна використовувати для створення функцій, що покращують користувальницький інтерфейс та досвід роботи на сайті); розширена функціональність; оновлення (ECMA International займається оновленням JavaScript щорічно).

Недоліки JavaScript:

- Захист на стороні клієнта. Оскільки код JavaScript виконується на стороні клієнта, помилки та недогляди іноді можуть бути використані для зловмисних цілей, що спонукає деяких користувачів повністю вимикати JavaScript.
- Підтримка браузера. Хоча сценарії на стороні сервера завжди дають однакові результати, різні браузери іноді інтерпретують код JavaScript по-різному, тому рекомендовано тестувати свої сценарії у всіх основних браузерах.

Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією, що є ефективною для швидкої розробки (за рахунок динамічної семантики та структури даних високого рівня) та поєднання існуючих компонентів (підтримує модулі та

пакети модулів, що сприяє модульності та повторному використанню коду). В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [36].

Переваги Python [16]: Open Source (відкритий код) та PEP (єдиний стандарт для написання коду); гнучкість; можливість розширення; простота та чистота синтаксису; інтерпретованість; переносимість програм; наявність великої кількості корисних модулів та можливість використання в діалоговому режимі; дружнє та приємне ком'юніті і т.ін.

Разом з тим, виділяють наступні недоліки Python [16]:

- низька продуктивність та погана реалізація багатопоточності (код Global Interpriter Lock не змінювався з першої версії мови та явно застарів);
- незвичний синтаксис при переході з іншої мови програмування;
- споживання більше ресурсів через динамічну типізацію;

PHP - одна з найпоширених скриптових мов програмування з відкритим програмним кодом (використовуються у сфері веб-розробок та підтримується переважною більшістю хостинг-провайдерів), що призначена для генерації HTML-сторінок на стороні веб-сервера [32]. PHP дозволяє автоматизувати роботу з сайтом. PHP - це серверна мова. Конструкції PHP, що вставлено в HTML-текст, виконуються сервером при кожному відвідуванні сторінки. Результат обробки конструкцій разом із звичайним HTML-текстом передається браузеру. Відмінність PHP від JavaScript, полягає в тому, що PHP-скрипт виконується на сервері, а клієнту передається результат роботи, тоді як в JavaScript-код повністю передається на клієнтську машину і тільки там виконується.

PHP працює як частина Web-сервера. У цій мові немає строгої типізації даних і немає необхідності в діях по виділенню / звільненню пам'яті.

Переваги PHP [37]: орієнтація на веб-розробку; є вільним програмним забезпеченням, поширюваним під особливою ліцензією (PHP license);

традиційність; низький поріг входу (простота та легкість в освоєнні на всіх етапах); ефективність; безпека; гнучкість; ухвалення стратегії OpenSource і безкоштовне розповсюдження початкових текстів PHP; розвинена підтримка баз даних (в PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase); наявність великої кількості бібліотек і розширень мови; велика кількість доступних фреймворків; можливість використання в ізольованому середовищі; достатньо висока швидкість обробки сценаріїв, в порівнянні з іншими інтерпретованими мовами; кросплатформність; підтримка веб-серверів; пропонує нативні засоби організації веб-сесій, програмний інтерфейс розширень; підтримується великою спільнотою користувачів і розробників; безперервність розвитку і т.ін.

Разом з тим, виділяють наступні недоліки PHP:

- вузькопрофільність (не підходить для створення десктопних додатків або системних компонентів);
- проблеми з обслуговуванням застарілих додатків;
- безпека;
- глобальні параметри конфігурації впливають на базовий синтаксис мови, що ускладнює налаштування сервера і розгортання додатків;
- протиріччя в коді;
- має слабкі засоби для роботи з винятками;
- об'єкти передаються за значенням, а не за посиланням, як це робиться в більшості інших мов.

Проведений аналіз найпопулярніших мов програмування виявив їх основні переваги та недоліки. Враховуючи всі особливості вищезазначених мов програмування, в даній кваліфікаційній роботі було вирішено використовувати мову PHP через її простоту та легкість в освоєнні, орієнтацію на веб-розробку, розвинену підтримку баз даних і веб-серверів, значну популярність та інформативність серед інших мов програмування.

Для написання клієнтських сценаріїв в роботі було вирішено застосувати бібліотеку jQuery через її переваги [29]: компактність коду (дозволяє писати

код більш компактно ніж на чистому JavaScript, тобто за набагато меншу кількість рядків коду); простий і зрозумілий синтаксис (значно спрощує написання багатьох речей, наприклад, таких як маніпулювання DOM елементами, обробку подій, додавання ефектів анімації на сторінку, AJAX запитів і т.ін.); кросбраузерність (код написаний на jQuery буде гарантовано працювати у всіх основних браузерах, у той час як код, написаний на чистому JavaScript треба буде однозначно перевіряти у всіх браузерах); відкритий код (бібліотека jQuery є повністю безкоштовною як для особистих, так і для комерційних проектів).

Бібліотека jQuery дозволяє дуже легко: вибирати елементи для виконання різних маніпуляцій над ними; створювати різні візуальні ефекти (наприклад, плавне відображення та приховування елементів); створювати складну анімацію, при цьому реалізовуючи це за набагато меншу кількість рядків коду ніж на чистому JavaScript; маніпулювати DOM елементами і їх атрибутами; реалізовувати AJAX для асинхронного обміну даними між клієнтом і сервером; переміщатися від поточного вузла до інших по ієрархічній структурі DOM дерева; виконувати кілька дій над елементом за допомогою одного рядка коду; отримувати або встановлювати розміри HTML елементам і т.ін.

Наступним етапом є вибір системи управління базою даних, що є невід'ємною частиною проекту, що розробляється.

Система управління базами даних (СУБД, англ. Database Management System) - набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних [20]. Надає можливості створення (здійснюється за допомогою мови визначення даних DDL); додавання, оновлення, видалення, пошуку, читання збереження інформації в базах даних (за допомогою мови маніпулювання даними DML); контролю доступу до бази даних (за допомогою системи забезпечення захисту від несанкціонованого доступу, системи управління паралельною роботою прикладних програм, системи відновлення до попереднього несуперечливого стану)

Основними характеристиками СУБД є: контроль за надлишковістю даних; несуперечливість даних; підтримка цілісності бази даних; незалежність прикладних програм від даних; спільне використання даних; підвищений рівень безпеки.

Найбільш поширеними реляційними СУБД є: MySQL, Microsoft SQL Server, Oracle Database, Firebird, DB2, MongoDB, PostgreSQL.

В якості СУБД в роботі було обрано MySQL з наступних причин [25]: простота у встановленні та використанні; багатопотоковість; оптимізація зв'язків з приєднанням багатьох даних за один прохід; записи фіксованої і змінної довжини; ODBC драйвер; ліцензія GNU (General Public License), що надає свободу запуску програми з будь-якою метою; гнучка система привілеїв і паролів; наявність простої і ефективної системи безпеки; інтерфейс з мовами C і Perl, PHP; гнучка підтримка форматів чисел, рядків змінної довжини і міток часу; швидка робота, масштабованість; сумісність з ANSI SQL; входить до базового пакету «Denwer»; хороша підтримка з боку провайдерів послуг хостингу; швидка підтримка транзакцій через механізм InnoDB4; безкоштовна в більшості випадків (для некомерційного використання MySQL є безкоштовною).

Для управління СУБД MySQL використовується phpMyAdmin - веб-застосунок з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування СУБД MySQL.

PhpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних (через дружній інтерфейс і з будь-якого комп'ютера, що під'єднаний до інтернету) без безпосереднього вводу SQL команд та необхідності встановлення додаткового програмного забезпечення [38].

В якості локального серверу в роботі було обрано Denwer.

Denwer [3] - набір дистрибутивів (локальний сервер WAMP) і програмна оболонка, призначені для створення і налагодження сайтів (веб-додатків, іншого динамічного вмісту інтернет-сторінок) на локальному ПК (без необхідності

підключення до мережі Інтернет) під керуванням ОС Windows. Відразу після установки Denwer доступний веб-сервер Apache (на якому може працювати безліч сайтів), що працює на локальному комп'ютері та дозволяє розробку та налагодження сценаріїв PHP без завантаження його файлів на віддалений сервер.

Особливістю, що відрізняє Denwer від інших WAMP - дистрибутивів, є автоматична правка системного файлу hosts (локальний аналог DNS-сервера), що дозволяє звертатися до локальних сайтів (які працюють під управлінням Denwer) по іменах, що збігається з ім'ям папки, розташованої в каталозі home Denwer.

Зараз існує велика кількість текстових редакторів (редакторів коду), що надають веб-розробнику безліч інструментів і функцій, які допоможуть редагувати код, мають підсвічування синтаксису для декількох мов, включають вбудовані завантаження файлів, звіти про помилки, пошук, заміну і багато іншого [4]. Основними з них є: Visual Studio Code, Atom, Sublime Text, IntelliJ IDEA, PyCharm, Brackets, Vim, Eclipse, Aptana Studio, Notepad++, Emacs, PhpStorm, WebStorm, NetBeans та ін.

В даній кваліфікаційній роботі в якості редактору коду було обрано Brackets. При виборі текстового редактору були враховані наступні критерії: підтримка необхідної операційної системи, можливості спільної розробки, підтримка мов програмування, вартість.

Brackets заточений спеціально для frontend-розробників і дизайнерів. Крім традиційних для подібних редакторів підсвічування синтаксису і підказок Brackets дозволяє редагувати файли в режимі реального часу (LivePreview): ви можете слідкувати за зовнішнім виглядом вашого проекту в міру внесення змін без необхідності перезавантаження сторінки. Brackets в реальному часі аналізує структуру проекту, будує дерево залежностей і дозволяє писати взаємозалежні ділянки коду, практично не залишаючи основного контексту файлу. З інших особливостей виділяється швидкий доступ до документації (правда, не настільки деталізований, як у LightTable) і JSLint з коробки. Також редактор

дуже добре працює з препроцесора, має дійсно розумний автокомпліт і деяку подобу парсинга PSD, що дозволяє зробити розмітку стилів сторінки. Multiple cursors дозволяє виділяти відразу кілька фрагментів коду для швидкої заміни.

Brackets підтримує безліч корисних розширень (плагінів) для PHP, зокрема: Integrated Development - інтерпретатор командного рядка, що дозволяє тестувати код; Codeoverview - допомагає швидко переміщатися по коду; Code Folding - дає можливість згортати код в блоки (приховувати); Epic Linter - підсвічування помилок прямо в коді; Emmet - прискорює введення тексту (автозаміна команд). На рис.2.4 наведено вікно редактору Brackets.

Рис.2.4. Вікно редактора коду Brackets із відкритим tpl-файлом

Підводячи підсумки проведеного аналізу наявних технологічних рішень реалізації поставленого завдання, в даній кваліфікаційній роботі доцільно використовувати:

– Apache HTTP Server в якості веб-сервера (найпоширеніший у світі веб-сервер з відкритим сирцевим кодом [31]);
– Denwer в якості локального веб-сервера (для зручної розробки і налагодження сайту на локальному ПК без необхідності підключення до мережі Інтернет);

– HTML5 і CSS3 для верстання сайту інформаційної системи (застосовується в основному блочний метод верстання сторінок, що надає можливість створювати адаптивні сторінки);
– СУБД MySQL - для розробки бази даних;
– PHP в якості мови програмування;
– бібліотеку jQuery для написання клієнтських сценаріїв;
– Brackets в якості текстового редактору.

2.5. Опис структури програми та алгоритмів її функціонування

На рис.2.5 наведено структуру інформаційної системи, де її структурні елементи розділено на окремі модулі, кожен з яких виконує свої функції.

Рис.2.5. Структура інформаційної системи

До складу інформаційної системи входять наступні модулі: шаблонізатор; клас управління; класи сторінок; класи для роботи з таблицями; абстрактний

клас для роботи з таблицями; клас для роботи з базою даних; база даних; клас конфігурації; допоміжні класи.

Розглянемо більш докладно кожен з модулів.

Шаблонізатор. Модуль системи (описаний в `class Template`) відповідає за виведення сторінок інформаційної системи, підставляючи потрібну інформацію.

Розглянемо програмний код `class Template`.

```
class Template {
 private $dir_tmpl;
 private $data = array();
 public function __construct($dir_tmpl) {
      $this->dir_tmpl = $dir_tmpl;
 }
 public function set($name, $value) {
      $this->data[$name] = $value;
 }
 public function delete($name) {
      unset($this->data[$name]);
 }
 public function __get($name) {
      if (isset($this->data[$name])) return $this-
>data[$name];
      return "";
 }
 public function display($template) {
      $template = $this->dir_tmpl.$template.".tpl";
      ob_start();
      include ($template);
      echo ob_get_clean();
 }
}
```

Шаблонізатор виводить вміст tpl-файлів, що необхідні для зберігання HTML-коду з елементами шаблонів, які підставляються в PHP-коді. Інформаційна система клієнтської частини містить наступні tpl-файли:

content_auth.tpl, content_discount_form.tpl, content_discounts.tpl, content_index.tpl, content_order_form.tpl, content_orders.tpl, content_product_form.tpl, main.tpl, menu.tpl, content_products.tpl, content_section_form.tpl, content_sections.tpl, content_statistics.tpl, message.tpl, pagination.tpl, програмний код яких приведено в ДОДАТКУ А.

Класи сторінок. Шаблонізатор звертається до класів сторінок. Кожна сторінка інформаційної системи має свій клас (відповідає за виведення відповідного контенту на сторінці), що визначають значення змінних, які шаблонізатор підставляє в tpl-файли. Батьківським класом для них є абстрактний (це клас, від якого не можна створити екземпляру; в ньому можуть бути описані абстрактні методи, реалізація яких буде описана в дочірньому класі) `class Modules`, для якого, в свою чергу, батьківським є абстрактний `class AbstractModules`.

Розглянемо фрагмент програмного коду абстрактного `class Modules`.

```
abstract class Modules extends AbstractModules {
 public function __construct() {
      parent::__construct();
      $this->setInfoCart();
      $this->template->set("index", $this->url->index());
      $this->template->set("cart_link", $this->url->cart());
      $this->template->set("about_link", $this->url->about());
      $this->template->set("link_contacts",         $this->url-
>contacts());
      $this->template->set("link_search",           $this->url-
>search());
      $this->template->set("items",             $this->section-
>getAllData());
      $this->template->display("main");
  }
```

Класи для роботи з таблицями. Реалізують SQL-запити до БД, отримують інформацію, трансформують її в зручну і безпечну форму, передають для виведення.

Абстрактний GlobalClass є батьківським класом для окремих класів, що працюють із таблицями бази даних, та містить загальні методи для всіх дочірніх класів (таких як class Product або class Section). В даних дочірніх класах перевизначаємо реалізацію вже описаних в батьківському класі методів, або описуємо притаманні лише їм методи.

Фрагмент програмного коду GlobalClass наведено нижче.

```
abstract class GlobalClass {
protected $db;
protected $table_name;
protected $format;
protected $config;
protected $check;
protected $url;
public function __construct($table_name) {
     $this->db = DataBase::getDB();
     $this->format = new Format();
     $this->config = new Config();
     $this->check = new Check();
     $this->url = new URL();
     $this->table_name = $this->config->db_prefix.$table_name;
}
public function add($data) {
     if (!$this->check($data)) return false;
     $query = "INSERT INTO `".$this->table_name."` (";
     foreach ($data as $field => $value) $query .= "`$field`,";
     $query = substr($query, 0, -1);
     $query .= ") VALUES (";
     foreach ($data as $value) $query .= $this->config->sym_query.",";
     $query = substr($query, 0, -1);
     $query .= ")";
     return $this->db->query($query, array_values($data));
```

```
    }
```

Фрагмент програмного коду class Product наведено нижче.

```
class Product extends GlobalClass {
    public function __construct() {
        parent::__construct("products");
    }
    public function getAllData($count) {
        return $this->transform($this->getAll("date", false,
$count));
    }
    public function getAllTable() {
        return $this->getAll("id");
    }
    public function getTableData($section_table, $count, $offset)
{
        $l = $this->getL($count, $offset);
        $query = "SELECT `".$this->table_name."`.`id`,
        `".$this->table_name."`.`section_id`,
        `".$this->table_name."`.`img`,
        `".$this->table_name."`.`title`,
        `".$this->table_name."`.`price`,
        `".$this->table_name."`.`description`,
        `$section_table`.`title` as `section`
        FROM `".$this->table_name."`
        INNER JOIN `$section_table` ON `$section_table`.`id` =
`".$this->table_name."`.`section_id`
        ORDER BY `date` DESC $l";
        return $this->transform($this->db->select($query));
    }
```

Фрагмент програмного коду class Section наведено нижче.

```
class Section extends GlobalClass {

    public function __construct() {
        parent::__construct("sections");
    }
```

```
    public function getAllData() {
        return $this->transform($this->getAll("id"));
    }

    public function getTableData($count, $offset) {
        return $this->transform($this->getAll("id", true,
$count, $offset));
    }

    public function get($id) {
        return $this->transform(parent::get($id));
    }

    protected function transformElement($section) {
        $section["link"] = $this->url->section($section["id"]);
        $section["link_admin_edit"] = $this->url-
>adminEditSection($section["id"]);
        $section["link_admin_delete"] = $this->url-
>adminDeleteSection($section["id"]);
        return $section;
    }
    protected function checkData($data) {
        if (!$this->check->title($data["title"])) return
"ERROR_TITLE";
        return true;
    }
}
```

Клас конфігурації (class Config). Описано основні змінні, які часто використовуються в коді (назва системи, адреса, логін і пароль для доступу до бази даних, кількість товару і т. ін.). Class Config створено для зручності розробки у випадку редагування значень змінної.

```
class Config {
 public $secret = "DFSJLFSDLJG";
 public $sitename = 'JackBlack';
```

```
    public $address = "/";
    public $address_admin = "http://JackBlack/admin/";
    public $db_host = "localhost";
    public $db_user = "root";
    public $db_password = "root";
    public $db_name = "JackBlack_local";
    public $db_prefix = "adb_";
    public $sym_query = "{?}";
    public $admname = "C";
    public $admemail = "JackBlack@live.com";
    public $adm_login = "JackBlack";
    public $adm_password = "6deaff5c019acfc2800e7f83a1a7456f";
    public $count_on_page = 9;
    public $pagination_count = 10;
    public $dir_text = "lib/text/";
    public $dir_tmpl = "tmpl/";
    public $dir_tmpl_admin = "admin/tmpl/";
    public $dir_img_products = "images/products/";
    public $max_name = 255;
    public $max_title = 255;
    public $max_text = 65535;
    public $max_size_img = 102400;
}
```

Допоміжні класи (class Format, class URL, class Check, class GlobalMessage). Виконують допоміжні функції, зокрема: class Format - відповідає за коректність представлення всіх даних; class URL - містить методи для роботи з URL; class Check - відповідає за реалізацію різноманітних перевірок на коректність; class GlobalMessage та class Message - відповідають за обробку системних повідомлень. Програмні коди вказаних класів наведено в ДОДАТКУ А.

Клас управління (class Manage). Відповідає за маніпуляції з даними (програмний код наведено в ДОДАТКУ А). В конструкторі цього класу розпочинається нова сесія та створюються екземпляри класів.

```
public function __construct() {
```

```
        session_start();
        $this->config = new Config();
        $this->format = new Format();
        $this->product = new Product();
        $this->order = new Order();
        //$this->discount = new Discount();
        $this->sm = new SystemMessage();
        $this->mail = new Mail();
        $this->url = new URL();
        $this->data = $this->format->xss($_REQUEST);
        $this->saveData();
    }
```

Далі прописані методи, що відповідають за виконання наступних дій: додавання елементу до кошику, видалення його звідти, оновлення кошику, додавання нового замовлення, виведення деяких системних повідомлень і т. ін.). Розглянемо приклад функції, що відповідає за додавання елементу до кошику.

```
    public function addCart($id = false) {
        if (!$id) $id = $this->data["id"];
        if (!$this->product->existsID($id)) return false;
        if ($_SESSION["cart"]) $_SESSION["cart"] .= ",$id";
        else $_SESSION["cart"] = $id;
    }
```

База даних і клас для роботи з базою даних (class DataBase). База даних (англ. *database*) – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами, що призначені для задоволення інформаційних потреб користувачів [1]. База даних забезпечує гарантоване збереження значних обсягів інформації (записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки (управління). За роботу із БД відповідає написаний class DataBase, програмний код якого наведено в ДОДАТКУ А.

На рис.2.6 наведено структуру бази даних інформаційної системи, що складається з трьох таблиць, що відповідають за список продукції (**adb_products**), замовлення (**adb_orders**), розділи продукції (**adb_sections**). Даної кількості таблиць бази даних цілком достатньо для ефективної роботи інформаційної системи.

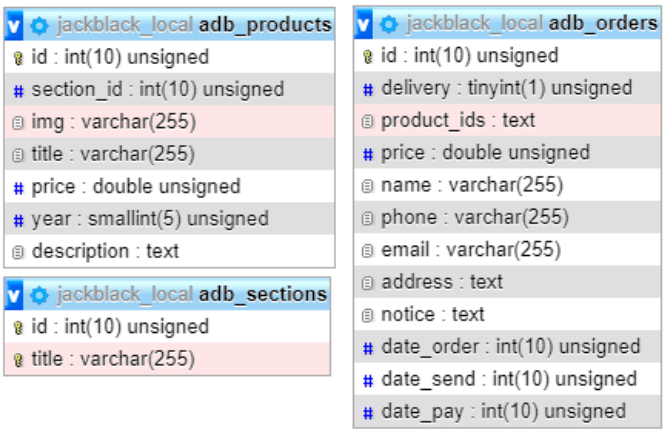

Рис.2.6. Структура БД інформаційної системи

Для забезпечення безпеки даних (зловмисник не зможе отримати доступ до таблиці, не знаючи її повної назви) назва кожної з таблиць починається із префікса «adb_». Розглянемо більш докладно кожну із таблиць.

Таблиця «**adb_sections**» містить два атрибути: «id» і «title» для запису ідентифікатора і назви розділу відповідно.

Таблиця «**adb_products**» містить наступні атрибути:

–«id», «section_id» - для запису ідентифікаторів товару і розділу, якому він належить;

–«img» - містить назву зображення товару (шлях до папки «images/products/» + назва зображення «product_img»). Для зручності шлях до розділу із зображення винесено окремо (при зміни папки потрібно буде відредагувати лише значення змінної, яка зберігає цей шлях);

–«title», «price», «year» і «description» - містять назву товару, ціну, рік виробництва і опис товару відповідно.

Таблиця «**adb_orders**» містить наступні атрибути:

«id» - містить ідентифікатор замовлення;

«delivery» - визначає спосіб доставки (приймає значення «0» - доставка, або «1» - самовивіз.

«product_ids» містить список ідентифікаторів придбаних товарів, наприклад, якщо клієнт замовив три товари з «id» = 2 і два товари з «id» = 4, то запис в полі «product_ids» буде мати вигляд «2,2,2,4,4».

«price», «name», «phone», «email», «address», «notice», «date_order», «date_send» і «date_pay» - містять інформацію стосовно ціни замовлення, ПІП клієнта, телефону, E-Mail, адреси, приміток замовлення, дати замовлення, дати відправлення і дати оплати відповідно.

Для всієї БД використовується кодування UTF-8. Таблиці збережено з використанням механізму InnoDB, що забезпечує більш надійне зберігання даних за рахунок транзакційності і блокування даних на рівні рядка в порівнянні з механізмом MyISAM.

Для написання клієнтських сценаріїв було підключено бібліотеку jQuery з CDN. CDN (Content Delivery Network) - це технологія, яка дозволяє збільшити швидкість доставки контенту кінцевим користувачам. Складається вона з великої кількості серверів, географічно розташованих в різних точках світу, на кожному з яких розташовується кеш контент. При цьому його доставка кінцевому користувачеві здійснюється зазвичай з того сервера, який ближче інших розташований до нього. В результаті чого скорочується час його завантаження, прискорюється відгук, збільшується продуктивність сайту, а також знижується навантаження на оригінальний сервер. Тобто CDN надає ще один спосіб підключити бібліотеку jQuery. При цьому безпосередньо завантажувати його собі на сервер не потрібно, він буде братися з CDN [29].

Завантаження jQuery з CDN надають безліч компаній, таких як: Google, Microsoft, Cloudflare, jQuery, Yandex і т.ін. В даній роботі було здійснено онлайн підключення jQuery останньої версії (3.5.1) з Google CDN в файлі main.tpl.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min
.js"></script>
<script>
    $(function(){
        var header = $("#menu"),
            introH = $("#intro").innerHeight(),
            scrollOfset = 0;

        $(window).on("scroll", function(){
           scrollOfset = $(this).scrollTop();
            /*console.log(introH);
            console.log(scrollOfset);*/
            if(scrollOfset > introH){
                menu.style.display="block";
                menu.style.width="auto";
                menu.style.marginTop="30px";
            }
            else{
                menu.style.display="flex";
                menu.style.width="100%";
                menu.style.marginTop="0px";
            }
        });
    });
</script>
```

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Програмне забезпечення отримує вхідні дані шляхом завантаження інформації із бази даних та сховищ даних, та через передачу значень від інших підсистем через змінні інформаційної системи.

За введення вхідних даних відповідають наступні модулі системи:

- шаблонізатор. Відповідає за виведення сторінок інформаційної системи, підставляючи потрібну інформацію. Описаний в class Template;
- класи сторінок. Батьківським класом для них є абстрактний class Modules, в якому встановлюємо значення змінних, які будуть підставлятися шаблонізатором;
- класи для роботи з таблицями. Батьківським класом для них є абстрактний GlobalClass, що містить загальні методи для всіх дочірніх класів, таких як Product або Section;
- клас конфігурації. В class Config описано основні змінні, котрі часто використовуються в коді (назва інформаційної системи або її адреса, логін і пароль для доступу до бази даних, кількість товару, що одночасно виводиться на сторінці і т. ін.);
- клас для роботи з БД (class DataBase);
- клас конфігурації. В class Config описано основні змінні, котрі часто використовуються в коді (назва системи або її адреса, логін і пароль для доступу до бази даних, кількість товару, що одночасно виводиться на сторінці і т. ін.);
- Допоміжні класи (class Format, class URL, class Check, class GlobalMessage).

Вхідні дані:

- шаблони списків;
- шаблони таблиць;
- шаблони індивідуальних сторінок системи та товарів;
- графічні і текстові матеріали про товар;

Вихідні дані:

- веб-сторінки інформаційної системи веб-сайт «JackBlack», який реалізує елітні алкогольні напої.

2.7. Опис розробленого програмного продукту

2.7.1. 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.

2.7.2. Використані програмні засоби

Проект реалізований на мові програмування PHP. Для верстання сторінок застосовано сучасні технології HTML5 та CSS3 (застосовується в основному блочний метод верстання сторінок, що надає можливість створювати адаптивні сторінки). Для написання клієнтських сценаріїв - бібліотеку jQuery. В якості СУБД була обрана MySQL - реляційна СУБД з вільною ліцензією. Інформаційна система працює на веб-сервері Apache HTTP Server. Для зручної розробки і налагодження сайту на локальному ПК без необхідності підключення до мережі Інтернет, було застосовано програмну оболонку Denwer

в якості локального веб-сервера. Розробка проводилася із використанням текстового редактора Brackets. Для коректної роботи розробленого додатка необхідний довільний веб-браузер.

Необхідні програмні засоби:

Для клієнтського комп'ютера:

– будь-який мобільний або стаціонарний пристрій, підключений до мережі Інтернет, на якому можна запустити довільний веб-браузер.

Для сервера:

– серверна ОС сімейства Microsoft Windows, Linux, FreeBSD;
– СУБД MySQL;

2.7.3. Виклик та завантаження програми

Для отримання доступу до інформаційної системи необхідно в адресному рядку браузера ввести її адресу JackBlack та натиснути клавішу «Enter» (рис.2.7).

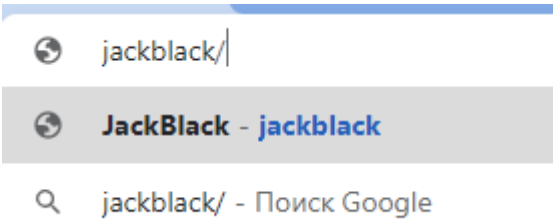

Рис.2.7. Виклик та завантаження програми

2.7.4. Опис інтерфейсу користувача

2.7.4.1. Клієнтська частина

Набравши в адресному рядку браузера веб-адресу інформаційної системи, користувач потрапляє на головну сторінку (рис.2.8).

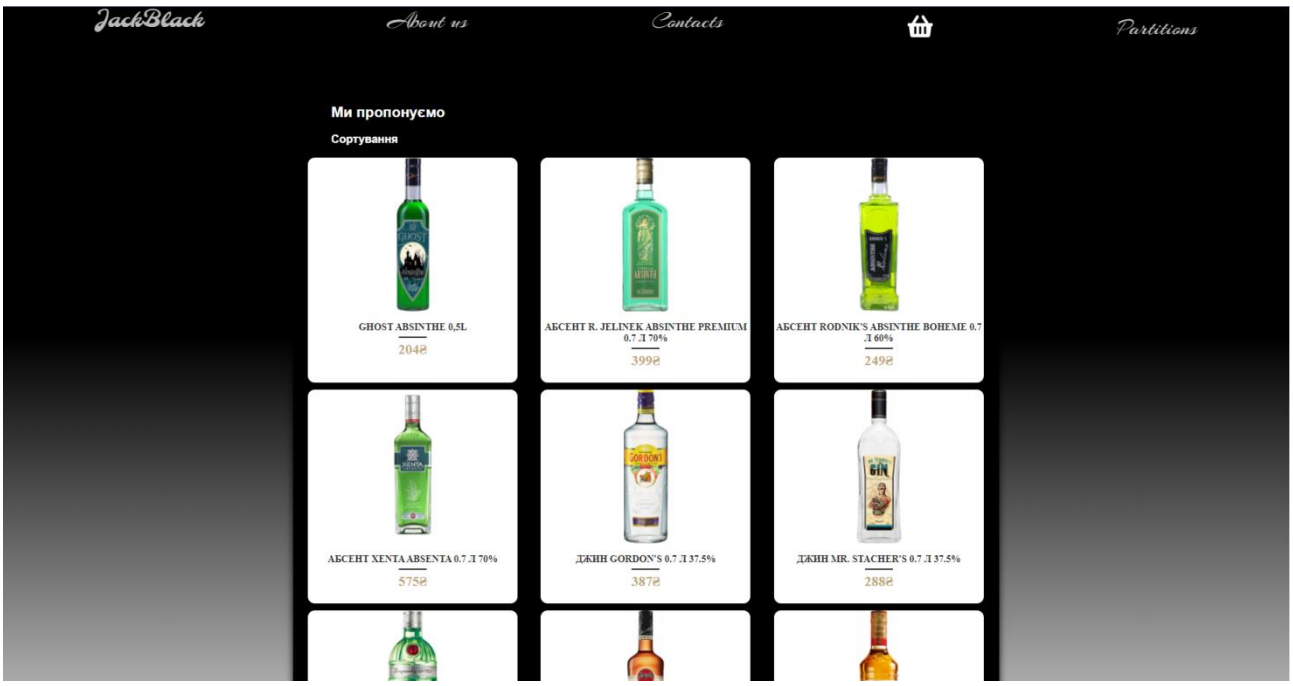

Рис.2.8. Головна сторінка інформаційної системи

В верхній частині розміщено переходи на наступні розділи: «Головна сторінка», «Про компанію», «Контакти компанії», «Кошик» та «Розділи», в якому можна обрати необхідний розділ товарів. В центральній частині розміщені товари, які представлено на сайті, перехід до яких здійснюється натисканням на відповідній області (рис.2.9).

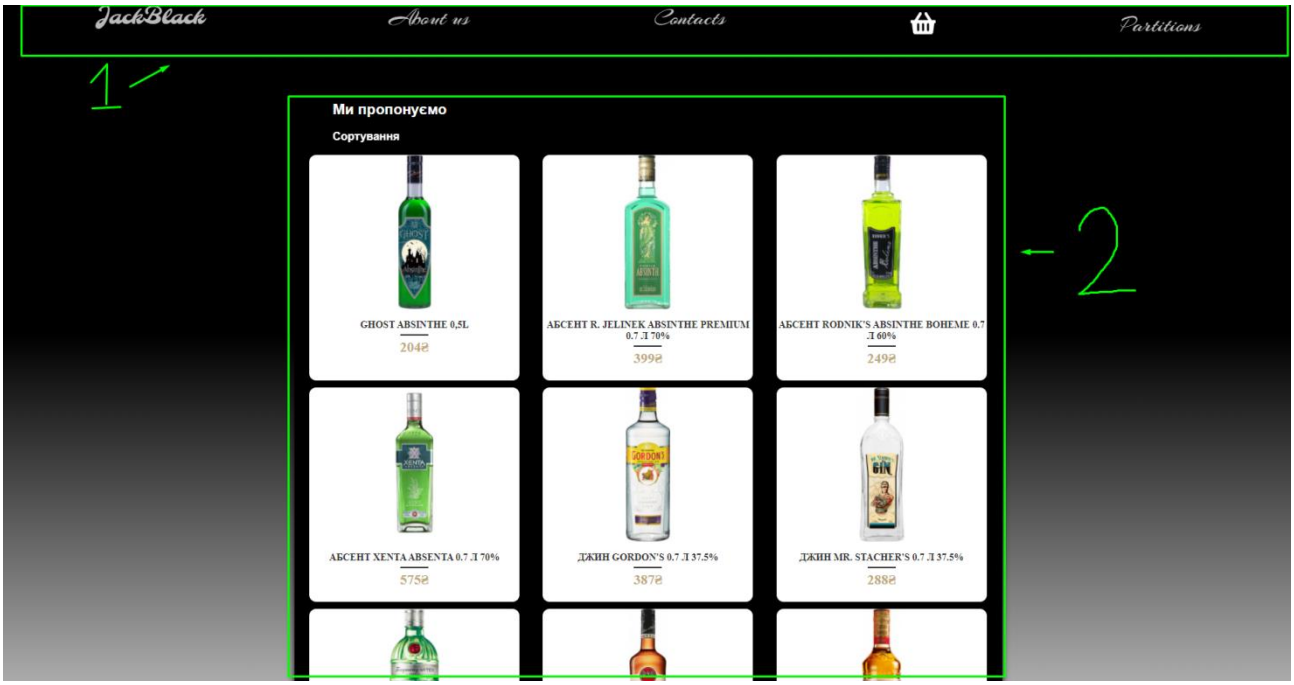

Рис.2.9. Основні шляхи навігації по інформаційній системі

При натисканні на пункти меню «About us» і «Contacts» користувачу надаються сторінки системи з відомостями про компанію і контакти, відповідно (рис.2.10-2.11). Дана інформація є тестовою, а тому тимчасовою і недостовірною.

Рис.2.10. Сторінка системи «About us»

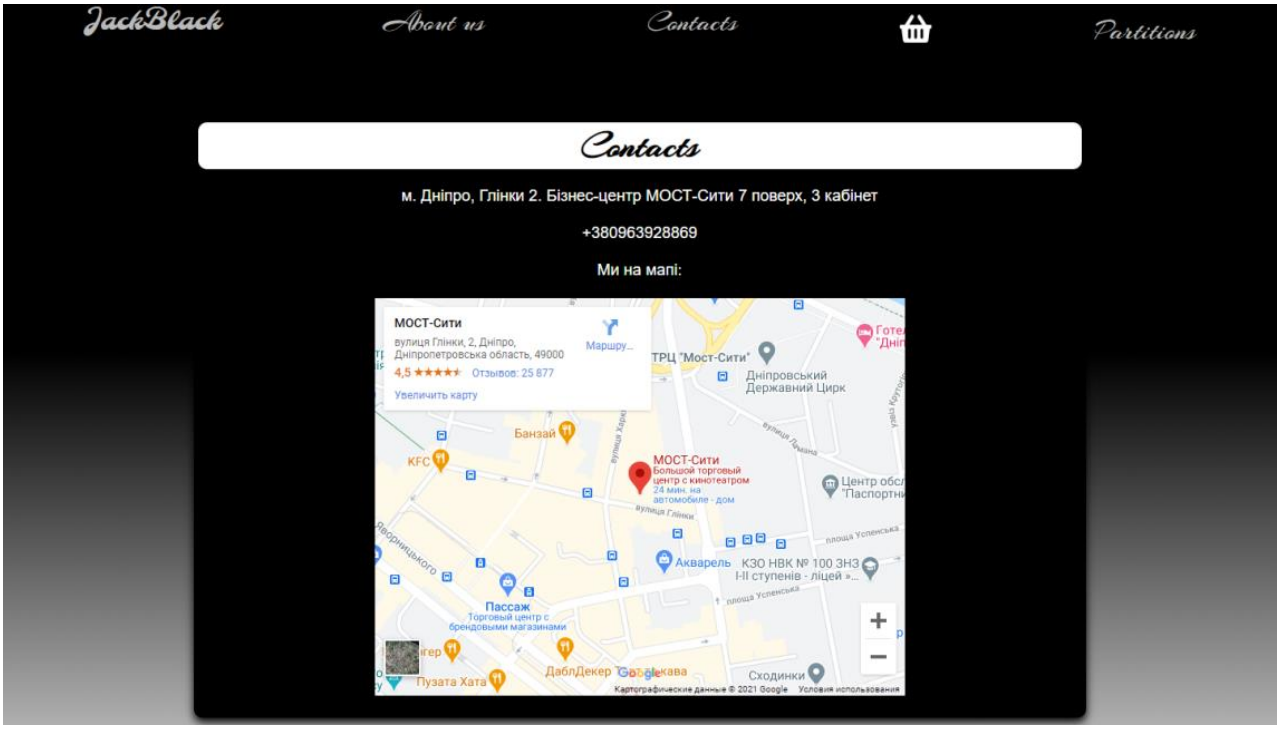

Рис.2.11. Сторінка системи «Contacts»

При прокручуванні сторінок головне меню навігації системи змінює своє положення, та розміщується ліворуч, як зображено на рис.2.12.

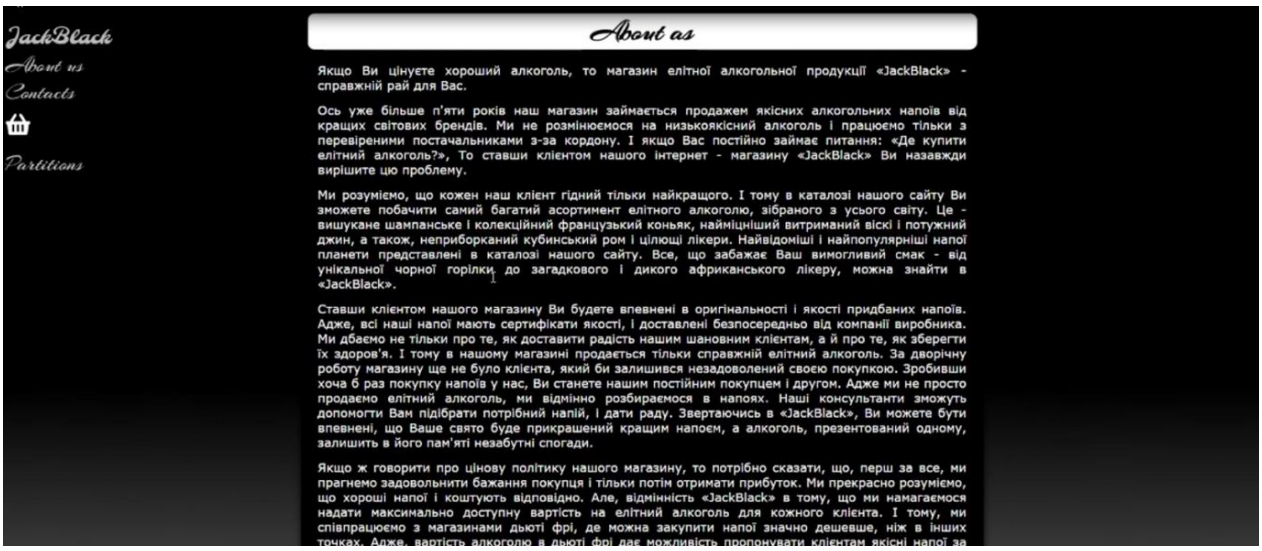

Рис.2.12. Зміна положення навігаційного меню

В системі передбачено сортування товарів (як на головній сторінці так і в обраних розділах товарів) за наступними критеріями: "За зростанням ціни", "За спадом ціни", "За алфавітом", "Проти алфавіту" (рис.2.13). Дані можливості з'являються при наведенні курсору на заголовок "Сортування".

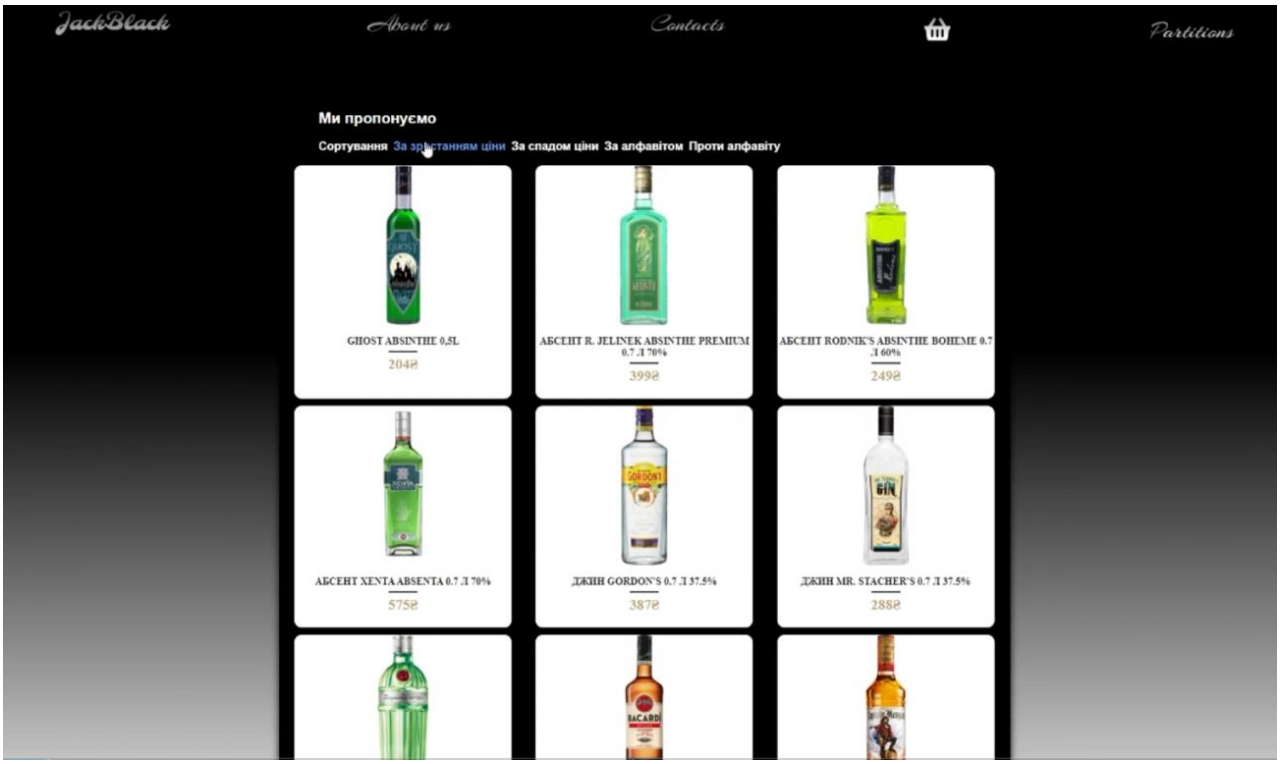

Рис.2.13. Сортування товару

При необхідності придбання товарів з головної сторінки достатньо навести курсор на відповідний товар та натиснути на спливаючу кнопку "В КОРЗИНУ" (рис.2.14).

Рис.2.14. Придбання товару

Розділ "Partitions" (рис.2.15) містить розділи продукції, що було сформовано згідно з класифікаційною ознакою.

Рис.2.15. Вміст меню «Partitions»

Натиснувши на будь-який із пунктів меню "Partitions", користувач потрапить на сторінку із продукцією саме цього розділу (рис. 2.16).

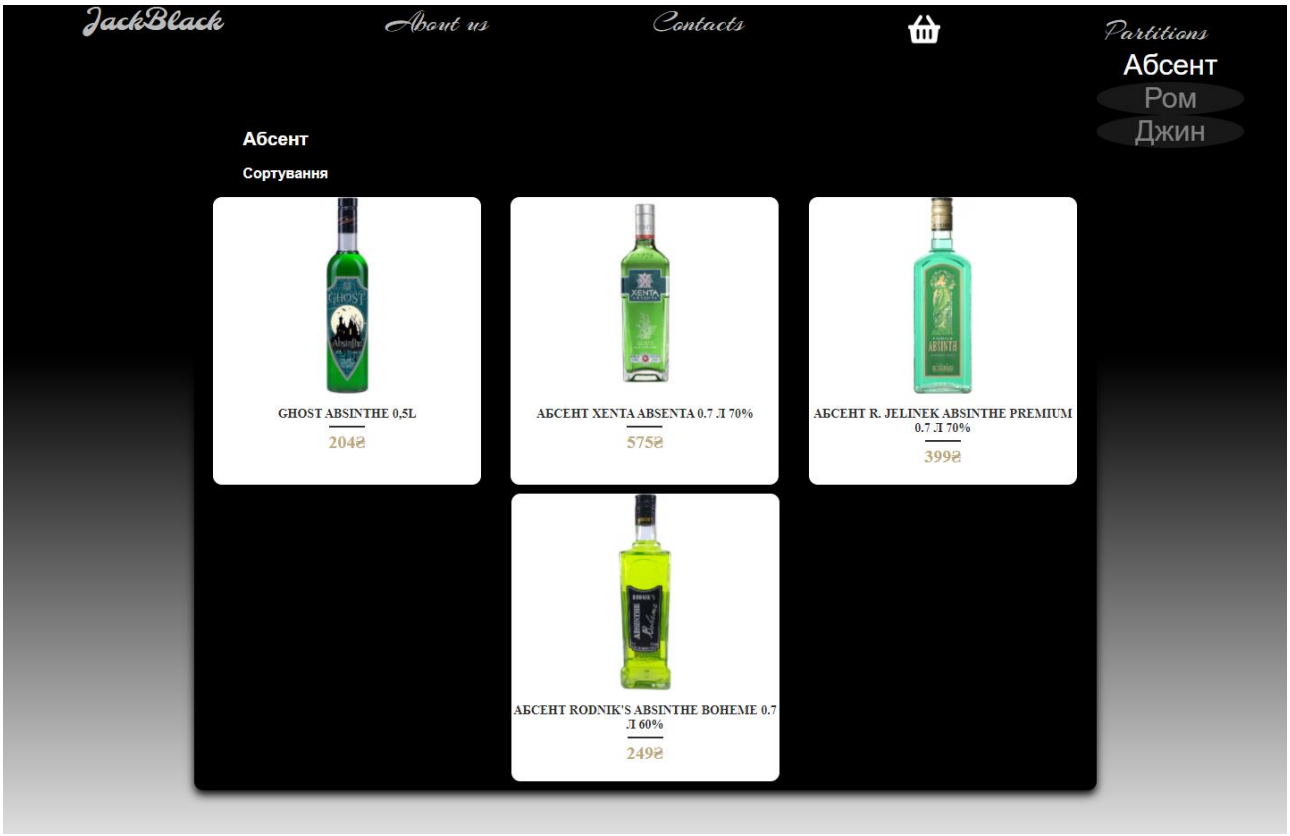

Рис.2.16. Сторінка обраного розділу

Для отримання більш детальної інформації про наведену продукцію, необхідно натиснути на сторінку опису обраного товару (рис.2.17).

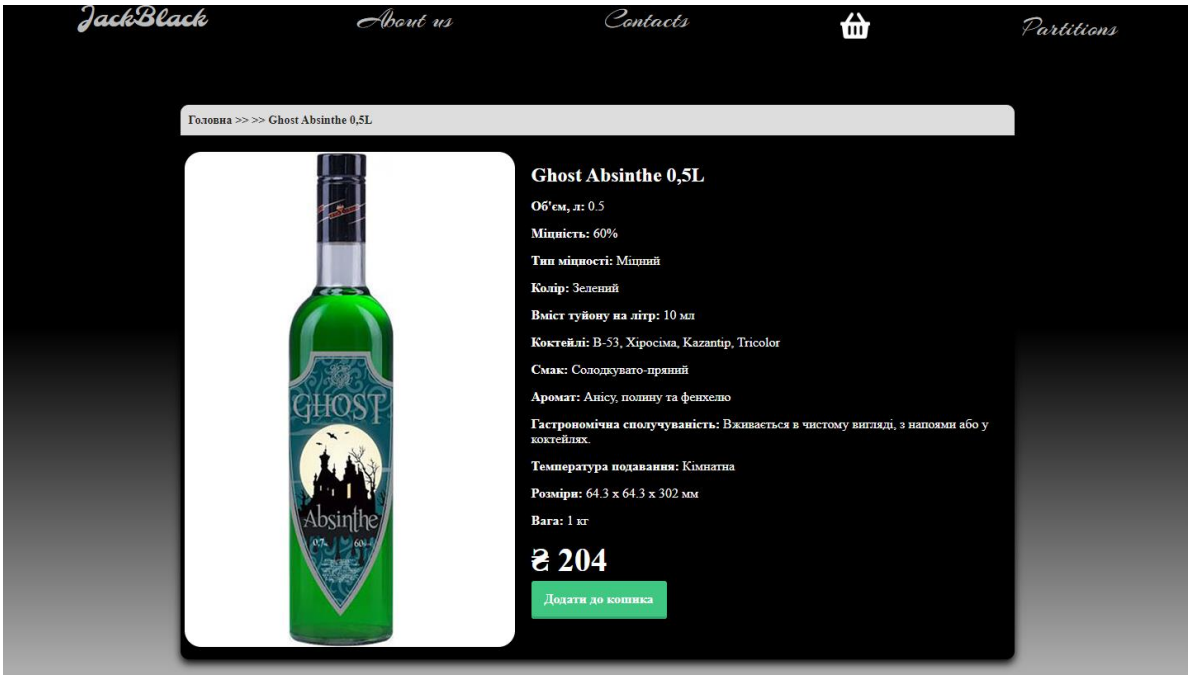

Рис.2.17. Сторінка обраного товару

Також на даній сторінці є можливість здійснити покупку даного товару, натиснувши на кнопку "Додати до кошика" (рис.2.18).

Рис.2.18. Придбання обраного товару

Натиснувши на стилізоване зображення кошика (знаходиться в заголовку системи) - рис.2.19, користувач потрапляє на сторінку «Кошик» (рис.2.20), де відображається інформація стосовно обраного до покупки товару (зображення, назва, ціна, кількість, загальна ціна товару). За необхідності користувач має можливість змінити кількість обраного товару (ввівши нове значення у відповідній формі) та видалити обраний товар із списку.

Рис.2.19. Стилізоване зображення кошику

Рис. 2.20. Сторінка «Кошик»

Для подальшої роботи із системою користувач натискає на кнопку "Оформити замовлення" (рис.2.21), де йому пропонується ввести необхідні контактні дані (ПІП, контактний телефон, e-mail, спосіб доставки, адресу) та примітки до замовлення у відповідні поля запропонованої форми (рис.2.22).

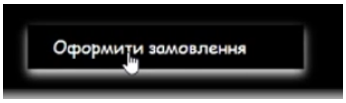

Рис.2.21. Стилізоване зображення кнопки "Оформити замовлення"

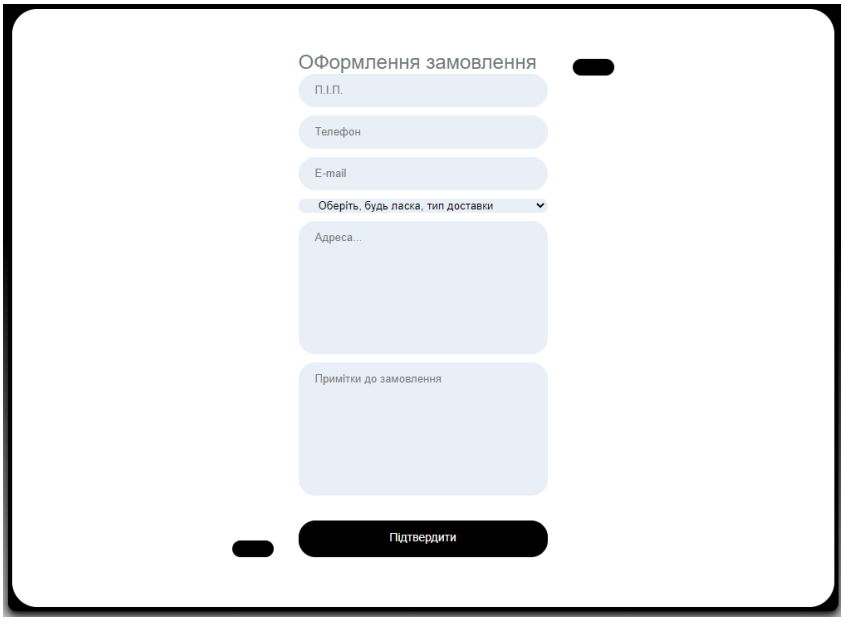

Рис. 2.22. Сторінка "Оформлення замовлення"

Для підтвердження введеної інформації необхідно натиснути кнопку "Підтвердити". Після чого користувачу буде надіслано повідомлення про успішне отримання замовлення (рис.2.23).

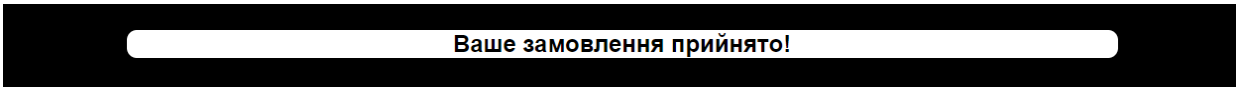

Рис.2.23. Успішне отримання замовлення

Щоб повернутися на головну сторінку можна скористатися різними посиланнями: натиснувши на логотип компанії у верхньому лівому кутку екрану, або натиснувши на слово «Головна» у шляху товару (рис.2.24).

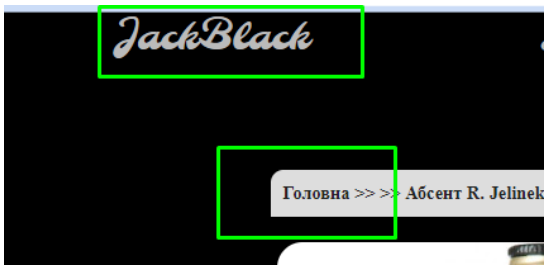

Рис.2.24. Повернення на головну сторінку

2.7.4.2. Адміністрування системою

Для управління роботою інформаційної системи і її вмістом було розроблену адміністративну панель, що надає можливості адміністратору системи отримати повний контроль над системою.

Для отримання доступу до admin-панелі необхідно в адресному рядку браузера ввести її адресу http://<адреса>/admin/ та натиснути клавішу «Enter».

При першому вході перед адміністратором з’являється сторінка входу в admin-панель (рис.2.25), де необхідно підтвердити права адміністратора, ввівши логін і пароль.

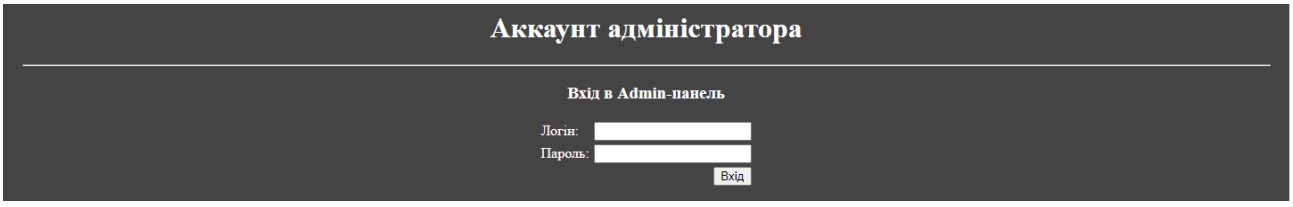

Рис.2.25. Сторінка входу в admin-панель

При успішній авторизації, адміністратор системи потрапляє до головного меню адміністративної панелі, де отримує доступ до управління (перегляд, редагування, додавання та видалення інформації) інформаційною системою (рис.2.26).

Рис.2.26. Головна сторінка admin-панелі

Адміністратор системи має права управління наступними розділами адміністративного меню: "Товари", "Замовлення", "Розділи", "Статистика".

На рис.2.27 представлено сторінку управління "Розділи".

Рис.2.27. Сторінка управління розділами

За необхідності, адміністратор має можливості редагування і видалення зазначених розділів системи. На рис.2.28-2.30 наведено механізм редагування розділу "Абсент".

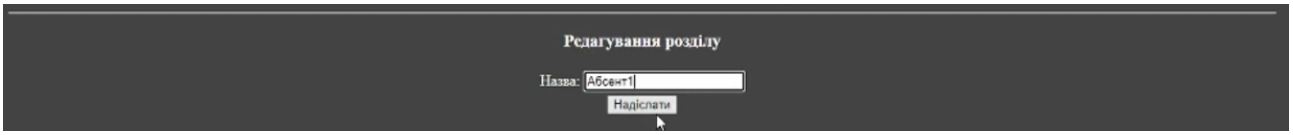

Рис.2.28. Меню "Редагування розділу"

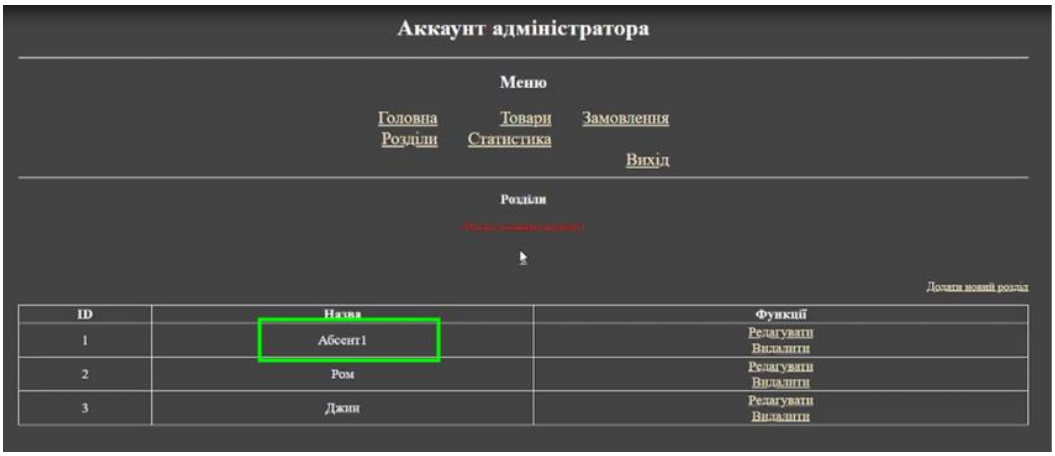

Рис.2.29. Зміна назви розділу

Рис.2.30. Зображення зміни назву розділу

На рис.2.31-2.32 зображено сторінку "Замовлення" admin-панелі та механізм редагування замовлення, відповідно.

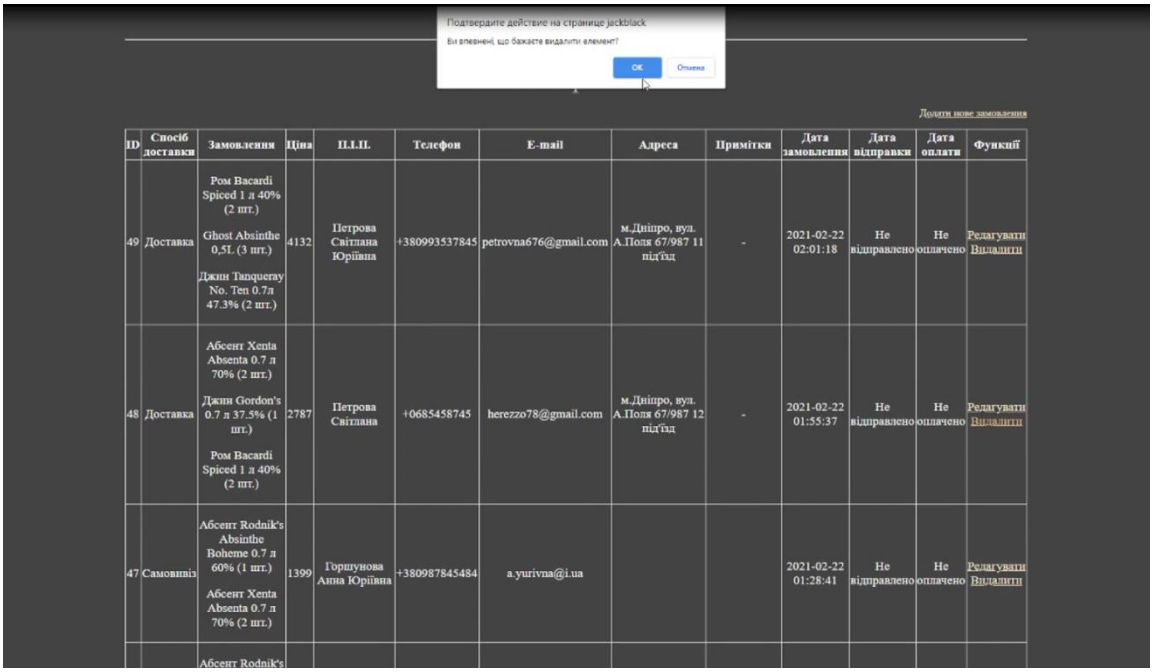

Рис.2.31. Сторінка "Замовлення" admin-панелі

На рис.2.33 наведено сторінку "Статистика" admin-панелі, де адміністратор має можливість переглянути статистику замовлень за певний період (виставляється у відповідних полях).

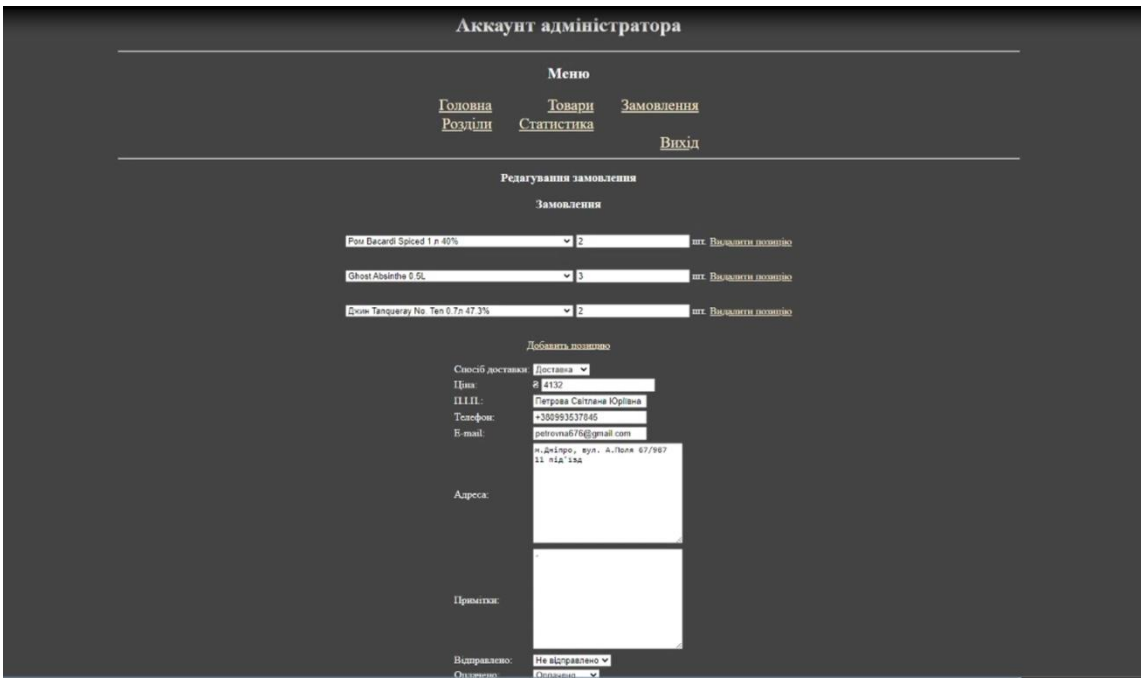

Рис.2.32. Форма "Редагування замовлення" admin-панелі

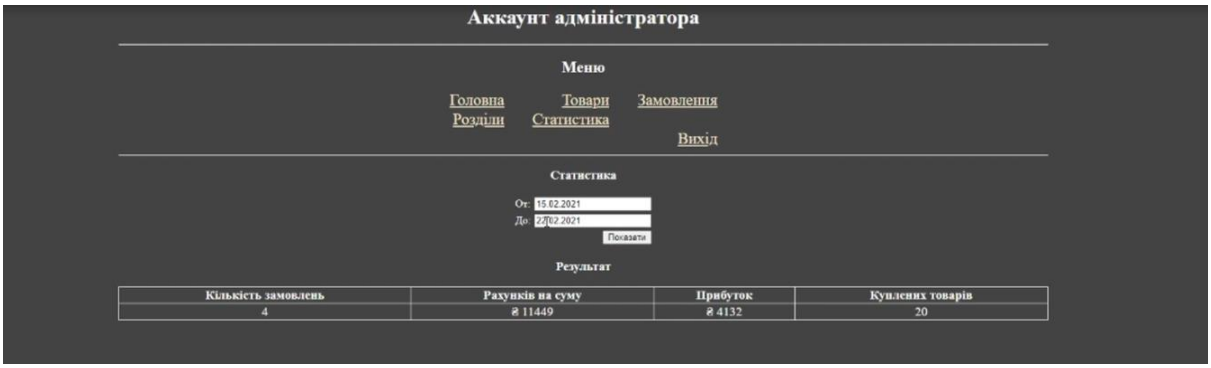

Рис.2.33. Сторінка "Статистика" admin-панелі

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. Передбачуване число операторів програми - 700;
2. Коефіцієнт складності програми (1,25...2,0) - 1,6;
3. Коефіцієнт корекції програми в ході її розробки (0,05...0,1) - 0,06;
4. Годинна заробітна плата програміста - 84 грн/год;
5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2...1,5) - 1,2;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності (від 3-х до 5 років) – 1,1;
7. Вартість машино-години ЕОМ - 14 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:

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

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 програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q - передбачуване число операторів програми (q =700);

C - коефіцієнт складності програми (C =1,6);

p - коефіцієнт корекції програми в ході її розробки (p =0,06).

Звідси умовне число операторів в програмі:

$$Q = 700 \cdot 1{,}6 \cdot (1 + 0{,}06) = 1187{,}2.$$

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

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,} \tag{3.3}$$

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 1,1.

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

$$t_u = (1187{,}2 \cdot 1{,}2) / (80 \cdot 1{,}1) = 16{,}19 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25) \cdot k}\ \text{, людино-годин,} \tag{3.4}$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), отримаємо:

$$t_a = 1187{,}2 / (21 \cdot 1{,}1) = 51{,}39 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,} \qquad (3.5)$$

t_n = 1187,2 / (25 · 1,1) = 43,17 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин,} \qquad (3.6)$$

$t_{отл}$= 1187,2 / (5 · 1,1) = 215,85 людино-годин.

– за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5\cdot t_{отл}\text{, людино-годин,} \qquad (3.7)$$

$t^k_{отл}$= 1,5 · 215,85 = 323,77 людино-годин.

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}\ \text{, людино-годин,} \qquad (3.8)$$

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

$$t_{др} = \frac{Q}{(15..20)\cdot k}\ \text{, людино-годин,} \qquad (3.9)$$

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

$$t_{до} = 0{,}75\cdot t_{др}\ \text{, людино-годин.} \qquad (3.10)$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 1187,2 / (18 · 1,1) = 59,96 людино-годин.

$t_{до}$ = 0,75 · 59,96 = 44,97 людино-годин.

$t_д$= 59,96 + 44,97 = 104,93 людино-годин.

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

t = 50 + 16,19 + 51,39 + 43,17 + 215,85 + 104,93 = 481,53 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 витрати машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.11)$$

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 60 грн / год, отримуємо:

$$З_{ЗП} = 481{,}53 \cdot 60 = 28891{,}8 \text{ грн.}$$

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

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.12)$$

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

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (14 грн/год).

Підставивши в формулу (3.12) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 215{,}85 \cdot 14 = 3021{,}9 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 28891{,}8 + 3021{,}9 = 31913{,}7 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс,} \quad (3.13)$$

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:

$$T = 481{,}53 / 1 \cdot 176 = 2{,}73 \text{ міс.}$$

Висновок

Програмне забезпечення призначене для реалізації веб-орієнтованої інформаційної системи з метою автоматизації діяльності компанії в сфері електронної комерції. Вартість даного програмного забезпечення становить 31913,7 грн. і не вимагає додаткових витрат при впровадженні та експлуатації програми. Очікуваний час розробки становить приблизно 3 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

Метою кваліфікаційної роботи бакалавра є розробка програмного забезпечення веб-орієнтованої інформаційної системи для автоматизації діяльності компанії в сфері електронної комерції.

Інформаційна система представляє собою веб-сайт типу інтернет-магазин, що призначений для:

- об’єднання елементів прямого маркетингу та традиційної торгівлі;
- надання універсального інструменту управління візуальним контентом та адміністративної частини проекту;
- забезпечення відвідувачам сайту максимальної зручності роботи із системою, простоти і комфортності доступу до каталогу товарів за рахунок оптимальних параметрів візуалізації його вмісту;
- підвищення ефективності роботи компанії, за рахунок автоматизації управління бізнес-процесами.

Для управління роботою інформаційної системи і її вмістом було розроблену адміністративну панель, що надає можливості адміністратору системи отримати повний контроль над системою (у разі успішної авторизації), що недоступно звичайному користувачу. Адміністратор системи має права управління (перегляд, редагування, додавання та видалення інформації) наступними розділами адміністративного меню: "Товари", "Замовлення", "Розділи", "Статистика".

Актуальність розробки інформаційної системи для автоматизації діяльності компанії в сфері електронної комерції не викликає сумніву та визначається великим попитом на подібні системи, оскільки електронна комерція робить торгівлю більш гнучкою, зрозумілою, відкритою й стандартизованою, а також надає значні переваги (що особливо відчутні в умовах COVID-19): мобільність, безконтактність, ціна, масштабованість, оперативність реагування на зміни ринку товарів і послуг, швидкість і ефективність здійснення комерційних операцій і т.ін.

Розроблене програмне забезпечення інформаційної системи призначене для застосування в будь-якій компанії з подібним родом діяльності і схожими функціональними вимогами.

Створена система дозволить оптимізувати та спростити дії по веденню операцій онлайн-продажів; скоротити час на оформлення продажу; підвищити ефективність діяльності компанії за рахунок автоматизації його діяльності в сфері електронної комерції та можливості моніторингу стану системи і аналізу наявних даних (за рахунок наявності адміністративної частини проекту).

Проект реалізований на мові програмування PHP. Для верстання сторінок застосовано сучасні технології HTML5 та CSS3 (застосовується в основному блочний метод верстання сторінок, що надає можливість створювати адаптивні сторінки). Для написання клієнтських сценаріїв - бібліотеку jQuery. В якості СУБД була обрана MySQL - реляційна СУБД з вільною ліцензією. Інформаційна система працює на веб-сервері Apache HTTP Server. Для зручної розробки і налагодження сайту на локальному ПК без необхідності підключення до мережі Інтернет, було застосовано програмну оболонку Denwer в якості локального веб-сервера. Розробка проводилася із використанням текстового редактора Brackets. Для коректної роботи розробленого додатка необхідний довільний веб-браузер.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (481,53 людино-годин), підраховані витрати на створення програмного забезпечення (31913,7 грн.) і гаданий період розробки (приблизно 3 місяці).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. База даних [Електронний ресурс]//Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: https://uk.wikipedia.org/ База_даних (дата звернення: 06.04.2021).

2. Дронов В. А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В.А.Дронов - СПб.: БХВ-Петербург, 2011. - 416 с.: ил. - (Профессиональное программирование).

3. Денвер (программа) [Електронний ресурс] // Википедия. Свободная энциклопедия - Режим доступу до ресурсу:https://ru.wikipedia.org/wiki/ Денвер_(программа) (дата звернення: 06.04.2021).

4. Десять лучших IDE и редакторов кода для веб-разработчиков [Електронний ресурс]/ URL: https://www.reg.ru/blog/10-luchshih-ide-i-redaktorov-koda-dlya-veb-razrabotchikov/ (дата звернення: 06.04.2021).

5. Електронна комерція [Електронний ресурс]// Вікіпедія. Вільна енциклопедія/ URL: https://uk.wikipedia.org/wiki/Електронна_комерція (дата звернення: 06.04.2021).

6. Загрози інформаційної безпеки [Електронний ресурс] / URL: https://uk.wikipedia.org/wiki/Загрози_інформаційної_безпеки (дата звернення: 06.04.2021).

7. Інформаційна безпека [Електронний ресурс]//Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Інформаційна_безпека (дата звернення: 06.04.2021).

8. Інформаційна система [Електронний ресурс] // Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Інформаційна_система (дата звернення: 06.04.2021).

9. Информационная безопасность веб-приложений: как защитить свой интернет-магазин [Електронний ресурс] / URL: https://astrio.ru/blog/magento-security/ (дата звернення: 06.04.2021).

10. Исследование популярности CMS за 2021 год [Електронний ресурс]/ URL: https://itrack.ru/research/cmsrate/(дата звернення: 06.04.2021).

11. Класифікація веб-сайтів [Електронний ресурс] / URL: https://studopedia.com.ua/1_10505_klasifikatsiya-veb-saytiv.html (дата звернення: 06.04.2021).

12. Концепция Web-2.0// URL:http://x-site.by/info/concept-of-web-20 (дата звернення: 06.04.2021).

13. Організація і технологія роботи Internet-магазину [Електронний ресурс]/URL:https://pidru4niki.com/10931123/informatika/organizatsiya_tehnologiya_roboti_internet-magazinu (дата звернення: 06.04.2021).

14. Пасічник О.Г. Основи веб-дизайну /О.Г.Пасічник, О.В.Пасічник, І.В.Стеценко: [Навч.посіб.].-К.:Вид. група BHV.- 2009 .- 336 с.: іл.

15. Переваги та недоліки JavaScript [Електронний ресурс] // URL: https://hackit-ukraine.com/627-the-advantages-and-disadvantages-of-javascript (дата звернення: 06.04.2021).

16. Преимущества и недостатки языка Python [Електронний ресурс]/ URL: https://blog.ithillel.ua/articles/preimushchestva-i-nedostatki-yazyka-python (дата звернення: 06.04.2021).

17. Призначення та особливості CMS [Електронний ресурс]/ URL: https://armedsoft.com/ua/blog/ pryznachennya-ta-osoblyvosti-cms (дата звернення: 06.04.2021).

18. Росс В. С. Создание сайтов: HTML, CSS, PHP, MySQL/ В.С.Росс [Учебное пособие, ч. 1]- МГДД(Ю)Т, М.:2010 - 107 с.

19. Система управления содержимым [Електронний ресурс] // Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Система_ управления_ содержимым (дата звернення: 06.04.2021).

20. Система управління базами даних [Електронний ресурс] // Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу:

https://uk.wikipedia.org/wiki/Система_управління_базами_даних (дата звернення: 06.04.2021).

21. Скриптова мова [Електронний ресурс] // Вікіпедія. Вільна енциклопедія - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Скриптова_мова (дата звернення: 06.04.2021).

22. Статистика GitHub за 2020 рік - яка найпопулярніша мова програмування [Електронний ресурс]. - Режим доступу до ресурсу: https://nachasi.com/2020/12/03/github-statistics-2020/ (дата звернення: 06.04.2021).

23. Статичні та динамічні web-сайти/ URL: https://armedsoft.com/ua/blog/statychni-ta-dynamichni-web-sayty (дата звернення: 06.04.2021).

24. Ташков П. А. Веб-мастеринг на 100 %: HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка /П.А.Ташков. - СПб.: Питер, 2010. - 512 с.: ил. - (Серия «На 100%»).

25. СУБД MySQL [Електронний ресурс]/ URL: https://www.methodlab.ru/technology/mysql.shtml (дата звернення: 06.04.2021).

26. Тардаскіна Т.М. Електронна комерція: Навчальний посібник / Т.М.Тардаскіна, Є.М.Стрельчук, Ю.В.Терешко – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 244 с.

27. Трёхуровневая архитектура [Електронний ресурс] // Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Трёхуровневая_архитектура (дата звернення: 09.02.2021).

28. Топ-10 актуальних вразливостей від OWASP у 2020 році [Електронний ресурс]/ URL: https://qagroup.com.ua/publications/tpo-10-vulnerability-owasp/ (дата звернення: 06.04.2021).

29. Что такое jQuery? Как его скачать и подключить к сайту? [Електронний ресурс]/ URL: https://itchief.ru/javascript/jquery-introduction (дата звернення: 06.04.2021).

30. Шалева О. І. Електронна комерція. Навч. посіб. – К.: Центр учбової літератури, 2011. – 216 с.

31. Apache HTTP Server [Електронний ресурс] // Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Apache_HTTP_Server (дата звернення: 06.04.2021).

32. FORUM.PHP.SU [Електронний ресурс] - Режим доступу до ресурсу: http://www.php.su/learnphp/ (дата звернення: 06.04.2021).

33. JavaScript [Електронний ресурс] // Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/JavaScript (дата звернення: 06.04.2021).

34. O'Reilly Tim What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software [Virtual resource]. – 2005. – Oct. 30. – Access path: www.oreillynet.com/pub/a/oreilly /tim/news/2005/09/30/what-is-web-20.html.

35. OWASP [Електронний ресурс] //Википедия. Свободная энциклопедия - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/OWASP (дата звернення: 06.04.2021).

36. Python [Електронний ресурс] // Википедия. Свободная энциклопедия- Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Python (дата звернення: 06.04.2021).

37. Php достоинства и недостатки [Електронний ресурс]/ URL: https://web-shpargalka.ru/php-dostoinstva-i-nedostatki.php (дата звернення: 06.04.2021).

38. PhpMyAdmin [Електронний ресурс]/ URL: https://ru.wikipedia.org/wiki/PhpMyAdmin (дата звернення: 06.04.2021).

39. Three-Tier Architecture [Електронний ресурс] // Techopedia/ URL: https://www.techopedia.com/definition/24649/three-tier-architecture (дата звернення: 06.04.2021).

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг файлу index.php

```
<?php
	require_once "start.php";

	require_once "url_class.php";

	$url = new URL();
	$view = $url->getView();

	$class = mb_strtolower($view."Content");
	if ($url->fileExists($class."_class.php")) {
		require_once $class."_class.php";
		new $class();

	$url1 = ((!empty($_SERVER['HTTPS'])) ? 'https' : 'http') . '://' . $_SERVER['HTTP_HOST'] .
$_SERVER['REQUEST_URI'];

	//echo $url1;
	if($url1=="http://jackblack/" /*false*/)
	{
	   echo '<script>location.replace("http://jackblack/?sort=title&up=1");</script>';
	}
	//exit;
	}
	else {
		header("Location: ".$url->notFound());
		exit;
	}

?>
```

Лістинг файлу functions.php

```
<?php
	require_once "start.php";

	require_once "manage_class.php";
	require_once "url_class.php";

	$manage = new Manage();
	$url = new URL();
	$func = $_REQUEST["func"];
	if ($func == "add_cart") {
		$manage->addCart();
	}
	elseif ($func == "delete_cart") {
		$manage->deleteCart();
	}
	elseif ($func == "cart") {
		$manage->updateCart();
	}
	elseif ($func == "order") {
		$success = $manage->addOrder();
	}
	elseif ($func == "success_pay") {
		$success = $manage->successPay();
```

```
    }
    elseif ($func == "fail_pay") {
        $success = $manage->failPay();
    }
    elseif ($func == "status_pay") {
        $success = $manage->statusPay();
    }
    else exit;
    if ($success) {
        $link = $url->message();
    }
    else {
        $link = ($_SERVER["HTTP_REFERER"] != "")? $_SERVER["HTTP_REFERER"]: $url->index();
    }
    header("Location: $link");
    exit;
?>
```

Фрагмент лістингу файлу manage_class.php

```
<?php

class Manage {

    protected $config;
    protected $format;
    protected $product;
    protected $order;
    protected $discount;
    protected $url;

    public function __construct() {
        session_start();
        $this->config = new Config();
        $this->format = new Format();
        $this->product = new Product();
        $this->order = new Order();
        //$this->discount = new Discount();
        $this->sm = new SystemMessage();
        $this->mail = new Mail();
        $this->url = new URL();
        $this->data = $this->format->xss($_REQUEST);
        $this->saveData();
    }

    private function saveData() {
        foreach ($this->data as $key => $value) $_SESSION[$key] = $value;
    }

    public function addCart($id = false) {
        if (!$id) $id = $this->data["id"];
        if (!$this->product->existsID($id)) return false;
        if ($_SESSION["cart"]) $_SESSION["cart"] .= ",$id";
        else $_SESSION["cart"] = $id;
    }

    public function deleteCart() {
        $id = $this->data["id"];
        $ids = explode(",", $_SESSION["cart"]);
        $_SESSION["cart"] = "";
        for ($i = 0; $i < count($ids); $i++) {
                if ($ids[$i] != $id) $this->addCart($ids[$i]);
        }
```

```
    }

    public function updateCart() {
        $_SESSION["cart"] = "";
        foreach ($this->data as $k => $v) {
                if (strpos($k, "count_") !== false) {
                        $id = substr($k, strlen("count_"));
                        for ($i = 0; $i < $v; $i++) $this->addCart($id);
                }
        }
        $_SESSION["discount"] = $this->data["discount"];
    }

    public function addOrder() {
        $temp_data = array();
        $temp_data["delivery"] = $this->data["delivery"];
        $temp_data["product_ids"] = $_SESSION["cart"];
        $temp_data["price"] = $this->getPrice();
        $temp_data["name"] = $this->data["name"];
        $temp_data["phone"] = $this->data["phone"];
        $temp_data["email"] = $this->data["email"];
        $temp_data["address"] = $this->data["address"];
        $temp_data["notice"] = $this->data["notice"];
        $temp_data["date_order"] = $this->format->ts();
        $temp_data["date_send"] = 0;
        $temp_data["date_pay"] = 0;
        $id = $this->order->add($temp_data);
        if ($id) {
                $send_data = array();
                $send_data["products"] = $this->getProducts();
                $send_data["name"] = $temp_data["name"];
                $send_data["phone"] = $temp_data["phone"];
                $send_data["email"] = $temp_data["email"];
                $send_data["address"] = $temp_data["address"];
                $send_data["notice"] = $temp_data["notice"];
                $send_data["price"] = $temp_data["price"];
                $to = $temp_data["email"];
                $this->mail->send($temp_data["email"], $send_data, "ORDER");
                header("Location: ".$this->url->addOrder($id));
                exit;
        }
        return false;
    }

}
?>
```

Фрагмент лістингу файлу global_class.php

```
<?php
require_once "database_class.php";
require_once "config_class.php";
require_once "check_class.php";
require_once "url_class.php";
require_once "systemmessage_class.php";

abstract class GlobalClass {

    protected $db;
    protected $table_name;
    protected $format;
    protected $config;
    protected $check;
```

```
    protected $url;

    public function __construct($table_name) {
        $this->db = DataBase::getDB();
        $this->format = new Format();
        $this->config = new Config();
        $this->check = new Check();
        $this->url = new URL();
        $this->table_name = $this->config->db_prefix.$table_name;
    }

    public function add($data) {
        if (!$this->check($data)) return false;
        $query = "INSERT INTO `".$this->table_name."` (";
        foreach ($data as $field => $value) $query .= "`$field`,";
        $query = substr($query, 0, -1);
        $query .= ") VALUES (";
        foreach ($data as $value) $query .= $this->config->sym_query.",";
        $query = substr($query, 0, -1);
        $query .= ")";
        return $this->db->query($query, array_values($data));
    }


    public function delete($id) {
        $query = "DELETE FROM `".$this->table_name."` WHERE `id`=".$this->config->sym_query;
        return $this->db->query($query, array($id));
    }

    private function check($data) {
        $result = $this->checkData($data);
        if ($result === true) return true;
        $sm = new SystemMessage();
        return $sm->message($result);
    }

    protected function getOnField($field, $value) {
        $query = "SELECT * FROM `".$this->table_name."` WHERE `$field` = ".$this->config->sym_query;
        return $this->db->selectRow($query, array($value));
    }

    protected function getAllOnField($field, $value, $order = false, $up = true, $count = false, $offset = false) {
        $ol = $this->getOL($order, $up, $count, $offset);
        $query = "SELECT * FROM `".$this->table_name."` WHERE `$field` = ".$this->config->sym_query." $ol";
        return $this->db->select($query, array($value));
    }

    public function get($id) {
        if (!$this->check->id($id)) return false;
        return $this->getOnField("id", $id);
    }

    protected function getField($field_in, $value_in, $field_out) {
        $query = "SELECT `$field_out` FROM `".$this->table_name."` WHERE `$field_in` = ".$this->config-
>sym_query;
        return $this->db->selectCell($query, array($value_in));
    }

    public function search($q, $fields, $order = false, $up = false) {
        if (count($fields) == 0) return false;
        $q = trim($q);
        if ($q === "") return false;
        $q = preg_replace("/\s+/", " ", $q);
```

```
            $q = mb_strtolower($q);
            $array_words = explode(" ", $q);
            $logic = " AND ";
            $params = array();
            foreach ($array_words as $key => $value) {
                        if (isset($array_words[$key - 1])) $where .= $logic;
                        for ($i = 0; $i < count($fields); $i++) {
                                    $where .= "`".$fields[$i]."` LIKE ".$this->config->sym_query;
                                    $params[] = "%$value%";
                                    if (($i + 1) != count($fields)) $where .= " OR ";
                        }
            }
      protected function getOL($order, $up, $count, $offset) {
            if ($order) {
                        $order = "ORDER BY `$order`";
                        if (!$up) $order .= " DESC";
            }
            $limit = $this->getL($count, $offset);
            return "$order $limit";
      }

      protected function setField($field_in, $value_in, $field, $value) {
            $query = "UPDATE `".$this->table_name."` SET `$field` = ".$this->config->sym_query." WHERE
`$field_in` = ".$this->config->sym_query;
            return $this->db->query($query, array($value, $value_in));
      }

      ?>
```

Фрагмент лістингу файлу database_class.php

```
<?php
require_once "config_class.php";

class DataBase {

      private static $db = null;
      private $config;
      private $mysqli;

      public static function getDB() {
            if (self::$db == null) self::$db = new DataBase();
            return self::$db;
      }

      public function select($query, $params = false) {
            $result_set = $this->mysqli->query($this->getQuery($query, $params));
            if (!$result_set) return false;
            return $this->resultSetToArray($result_set);
      }

      public function selectRow($query, $params = false) {
            $result_set = $this->mysqli->query($this->getQuery($query, $params));
            if ($result_set->num_rows != 1) return false;
            return $result_set->fetch_assoc();
      }

      private function resultSetToArray($result_set) {
            $array = array();
            while (($row = $result_set->fetch_assoc()) != false) {
                        $array[] = $row;
            }
```

```
        return $array;
    }

    public function __destruct() {
        if ($this->mysqli) $this->mysqli->close();
    }

}

?>
```

Лістинг файлу product_class.php

```
<?php
require_once "global_class.php";

class Product extends GlobalClass {
    public function __construct() {
        parent::__construct("products");
    }
    public function getAllData($count) {
        return $this->transform($this->getAll("date", false, $count));
    }
    public function getAllTable() {
        return $this->getAll("id");
    }
    public function getTableData($section_table, $count, $offset) {
        $l = $this->getL($count, $offset);
        $query = "SELECT `".$this->table_name."`.`id`,
        `".$this->table_name."`.`section_id`,
        `".$this->table_name."`.`img`,
        `".$this->table_name."`.`title`,
        `".$this->table_name."`.`price`,
        `".$this->table_name."`.`description`,
        `$section_table`.`title` as `section`
        FROM `".$this->table_name."`
        INNER JOIN `$section_table` ON `$section_table`.`id` = `".$this->table_name."`.`section_id`
        ORDER BY `date` DESC $l";
        return $this->transform($this->db->select($query));
    }
    protected function transformElement($product) {
        $product["img"] = $this->config->address.$this->config->dir_img_products.$product["img"];
        $product["link"] = $this->url->product($product["id"]);
        $product["link_cart"] = $this->url->addCart($product["id"]);
        $product["description"] = str_replace("\n", "<br />", $product["description"]);
        $product["link_delete"] = $this->url->deleteCart($product["id"]);
        $product["link_admin_edit"] = $this->url->adminEditProduct($product["id"]);
        $product["link_admin_delete"] = $this->url->adminDeleteProduct($product["id"]);
        return $product;
    }
    private function checkSortUp($sort, $up) {
        return ((($sort === "title") || ($sort === "price")) && (($up === "1") || ($up === "0")));
    }
    public function getAllOnSectionID($section_id, $sort, $up) {
        if (!$this->checkSortUp($sort, $up)) return $this->transform($this->getAllOnField("section_id", $section_id));
        return $this->transform($this->getAllOnField("section_id", $section_id, $sort, $up));
    }
    public function getAllSort($sort, $up, $count) {
        if (!$this->checkSortUp($sort, $up)) return $this->getAllData($count);
        $l = $this->getL($count, 0);
        $desc = "";
        if (!$up) $desc = "DESC";
        $query = "SELECT * FROM `".$this->table_name."` ORDER BY `$sort` $desc $l";
```

```
        return $this->transform($this->db->select($query));
    }

    public function getAllOnIDs($ids) {
        $query_ids = "";
        $params = array();
        for ($i = 0; $i < count($ids); $i++) {
                $query_ids .= $this->config->sym_query.",";
                $params[] = $ids[$i];
        }
        $query_ids = substr($query_ids, 0, -1);
        $query = "SELECT * FROM `".$this->table_name."` WHERE `id` IN ($query_ids)";
        return $this->transform($this->db->select($query, $params));
    }
    public function getPriceOnIDs($ids) {
        $products = $this->getAllOnIDs($ids);
        $result = array();
        for ($i = 0; $i < count($products); $i++) {
                $result[$products[$i]["id"]] = $products[$i]["price"];
        }
        $summa = 0;
        for ($i = 0; $i < count($ids); $i++) {
                $summa += $result[$ids[$i]];
        }
        return $summa;
    }
    public function getDate($id) {
        return $this->getFieldOnID($id, "date");
    }
    public function getImg($id) {
        return $this->getFieldOnID($id, "img");
    }

    protected function checkData($data) {
        if (!$this->check->id($data["section_id"])) return "UNKNOWN_ERROR";
        if (!$this->check->title($data["title"])) return "ERROR_TITLE";
        if (!$this->check->amount($data["price"])) return "ERROR_PRICE";
        if (!$this->check->year($data["year"])) return "ERROR_YEAR";
        if (!$this->check->text($data["description"])) return "ERROR_DESCRIPTION";
        if (!$this->check->title($data["img"])) return "ERROR_IMG";
        if (!$this->check->ts($data["date"])) return "UNKNOWN_ERROR";
        return true;
    }
}
?>
```

Лістинг файлу content_product.tpl

```
<div id=product>
    <div id=hornav>
        <p>
                <a href="<?=$this->index?>">Головна<a>
                >>
                <a href="<?=$this->link_section?>"><?=$this->product["section"]?><a>
                >>
                <?=$this->product["title"]?>
        </p>
    </div>
    <div class="cont_block">
    <div id=product_img_big>
      <div id="block_product">
       <img src="\images\products\<?=$this->product["img"]?>" alt="<?=$this->product["title"]?>">
```

```
    </div>
  </div>
  <div id=product_desc>
    <h2><?=$this->product["title"]?></h2>
    <!--<p><b>Категорія:</b> <span><?=$this->product["section"]?></span></p>
    <p><b>Рік випуску:</b> <?=$this->product["year"]?></p>-->
    <p><!--<b>Опис:</b>--> <?=$this->product["description"]?></p>

    <p id=price_big>₴ <?=$this->product["price"]?></p>
    <p > <a id=button7 href="/functions.php?func=add_cart&id=<?=$this->product["id"]?>">Додати до
кошика</a></p>
  </div>
  </div>

</div>
```

Лістинг файлу modules_class.php

```
<?php
require_once "lib/abstractmodules_class.php";

abstract class Modules extends AbstractModules {

    public function __construct() {
        parent::__construct();

        $this->setInfoCart();
        $this->template->set("content", $this->getContent());
        $this->template->set("action", $this->url->action());
        $this->template->set("title", $this->title);
        $this->template->set("meta_desc", $this->meta_desc);
        $this->template->set("index", $this->url->index());
        $this->template->set("cart_link", $this->url->cart());
        $this->template->set("about_link", $this->url->about());
        $this->template->set("link_contacts", $this->url->contacts());
        $this->template->set("link_search", $this->url->search());
        $this->template->set("items", $this->section->getAllData());
        $this->template->display("main");
    }

    private function setInfoCart() {
        if ($_SESSION["cart"]) {
                $ids = explode(",", $_SESSION["cart"]);
                $summa = $this->product->getPriceOnIDs($ids);
                $this->template->set("cart_count", count($ids));
                $this->template->set("cart_summa", $summa);
                $words = array("товар", "товари", "товарів");
                $this->template->set("cart_word", $this->getWord(count($ids), $words));
        }
        else {
                $this->template->set("cart_count", 0);
                $this->template->set("cart_summa", 0);
                $this->template->set("cart_word", "товарів");
        }
    }

    protected function setLinkSort() {
        $this->template->set("link_price_up", $this->url->sortPriceUp());
        $this->template->set("link_price_down", $this->url->sortPriceDown());
        $this->template->set("link_title_up", $this->url->sortTitleUp());
        $this->template->set("link_title_down", $this->url->sortTitleDown());
    }
```

```
    private function getWord($number, $words) {
        $keys = array(2, 0, 1, 1, 1, 2);
        $mod = $number % 100;
        $word_key = ($mod > 7 && $mod < 20)? 2: $keys[min($mod % 10, 5)];
        return $words[$word_key];
    }

    protected function getDirTmpl() {
        return $this->config->dir_tmpl;
    }
}

?>
```

Лістинг файлу main.tpl

```
<!DOCTYPE html>
<html>
    <head>
    <p id="intro"></p>
        <title><?=$this->title?></title>
        <meta charset=utf-8>
        <meta name=description content="<?=$this->meta_desc?>">
        <link rel=stylesheet href=styles/main.css>
        <link rel=stylesheet href=styles/cart.css>
        <link rel=icon href=favicon.png>
        <link rel="shortcut icon" href="favicon.png" type="image/png">
        <link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Molle:ital@1&display=swap" rel="stylesheet">
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?family=Arizonia&display=swap" rel="stylesheet">
    </head>
    <body>
        <nav>
                <div class="middle main__block block" id=menu>
                        <div id="main_menu_bl"><a id="Llogo" href=<?=$this->index?>>JackBlack</a></div>
                        <div ><a id="Llogo1" href=<?=$this->about_link?>>About us</a></div>
                        <div><a id="Llogo1" href=<?=$this->link_contacts?>>Contacts</a></div>
         <div id=cart_pict>
                                <a href=<?=$this->cart_link?>><img src=images/imgonline.jpg alt=cart
title=cart></a>
                        </div>
                        <div class="rozd__prod">
                        <div><a id="Llogo1" href=<?=$this->index?>>Partitions</a></div>
                        <ul>
                        <?php for ($i = 0; $i < count($this->items); $i++) { ?>
                                        <li>
                                                <a href="<?=$this->items[$i]["link"]?>"><?=$this-
>items[$i]["title"]?></a>
                                        </li>
                                <?php } ?>
         </ul>

         </div>

                </div>
        </nav>
        <div id=content>
                <section class=block_section>
                        <?php include "content_".$this->content.".tpl"; ?>
                </section>
```

```
            </div>
            <footer>

            </footer>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
   $(function(){
      var header = $("#menu"),
         introH = $("#intro").innerHeight(),
         scrollOfset = 0;

      $(window).on("scroll", function(){
        scrollOfset = $(this).scrollTop();
         /*console.log(introH);
         console.log(scrollOfset);*/
         if(scrollOfset > introH){
            menu.style.display="block";
            menu.style.width="auto";
            menu.style.marginTop="30px";


         }
         else{
            menu.style.display="flex";
            menu.style.width="100%";
            menu.style.marginTop="0px";

         }
      });
   });
</script>

      </body>
</html>
```

Фрагмент лістингу файлу main.css

```
      /*--- Меню ---*/

#menu {
      margin: 15px auto 0;
      min-width: 800px;
   width: 85%;
      /max-width: 1140px;
      display: flex;
      flex-flow: row || nowrap;
      justify-content: space-around;
}

#menu a {
      border-radius: 10px;
      padding: 10px;
      text-decoration: none;
      color: black;
      font-size: 200%;
}
/*
#menu a:hover {
      background-color: #008500;
      color: white;
      border-radius: 0 50%;
```

```
    padding: 5px 10px;
}
*/

        /*--- Контент ---*/

#content {
        margin: 15px auto 0;
        min-width: 1024px;
        display: flex;
        justify-content: center;
}

section {
        padding: 5px;
    width: 80%;
    max-width: 1000px;
}


.middle a {
    text-decoration: none;
    font-size: 1em;
    position: relative;
    transition: all 0.6s;

}

.middle a:before {
    background-color: #fff;
    margin-bottom: 2px;
    content: "";
    width: 0;
    height: 0.1em;
    position: absolute;
    bottom: 0;
    left: 50%;
    transition: all 0.3s;
}
#menu a:hover {
    color: #fff;
}
#menu li a:hover {
    color: #fff;
}

.middle a:hover:before {
    width: 100%;
    left: 0;

}

.rozd__prod{
    margin-top: 10px;
}

.rozd__prod ul{
    position: absolute;
    display: none;
    transition: all 0.3s;
}
.rozd__prod:hover ul{
    display: block;
```

```
    transition: all 0.3s;
}

.rozd__prod ul li{
    background-color: #222;
    padding: 0 20px;
    border-radius: 50%;
    opacity: 0.7;

    text-align: center;

    list-style-type: none;
    transition: all 0.3s;
}

.main_menu_bl{

}

.rozd__prod ul li:hover{
    color: white;
    background-color: #000;
    border-radius: 20% 50%;
    opacity: 1;
    list-style-type: none;
    transition: all 0.3s;
}


.rozd__prod ul li a:hover:before{
    width: 0;
    transition: all 0.3s;
}

#cart_pict img{
    height: 36px;
    width: 36px;
    margin: 10px;
    transition: all 0.3s;
}

#cart_pict a{
    padding: 0;
}

#cart_pict img:hover{
    margin: 5px;
    height: 46px;
    width: 46px;
    transition: all 0.3s;
}

.product{
    transition: all 0.3s;
}

.product:hover{
    box-shadow: 0 5px 10px;
    transition: all 0.3s;
}

.block_content{
    font-family: 'Verdana';
```

```
}


img{
   border-radius: 5px;
}

#product{
   font-family:none;
}

/* sort */
.three{
   display: flex;
   position: absolute;
}
.three .submenu{
   display: flex;
}
.three .submenu a {
  color: #74924C;
}
.three .submenu {
  visibility: hidden;
  opacity: 0;
   transition: all 0.5s;
}
.three:hover .submenu {
  visibility: visible;
  opacity: 1;
   transition: all 0.5s;
}

.three div {
   padding: 0px 4px;
}
```

Лістинг файлу content_order.tpl

```
        <div id="order">

           <?php include "message.tpl"; ?>
           <form name="order" action="<?=$this->action?>" method="post" class="decor">
               <div class="form-left-decoration"></div>
               <div class="form-right-decoration"></div>
               <div class="circle"></div>
               <div class="form-inner">
               <h3>ОФормлення замовлення</h3>
               <input type="text" name="name" value="<?=$this->name?>" placeholder="П.І.П."/>
               <input type="text" name="phone" value="<?=$this->phone?>" placeholder="Телефон"/>
               <input type="text" name="email" value="<?=$this->email?>" placeholder="E-mail"/>
               <select name="delivery" onchange="changeDelivery(this)">
                  <option value="">Оберіть, будь ласка, тип доставки</option>
                  <option value="0" <?php if ($this->delivery == "0") { ?>selected="selected"<?php
}?>>Доставка</option>
                  <option value="1" <?php if ($this->delivery == "1") { ?>selected="selected"<?php
}?>>Самовивіз</option>
               </select>
               <textarea name="address" cols="80" rows="4" placeholder="Адреса..."><?=$this->address?></textarea>
               <textarea name="notice" cols="80" rows="4" placeholder="Примітки до замовлення"><?=$this-
>notice?></textarea>
               <input type="submit" value="Підтвердити">
```

```
            <input type="hidden" name="func" value="order" />


          </form>
        </div>
        <style>
          * {
          box-sizing: border-box;
        }
        .decor {
          position: relative;
          max-width: 400px;
          margin: 50px auto 0;
          background: white;
          border-radius: 30px;
        }
        .form-left-decoration, .form-right-decoration {
          content: "";
          position: absolute;
          width: 50px;
          height: 20px;
          background: #000;
          border-radius: 20px;
        }
        .form-left-decoration {
          bottom: 60px;
          left: -30px;
        }
        .form-right-decoration {
          top: 60px;
          right: -30px;
        }
        .form-left-decoration:before, .form-left-decoration:after, .form-right-decoration:before, .form-right-
decoration:after {
          content: "";
          position: absolute;
          width: 50px;
          height: 20px;
          border-radius: 30px;
          background: white;
        }
        .form-left-decoration:before {
          top: -20px;
        }
        .form-left-decoration:after {
          top: 20px;
          left: 10px;
        }
        .form-right-decoration:before {
          top: -20px;
          right: 0;
        }
        .form-right-decoration:after {
          top: 20px;
          right: 10px;
        }
        .circle {
          position: absolute;
          bottom: 80px;
          left: -55px;
          width: 20px;
          height: 20px;
          border-radius: 50%;
```

```
        background: white;
      }
      .form-inner {
        padding: 50px;
      }
      .form-inner input, .form-inner textarea, .form-inner select {
        display: block;
        width: 100%;
        padding: 0 20px;
        margin-bottom: 10px;
        background: #E9EFF6;
        line-height: 40px;
        border-width: 0;
        border-radius: 20px;
        font-family: 'Roboto', sans-serif;
      }
      .form-inner input[type="submit"] {
        margin-top: 30px;
        background: #000;
        border-bottom: 4px solid #000;
        color: white;
        font-size: 14px;
      }
      .form-inner textarea {
        resize: none;
      }
      .form-inner h3 {
        margin-top: 0;
        font-family: 'Roboto', sans-serif;
        font-weight: 500;
        font-size: 24px;
        color: #707981;
      }
</style>
```

ДОДАТОК Б

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім'я файла	**Опис**
Пояснювальні документи	
Диплом_ Качан.doc	Пояснювальна записка до дипломного проекту. Документ Word.
Диплом_ Качан.pdf	Пояснювальна записка до дипломного проекту в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_ Качан.pptx	Презентація дипломного проекту