Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка веб-додатку для надання послуг з продажу електроніки»

Виконав: студент групи ІПЗ20-1
Спеціальність 121 «Інженерія програмного забезпечення»
Камишан Кирило Едуардович
(прізвище та ініціали)
Керівник к.т.н., доцент, Мала Ю.А.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
доцент кафедри інформаційних технологій
(посада)

(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Камишан К. Е. Розробка веб-додатку для надання послуг з продажу електроніки.

Кваліфікаційна робота бакалавра на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

Дипломна робота бакалавра присвячена розробці веб-додатку для надання послуг з продажу товару. У сучасному цифровому світі електронна комерція стає все більш популярною, тому важливо мати зручний та ефективний інструмент для проведення торгівлі в мережі Інтернет за допомогою мобільних пристроїв або сучасних комп'ютерів. Метою роботи було створення програмного рішення у вигляді веб-додатку для персонального комп'ютера, який допоможе бізнесу ефективно керувати взаємодією з клієнтами, надасть користувачам зручний інтерфейс для вибору замовлення та оплати товарів онлайн, що в свою чергу оптимізує процеси продажів товарів, а також підвищить ефективність взаємовідносин з клієнтами.

У процесі розробки досліджуватимуться сучасні технології веб-розробки та способи використання сучасних баз даних, та інструменти керування продажами товарів, і взаємовідносин з клієнтами. Додаток буде розроблений з використанням інструментів та технологій, таких як HTML, CSS, JavaScript - для розробки front-end та язика програмування Python з фреймворком Django - для розробки back-end.

Результатом цієї роботи буде функціональний та надійний веб-додаток, який зможе задовольнити потреби як користувачів, так і власників бізнесу в галузі електронної комерції. Додаток було протестовано з метою перевірки його ефективності та надійності перед початком використання.

Ключові слова: HTML, CSS, JavaScript, Python, Django, SQLite3.

ABSTRACT

Kamishan K. E. Development of a web application for the provision of services for the sale of electronics.

Qualification of a bachelor's degree for a bachelor's degree in specialty 121 "Software Security Engineering" – University of Mining and Finance, Dnipro, 2024.

The bachelor's thesis is dedicated to the development of a web application for the provision of services for the sale of goods. In today's digital world, e-commerce is becoming increasingly popular, so it is important to have a handy and effective tool for trading online via mobile devices or current computers. The robot was created to create a software solution in the form of a web-based add-on for a personal computer, which will help businesses effectively interact with clients and provide customers with a manual interface for selecting and paying for goods online, which in turn optimizes product sales processes and also promotes effectiveness of relationships with clients.

The development process is based on current web development technologies, methods for retrieving current databases, tools for managing product sales, and interacting with clients. The addition will include the use of various tools and technologies, such as HTML, CSS, JavaScript - for the development of the front-End and the Python programming language with the Django framework - for the development of the back-End.

The result of this work will be a functional and reliable web application that can satisfy the needs of both merchants and business owners in the e-commerce industry. The supplement was tested to verify its effectiveness and reliability cob vikoristannya.

Keywords: HTML, CSS, JavaScript, Python, Django, SQLite3.

ЗМІСТ

ВСТУП

Актуальність дослідження даної дипломної роботи полягає в тому, що сучасний світ важко уявити без цифрових технологій. Технологічний процес перебуває в постійному розвитку. Сьогодні доступ до інтернет мережі можна получити в будь-якій точці Землі.

Інтернет-магазин є основою сучасного бізнесу. Це є електроні майданчики, які дають можливість вести процес реалізації фізичних та не фізичних товарів. Вони дають можливість дистанційно оформити інтернет-замовлення. Електроні магазини не замінять повністю традиційні магазини, але вони розширюють ринок та сферу застосування та впливу, особливо в економічну кризу.

Економічна криза штовхає підприємців до розвитку онлайн-торгівлі. Бо традиційний продаж товару з прилавка несе за собою більші витрати: оренда, витрати на персонал тощо. Ще однією відмінною рисою інтернет-магазину є те, що інтернет-магазин може запропонувати значно більшу кількість товару та послуг, та забезпечити покупця значно більшим обсягом інформації, яка необхідна для прийняття рішення по покупці. Крім того, онлайн магазин надає можливість персоналізований підхід до кожного клієнта, на основі історії відвідування минулих покупок. Також є перевага інтернет-магазину в тому, що він працює цілодобово, а при повній автоматизації продає товар, навіть без участі продавця. До того ж не треба покупати товар заздалегідь та тримати його на складських приміщеннях, а достатньо домовитися с постачальником товару, та викупити його в потрібний момент, це все також економія коштів.

Наявність веб-додатка інтернет-магазину надає продавцю такі переваги, як відстежування потоку покупців, що надає можливість побачити конверсію сайту та оцінити ефективність роботи інтернет-магазину, відстеження кроків покупця до кінцевої точки – покупки, проведення реклами та акцій. Такі переваги підвищують конкурентоспроможність бізнесу.

Останнім часом інтернет-торгівля в Україні дуже поширена і продовжує розвиватися досить успішно, тому що це дійсно вигідно і зручно для покупця.

Актуальність розробки веб-додатку полягає в наступних факторах:

1) Швидкість подачі інформації широкому колу осіб;
2) Покращення іміджу магазину та підвищення популярності на ринку;
3) Організація маркетингових досліджень;
4) Реклама та залучення клієнтів.

Враховуючи, що ринок інформаційних послуг не стоїть на місці, адже людям постійно потрібна інформація, ця частина інформаційних потоків є важливою для розгляду та вирішення питань.

Метою дипломної роботи є розробка макету інформаційної системи зручного інтернет-магазина у вигляді веб-додатка з урахуванням сучасних засобів.

Відповідно до мети виділені наступні завдання дипломної роботи:

1) Аналіз сучасних рішень для поставленої задачі;
2) Розробка бази даних для програмного продукту;
3) Вибір необхідних засобів для реалізації потрібного функціоналу;
4) Розробка веб-додатку для інтернет-магазину.

Методи дослідження

1) Актуальність веб-додатків в сучасному комерційному світі;
2) Дослідження різновиду веб-додатків в залежності від їх призначення та вимог до майбутнього проекту;
3) Процес створення веб-додатків;
4) Дослідження портрету майбутнього користувача веб-додатка, його характерні особливості;
5) Зручний інтерфейс. Дослідження правил юзабіліті – інтуїтивно зрозумілого дизайну;
6) Адаптивний дизайн та його переваги;
7) SEO-аналіз, його важливість для рейтингу сайту у пошукових системах;
8) Концепція вибору клієнт-серверної архітектури;
9) Дослідження вимог до функціоналу веб-додатка інтернет-магазину;
10) Аналіз бібліотек фреймворків, їх переваги та недоліки;

11) Дослідження переваг нативного коду для унікальності проекту;

12) Дослідження мов програмування, їх переваги, та призначення;

13) Дослідження сучасних баз даних.

Об'єктом дослідження є веб-додаток, який буде розроблений з метою надання користувачам зручного та ефективного інструменту для організації процесу покупок. Що має в собі вивчення принципів проектування веб-ресурсів, фреймворків та систем управління базами даних, а також реалізацію ключових функцій веб-додатку, таких як навігація, фільтрація товарів, робота з кошиком покупок тощо.

Предметом дослідження є методи та інструменти розробки веб-додатка інтернет-магазину.

Практичне значення одержаних результатів - підвищення конкурентної спроможності бізнесу за допомогою унікальності веб-додатка інтернет-магазину.

Структура роботи:

Розділ 1. Аналіз технологій розробки веб-додатків. В даному розділі буде виконуватися постановка задач та аналіз методів їх вирішення, за допомогою публікацій стосовно теми веб-розробки.

Розділ 2. Проектування веб-додатку для надання послуг продажу електроніки. В даному розділі буде проаналізовано та визначено технології для розробки веб-додатка інтернет-магазину.

Розділ 3. Програмна реалізація веб-додатку. В даному розділі буде описано розробка інтернет-магазину: back-end, front-end та бази даних.

Робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел з 22 найменувань. Обсяг роботи 58 сторінок, 19 рисунків.

РОЗДІЛ 1. АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБКИ ВЕБ-ДОДАТКІВ

1.1 Опис предметної області

У сучасному цифровому світі веб-додатки, можна сказати, є най поширеним інструментом для бізнесу. Він дає можливість надавати клієнтам зручні та ефективні сервіси з пошуку клієнтів та товарів, отримання порад, та відгуків про товари та клієнтів. Для продажу сучасної електроніки веб-додаток – це важливий крок для компанії, яка бажає збільшити обсяги своїх продаж та покращити взаємодію з клієнтами. Цей додаток зможе не лише оптимізувати процес продажу, але й надає користувачам можливість легко користуватись інтерфейсом – швидко знаходити та купувати необхідні товари та послуги. У роботі буде детально розглянуто всі аспекти створення такого додатка, починаючи з початкового планування, розробці дизайну, наповнення бази даних інформацією розробці самого веб-додатку і до впровадження та тестування.

Головна мета роботи – створити веб-додаток з інтуїтивно зрозумілим і зручним інтерфейсом для замовників та користувачів. Забезпечити безперебійну роботу веб-додатку з можливістю обробки великої кількості запитів до бази даних, та видачу інформації о замовлення на сайті веб-додатку. Підвищення рівня задоволеності клієнтів через швидкий та безпечний процес покупки також буде одним з основних пріоритетів роботи веб-додатку. Це надає бізнесу збільшити зручність продаж та обсяги продажу.

Для того, щоб досягти поставлених цілей, перш за все, потрібно створити структуру бази даних товарів та послуг певного магазину, яка дозволяє легко шукати товари. Для цього буде зручно впровадити сучасний стандартний функціонал кошика покупок та систему обробки замовлень, а також будемо інтегрувати різні сервіси платіжних систем для зручності розрахунку користувачів з бізнесом. Також потрібно розробити систему адміністрування, для зручності продавця.

Стратегічно важливим для бізнесу є розвиток сучасної електронної комерції, вплив якої зростає з кожним днем і є не одмінним аспектом життя

суспільства. Розробка веб-додатку для надання послуг з продажу сучасної електроніки забезпечить бізнесу низку переваг, які важко перебільшити.

Тобто сучасна електронна комерція має багато переваг. І основною її перевагою є розширення географії потенційних покупців, що надає доступ до товарів для користувачів у будь-якому куточку світу. Наприклад у регіонах з обмеженим асортиментом або де відсутні магазини електроніки, веб-додаток стане провідником у світ сучасних технологій для клієнтів.

Також веб-додаток надає безліч можливостей для маркетингових ініціатив, аналізу воронки конверсії, тобто дає можливість відстеження та аналізу кроків, які роблять відвідувачі на сайті, щоб досягти певної мети. Ще веб-додаток надає можливість проводити рекламні кампанії, використовувати систему знижок, акції та бонусні програми. Ці інструменти не лише приваблюють нових клієнтів, але й стимулюють повторні покупки, та підвищують лояльність.

Простий інтуїтивний інтерфейс додатку спрощує процес купівлі за декілька кроків, що зменшує бар'єри для виконання операцій покупки товарів та підвищує конверсію.

Веб-додаток зможе забезпечувати збір та аналіз даних про взаємодію користувачів з платформою, що дозволить постійно вдосконалювати веб-додаток, його функціональність та інтерфейс. Це призведе до покращення користувацького досвіду та підвищення ефективності продажів для бізнесу.

1.2 Види веб-додатків та процес створення веб-додатків

Веб-додаток – це програмне забезпечення, яке працює на веб-сервері, а доступ до нього здійснюється через веб-сайт, до якого надається доступ користувачу через веб-браузер. Веб-додаток надає розширені функції, наприклад: автоматизація процесів, зберігання даних. Сам веб-сайт складається зі статичного контенту (зображення, відео, статті, посилання), то ж їм бракує можливостей для автоматизації процесів, зберігання даних. Різновиди веб-

додатків представлені на рис. 1.

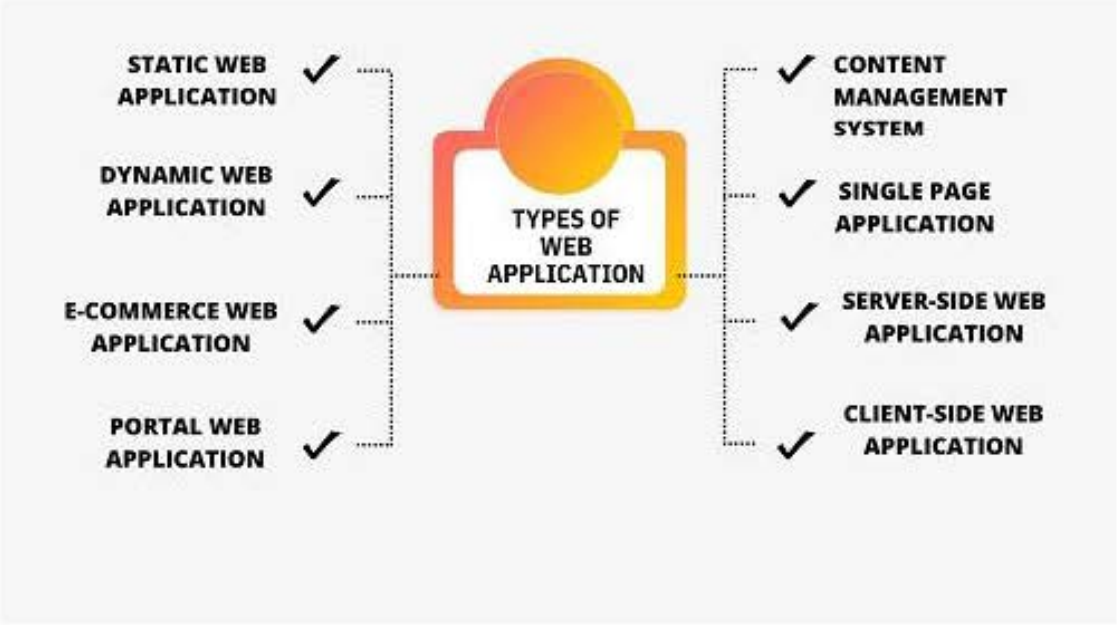

Рисунок 1 – Різновид веб-додатків [18]

Тобто веб-додаток - це програмне забезпечення або програма, яку можна відкрити за допомогою будь-якого браузера. Зовнішній інтерфейс – front-end, який бачить користувач, розробляється за допомогою таких мов програмування, як: HTML, CSS, JavaScript. Ці мови підтримуються на будь-якому браузері. А от в написанні серверної частини – back-end – можна використовувати будь-які інші мови програмування або фреймворки (Python, PhP, Ruby, Java).

Веб-додаток має певні переваги:

По-перше, веб-додаток можна застосовувати на будь-якій операційній системі. (Linux, Mac, Windows)

По-друге, тому що у веб-додатку використовується той самий код, якщо зрівнювати с desktop додатками, то веб-додаток набагато легше підтримувати.

По-третє, веб-додаток простіше програмувати, тому що він не містить багато роботи з елементами ПК (ядро, процесор, відеокарта)

Четверте, на відміну від мобільних застосунків, для веб-додатків не потрібно узгодження жодних платформ, щоб випустити свою програму.

І п’яте, веб-додаток є більш економічний варіант для будь-якого

підприємства, тому що веб-програми не потребують підписки або покупки ліцензій, і можуть використовуватися, як SaaS-сервіс, що є дешевше.

Створення веб-додатків може здійснюватися різними способами. Зважаючи на потреби та вимоги проекту, його кошторис і спроможність замовника. Вибір підходящого методу може значно вплинути на його результат і його можливості. Ось декілька можливих шляхів розробки веб-додатків:

Традиційний підхід з використанням серверного програмування: У цьому випадку, команда розробників використовує мови програмування, такі як Python або Node.js для створення серверної логіки. Вони також можуть використовувати фреймворки, такі як Django або Express.js, щоб прискорити процес розробки.

SPA (Односторінкова програма): Цей підхід дозволяє створювати додатки, які використовуються на одній сторінці. В односторінковому веб-додатку користувач, перемикаючись між вкладками залишається на одній сторінці. При цьому завантажуються та оновлюються лише необхідні частини контенту. Пріоритет SPA – це швидкість розробки, та реактивна зміна контенту.

JavaScript фреймворки та бібліотеки, такі як React, Angular або Vue.js, дозволяють створювати динамічний контент, який оновлюється без перезавантаження сторінки. Невеликий односторінковий додаток можна зробити за допомогою бібліотеки jQuery. Однак цей варіант не підходить для завеликих проектів. У JavaScript фреймворках "реактивність" стосується концепції автоматичного оновлення користувацького інтерфейсу (UI) у відповідь на зміни в стані даних. Це дозволяє розробникам створювати інтерактивні та динамічні веб-додатки, де зміни даних автоматично відображаються в інтерфейсі користувача без необхідності ручного оновлення DOM. Нижче наведено короткий огляд реактивності у кількох популярних JavaScript фреймворках:

Статичні сайти: Для простих веб-додатків, які не потребують складної серверної логіки, можна використовувати генератори статичних сайтів, такі як Jekyll, Hugo або Gatsby. Вони дозволяють створювати веб-сторінки з використанням HTML, CSS та JavaScript, а потім публікувати їх на сервері.

Serverless архітектура: Цей підхід дозволяє розробникам створювати та

запускати додатки без управління інфраструктурою серверів. Вони можуть використовувати платформи, такі як «AWS» «Lambda», «Google» «Cloud Functions» або «Azure Functions», для виконання коду.

Ці способи розробки веб-додатків мають свої переваги та недоліки, і вибір залежить від конкретних потреб та обставин проекту.

Зокрема технічної класифікації веб-додатків важливість також має і їх призначення. Є декілька найпопулярніших видів інформаційних систем для бізнесу:

E-commerce системи. Ці системи дають можливість клієнтам замовляти і продавати товари без сторонніх осіб. Тільки дві людини в ланцюжку продажу. Наприклад: інтернет-магазини, онлайн-каталоги.

CRM-системи. Дані системи розробляються для автоматизації відділу продажу та всіх заявок, котрі надходять до магазину. Внаслідок цієї системи компанія може бачити, відслідковувати всю конверсійну воронку продажів. Також компанія може бачити всю історію взаємодії з усіма клієнтами.

ERP-системи. Такі системи включають в себе не тільки автоматизацію відділу продажу магазину, а також і автоматизацію інших підрозділів компанії.

Корпоративні портали. Це веб-програми для соціального модуля корпорації. Завдяки їй можна робити швидке інформування всіх співробітників компанії, проводити корпоративне навчання, та контролювати співробітників [2].

Розробку веб-додатку можна поділити на 6 етапів:

Постанова мети. Треба зрозуміти власника бізнесу, його сферу діяльності, та які цілі він переслідує при використанні майбутнього веб-додатку. Повне розуміння мети та остаточного результату допоможе збудувати структуру проекту та осмислити, які певні етапи створення сайту необхідні для досягнення мети. Це все обговорюється на початковому етапі у повній взаємодії з замовником. Замовник дуже часто не може висловити загальну ідею. Тож треба розробляти концепцію за допомогою переговорів і сформулювати основні цілі. Тільки коли є досягнення в порозумінні та визначенні основних пріоритетів,

можна переходити до складання технічного завдання.

Складання технічного завдання. На цьому етапі заповнюється технічна документація, технології, які будуть використовуватись в додатку, а також стек всіх технічних елементів, які треба розробити. Технічне завдання – є офіційний документ та основа для подальшої роботи. В ньому прописуються структура та мапа сайту (кількість сторінок, кількість розділів, блоків, категорій тощо.) Прописують вимоги до дизайну, функціоналу, контенту та інші технічні можливості у продовж роботи над додатком. Технічне завдання – це інструкція, яка буде постійно використовуватись. Тільки після узгодження всіх питань технічного завдання можна переходити до основних видів робіт.

Прототипування. Створюють черговий макет веб-додатка, який перевтілить ідеї у реальний об'єкт, де можна буде побачити взаємодію системи з користувачем. Цей етап допоможе більш нагляданіше і детальніше обговорити технічні нюанси майбутнього веб-застосунку з клієнтом. Після обговорення прототипу з замовником - прототип узгоджують. При необхідності вносяться зміни, поки проект не буде ухвалений остаточно.

Створення дизайну. Ескізний дизайн застосовують вже на етапі прототипування для більш наглядного макета веб-додатка. Дизайн створюється на основі побажань клієнта, тенденцій сучасного UX/UA дизайну, та призначення веб-додатку. Це все призведе до того, що додатком буде зручно користуватися, до того ж додаток буде більш впізнаваним. Найбільшу увагу приділяють правильному розташуванню елементів за правилами юзабіліті та інших технічних особливостей.

Програмування. Програмісти створюють всі веб-сторінки сайту, а також логіку, яка буде виконуватись у веб-додатку (фільтрація товару, відображення цін, місцеперебування товару тощо). Після цього йде процес з'єднання front-end з back-end. За більшістю випадків програмування здійснюється на основі CMS, такі як WordPress, але іноді веб-додаток потребує написання коду з нуля, щоб розробити унікальний функціонал.

Тестування. Після написання коду важливо зробити перевірку, щоб всі

елементі працювали стабільно і вірно. Важливо зробити обидва види тестування (Manual і Unit), тому що найменша помилка може коштувати компанії великих коштів. Після тестування сайт переносяться на хостинг та вибирають доменне ім'я. Після розміщення сайту в інтернеті – проводять фінальне тестування.

Здача готового проекту. На даному етапі розробляється документація до роботи з сайтом. Проводяться навчальні роботи з персоналом. Це надасть можливість клієнту самостійно оновлювати інформацію на сайті, збирати аналітику та вносити зміни. До того ж клієнт може продовжити роботу з розробником, тому що ресурс може потребувати подальшого розвитку, підтримки та просування. Наприклад інтернет-магазин повинен постійно оновлюватися та утримувати свої позиції у пошукових системах.

1.3 Принципи проектування веб-ресурсів

Проектування веб-ресурсів - це складний та захоплюючий процес, що вимагає ретельного розуміння потреб користувачів, особливостей платформи та ефективних стратегій залучення аудиторії. Цей процес поєднує в собі технічні знання, творчий підхід та увагу до деталей з метою створення веб-ресурсу, який буде ефективним та привабливим для користувачів.

Три основні принципи проектування веб-ресурсів:

1.3.1 Фокус на користувачі

З питань розуміння потреб користувача проводяться дослідження за певною аудиторією. Спілкування та аналіз поведінки цієї аудиторії. Це все робиться для визначення, що саме потребує певний користувач від веб-додатку. Тобто, якщо створюється веб-додаток по продажу електроніки, то треба володіти інформацією про найпопулярніші бренди продукції. Аспекти використання одного чи іншого бренду. Наприклад бренд «Huawei» має власну операційну систему з власними додатками, але деякі додатки «Google» в приладах «Huawei»

– не працюють.

Також визначається основний вік користувача. Тобто ваша продукція буде ціле направлена більш на молодіжну аудиторію, чи людей похилого віку. Це теж може дати поштовх в ту чи іншу сторону дизайну та інтерфейсу веб-додатка.

Також можна визначити потреби користувача способом персоналізації контенту. Тобто завдяки аналізу історії покупок користувача на основі рекомендаційного алгоритму, або завдяки сторінці з персоналізованим змістом, яка відображається після входу на сайт. Головне визначити характерні особливості користувача і використати їх, щоб розділити аудиторію на сегменти. Саме уявлення, кому надається товар чи послуги допоможе групувати користувачів за різними категоріями та створювати пропозиції, які найкраще підходять під одну чи іншу групу користувачів.

На основі цих аспектів потрібно скласти портрет основного покупця. Це допоможе зрозуміти хто головні користувачі веб-додатка і що вони шукають. Але портрет користувача має бути в постійному перегляді і коригуванні. Він може змінюватись за будь-яких змін: це може бути – нова колекція, зміна цінової політики та інше.

Є певні ознаки якості з погляду користувача: зовнішній вигляд, простота навігації на сайті, зручність системної конструкції сайту, зручність інтерфейсу, своєчасне оновлення інформації, дотримання авторських прав на використання інформації.

1.3.2 Забезпечення зручності використання

Відвідувачі сайту у більшості випадків покидають його, якщо сайт має не зрозумілий дизайн. Тобто інтуїтивно зрозумілий дизайн – це той продукт, яким легко користуватися. Також є фактом, того що на сайті буде зроблений дійсно інтуїтивно зрозумілий дизайн, то користувач просто не зверне на це увагу. Але з часом, коли йому знов буде необхідний товар або послуга, він знову відвідає той сайт на якому він легко і зручно замовив товар.

Цифровий світ існує дуже давно, тому в дизайні з'явились певні правила:

1) зворотний зв'язок з користувачем, тобто сповіщення користувачів про те, що на сторінці відбулися зміни після певних дій;
2) на сайті треба використовувати слова, поняття, які знайомі користувачу;
3) потрібно вказувати користувачеві на помилки і надати швидку можливість це виправити або вийти з процесу;
4) використовувати досвід користувача, який він отримав на інших цифрових продуктах. Звичайні речі повинні залишатися такими, як і в інших цифрових приладах;
5) повідомляти користувача про помилки;
6) зробити елементи додатку та його функціонал видимим та зрозумілим, а при потребі легко вилученим;
7) зрозумілий дизайн може задовільнити будь-якого користувача;
8) контент та візуальний дизайн, орієнтовані на головне, а візуальні елементи інтерфейсу підтримують основні цілі користувача;
9) повідомлення про помилку виражається простою мовою;
10) якщо потрібна документація, то потрібно допомогти користувачеві зрозуміти, як виконати поставлені задачі. Документація повинна бути простою у пошуку, короткою і орієнтована на задачу користувача [7].

Всі ці аспекти дизайну в своїх дослідженнях сконсолідував Якоб Нільсен. Він вивів 10 основних правил юзабіліті. Вони не є остаточним і кінцевим висновком, але є основою в сучасному дизайні веб-додатків.

Кожний пристрій має свій розмір. Щоб зручно було користуватися веб-додатком на різних пристроях – використовують адаптивний дизайн. Адаптивний дизайн – це дизайн веб-сторінок, що надає можливість оптимального відображення та взаємодію сайту з користувачем незалежно від формату пристрою. Це дозволяє користувачеві отримати інформацію незалежно від того, яким пристроєм він користується.

Переваги адаптивного дизайну це:

1) Суворий контроль, як саме сайт працює на певному пристрої;

2) Веб-додаток швидше завантажується, тому що використовуються лише необхідні для певного пристрою елементи сторінки.

Також треба зауважити інші плюси адаптивного дизайну:

1) пошукова система «Google» віддає перевагу мобільним версіям сторінок. Це впливає на позицію сайту в пошукових результатах. Тож сайт повинен правильно відображатися та бути зручним у мобільній версії – це і надає адаптивний дизайн;
2) на разі зростає популярність мобільних пристроїв, завдяки яким виконується підвищення конверсії, а також прибутку, що підвищує попит до адаптивного дизайну;
3) користувач більш лояльний до веб-додатків, які працюють правильно на зручних для них пристроях.

Компанія «Google» проводила дослідження, які показують, що користувачі надають перевагу адаптованим сайтам:

1) 48% дратують сайти, які погано відображаються на мобільних пристроях;
2) 50% скоріше не звернуться до компанії, яка їм подобається, але яка не має адаптивного сайту;
3) 36% вважають, що неадаптивні сайти витрачають їх час;
4) 48% відповіли, що компанії, які мають неадаптовані під мобільні пристрої сайти, байдужі до власного бізнесу.

Створення будь-якого веб-сайту починається зі створення інформаційної моделі сайту. Варто розуміти та детально проаналізувати яка інформація буде на сайті, та її кількість. Процес створення проекту сайту, треба добре продумати особливі його моменти такі, як: загальну структуру, зміст інформації та посилання. Від загальної структури залежить навігація сайту, яка повинна бути простою в розумінні для користувача.

При розробленні структури сайту визначаються з кількістю сторінок та встановлюють зв'язки між ними. Розрізняють лінійну, ієрархічну та довільну структури сайтів рис. 2.

Лінійну структуру веб-додатка використовують в послідовному додаванні інформації, наприклад у навчальному матеріалі. Перегляд здійснюється послідовно та кожна сторінка має посилання тільки на одну – наступну сторінку. Також додається посилання на попередню сторінку, для зручності навігації.

Ієрархічна структура складається з однієї сторінки (головної), яка не має попередніх, та решта сторінок мають лише одну попередню сторінку. При такій структурі кожна сторінка може мати посилання на довільну кількість сторінок сайту. Ієрархічна структура більше підходить для сайтів, які містять різну за тематикою інформацію, наприклад: каталоги, збірка статей на різні теми або добірки.

Та найчастіше використовують довільну структуру сайту. При цій структурі сторінки сайту пов'язані між собою довільним чином. В сайтах довільної структури можуть бути присутні фрагменти лінійної та ієрархічної структури. Прикладом довільної структури з вікіпедії [**Ошибка! Источник ссылки не найден.**, с.145-146].

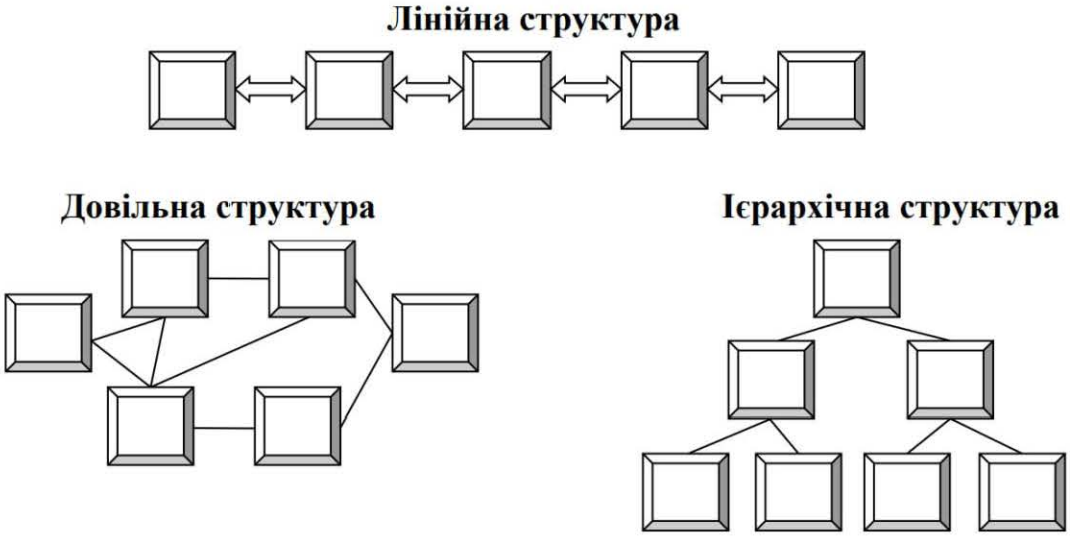

Рисунок 2 – Структури веб-додатків [5, с.146]

Структура веб-додатка пов'язана з навігацією майбутнього сайту. Навігація є частина архітектури сайту і розробляється ще на етапі розробки. Це є дорожня карта магазина, яку покупець сприймає інтуїтивно. Будь-які труднощі які користувач зустріне під час роботи с веб-додатком, зіб'ють користувача зі

шляху до мети і приведе його до конкурентів. Саме тому навігація повинна бути зрозумілою, простою та інтуїтивно підштовхувати користувача в потрібному напрямку. Тобто логічна організація структури веб-сайту та відповідна до структури навігація допомагає користувачеві швидко находити інформацію. Наприклад: меню буде зрозумілим, якщо воно буде містити категорії, які точно відображають структуру веб-додатка.

1.3.3 Забезпечення ефективності ресурсу

Якщо веб-додаток довго завантажується, то більшість користувачів покинуть його. Якщо обслуговування буде швидким – користувач буде повертатися. Буває, що сервер часто недоступний, та час відгуку від сервера дуже тривалий, у такому випадку краще знайти інший сервер, який належно забезпечить швидку роботу сайту. Швидкість завантаження сторінок допомагає зберегти увагу користувача і покращити його загальний досвід від використання веб-додатка.

Витік інформації впливає на процеси бізнесу і стає міцною зброєю руках конкурентів. Безпека – це оцінка ризиків. Конфіденційність і безпека є важливими аспектами для будь-якого сайту будь-якого розміру. За даними Ponemon Institute, 60% компаній, що зіткнулися з будь-яким інцидентом порушення безпеки, припиняють свою діяльність за 6 місяців. Це трапляється, тому що при зломі раптово втрачаються кошти, трафік та клієнти. Пошукові системи теж можуть бачити, що на сайті є загроза і вводити свої санкції, котрі будуть сповіщати клієнтів, що на сайті є загроза. Такі повідомлення знищують трафік веб-додатку. Тому потрібно застосовувати заходами безпеки, таких як захищений протокол передачі даних (HTTPS), регулярне оновлення програмного забезпечення та застосування кращих практик зберігання та обробки конфіденційної інформації клієнтів, що забезпечує їхню безпеку під час відвідування веб-додатку.

Ефективність роботи сайту може падати, якщо сайт працює не належним

чином. Тому в підтримці сайту залучають SEO – спеціалістів, які проводять аналіз роботи сайту. Аналіз повинен бути регулярним, інакше не можливо оцінити, чи повністю реалізується потенціал сайту.

Аналіз сайту складається з декілька складових: SEO-аналіз, аналіз відвідуваності веб-додатка, юзабіліті аналіз, технічний та інше.

SEO-аналіз – допоможе визначити, як пошукові системи «бачать» веб-додаток. Також SEO-аналіз виявить сильні та слабкі місця веб-додатку у пошуковому просуванні. Тобто за допомогою SEO-аналізу можна дізнатися за якими словами веб-додаток перебуває в верхніх рядках пошукової системи, на скільки є релевантними ті чи інші сторінки веб-додатку по пошуковим запитам. Також визначається взаємозв'язок між внутрішніми сторінками веб-додатку.

Тож робота SEO полягає в комплексі робіт, які повинні підвищити рейтинг сайту в пошукових системах. Метою SEO – є досягнення першої сторінки результатів пошуку. Це здійснюється за використання SEO-стратегій [**Ошибка! Источник ссылки не найден.**, с.234].

1.4 Клієнт-серверна архітектура

Практично всі веб-додатки та інтернет-сервіси побудовані на основі Клієнт-серверної архітектури (рис. 3).

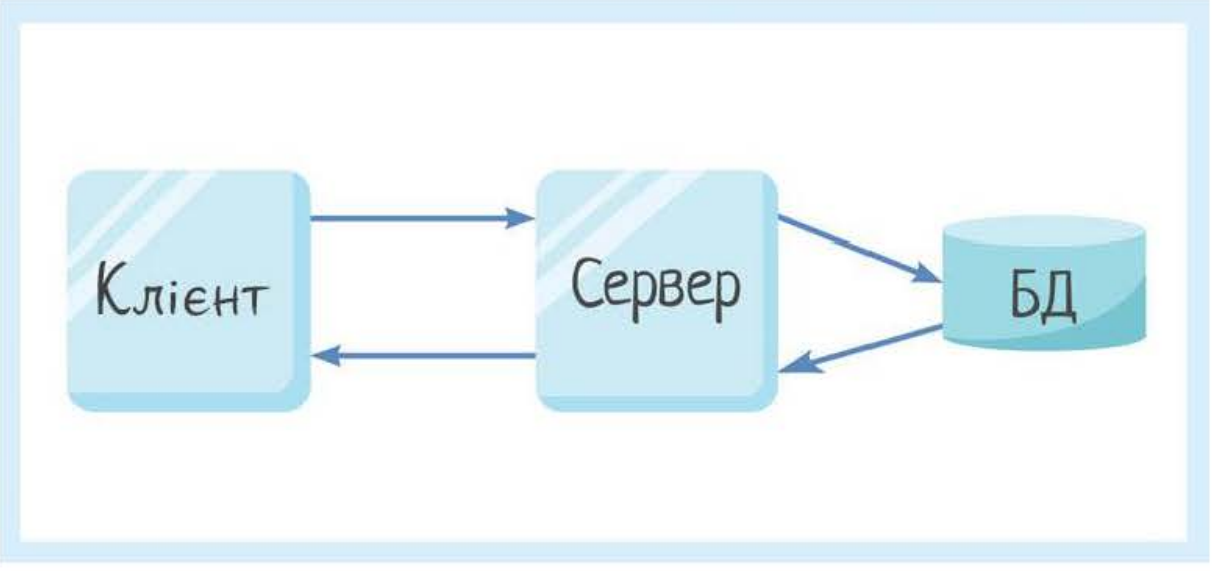

Рисунок 3 – Клієнт-серверна архітектура [11]

Клієнт-серверна архітектура є однією з найпоширеніших моделей в розробці програмного забезпечення та комп'ютерних мережах. В її основі лежать дві компоненти: клієнта та сервера.

Клієнт – це пристрій на стороні користувача, який робить запит до сервера за для отримання певної інформації або виконання певних дій.

Сервер – це більш сильніший цифровий прилад, він призначений для вирішення певних цілей з виконання програмних кодів, виконання серверних функцій за запитом користувача, надання користувачеві доступу до певних ресурсів, зберігання інформації та бази даних. Сервер чекає на запити від клієнтів, обробляє їх і надсилає відповіді назад. Сервер може обробляти кілька клієнтів одночасно. Якщо одночасно приходить декілька запитів, то вони стають в чергу і виконуються сервером послідовно. Можуть бути і пріоритетні запити, які виконуються раніше.

Клієнти та сервери взаємодіють через мережу, яка може бути локальною або глобальною.

Функції сервера:

1) зберігання, доступ, захист і резервне копіювання даних;
2) обробка клієнтського запиту;
3) відправлення відповіді.

Функції, які реалізуються на стороні клієнта:

1) надання інтерфейсу;
2) формування запиту до сервера та його відправка;
3) отримання результатів запита та відправка додаткових команд за потребою (видалення даних або оновлення тощо) [12].

Взаємодія сервера та клієнта може відбуватися через різні протоколи, такі як HTTP для веб-серверів, TCP/IP для загального мережевого з'єднання тощо. Тобто правила та взаємодії спілкування між сервером та клієнтом визначені в протоколі.

Мережевий протокол – це набір правил, за якими відбувається взаємодія між комп'ютерами в мережі.

Концепції побудови системи клієнт – сервер:

1) Слабкий клієнт – потужний сервер. Така модель означає, що обробка інформації повністю перенесена на сервер, а у клієнта права доступу суворо обмежені;
2) Сильний клієнт – це модель, в якій частина обробки інформації надається клієнтові. Тобто сервер виступає сховищем даних, а обробка та подання інформації переноситься на комп'ютер клієнта.
3) Існують дворівнева і трирівнева клієнт-серверна архітектура.
4) Дворівнева архітектура відбувається на основі двох вузлів:
5) Сервер, який відповідає за отримання запитів та відправку відповідей і використовує при цьому лише власні ресурси;
6) Клієнт, який в цій моделі є користувацький інтерфейс.

Та принцип роботи такої моделі є в тому, що сервер отримує запит, обробляє та відповідає, без використання сторонніх ресурсів.

Трирівнева архітектура складається з:

1) Представлення даних – призначений інтерфейс для користувача;
2) Прикладний компонент – сервер додатків;
3) Керування ресурсами – сервер бази даних, який надає інформацію.

Тобто робота полягає в тому, що декілька серверів оброблять запит. Такий розподіл операцій знижує навантаження на сервер. Таку архітектуру можна розширити до багаторівневої завдяки додаткових серверів, що в свою чергу підвищить ефективність роботи інформаційної системи.

Клієнт-серверна архітектура є основою для багатьох розподілених систем, від веб-додатків до корпоративних мережевих рішень. Її простота та ефективність роблять її важливим компонентом сучасних інформаційних технологій.

1.5 Висновки до першого розділу

У рамках першого розділу були розглянуті та проаналізовані головні можливості, які потрібні для розробки веб-додатка, та варіанти цих можливостей для тих чи інших рішень.

Було зауважено, що головною метою дипломної роботи є - розробка веб-додатка інтернет-магазину з подальшим впровадженням його у бізнес. Для цього були проаналізовані переваги веб-додатка та розглянуті види веб-додатків.

Розгорнуто описані 6 етапів розробки веб-додатку: ТЗ, прототип, дизайн, програмування, тестування, задача проекту.

Також розглянуті принципи проектування веб-ресурсів, такі як:

1) фокус на користувачі – де було обговорено всі нюанси, які торкаються майбутнього користувача сайту;
2) забезпечення зручності використання – було зачеплено дизайн, та описані дизайнерські сучасні принципи. Проаналізована адаптивний дизайн та його переваги. Описані можливі інформаційні моделі, можливі структури сайту та його навігація, які залежать від поставленої задачі;
3) забезпечення ефективності ресурсу – розглянуто SEO- аналіз, та його переваги для бізнесу.

Проаналізовані варіанти де клієнт має сильний або слабкий прилад. Описані можливі функції для інтернет-магазину.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ НАДАННЯ ПОСЛУГ ПРОДАЖУ ЕЛЕКТОРНІКИ

Аналіз вимог до системи для веб-додатку з продажу електроніки розпочинається зі збору та уважного вивчення потреб користувачів. Кожен етап цього процесу охоплює інтенсивне дослідження та аналіз, спрямований на забезпечення того, щоб додаток відповідав вимогам і очікуванням клієнтів.

Після вивчення потреб клієнтів та визначення ключових функцій, формуються технічні вимоги. Це включає вибір мов програмування, технологій баз даних та інфраструктури хостингу, які забезпечать ефективну роботу системи.

2.1 Визначення цільової аудиторії

Цільова аудиторія для веб-додатку з продажу електроніки включає в себе різноманітні групи користувачів, кожна з яких має власні потреби та очікування:

Технологічно освічений споживач - це користувач, який активно цікавиться новинками в сучасному світі електроніки, він завжди оновлює свої гаджети та шукає нові рішення для полегшення свого життя. Такий споживач буде зацікавлений в придбанні нових гаджетів.

Споживача можна поділити на декілька груп:

1) Геймери - вони шукають потужні комп'ютери, консолі, більш розвинені аксесуари для геймерських потреб. Для них важлива графіка, швидкість обробки інформації та інше.
2) Бізнесмени це – фрілансери, підприємці, студенти, які потребують спеціалізованої електроніки типу: сервера, камери, ксерокси та інше.
3) Бюджетники – вони потребують електроніку по більш низьким цінам, відносно свого бюджета. Вони шукають бренди з доступними цінами або пристрої попереднього покоління.
4) Технічно необізнаний користувач – основа цієї групи це люди похилого віку. Вони потребують додаткової підтримки та консультацій. Їм

потрібен простий і зрозумілий інтерфейс, який допоможе їм зробити свій вибір.

Кожна окрема група користувачів є унікальною і потребує певні візуальні рішення та різноманітні функціональні можливості. Тобто розробка веб-додатка повинна бути спрямована на забезпечення різноманітних можливостей.

2.2 Визначення ключових функцій веб-додатка

Метою майбутнього проекту є розробка інтернет-магазину електроніки, який складе в собі конкурентні властивості та спроможність бути легко функціональним для користувача, що повинно розширити воронку відвідуваності сайту.

Вимоги до функціоналу мають в собі:

1) унікальність проекту;
2) перегляд каталогу товарів з можливістю фільтрації та пошуку;
3) додавання товарів до кошика, зміна кількості товарів;
4) реклама та акції на головні сторінці;
5) оформлення замовлення з вказанням доставки та способу оплати;
6) можливість додавання, редагування та видалення товарів адміністратором.

2.3 Аналіз та вибір системи інформації

Будь-який цифровий продукт можна написати з допомогою чистого (нативного) коду або за допомогою фреймворків. Наприклад: WordPress.

Фреймворк – це конструктор за допомогою якого можна створювати сайти з нуля і керувати ними без технічних навиків. Тобто це онлайн-сервіси, які дають необізнаним користувачам створити сайт. Дизайн, заповнення та редагування контенту відбувається в браузері комп'ютера за допомогою галереї і дизайнерських інструментів. Зображення та інші модулі можна пересувати по

сайту.

Більшість сучасних фреймворків є платними. Інші фреймворки є безкоштовними. Деякі – можуть надавати певний функціонал безкоштовно, а низко додаткових послуг і можливості є платні. Є такі фреймворки, котрі надають безкоштовні послуги, лише на певний термін, а надалі вимагають оплату.

Переваги фреймворків:

1) Ідеальна платформа для початківців веб-майстрів;
2) Значне спрощення розробки та її прискорення;
3) Зменшення кількості помилок під час розробки;
4) Швидкість виконання роботи.

Але фреймворки мають багато недоліків:

1) Складана структура бази даних;
2) Великий обсяг пам'яті, потрібний для роботи даної CMS;
3) Обмежена базова функціональність – надає тільки стандартний набір опцій. Для більших можливостей потрібно закачувати плагіни;
4) І головний недолік – це повна втрата унікальності.

Тобто фреймворки ставлять рамки, позбавляючи свободи. І створення унікального продукту близиться до нуля. Кожний фреймворк має свою вбудовану логіку, якщо потрібно буде перейти з одного язика на інший або з одного фреймворка на інший - потрібен буде час для того, щоб почати розуміти новий фреймворк. Якщо потрібно буде переносити продукт, прийдеться заново його створювати. Ці всі мінуси нівелює нативний код.

Нативний код надає поекту унікальність, що в свою чергу, надасть бізнесу свої плюси, наприклад впізнаваність. Швидкість виконання роботи обумовлено наявністю бібліотек, які на відміну від обмежень фреймворків - не нав'язують певну архітектуру. Можна костомізувати застосунок без обмежень структури фреймворка і використати любий потрібний компонент бібліотеки. І ще однією важливою перевагою нативного коду є – розвантаження пам'яті, це важливо якщо проект великий і потребує постійного росту та підтримки в подальшому

часі.

Майбутній проект повинен бути великим за обсягом інформації, унікальним і багатофункціональним за вимогою клієнта, з можливістю подальшого розвитку і підтримки. Ознайомившись із запитами та вимогами майбутнього проекту, беремо за основу для розробки клієнтської частини веб-додатку - нативний код.

2.4 Вибір клієнт-серверної архітектури веб-додатку

Основні концепції побудови системи клієнт серверу, це:

1) Слабкий комп'ютер клієнта – потужний сервер. Обробка всієї інформації робиться на сервері, клієнт має тільки права доступу, які йому надає система – веб-додаток;
2) Сильний комп'ютер клієнта – така концепція надає можливість обробляти частину інформації безпосередньо у клієнта. Віддалений сервер в такому випадку виступає сховищем даних.

В системну взаємодію входять такі компоненти, як: представлення даних, прикладний компонент, а також управління ресурсами та їх зберігання.

Дворівнева архітектура складається з двох вузлів: сервер, клієнт. Сервер отримує запит, робить його обробку та надає відповідь на цей запит.

Трирівнева складається з: представлення даних, прикладний компонент, керування ресурсами. Роботу виконують декілька серверів, які обробляють запит клієнта, такий розподіл операцій знижує навантаження на сервер. Трирівневу архітектуру можна розширити до багаторівневої архітектури, за допомогою додаткових серверів.

Найчастіше у клієнтів комп'ютер може бути малої або середньої потужності. Тому для охоплення більшої кількості користувачів та для більш оптимізованої і швидкої обробки даних вибрана клієнт-серверна архітектура де сервер є головним, який відповідає за дані, а клієнтська частина відповідає лише за отримання даних, відображення даних та зміну й очищення даних.

2.5 Вибір мови програмування

Вибір мови програмування цілеспрямовано пов’язана з поставленими задачами. Кожна мова має свою унікальність та призначення для розв'язання певних задач, а також свої обмеження. Дуже часто використовується комбінація мов, щоб створити унікальний функціонал.

C – мова загального призначення. Головна ціль – можливість прямолінійної реалізації компіляції. Забезпечує низькорівневий доступ до оперативної пам’яті і не вимагає великої динамічної підтримки. Підходить для більшості системного програмного забезпечення. Мову C використовують як проміжну мову деякими високорівневими мовами програмування. Використання мови C вимагає від програміста більше навичок, досвіду та зусиль, ніж при інших мовах.

C++ – мова розроблена на основі мови C, унаслідок додавання до неї об’єктно-орієнтованої функціональності. Вона перейняла синтаксис мови C. Використовується в розробці комп’ютерних ігор, крім цього на ній можна створювати операційні системи та різні прикладні програми. Завдяки широкому набору інструментів ця мова легко адаптується для застосування в різноманітних сферах життя, будь то банківська сфера, чи комп’ютерна гра.

C# – мова C# розроблена на основі мови C. Вона також універсальна. За допомогою неї розробляються ігри та різні ПЗ для бізнесу. Наприклад з мовою C# працює «Microsoft».

Python – високорівнева мова програмування. За допомогою Python можна написати навіть інші мови. Головна перевага - наявність нейронних систем і відповідних бібліотек.

«Python також має простий та інтуїтивний зрозумілий синтаксис, це означає, що розробники можуть писати код скоріше і з меншою кількістю помилок.

Ще одна причина популярності мови – її універсальність. Python можна

використовувати для широкого спектру додатків, включаючи веб-розробку, наукові розрахунки, штучний інтелект та аналіз даних.

Python – інтерпретована мова, а це означає, що його інтерпретатор виконує код напряму, без необхідності компіляції. Це робить код Python легким для тестування і налагодження, а також дозволяє розробникам швидко повторювати коді експериментувати з ним.» [20, с.21]

Java – нативна мова Android. Мова затребувана для мобільних гаджетів. Простота коду, мультиплатформений, принципи об'єктно-орієнтованого програмування роблять цю мову дуже затребуваною. 90% десктопних бекендов розробляються з її допомогою.

«JavaScript – це язик програмування для інтернету. Він використовується практично на всіх сучасних веб-сторінках. Во всіх сучасних браузерах і клієнтських пристроях – настільних комп'ютерах, ігрових консолях, планшетах та смартфонах – є наявність інтерпетаторів JavaScript, що робить його найпоширенішою мовою в історії програмування. Він входить до базової тріади технологій, які необхідно знати всім розробникам веб-додатків: HTML (визначення зміста веб-сторінок), CSS (визначення зовнішнього вигляду веб-сторінок) та JavaScript (визначення поведінки веб-сторінок). І, нарешті останнім часом з появою технології Node.js мова JavaScript стала важливим засобом створення сучасних веб-серверів.» [**Ошибка! Источник ссылки не найден.**, с.14]

Мова JavaScript дозволяє не тільки створювати сайти, а також писати розширення для веб-платформ, також активно застосовується для розробки додатків. Має досить простий синтаксис та велику кількість готових бібліотек: React, Angular або Vue.js, які дозволяють створювати динамічний контент, який оновлюється без перезавантаження сторінки.

GO – розроблена корпорацією «Google». Мова GO є компільована та багатопотокова. Особливість цієї мови в тому, що вона заточена під багатоядерні процесори і надає можливість писати код в режимі мультизадачності. Це надає суттєву економію пам'яті, що позначається на швидкості, як дії так і відгуку. Це

мова створення високоефективних програм, але в основному вона підходить для back-end.

Для виконання розробки веб-додатка в якості front-end була вибрана мова JavaScript, як багато функціональна, що надасть сайту унікальності, для підтримки зацікавленості користувача.

Сторона back-end – вибрана мова Python за її гнучкість, можливість розширення. Існування бібліотек і фреймворків під будь-який тип завдань. Також за інтерпретованість для всіх популярних платформ та за єдиний стандарт для написання коду, що робить код підтримуваним та читабельним навіть при переході до іншого програміста.

Принцип виконання коду у мовах програмування JavaScript та у Python схожий, тобто вони у своєму звичайному стані використовують один потік даних, що дає можливість писати в імперативному стилі (Імперативне програмування — це парадигма програмування, котрій характерні такі риси: У вихідному коді програми записуються інструкції (команди). Інструкції мають виконуватися послідовно. Дані, отримані під час попередніх інструкцій, можуть зчитатися з пам'яті наступними інструкціями).

2.6 Вибір системи управління базами даних

Буває три основні технології даних такі як: централізована децентралізована та змішана.

Централізована технологія передбачає централізований підхід до збору, обробки та використання даних через інформаційно-обчислювальний центр.

Децентралізована технологія заснована на використанні персональних комп'ютерів та локальних мереж, що дозволяє працівникам вводити та обробляти дані.

Змішана технологія поєднує централізовану та децентралізовану технології, використовує потужності обчислювальних центрів та персональних комп'ютерів.

Обираючи базу даних, особа може вибрати SQLite3 через його здатність працювати як у централізованих, так і у децентралізованих середовищах. SQLite3 - це база даних, яка забезпечує легка, ефективна та надійна керуванням даними, який не потребує окремого сервера, що робить його ідеальним варіантом для використання на персональних комп'ютерах.

2.7 Висновки до другого розділу

Було проаналізовано та визначено цільового користувача, його різноманітність та вимоги від кожну групу.

На основі аналізів цільової аудиторії та оцінки конкурентної спроможності були визначені ключові функції веб-додатка.

Також для розробки front-end було вибрано нативний код, щоб підтримати унікальність веб-додатку і зробити його більш конкурентоспроможнім для бізнесу.

Тому що у клієнтів дуже часто слабкі комп’ютери, щоб надавати їм більш швидку роботу сайту, була вибрана клієнт-серверна архітектура де сервер є головним.

Для програмування веб-додатку на стороні front-end була вибрана мова програмування JavaScript, та на стороні back-end - мова Python

Надалі, для введення проекта в комерцію можна розширити функціонал:

1) Підтвердження та відстеження статусу замовлення для користувачів;
2) Створення облікових записів користувачів з можливістю авторизації;
3) Оплата замовлення онлайн через платіжну систему;
4) Доробка адміністрування сайту для автоматичного завантаження даних сайту до бази даних товарів;
5) Доробка аналізу статистики продаж по сайту(SEO).

Після узгодження з замовником обсягів виконаних робіт проводиться тестування усіх можливостей створеного сайту. Потім узгоджуються всі питання з замовником і надалі виконується хостинг, наповнюється база даних веб-

додатка та починається тестова експлуатація.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

На основі поставленої кваліфікаційної задачі, проведених аналізів клієнтів сайту, аналізу потрібної системи та архітектури сайту, були розроблені front-end і back-end веб-додатка.

3.1 Розробка клієнтської частини сайту (front-end)

На основі аналізу вибору язика програмування для цього веб-додатку частини front-end був обраний язик JavaScript

Інтерфейс веб-додатку – одна із головних частин, якій слід приділяти велику увагу. Головною метою інтерфейсу веб-додатку було зробити взаємодію з програмою зручною та ефективною для користувача.

На головній сторінці (рис. 4) розміщують основні модулі для зручного навігації та доступу до ключових функцій інтерфейсу.

1) Меню – дозволяє швидко переходити до різних розділів товару;
2) Пошукове поле – дозволяє користувачеві швидко знаходити потрібний товар;
3) Профіль користувача – доступ до особистого профілю користувача, де він може додавати та змінювати інформацію про себе;
4) Рекламні пости – пропонують різноманітні товари та послуги.

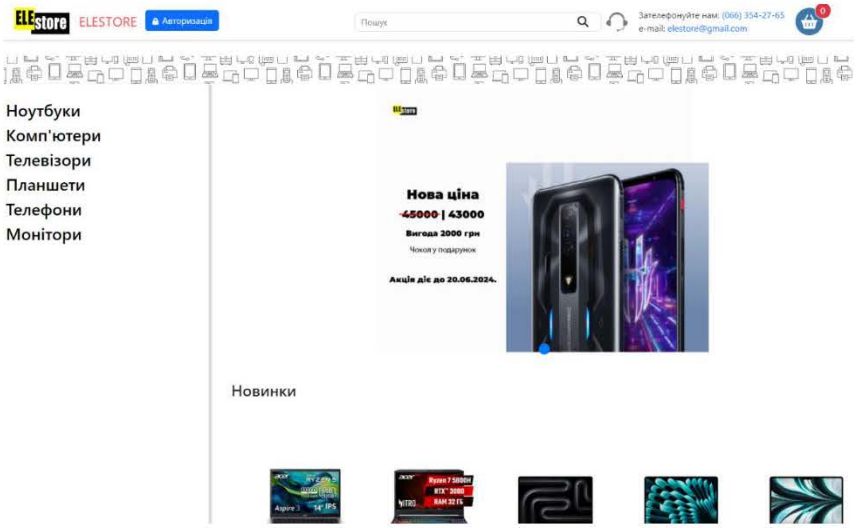

Рисунок 4 – Головна сторінка веб-додатку

У реалізації слайдера, використовується Swiper.js. Відмінно підходить для створення слайдерів, надаючи широкі можливості для налаштування та стилізації. З його допомогою можна швидко створювати інтерактивні слайдери, використовуючи готові рішення без необхідності великого обсягу коду (рис. 5).

```
<section class="swiper main-index__banner">
    <div class="swiper-wrapper main-index__banner__list">
        <div class="swiper-slide main-index__banner__slide">
            <div class="main-index__banner__slide__img__wrap">
                <img class="main-index__banner__slide__img" src="{% static 'assets/slider-main/slider-1.png' %}" alt="">
            </div>
        </div>
        <div class="swiper-slide main-index__banner__slide">
            <div class="main-index__banner__slide__img__wrap">
                <img class="main-index__banner__slide__img" src="{% static 'assets/slider-main/slider-2.png' %}" alt="">
            </div>
        </div>
        <div class="swiper-slide main-index__banner__slide">
            <div class="main-index__banner__slide__img__wrap">
                <img class="main-index__banner__slide__img" src="{% static 'assets/slider-main/slider-3.png' %}" alt="">
            </div>
        </div>
        <div class="swiper-slide main-index__banner__slide">
            <div class="main-index__banner__slide__img__wrap">
                <img class="main-index__banner__slide__img" src="{% static 'assets/slider-main/slider-4.png' %}" alt="">
            </div>
        </div>
        <div class="swiper-slide main-index__banner__slide">
            <div class="main-index__banner__slide__img__wrap">
                <img class="main-index__banner__slide__img" src="{% static 'assets/slider-main/slider-5.png' %}" alt="">
            </div>
        </div>
        <div class="swiper-slide main-index__banner__slide">
            <div class="main-index__banner__slide__img__wrap">
                <img class="main-index__banner__slide__img" src="{% static 'assets/slider-main/slider-6.png' %}" alt="">
            </div>
        </div>
    </div>
    <div class="swiper-pagination"></div>
</section>
```

Рисунок 5 – Реалізація слайдеру головної сторінки

В кожній сторінці, заголовок «Elestore» анімовано за допомогою JavaScript (рис. 6).

```
anime.timeline({loop: true})
  .add({
    targets: '.logo__text .line',
    scaleX: [0,1],
    opacity: [0.5,1],
    easing: "easeInOutExpo",
    duration: 1500
  }).add({
    targets: '.logo__text .letter',
    opacity: [0,1],
    translateX: [40,0],
    translateZ: 0,
    scaleX: [0.3, 1],
    easing: "easeOutExpo",
    duration: 1800,
    offset: '-=600',
    delay: (el, i) => 150 + 25 * i
  }).add({
    targets: '.logo__text',
    opacity: 0,
    duration: 3000,
    easing: "easeOutExpo",
    delay: 3000
  });
```

Рисунок 6 – Реалізація анімації заголовків

Користувач обирає вподобаний розділ з меню (рис. 7). Після цього він бачить рекламні банери, що були створені за допомогою бібліотеки Swiper.js. У нижній частині екрана розташовані категорії виробників.

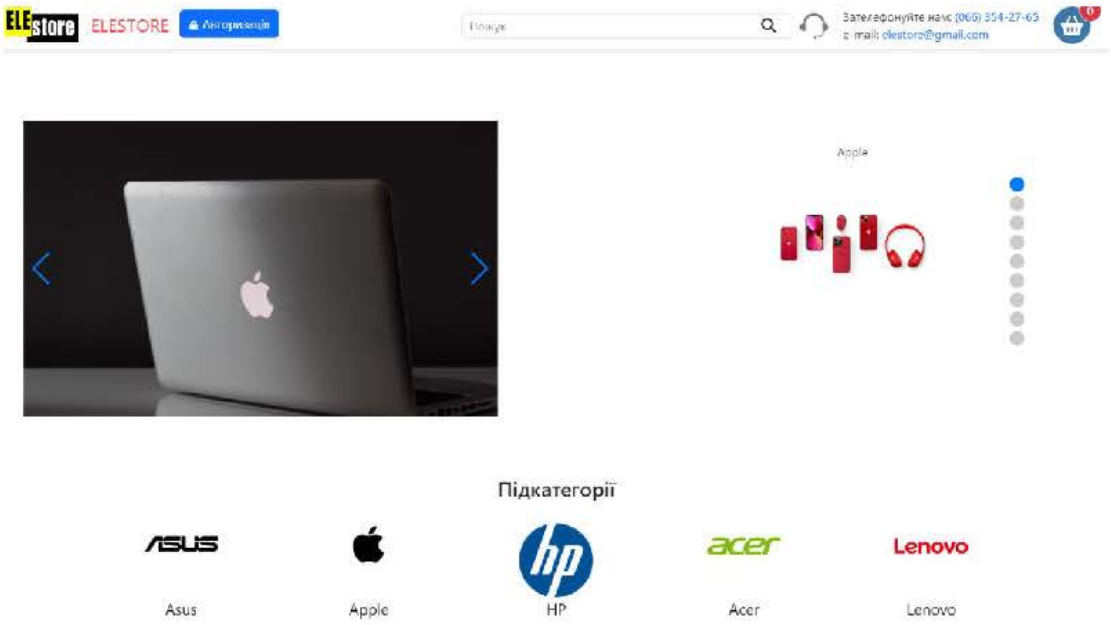

Рисунок 7 – Розділ виробників ноутбуків

Користувач вибирає виробника товару і обирає необхідний товар. (рис. 8).

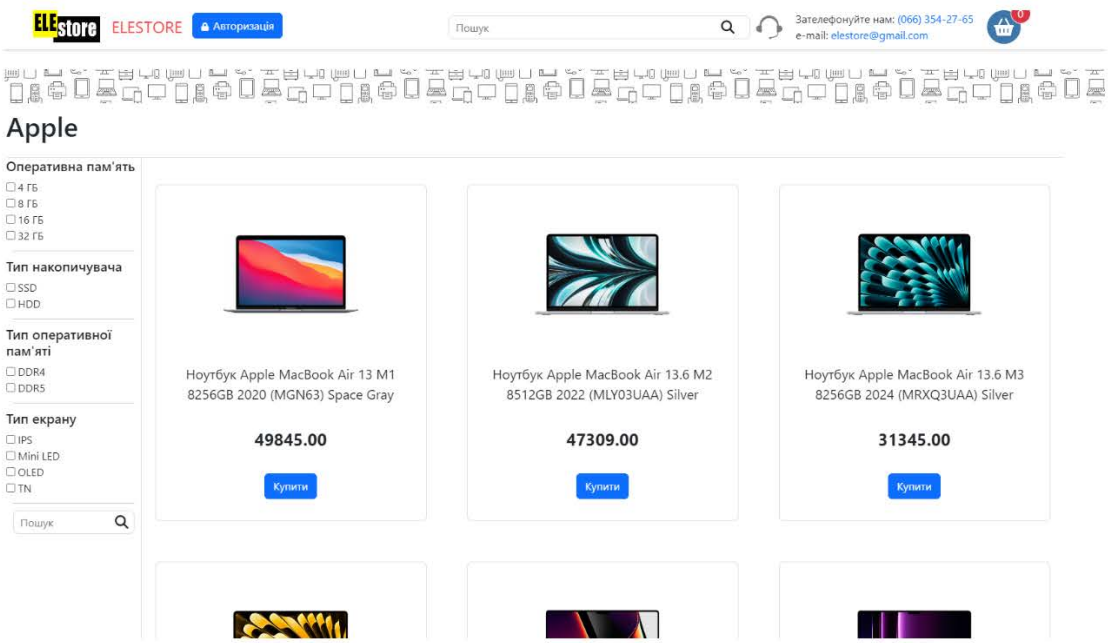

Рисунок 8 – Підрозділ ноутбуків «Apple»

Коли користувач натискає кнопку «Купити», він переходить до кошика (рис. 9). Де він може оформити свій товар.

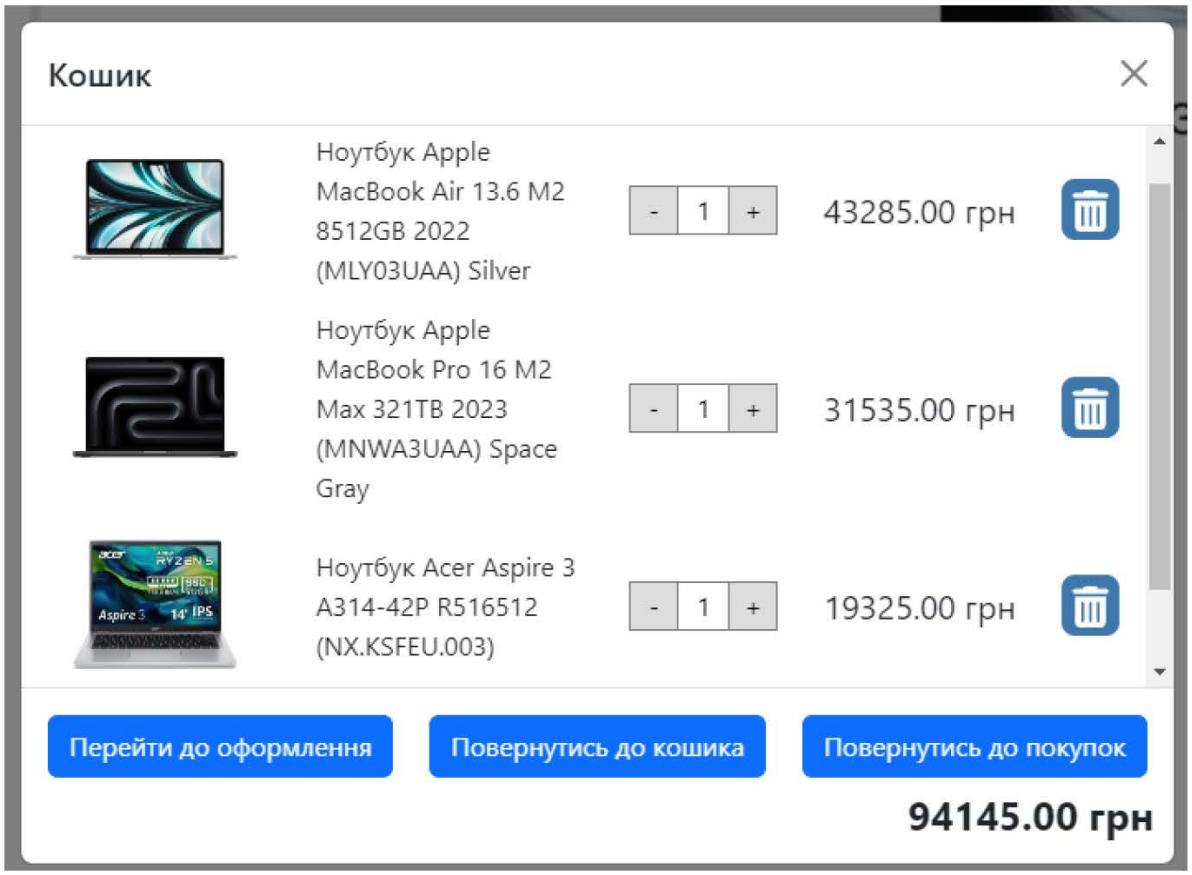

Рисунок 9 – Кошик для покупок

3.2 Розробка серверної частини додатку (back-end)

На основі аналізу вибору язика програмування для цього веб-додатку частини back-end був обраний язик Python з використанням фрейм-ворку Django.

Розглянемо код функції роботи додавання зміни та видалення товару з кошика зображено на малюнку (рис. 9).

Функція «cart_add» виконує запит на додавання товару в кошик (рис. 10). Коли вона отримує POST-запит, вона отримує ідентифікатор продукту з цього запиту. Після цього вона використовує цей ідентифікатор, щоб отримати відповідний об'єкт продукту з бази даних.

Якщо користувач, що виконав запит, функція «cart_add» перевіряє, чи вже є в кошику користувача товар з цим продуктом. Якщо такий товар вже існує, функція збільшує його кількість у кошику на одиницю. В іншому випадку вона створює новий запис у кошику користувача з цим продуктом.

Якщо користувач не аутентифікований, функція проводить аналогічні операції, але замість користувача використовує ідентифікатор сесії.

Після цього функція формує HTML-код для оновлення кошика на основі даних про товари у кошику. HTML-код включає оновлену інформацію про кількість товарів та загальну вартість кошика, що дозволяє динамічно відобразити зміни на веб-сторінці без її повного перезавантаження.

Крім цього, фреймворк Django забезпечує легку інтеграцію з різними базами даних, що робить його універсальним інструментом для розробки веб-додатків різного масштабу. Це дозволяє команді розробників ефективно управляти даними і швидко впроваджувати нові функції.

Використання мови Python сприяє більшій читабельності коду, що полегшує його підтримку та розвиток. Це особливо важливо в умовах швидкого зростання та змін, коли необхідно швидко адаптуватися до нових вимог і викликів.

```python
def cart_add(request):

    product_id = request.POST.get("product_id")

    product = Products.objects.get(id=product_id)

    if request.user.is_authenticated:
        carts = Cart.objects.filter(user=request.user, product=product)

        if carts.exists():
            cart = carts.first()
            if cart:
                cart.quantity += 1
                cart.save()
        else:
            Cart.objects.create(user=request.user, product=product, quantity=1)

    else:
        carts = Cart.objects.filter(
            session_key=request.session.session_key, product=product)

        if carts.exists():
            cart = carts.first()
            if cart:
                cart.quantity += 1
                cart.save()
        else:
            Cart.objects.create(
                session_key=request.session.session_key, product=product, quantity=1)

    user_cart = get_user_carts(request)
    cart_items_html = render_to_string(
        "carts/includes/included_cart.html", {"carts": user_cart}, request=request)

    response_data = {
        "message": "Товар добавлен в корзину",
        "cart_items_html": cart_items_html,
    }

    return JsonResponse(response_data)
```

Рисунок 10 – Функція «cart_add»

Функція «cart_change» виконує запит на зміну кількості товару у кошику (рис. 11).

Коли вона отримує POST-запит, вона отримує ідентифікатор кошика і нову кількість товару з цього запиту. За допомогою цього ідентифікатору отримати відповідний об'єкт кошика з бази даних. Після цього вона змінює кількість товару у кошику на нову кількість та зберігає зміни в базі даних.

Після цього функція формує HTML-код для оновлення кошика на основі даних про товари у кошику. Вона використовує цей HTML-код разом з повідомленням про те, що кількість товару була успішно змінена, для створення відповіді, яку вона повертає у форматі JSON.

```python
def cart_change(request):
    cart_id = request.POST.get("cart_id")
    quantity = request.POST.get("quantity")

    cart = Cart.objects.get(id=cart_id)

    cart.quantity = quantity
    cart.save()
    updated_quantity = cart.quantity

    cart = get_user_carts(request)
    cart_items_html = render_to_string(
        "carts/includes/included_cart.html", {"carts": cart}, request=request)

    response_data = {
        "message": "Количество изменено",
        "cart_items_html": cart_items_html,
        "quaantity": updated_quantity,
    }

    return JsonResponse(response_data)
```

Рисунок 11 – Функція «cart_change»

У функції cart_remove відбувається обробка запиту на видалення товару з кошика (рис. 12). Після отримання POST-запиту функція спочатку отримує ідентифікатор товару у кошику, який потрібно видалити.

Потім вона отримує об'єкт кошика за допомогою цього ідентифікатора та зберігає кількість товару перед його видаленням. Після цього об'єкт кошика видаляється з бази даних.

Після видалення об'єкта кошика формується HTML-код для оновлення кошика з урахуванням актуальних даних. Створюється JSON-відповідь, яка містить повідомлення про успішне видалення товару, HTML-код для оновлення кошика та кількість видалених одиниць товару.

Функція також перевіряє, чи залишилися інші товари у кошику після видалення. Якщо кошик порожній, функція може додатково відправити повідомлення або оновити інтерфейс, щоб відобразити, що кошик порожній. Це забезпечує користувачеві більш інтуїтивний досвід взаємодії з веб-додатком.

```python
def cart_remove(request):

    cart_id = request.POST.get("cart_id")

    cart = Cart.objects.get(id=cart_id)
    quantity = cart.quantity
    cart.delete()

    user_cart = get_user_carts(request)
    cart_items_html = render_to_string(
        "carts/includes/included_cart.html", {"carts": user_cart}, request=request)

    response_data = {
        "message": "Товар удален",
        "cart_items_html": cart_items_html,
        "quantity_deleted": quantity,
    }

    return JsonResponse(response_data)
```

Рисунок 12 – Функція «cart_remove»

3.3 Інтеграція з базою даних

На основі аналізу вибору бази даних для веб-додатку було вирішено скористатися SQLite3. Для забезпечення цілісності даних між таблицями прийняли рішення застосувати зовнішні ключі (foreign keys). Модель даних (рис. 13) охоплює інформацію про продукцію, фільтри, корзини та категорії. Структура забезпечує цілісність даних і дозволяє ефективно керувати інформацією про продукти, фільтри, корзини та категорії веб-додатку.

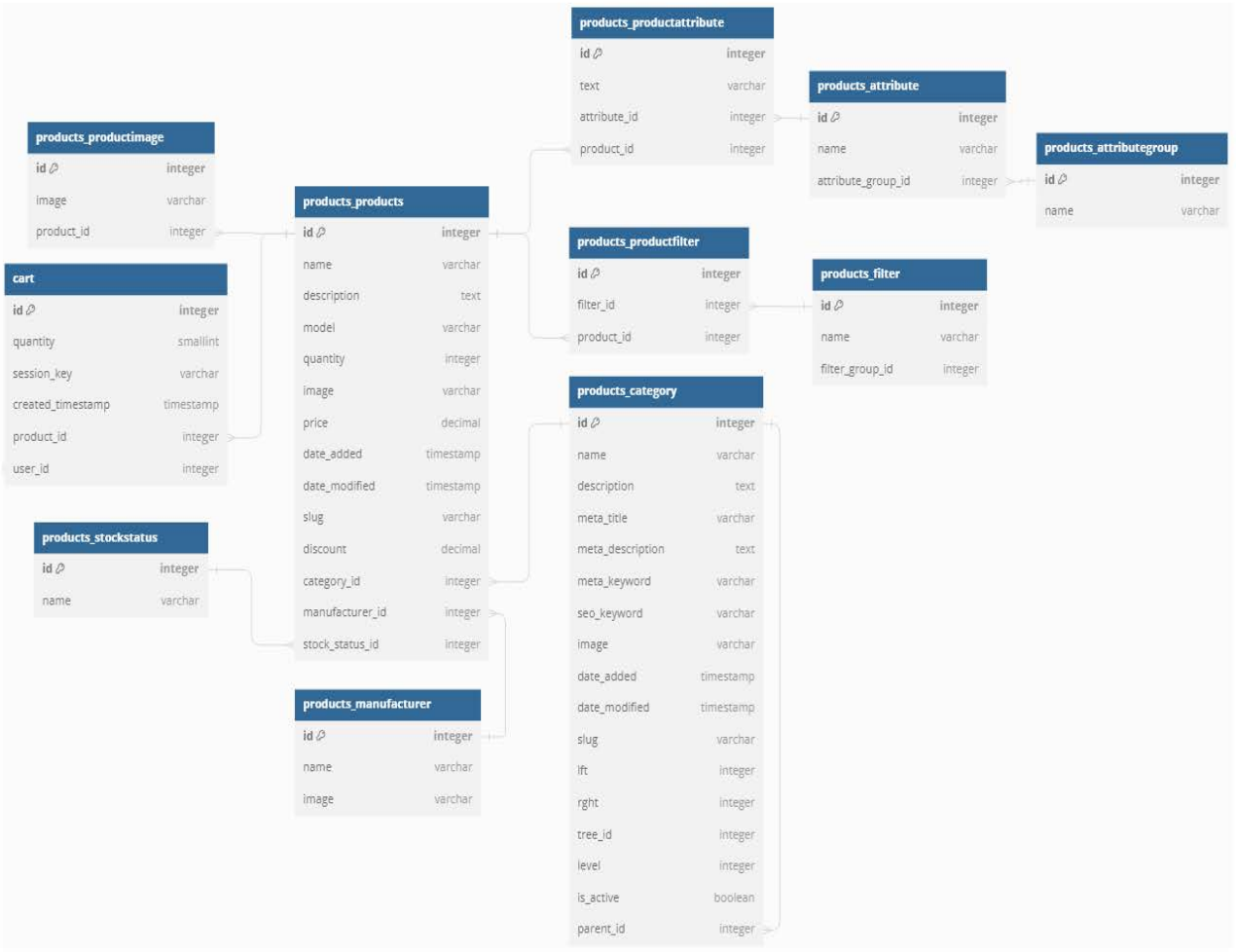

Рисунок 13 – Модель бази даних продукції

Модель бази даних складається з таблиць: «products_products», «cart», «products_productimage», «products_stockstatus», «products_manufacturer», «products_category», «products_productfilter», «products_filter», «roducts_productattribute», «products_attribute», «products_attributegroup».

Структура даних таблиці «products_products» представлена наступним чином (таблиця 1):

Таблиця 1

Структура даних таблиці «products_products»

Ім’я	Тип та розмір поля	опис
id	integer	первинний ключ
name	varchar(299)	назва
description	text	опис
model	varchar(255)	модель
quantity	integer	кількість
image	varchar(300)	зображення
price	decimal	ціна
date_added	datetime	дата додавання
date_modified	datetime	дата змінення
slug	varchar(299)	жетон привязування
discount	decimal	знижка
category_id	bigint	ідентифікатор категорії
manufacturer_id	bigint	ідентифікатор виробника
stock_status_id	bigint	ідентифікатор наявності

Таблиця «products_products» вона складається з полів: ідентифікатор, назва, опис, модель, кількість, зображення, ціна, дата додавання, дата змінення, жетон прив’язування, знижка, ідентифікатор категорії, ідентифікатор виробника, ідентифікатор наявності.

Таблиця «products_category» має зв’язок з таблицею «products_products» по індифікатору категорії (таблиця 2).

Таблиця 2

Структура даних таблиці «products_category»

Ім’я	Тип та розмір поля	Опис
id	integer	первинний ключ
name	text	назва
meta_title	varchar(255)	назва мети
meta_description	text	мета опис
meta_keyword	varchar(255)	мета ключове слово
seo_keyword	varchar(255)	seo ключове слово
image	varchar(300)	зображення
date_added	datetime	дата додавання
date_modified	datetime	дата змінення
slug	varchar(299)	жетон привязування
lft	integer unsigned	відступ правої сторони
rght	integer unsigned	відступ лівої сторони
tree_id	integer unsigned	ідентифікатор дерева
level	integer unsigned	рівень
is_active	bool	активний
parent_id	bigint	батьківський ідентифікатор

Таблиця «products_category» містить такі поля як: ідентифікатор, назва, назва мети, мета опис, мета ключове слово, seo ключове слово, зображення, дата додавання, дата змінення, жетон прив’язування, відступ правої сторони, відступ лівої сторони, ідентифікатор дерева, рівень, активний, батьківський ідентифікатор.

Таблиця «products_productimage» має зв’язок з таблицею «products_products» по індифікатору продукту (таблиця 3).

Таблиця 3

Структура даних таблиці «products_productimage»

Ім’я	Тип та розмір поля	Опис
id	integer	первинний ключ
image	varchar(300)	зображення
product_id	bigint	ідентифікатор продукту

Таблиця «products_productimage» містить такі поля як: ідентифікатор, зображення, автор, ідентифікатор продукту.

Таблиця «card» має зв’язок з таблицею «products_products» по індифікатору продукту (таблиця 4).

Таблиця 4

Структура даних таблиці «card»

Ім’я	Тип та розмір поля	Опис
id	integer	первинний ключ
quanity	smallint unsigned	кількість
session_key	carchar(32)	ключ сеансу
created_timestamp	datetime	тимчасова створена мітка
product_id	bigint	ідентифікатор продукту
user_id	bigint	ідентифікатор користувачу

Таблиця «card» містить такі поля як: ідентифікатор, кількість, ключ сеансу, тимчасова створена мітка, ідентифікатор продукту, ідентифікатор користувачу.

Таблиця «products_manufacturer» має зв’язок з таблицею «products_products» по індифікатору виробника продукції (таблиця 5).

Таблиця 5

Структура даних таблиці «products_manufacturer»

Ім’я	Тип та розмір поля	Опис
id	integer	первинний ключ
name	text	назва
image	varchar(300)	зображення

таблиця «products_manufacturer» містить такі поля як: ідентифікатор, назва, зображення.

Таблиця «products_productfilter» має зв’язок з таблицею «products_products» по індифікатору продукту (таблиця 6).

Таблиця 6

Структура даних таблиці «products_productfilter»

Ім’я	Тип та розмір поля	Опис
id	integer	первинний ключ
filter_id	bigint	ідентифікатор фільтру
product_id	bigint	ідентифікатор продукту

Таблиця «products_productfilter» містить такі поля як: ідентифікатор, текст, ідентифікатор фільтру, ідентифікатор продукту.

Таблиця «products_filter» має зв’язок з таблицею «products_productfilter» по індифікатору фільтра (таблиця 7).

Таблиця 7

Структура даних таблиці «products_filter»

Ім’я	Тип та розмір поля	Опис
id	integer	первинний ключ
name	text	назва
filter_group_id	bigint	ідентифікатор групи фільтру

Таблиця «products_filter» містить такі поля як: ідентифікатор, текст, ідентифікатор фільтру, ідентифікатор продукту.

Таблиця «products_productattribute» має зв'язок з таблицею «products_products» по індифікатору продукту (таблиця 8).

Таблиця 8

Структура даних таблиці «products_productattribute»

Ім'я	Тип та розмір поля	Опис
id	integer	первинний ключ
text	varchar(255)	текст
attribute_id	bigint	ідентифікатор атрибута
product_id	bigint	ідентифікатор продукту

Таблиця «products_manufacturer» містить такі поля як: ідентифікатор, текст, ідентифікатор атрибута, ідентифікатор продукту.

Таблиця «products_attribute» має зв'язок з таблицею «products_productattribute» по індифікатору атрибута (таблиця 9).

Таблиця 9

Структура даних таблиці «products_attribute»

Ім'я	Тип та розмір поля	Опис
id	integer	Первинний ключ
name	text	Назва
attribute_group_id	bigint	Ідентифікатор групи атрібуту

Таблиця «products_attribute» містить такі поля як: ідентифікатор, назва, ідентифікатор групи атрибуту.

Таблиця «products_attributegroup» має зв'язок з таблицею «products_attribute» по індифікатору атрибутики групи (таблиця 10).

Таблиця 10

Структура даних таблиці «products_attributegroup»

Ім’я	Тип та розмір поля	опис
id	integer	первинний ключ
name	text	назва

Таблиця «products_attributegroup» містить такі поля як: ідентифікатор та назву.

3.4 План дій до запуску веб-додатку

1. Для початку запуску веб-додатку треба встановити Visual Studio Code, Node.js та Python 3.8.

Visual Studio Code - інтегроване середовище розробки що призначене для написання, редагування програмного коду.

Node.js - це вільне та відкрите середовище для виконання JavaScript, яке дозволяє запускати код JavaScript поза браузером, на сервері.

2. Відкриваємо проект в Visual Studio Code.

3. В терміналі вписуємо команду - «python -m venv venv» (рис. 14).

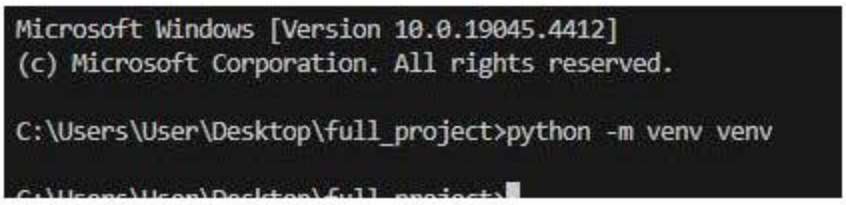

Рисунок 14 – Створення віртуального середовища

4. Команда «python -m venv venv» створює нове віртуальне середовище Python в поточному каталозі з назвою "venv". Віртуальне середовище дозволяє ізолювати залежності та бібліотеки Python для проекту, що спрощує управління залежностями між проектами. В результаті виконання цієї команди створюється каталог "venv", який містить необхідні файли для віртуального середовища.

5. Активуємо віртуальне середовище за допомогою команди -

«venv\Scripts\activate» (рис. 15).

```
C:\Users\User\Desktop\full_project>venv\Scripts\activate

(venv) C:\Users\User\Desktop\full_project>
```

Рисунок 15 – Активація віртуального середовища

Команда «venv\Scripts\activate» активує віртуальне середовище Python, що було створене за допомогою модуля venv. Після виконання цієї команди командний рядок буде змінений таким чином, що він вказуватиме на активоване віртуальне середовище.

6. Встановлюємо пакет «django-cors-headers» за допомогу команди «pip install django-cors-headers» (рис. 16).

```
(venv) C:\Users\User\Desktop\full_project>pip install django-cors-headers
Requirement already satisfied: django_cors_headers in c:\users\user\desktop\full_project\venv\lib\site-packages (4.4.1)
Requirement already satisfied: asgiref>=3.6 in c:\users\user\desktop\full_project\venv\lib\site-packages (from django-cors-headers) (3.8.1)
Requirement already satisfied: Django>=4.2 in c:\users\user\desktop\full_project\venv\lib\site-packages (from django-cors-headers) (4.2.11)
Requirement already satisfied: typing-extensions>=4; python_version < "3.11" in c:\users\user\desktop\full_project\venv\lib\site-packages (from a
sgiref>=3.6->django-cors-headers) (4.11.0)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\user\desktop\full_project\venv\lib\site-packages (from Django>=4.2->django-cors-header
) (0.5.0)
Requirement already satisfied: tzdata; sys_platform == "win32" in c:\users\user\desktop\full_project\venv\lib\site-packages (from Django>=4.2->dja
ngo-cors-headers) (2024.1)
Requirement already satisfied: backports.zoneinfo; python_version < "3.9" in c:\users\user\desktop\full_project\venv\lib\site-packages (from Djang
o>=3.2->django-cors-headers) (0.2.1)
WARNING: You are using pip version 20.2.3; however, version 24.0 is available.
You should consider upgrading via the 'c:\users\user\desktop\full_project\venv\scripts\python.exe -m pip install --upgrade pip' command.
```

Рисунок 16 – Встановлення пакету «django-cors-headers»

Команда «pip install django-cors-headers» встановлює пакет Django CORS Headers, який дозволяє управляти політикою обміну ресурсами між додатками на Django та іншими джерелами, щоб уникнути проблем, пов'язаних із змішаним вмістом на сторінках веб-додатків.

7. Встановлюємо пакети Python з «requirements.txt» файлу - «pip install -r requirements.txt» (рис. 17).

```
(venv) C:\Users\User\Desktop\full_project>pip install -r requirements.txt
Collecting asgiref==3.8.1
  Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Collecting backports.zoneinfo==0.2.1
  Using cached backports.zoneinfo-0.2.1-cp38-cp38-win_amd64.whl (38 kB)
Collecting Django==4.2.11
  Using cached Django-4.2.11-py3-none-any.whl (8.0 MB)
Collecting django-cors-headers==4.3.1
  Using cached django_cors_headers-4.3.1-py3-none-any.whl (12 kB)
Collecting django-js-asset==2.2.0
  Using cached django_js_asset-2.2.0-py3-none-any.whl (4.7 kB)
Collecting django-mptt==0.14.0
  Using cached django_mptt-0.14.0-py3-none-any.whl (115 kB)
Collecting pillow==10.3.0
  Using cached pillow-10.3.0-cp38-cp38-win_amd64.whl (2.5 MB)
Collecting sqlparse==0.5.0
  Using cached sqlparse-0.5.0-py3-none-any.whl (43 kB)
Collecting typing-extensions==4.11.0
  Using cached typing_extensions-4.11.0-py3-none-any.whl (34 kB)
Collecting tzdata==2024.1
  Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Installing collected packages: typing-extensions, asgiref, backports.zoneinfo, sqlparse, tzdata, Django, django-cors-headers, django-js-asset, dja
ngo-mptt, pillow
Successfully installed Django-4.2.11 asgiref-3.8.1 backports.zoneinfo-0.2.1 django-cors-headers-4.3.1 django-js-asset-2.2.0 django-mptt-0.14.0 pil
low-10.3.0 sqlparse-0.5.0 typing-extensions-4.11.0 tzdata-2024.1
WARNING: You are using pip version 20.2.3; however, version 24.0 is available.
You should consider upgrading via the 'c:\users\user\desktop\full_project\venv\scripts\python.exe -m pip install --upgrade pip' command.

(venv) C:\Users\User\Desktop\full_project>
```

Рисунок 17 – Встановлення пакетів Python з «requirements.txt» файлу

Команда «pip install -r requirements.txt» встановлює всі пакети Python, перераховані у файлі requirements.txt, з урахуванням їх версій.

8. Переходимо до папки static та запускаємо (Forn-end) сайту - «npm i» (рис. 18).

```
(venv) C:\Users\User\Desktop\full_project>cd static

(venv) C:\Users\User\Desktop\full_project\static>npm i

up to date, audited 8 packages in 2s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Рисунок 18 – Перехід до папки static та запуск Forn-end

Команда «npm i» встановлює всі залежності, перераховані у файлі package.json вашого проекту. npm - це менеджер пакетів Node.js, і використовується для управління залежностями JavaScript для вашого проекту.

9. Повертаємось та запускаємо (back-end) сайту - «python manage.py runserver». (рис. 19).

```
(venv) C:\Users\User\Desktop\full_project>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 15, 2024 - 20:04:03
Django version 4.2.11, using settings 'artmagic.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 19 – Запуск back-end сайту

Команда «python manage.py runserver» використовується для запуску локального веб-сервера Django. Після виконання цієї команди Django розпочне слухати вказаний порт на локальній машині і надавати доступ до вашого веб-додатка.

3.5 Висновки до третього розділу

Було детально описано процес розробки веб-додатку, включаючи реалізацію клієнтської (front-end) та серверної (back-end) частин, а також інтеграцію з базою даних. На основі проведеного аналізу були зроблені наступні висновки.

Для реалізації клієнтської частини було обрано мову програмування JavaScript, що забезпечило створення динамічного та інтерактивного інтерфейсу користувача. Використання бібліотеки Swiper.js дозволило легко реалізувати слайдери для демонстрації товарів, що значно підвищило зручність використання та візуальну привабливість додатку. Анімація заголовків та інтерактивні елементи інтерфейсу зробили взаємодію користувача з веб-додатком більш привабливою та зручною.

Для серверної частини було обрано мову програмування Python з використанням фреймворку Django, що забезпечило високу продуктивність та простоту в реалізації CRUD (Create, Read, Update, Delete) операцій. Реалізовані функції для додавання, зміни та видалення товарів з кошика, які забезпечують необхідну функціональність для управління замовленнями користувачів.

Використання сесій для ідентифікації користувачів дозволяє забезпечити збереження стану кошика навіть без реєстрації.

Для збереження даних на сайті було обрано СУБД SQLite3, що забезпечило простоту в налаштуванні та використанні для невеликих проектів. Використання зовнішніх ключів (foreign keys) забезпечило цілісність даних та ефективне управління зв'язками між таблицями. Структура бази даних була ретельно спроектована для зберігання інформації про продукти, кошики, категорії та виробників, що дозволило легко масштабувати систему при зростанні обсягів даних.

Докладний план дій для налаштування середовища розробки та запуску додатку допоміг забезпечити безперебійну роботу на етапі деплойменту.

Розроблений веб-додаток відповідає поставленим вимогам та забезпечує необхідну функціональність для кінцевих користувачів. Завдяки вибору відповідних інструментів та технологій, вдалося створити зручний, ефективний та масштабований продукт.

ВИСНОВОК

За результатами дипломної роботи було розроблено макет інтернет-магазину у вигляді веб-додатку з можливістю подальшого його впровадження у бізнес.

Здійснено аналіз сучасних рішень для розробки інтернет-магазинів та етапи виконання розробки веб-додатка. Розглянуті основні принципи проектування веб-додатка: фокус на користувачеві, зручність використання, SEO.

На основі аналізу сучасних рішень розробки веб-додатків, було розглянуто можливості різноманітних мов програмування таких, як C, C++, C#, Python, JAVA, JavaScript, Go. Дивлячись на сучасну тенденцію веб-розробок, було обрано дві мови програмування Python та JavaScript для виконання поставлених завдань, через те, що вони є мовами високого рівня та мають великий обсяг бібліотек. Мова Python для back-end, JavaScript – front-end.

Мова Python є високорівневою мовою програмування загального призначення і орієнтована на підвищення продуктивності розробника і читання коду з мінімалістичним синтаксисом ядра. Має велику кількість вбудованих бібліотек.

Було обрано бібліотеку фреймворк - Django, що має у своєму складі доволі великий перелік стандартних процедур та функцій, та модулів. Також в роботі було використано плагін до бібліотеки Django - admin панель, що призвів до скорочення часу розробки управління та наповнення клієнтських даних веб-додатку. Завдяки чому можна змінювати контент веб-додатка, моніторити замовлення та змінювати права користувачів, тощо.

Мова програмування JavaScript – динамічна, об'єктно-орієнтована, прототипна. Ця мова програмування дуже сильно розвинена, як було згадано раніше та у неї є певні бібліотеки-фреймворки, та великий вибір додаткових функцій.

Проте, дивлячись на нові тенденції було використано нативну мову

програмування (тобто чистий JavaScript), що також доволі сильно розвивається і якщо поглибитися, то можна на ньому написати навіть доволі низькорівневі процедури та функції, щоб мати велику швидкість виконання коду веб-додатку. Тобто fornt-end у проекті було реалізовано за допомогою власно написаних функцій та модулей на нативній мові програмування JavaScript, щоб під лаштуватися під стиль, та певні принципи написання коду сучасного веб-додатку.

На стороні front-end реалізовані такі функції:

1) Адаптивний дизайн;
2) Головна сторінка веб-додатку;
3) Рекламні слайдери;
4) Навігація сайту;
5) Фільтр по товарам і виробникам;
6) Пошук;
7) Комунікаційні функції з користувачем.

На клієнтській частині веб-додатка, для деяких сторінок було використано підхід, що використовується у SPA (Single Page Application – це тип веб-додатку, в якому вся інтерактивність і відображення даних відбувається на одній веб-сторінці, без необхідності завантаження додаткових сторінок або перезавантаження браузера;). Тобто, при натисканні кнопки на панелях сторінок веб-додатка, не перезавантажується сама сторінка повністю, а навпаки, завантажується лише необхідні дані.

Це дає можливість не навантажувати клієнтську сторону веб-додатка додатковими запросами, а використовується підхід деструктуризації (Деструктурізація - це коли щось велике розбиваємо на більш дрібні частинки, як наприклад велика задача, на якісь маленькі 5 задач, при виконанні, яких велика задача буде виконана швидше).

Такий підхід, дає можливість більш ефективно обмінюватись даними серверу із клієнтською частиною, вирішить проблему швидкості завантаження даних.

1) На стороні back-end реалізовані такі функції:
2) Функція додавання, видалення та зміна товару в магазині;
3) Функція додавання, зміни та видалення товарів у кошику.

За основу бази даних було узято СУБД SQLite3, через те, що з нею доволі легко працювати, та вона має доволі зручний механізм керування, що у свою чергу дає більш свободи до реалізації необхідного функціоналу веб-додатку для оптимізації виконання пошуку та обміну даних з веб-додатком.

Практичне значення отриманих результатів розробки веб-додатка інтернет-магазину сприяє для підвищення конкурентної спроможності бізнеса.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 – «Інженерія програмного забезпечення» і демонструє володіння такими навиками як:

1) Здатність проектувати та розробляти програмне забезпечення із застосуванням різних способів програмування: узагальненого об'єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень структурами даних і механізмами управління.
2) Здатність реалізувати багаторівневу обчислювальну модель на основі архітектури клієнт-сервер.

Серед результатів навчання, визначених стандартом, кваліфікаційна робота реалізовує наступні:

1) Використовувати інструментальні засоби розробки клієнт-серверних застосувань.
2) Володіти мовами системного програмування та методами розробки програм, що взаємодіють з компонентами комп'ютерних систем.
3) Розробляти програмні моделі програмування та методи розробки програм, що взаємодіють з компонентами комп'ютерних систем.
4) Розробляти програмні моделі предметних середовищ, вибирати методи програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв'язання задач в галузі комп'ютерних наук.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. П'юрівал С. П. Основи розробки веб-додатків курс лекцій. Київ: Питер, 2015. 272 с.

2. Webcase. Спеціалізація на розробці програмного забезпечення для автоматизації тестування веб-додатків. URL: https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 16.04.2024).

3. Webtune. Сервіс, який пов'язаний із веб-розробкою або оптимізацією веб-сайтів. URL: https://webtune.com.ua/statti/web-rozrobka/etapy-stvorennya-veb-sajtiv/ (дата звернення: 16.04.2024).

4. Трофименко О. Г. Веб-дизайн та HTML-програмування. Одеса: Фенікс, 2018. 194 с.

5. Трофименко О. Г. Веб-технології та веб-дизайн : навч. посібник / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачінда. – Одеса : Фенікс, 2019. 284 с.

6. Turumburum. Персоналізація в маркетингу: закордонні кейси та практичні поради. URL: https://turumburum.ua/blog/personalizaciya-v-marketingu-zakordonni-keysi-ta-praktichni-poradi-dlya-ecommerce (дата звернення: 18.04.2024).

7. Wikipedia. Метод оцінки, який використовує експертні знання та досвід для швидкого та приблизного прийняття рішень або оцінки ситуації. URL: https://uk.wikipedia.org/wiki/Евристичне_оцінювання (дата звернення: 19.04.2024).

8. Genius space. Інтуїтивно зрозумілий веб-дизайн: як зробити ваш сайт простим та зручним у використанні. URL: https://genius.space/lab/intuyitivno-zrozumilij-veb-dizajn-yak-zrobiti-vash-sajt-intuyitivno-zrozumilim-u-vikoristanni/ (дата звернення: 21.04.2024).

9. Hostiq. Що таке адаптивний дизайн сайту та як його зробити. URL: https://hostiq.ua/blog/ukr/adaptive-design/ (дата звернення: 25.04.2024).

10. Web-promo. Навігація сайту: як оптимізувати, щоб клієнт "не загубився" на шляху до покупки. URL: https://web-promo.ua/ua/blog/navihatsiya-saytu-yak-optymizuvaty-shchob-kliyent-ne-zahubyvsya-na-shlyakhu-do-pokupky/ (дата звернення: 30.04.2024).

11. DOU. Розуміння Клієнт-Серверної Архітектури на прикладах. URL: https://dou.ua/forums/topic/44636/ (дата звернення: 04.05.2024).

12. Training-qatestlab. Клієнт-серверна архітектура. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 07.05.2024).

13. PPN. Погляд на фреймворки в IT: Визначення та значення термінів. URL: https://pnn.com.ua/ua/blog/detail/what-the-framework-in-simple-words (дата звернення: 08.05.2024).

14. Anywhere club. Бібліотеки та фреймворки JavaScript: як правильно їх вибрати. URL: https://aw.club/global/uk/blog/javascript-libraries-and-frameworks-how-to-choose (дата звернення: 10.05.2024).

15. Seo-evolution. Написання технічного завдання на розробку інтернет-магазину. URL: https://seo-evolution.com.ua/blog/razrabotka/napisannya-tehnichnogo-zavdannya-na-rozrobku-internet-magazinu (дата звернення: 11.05.2024).

16. Buklib. Технології обробки даних та основи проектування баз даних. URL: https://buklib.net/books/24451/ (дата звернення: 12.05.2024).

17. HillelBlog. Переваги і недоліки мови Python. URL: https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python (дата звернення: 13.05.2024).

18. BoscTech Labs. Розробка веб-додатків: повний посібник на 2023 рік. URL: https://bosctechlabs.com/a-comprehensive-guide-web-application-development/ (дата звернення: 15.05.2024).

19. Девід Фленаган JavaScript: Ларманий довідник, 3-є видання. Північне шосе Гравенстен: Reilly Media, 2012, 600 с.

20. Петр Левашов. Python с нуля. Мінськ: Питер, 2024. 448 с.

21. Лаура Грессер, Ван Лук Кенг. Глибоке навчання: теорія та практика мовою Python. Мінськ: Питер, 2022. 415 с.

22. Владимир Дронов. Django 2.1. Практика створення веб-застосунків на Python. Сант-Петербург: БХБ-Петербург, 2019. 672 с.