КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Програмна система електронної торгівлі товарами широкого вжитку»

Виконала _____
(Підпис)
Бортник Софія Сергіївна
(прізвище, ім'я, по батькові)

Керівник _Бойко Юлія Петрівна
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
_____ (Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри_____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Засвідчую, що у цій кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1 ст.
Завдання до дипломної роботи	1 ст.
Відомість дипломної роботи	1 ст.
Календарний план	1 ст.
Реферат (Анотація)	1 ст.
Анотація (іноземною мовою-англійською)	1 ст.
Зміст	1 ст.
Вступ	3 ст.
Розділ 1	11 ст.
Розділ 2	16 ст.
Розділ 3	16 ст.
Висновки	2 ст.
Перелік використаних джерел	2 ст.
Додатки	11 ст.

<table>
<tr><td></td><td></td><td></td><td></td><td colspan="3" rowspan="2">ДП XXXX 00.000.00</td></tr>
<tr><td></td><td>ПІБ</td><td>Підп.</td><td>Дата</td></tr>
<tr><td>Розробн.</td><td>Бортник С. С.</td><td></td><td></td><td rowspan="4">Відомість дипломної роботи</td><td>Лист</td><td>Листів</td></tr>
<tr><td>Керівн.</td><td>Бойко Ю. П.</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Н/контр.</td><td>Макаренко С.А.</td><td></td><td></td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Зав.каф.</td><td>Плескач В.Л.</td><td></td><td></td></tr>
</table>

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів бакалаврської роботи	Термін виконання етапів бакалаврської роботи	Відмітка про виконання
1)	Настановча групова співбесіда з бакалаврської роботи	01.12.2020	Виконано
2)	Затвердження плану бакалаврської роботи	18.02.2021	Виконано
3)	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	Виконано
4)	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	Виконано
5)	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	Виконано
6)	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	Виконано
7)	Подання роботи у першому варіанті	11.05.2021	Виконано
8)	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	Виконано
9)	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврську роботу)	11.06.2021	
10)	Захист бакалаврської роботи	23.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота: 68 с., 11 рис.,1 табл., 25 джерел, 3 дод.

Ця кваліфікаційна робота присвячена проектуванню та розробленню програмної системи електронної торгівлі товарами широкого вжитку.

Метою кваліфікаційної роботи підвищення ефективності обробки замовлень інтернет- магазину товарів широкого вжитку шляхом створення back-end системи веб-сервісу.

Для досягнення поставленої мети треба вирішити такі **завдання:**

1) Дослідити сучасні підходи до розроблення і впровадження системи електронної торгівлі (інтернет-магазин);
2) Провести аналіз архітектурних рішень і вибір програмних засобів для реалізації інтернет-магазинів;
3) Програмно реалізувати back-end систему для інтернет-магазину.

Об'єктом нашого дослідження є процес обробки замовлень інтернет-магазину.

Предметом є програмні засади створення back-end системи для інтернет-магазину.

Методи дослідження.

Вивчення наукової літератури по темі, аналіз відкритих джерел та публікацій в мережі інтернет, практичне порівняння систем розробки веб-додатків, дослідження найбільш популярних CMS та Node.js, включно з документацією. Аналіз архітектури наявних на ринку back-end систем для інтернет-магазину.

Ключові слова: електронна торгівля, інтернет-магазин, back-end система

ABSTRACT

Thesis: 68 pages, 11 figures, 1 tables, 25 sources, 3 appendices.

This thesis is devoted to the design and development of electronic trade system for usual commodity.

The purpose of the bachelor qualification paper is to increase the efficiency of e-commerce order processing by creating e-commerce back-end system.

In order to achieve this goal it is necessary to solve the following tasks:

1) Investigate modern approaches to the development and implementation of e-commerce;

2) Conduct an analysis of architectural solutions and the choice of software for the implementation of e-commerce;

3) Develop back-end system for an online store.

The object of our research is the e-commerce order processing system.

The subject is the software principles of e-commerce back-end system development.

Research methods.

Studying of scientific literature in field of thesis. Analising of open sources and internet publications, practical comparison of web application development platforms, researching in set of most popular CMS and Node.js platform, including official documentation. Analising of architecture of existing back-end systems foe e-commerce sites.

Keywords: e-commerce, online-shop, back-end system.

ЗМІСТ

ВСТУП

З появою інтернету, IT- технологій стрімко почала розвиватися електронна комерція. На цій хвилі попиту бізнес має великі можливості збільшити свої доходи, а саме через онлайн-торгівлю. Онлайн-торгівля – це насамперед реалізація товарів широкого вжитку через Інтернет. Наприклад, таких асортиментних груп, як одяг, взуття, домашня електроніка, книги і т.п. Онлайн-торгівля в усьому світі розвивається досить прискореними темпами, а в умовах пандемії особливо.

На сьогодні є очевидним те, що інтернет-магазини, які відповідають сучасним вимогам, є тим торговим каналом, який дає змогу компанії зміцнювати свої позиції як на ринках України, так і за її межами. Ефективність їх роботи залежить як від правильно розробленого зовнішнього вигляду, з яким безпосередньо взаємодіє користувач, так і від програмно-апаратної частини сайту. Однією із вагомих задач є адміністративне управління контентом інтернет-ресурсу, а саме, яким способом цей контент надається для сайту.

Більшість сайтів побудовані на базі монолітної (одно-ядерної) програмної архітектури та базуються на одному CMS, що включає у себе frontend частину, backend частину, базу даних та адміністративний інтерфейс. Це означає, що на стороні сайту, окрім власне обслуговування клієнтів та продажів, виконуються всі роботи з ведення товарного асортименту, формування цін, завантаження та імпорт значних даних. Очевидно, що це створює додаткове навантаження на менеджерів з одного боку, бо у більшості випадків підприємства вже мають впроваджені облікові системи, де ці ж менеджери так само вводять інформацію про товарний асортимент, залишки та ціни і роблять це у значно зручнішому, спеціально пристосованому для цього інтерфейсі.

Актуальність роботи. Запропонована у роботі backend система є розподіленою (двох-ядерною), такою, що не передбачає активного ведення товарного асортименту на сайті, побудованому на CMS або Node.js, а натомість спирається на вже наявні дані у обліковій базі даних і передбачає організацію ефективного асинхронного обміну між backend частиною сайту та обліковою

системою. При виборі програмних засобів для реалізації цієї архітектури виходили з того факту, що більшість середніх та крупних підприємств в Україні мають впроваджені рішення на базі платформи BAF/BAS. За деякими даними обсяг таких впровадження сягає біля 98% ринку. Окрім того, сучасні можливості платформи BAF, такі як вбудовані класи для створення веб-сервісів та засоби роботи зі стеком JSON/XML, а також клієнт-серверна архітектура роботи, дозволяє перетворити рішення на BAF у повноцінний веб-сервіс для обміну даними у мережі інтернет. Така концепція видається дуже перспективною з точки зору реальних впроваджень за умови доброї волі та взаємодії розробників різних частин такої розподіленої backend системи, зокрема спеціалістів з BAF та веб-студій, що працюють над сайтом в одній із популярних CMS.

Мета роботи: підвищення ефективності обробки замовлень інтернет-магазину товарів широкого вжитку шляхом створення back-end системи веб-сервісу.

Для досягнгення поставленої мети потрібно вирішити наступні задачі:

1. Дослідити сучасні підходи до розроблення і впровадження системи електронної торгівлі (інтернет-магазин);
2. Провести аналіз архітектурних рішень і вибір програмних засобів для реалізації інтернет-магазинів;
3. Програмно реалізувати back-end систему для інтернет-магазину.

Об’єктом нашого дослідження є процес обробки замовлень інтернет-магазину.

Предметом є програмні засади створення back-end системи для інтернет-магазину.

Новизна роботи. Новизна полягає у тому, що запропонована у роботі backend система є розподіленою (двох-ядерною), такою що не передбачає активного ведення товарного асортименту на сайті, а натомість спирається на вже наявні дані у обліковій базі даних і передбачає організацію ефективного асинхронного обміну між backend сайту та обліковою системою.

Перша частина - back-end система сайту реалізована на базі Node.js з використанням мови JavaScript, а інша як веб-сервіс BAF з використанням вбудованих класів для створення веб-сервісів.

РОЗДІЛ 1
ОГЛЯД СУЧАСНИХ ПІДХОДІВ ЩОДО ПРОЕКТУВАННЯ І ВПРОВАДЖЕННЯ СИСТЕМИ ЕЛЕКТРОННОЇ ТОРГІВЛІ

1.1 Загальна характеристика системи електронної торгівлі

Електронна торгівля – це ведення торгово - закупівельної діяльності через мережу Інтернет. Поняття електронна торгівля включає в себе : замовлення, оплата, доставка товарів або послуг лише через онлайн ресурси. Розвиток глобальної мережі інтернет та IT-технології в світі стимулювало стрімке зростання електронної торгівліт [14].

Згідно прогнозів, число онлайн-покупців в світі у 2021 році досягне 2,14 млрд осіб, обсяг інтернет-продажів становитиме біля $ 6 трлн. і всього на електронну торгівлю буде доводиться 18,1% від усіх роздрібних продаж в світі. За оцінками компанії EVO, в 2020 році споживачі українського ринку придбали через мережу інтернет товарів та послуг на суму 107 млрд грн. В порівнянні з 2019 роком обсяги продажу зросли на 41%., при цьому і на 50% зросла кількість онлайн-оплат. На сьогодні майже 9% всіх закупок споживачі роблять через інтернет-магазини та соціальні мережі. [10]

На рисунку 1.1 можна побачити як стабільно і впевнено прогнозується розвиток електронної торгівлі. Споживачі вибирають онлайн торгівлю з наступних причин:

- можливість детально вивчити інформацію в мережі перед великими покупками (81%);
- можливість порівняти ціни на товари в онлайн-додатку (34%);
- акції та знижки (41%);
- безкоштовна доставка (53%);
- простота повернення (33%);
- можливість прочитати відгуки (35%);
- економія власного часу (30%);
- швидкість оформлення замовлення (30%).

Рисунок 1.1 – Обсяг продажів в інтернет-торгівлі в світі

Переваги електронної торгівлі зі сторони комерційних структур:

- залучення великої кількості покупців;
- відсутні витрати на оренду приміщень;
- рівноправний доступ до ринку;
- великі можливості для аналізу ринку;
- зменшення витрат на рекламу;
- оперативність;
- зниження собівартості товарів та послуг;
- доступ до нових ринків збуту;
- персоналізація взаємодії;
- підсилення конкурентних позицій;
- створення нових продуктів та послуг;
- використання різних платіжних засобів.

Однак, слід зауважити, що в системі електронної комерції є й певні труднощі, зокрема, це питання безпеки як на технічному так і законодавчому рівні. Проблеми оподаткування електронного бізнесу, відсутність загальних правил та норм, узгоджених усіма сторонами, що беруть участь у розробці веб-ресурсів, нестача висококваліфікованих фахівців, недостатній економічний потенціал, недовір'я простого користувача. Також, на сьогодні держава повинна робити більші інвестиції в освіту, високі технології, кібербезпеку та інфраструктуру.

Учасників електронної торгівлі можна поділити на п'ять категорій: бізнес-бізнес; бізнес-споживач; бізнес-адміністрація; споживач-адміністрація; споживач-споживач.

Бізнес-бізнес (B2B). Найбільша система електронної торгівлі за обсягами оборотних коштів, де учасниками є комерційні структури.

Бізнес-споживач (B2C). Друга за активністю система. Це є взаємовідносини інтернет-магазину з користувачами.

Бізнес-адміністрація (B2A). Молода форма комерційних відносин між бізнесовими структурами та управлінськими органами.

Споживач-адміністрація (C2A). Взаємодія між споживачами та державними структурами (податкова та соціальні сфери).

Споживач-споживач (C2C). Комерційні відносини між фізичними особами безпосередньо.

Система B2B є найбільш перспективною і її можна розділити на такі типи: Корпоративний сайт компанії, інтернет-магазин, брокерські сайти, інформаційні сайти, служба закупівель, електронні торгові майданчики, інформаційні портали, електронні біржі. Призначений для спілкування через Інтернет даної компанії з партнерами та контрагентами - постачальниками і споживачами. Сайт, як правило, містить інформацію про компанію, її керівництво, персонал, ну і, звісно, каталоги пропонованих нею видів продукції та послуг.

Оплату покупки при електронній торгівлі можна здійснювати кредитною карткою, поштовим переказом, з рахунків юридичних осіб, переказ через будь-який банк, готівкою курєру, із застосуванням криптовалюти, післяплата.

Загальну структуру електронної торгівлі можна представити наступним чином. (рис. 1.2).

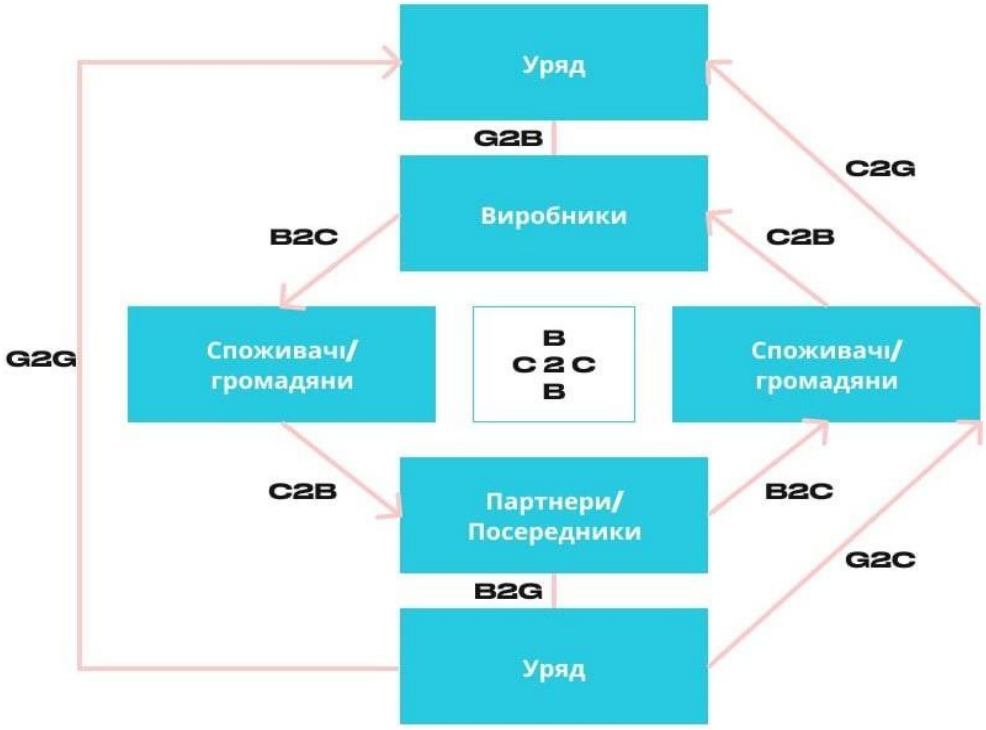

Рисунок 1.2 – Схематична структура систем електронної торгівлі

1.2 Тенденції розвитку системи електронної торгівлі

Незважаючи на пандемію, яка огорнула весь світ, ринок електронної комерції активно розвивається, стає більшим і різноманітнішим, як ніколи раніше. До цього спонукає стрімкий ріст новітніх технологій в світі. Які ж тенденції розвитку нам очікувати.

1. Ріст міжнародних роздрібних продаж. Обсяг роздрібних продажів в електронній торгівлі постійно зростає. За деякими даними до 2021 року на електронну торгівлю припадатиме близько 18,5% від загального обсягу роздрібної торгівлі в світі. А тому, сьогодні бізнес-структурам слід більшу увагу

приділити реформуванню інфраструктури під онлайн торгівлю. Підприємства, які вже на цьому ринку, повинні, певно, приділити більшу увагу новітнім технологіям у веб-розробках, для просування свого бренду.

2. Використання багатоканального просування товарів і послуг. Доведено, що чим більше магазинів відвідає людина, тим більше покупок вона зробить. Тому, ми бачимо, що з'являються нові соціальні мережі, розвиваються старі і чим більше каналів будуть використовувати покупці, тим більший оборот продажів. Проаналізовано, якщо користувач відвідає чотири торгових канали в співвідношенні до одного, то в середньому він зробить покупок на 9% більше. З іншої сторони, ви матимете кращу уяву про покупця, його потреби, а отже, як правильно просувати продукти на ринку та використовувати рекламний бюджет.

3. Популярність шопінга в соціальних мережах. Все більше охоплюють користувачів соціальні мережі. Якщо ми раніше мали змогу лише бачити рекламні ролики, рекламні оголошення, то тепер користувачі мають змогу без зайвих витрат і швидко купувати товари на самих популярних платформах. Зокрема, у Facebook, Twitter, Instagram вже є можливість використовувати каталоги товарів, що дало змогу збільшили продажі онлайн. Так, Instagram розробив функцію «shoppable post», яка дає можливість структурам відзначати товари стікерами в історіях або тегами (shopping tags) в постах . Коли користувачі натискають на тег товару або стікер в історії, їм відображається: зображення товару, ціна, коротка характеристика та, за допомогою натискання кнопки, можливість відразу придбати товар. Такий інструмент і приваблює користувачів і значно скорочує їх час.

4. Ядро сфери електронної комерції зміщується з Західної півкулі. Одним із факторів такого зміщення є і вдосконалення технологій та інфраструктури інших регіонів. За даними досліджень обсяги Сполучених Штатів в загальному світовому роздрібному ринку лише в минулому році скоротилися на 16,9%.. А отже, до електронної комерції бізнес-структури повинні застосовувати міжнародний підхід, аби забезпечити глобальну доступність для клієнтів у всьому світі. Вихід потрібно шукати у застосуванні новітніх технологій в

проектуванні і розробці веб-ресурсів для охоплення партнерів та просування продукту в інші регіони.

5. Розвиток міжнародного шопінгу. Як вказує статистика, сьогодні близько 57% онлайн-покупців здійснювали покупки онлайн в зарубіжних інтернет-магазинах. [22]

6. Ріст онлайн-торгівлі в B2B системі електронної торгівлі. Це вказує на те, що сектору B2B потрібно рухатися в сторону моделі B2C. Сьогодні аби купити товар, чи здійснити інші торгові операції за моделлю B2B, слід здійснити не один юридичний етап. А тому взаємодію продавець-покупець слід звести до мінімуму.

7. Персоналізація в електронній торгівлі стане стандартом. Покупці стають вимогливішими через збільшення ринку онлайн-торгівлі, та потребують до себе більшої уваги. За даними статистики, близько 33% користувачів припинили свої відносини з компаніями, де вони відули відсутність належного персонального ставлення до них. Персоналізація сьогодні складає цілу маркетингову стратегію. Це відслідковування історії покупок, його поведінка в соціальних мережах, складання різних пропозицій покупцю, нарахування бонусів та багатьох інших даних, що мають відношення до інтернет-торгівлі.

8. Ріст покупок з мобільних пристроїв. Покупці не хочуть тратити зайво свій час, хочуть робити покупки миттєво і вже. Вони не хочуть ,бути залежними від комп’ютера, а за допомогою мобільних пристроїв можуть переглядати і купувати товари будь де.

9.Голосова комерція на піку популярності. Нова технологія голосового управління, яка дала нові можливості для компаній, щоб продавати і розвивати свій бізнес . Впровадження голосових інструментів, таких як Google Home, Amazon Echo відкрило новий канал для взаємодії людей з продуктами.

10. Поява нових способів оплати. Наявність різних варіантів оплати є одним з тих факторів, що сприяють закупам через інтернет-мережу. Покупці хочуть зручності, мобільності, оперативності, а тому слід звернути увагу на впровадження нових форм оплати. Наразі, такі сервіси, як Google Pay, Apple,

Paypal або Samsung Pay. роблять процес здійснення замовлення зручнішим.

Електронна комерція - це постійно зростаюча сфера. З ростом купівельної спроможності споживачів у всьому світі, поширенням соціальних мереж і постійно розвиваючою інфраструктурою і технологіями майбутнє електронної комерції в наступні роки стане ще більш динамічним.

1.3 Класифікація сайтів та їх особливості

Для того, аби вдало просувався твій бізнес в інтернет мережі, перш за все потрібно мати уяву, як має виглядати твій сайт, які функції він має виконувати і, відповідно, яку структуру побудови обрати. Приведемо коротку характеристику сайтів. Основним критерієм класифікації сайтів зазвичай обирають вид діяльності. Тому, усі сайти можна класифікувати як представлено на рисунку 1.3.

Рисунок 1.3 – Типи сайтів

Сайт-візитка говорить сам за себе. Це не великий за обсягом сайт, який містить, зазвичай, основну інформацію про компанію: контакти, чим займається, місце знаходження, зворотній зв'язок. Користуються таким сайтом малі структури та фізичні особи.

Корпоративні сайти. Здебільшого дизайн такого сайту зроблений на основі корпоративного стилю з використанням логотипу. Окрім загальної інформації містить інформаційні сторінки, новини, конференції, системи взаємодії з партнерами.

Інтернет-вітрини. Це як прайс-лист на папері, лише електронний. Зручний пошук, немає обмежень на інформацію, можна перегляну, скачати та роздрукувати.

Інтернет-магазин. Як традиційний магазин, але із значно більшою оперативністю. Можна переглянути товар, ціни, опис товару і відразу здійснити або почати покупку через інтернет. Характерним для інтернет-магазинів є те, що покупець має значно більший вибір товарів та послуг, ніж реальні магазини. Переваги: Економія на складських приміщеннях, економія власного часу. Такі сайти використовують від малого до великого бізнесу.

Промо-сайти. Це рекламно-іміджевий сайт, який націлений привернути якомога більшу аудиторію до просування певного продукту на ринку. Такі сайти, зазвичай, маленькі цікаві та дизайнерські.

1.4 Етапи створення сайтів

Для створення якісного сайту необхідна продумана і відповідальна робота на кожній стадії її проектування, в якій задіяна група професіоналів. Кожен з проектів проходить наступні етапи розробки:

- Проектування сайту;
- Дизайн сайту;
- Верстка сторінок і шаблонів;
- Програмування;
- Наповнення;

- Тестування;
- Перенесення на хостинг.

Самий відповідальний етап – це перший, де закладається структура веб-ресурсу. Це дуже тісна співпраця фахівців із замовником. Перш за все слід чітко окреслити цілі та завдання сайту, яка в нього мета, які результати очікування, визначитися з типом сайту. Замовнику сайту важливо розуміти свою цільову аудиторію, їх інтереси та потреби. Чим більше ваш сайт буде наближений до інтересів потенційних споживачів, тим успішнішим буде ваш бізнес. Якщо користувач буде знаходити максимально відповіді на свої проблеми, тим більший шанс, що він поділиться ще й з іншим користувачем через соціальні мережі чи інший канал, тим більша конкурентноспроможність вашого ресурсу на ринку. На базі цього аналізу складається технічне завдання, де в письмовому вигляді детально виписуються вимоги замовника, узгоджується кожен аспект з окреслених робіт, усуваються неточності.

Дизайн сайту. Цей етап відповідає за візуальну складову ресурсу і є одним з найважливіших етапів проектування. Перше, що кидається у вічі користувачу потрапивши на сайт – це його оформлення. Він має бути і приємним, і зручним. А тому враховувати потрібно кожну деталь: колірну гамму, розмір шрифту, взаємне розташування елементів та багато іншого. Другий, не менш вагомий аспект- зручність і легкість користування сайтом. Особливо має бути зручна і надійна навігація по сторінках.

Верстка сторінок і шаблонів. Після узгодження дизайну проекту, HTML-верстальник створює робочий проект сайту. При цьому слід врахувати, щоб веб-ресурс був адаптований під будь-який браузер і пристрій.

Програмування. На цьому етапі програміст пише код з нуля, або за допомогою CMS – системи управління сайтом. Від якісної програмування у великій мірі залежить як працюватиме і індексуватиметься пошуковими системами сайт.

Наповнення. Ресурс наповнюється відповідним матеріалом – це статті, малюнки, відео, фотографії. Після чого відповідальний менеджер за контент

проводить внутрішню оптимізація сайту.

Тестування. На завершальному етапі перевіряють всі етапи розробки, тому що, як завжди, бувають якійсь недоречності. Зазвичай крім розробників на цьому етапі присутні дизайнер, користувач та власник.

Перенесення на хостинг. Хостинг - це послуга, яка забезпечує зберігання інформації на фізичному сервері. Завдяки хостингу ваш сайт стає доступним для користувачів. Наданням спектру послуг, необхідних для повноцінного та безперебійного функціонування сайту, займаються компанії-провайдери.

1.5 Веб-сервіси обміну даними

Сьогодні ми бачимо велике різноманіття веб-ресурсів, кожен з яких може працювати на різних апаратно-програмних платформах, використовувати різні технології, мови програмування. Виникла проблема обміну даними, що слугувало появі веб-сервісів. Веб-сервіси - це технологія обміну даними між різними додатками через мережу. Іншими словами, це посилання на адресу, за якою ми можемо отримати дані або виконати певну дію. Основна відмінність веб-сервісу від інших технологій передачі даних – це стандартизованість. Наведемо найбільш поширені способи реалізації веб-сервісів:

- SOAP (Simple Object Access Protocol);
- REST (Representational State Transfer);
- XML – RPC (Remote Procedure Call).

За протоколом SOAP – обмін відбувається у формате XML. Це універсальний засіб для обробки всіх типів даних. Перевагами даного протоколу є наявність чіткої специфікації, широка підтримка в продуктах Microsoft, однозначність, але є певна складність в реалізації. [3]

REST являє собою архітектурний стиль, а не протокол. За допомогою методів HTTP відбувається керування даними. У REST-сервісах акцент зроблений на доступ до ресурсів, а не на виконання віддалених сервісів і в цьому їх основна відмінність від SOAP-сервісів. Даний сервіс простий в реалізації,

значна економія ресурсів, водночас неоднозначність в керуванні даними і менша захищеність.

XML-RPC протокол є більш пізньою версією SOAP з використанням XML.

Тому, для більш складної архітектури застосовується SOAP, простішої задачі – REST, а можливий варіант і два одночасно. Загалом, переваги веб-сервісів в тому, що вони створюють умови для взаємодії програмних компонентів не залежно від платформ, за рахунок XML формату забезпечується простота формування та налаштування веб-сервісів та HTTP гарантує взаємодію системи за допомогою міжмережевого доступу. [4]

Висновок до розділу 1

В першому розділі ми дослідили сучасні підходи до розроблення і впровадження системи електронної торгівлі (інтернет-магазин).

Проаналізувавши ринок електронної торгівлі, виявили найбільш поширені способи реалізації веб-сервісів на сьогодні, проаналізували сучасні етапи розробки сайтів.

Також вказано на основні тенденції розвитку , які варто взяти до уваги при проектуванні та впровадженні програмних систем електронної торгівлі.

РОЗДІЛ 2
АНАЛІЗ АРХІТЕКТУРНИХ РІШЕНЬ І ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ІНТЕРНЕТ-МАГАЗИНІВ

2.1 Мови програмування та платформи CMS для реалізації веб-ресурсів

В розробці інтернет-магазину є декілька етапів, які відповідають за певну його складову. На етапі програмування слід виділити дві складові:

Front-end. Це все, що користувач може побачити і з чим може взаємодіяти на сторінці. Дизайн, візуальні елементи, схеми – все, що є на сторінці нашого сайту. Завдання front-end розробника - зробити сайт, який буде вирішувати проблеми і завдання власника, однаково правильно працювати на всіх пристроях, незалежно від браузера і пристрою і при цьому буде комфортним користувачеві. Основні інструменти з якими працюють розробники - це HTML, CSS і JavaScript. Front-end розробники мають за мету створити для користувачів інтуїтивно зрозумілий і зручний інтерфейс. Основні їхні завдання, які візуально відображаються на сторінці сайту:

- верстка сторінок (шрифт і розмір тексту);
- випадаюче меню;
- слайдер, написаний за допомогою HTML, CSS і JavaScript.

Всі графічні елементи, розташовані на сайті, розробляються цими програмістами, які пізніше можна побачити в звичному для нас вигляді за допомогою браузера. Це результат роботи декількох спеціалістів: front-end розробника з дизайнерами та UX-аналітиками.

Back-end. Невидима частина сайту. Back-end розробник відповідає за те, що приховано від очей користувача і працює на сервері. Його завданням є створення бази даних і програми, які будуть оперувати даними і записувати їх в базу. Також він шифрує паролі і персональну інформацію, займається налаштуванням доступів і системи резервного копіювання даних, розробляє програми, що обробляють інформацію, невидиму користувачеві.

Back-end складова сайту включає:

- сервера;
- додатки;
- бази даних.

Вони потрібні для збору і розміщення основної інформації, реєстрування нових користувачів, фіксування замовлень покупців і створення заявок для зворотного дзвінка. Діяльність розробників базується на логіці та архітектурі програмного забезпечення, яке в свою чергу виступає як створення підсумкового результату у вигляді працездатного і надійного веб-сайта.

Є декілька мов програмування для back-end розробника: PHP, Ruby, Python або JavaScript/Node.js. Для back-end розробки також необхідні системи управління базами даних. Наприклад: MySQL, PostgreSQL, SQLite або MongoDB.

HTML

Це мова гіпертекстової розмітки. Вона існує для того, щоб правильно побудувати структуру і зміст сторінки. З нею веб-розробник працює в редакторі коду за допомогою тегів. Мова використовує розширення .html, воно показує браузеру, що всередині файлу знаходиться код веб-сторінки. Браузер розбирає його структуру, визначає взаємне розташування елементів і показує їх на сторінці нашого сайту. Веб-стандарти HTML описані в специфікаціях. Це головне джерело інформації і для браузерів, і для розробників.

CSS

Це мова, яка використовується для стилізації нашої розмітки. Вона дає змогу зробити текст потрібного нам кольору, вирівняти текст по певну сторону, округлити кути зображення, змінити шрифти, змінювати відступи.

JavaScript

JavaScript - мова програмування для front-end розробки. Він використовується всіма популярними браузерами. Кожен сайт, кожен веб-додаток, яким ми користувалися чи бачили в браузерах містить JavaScript-код. Мова, яку часто рекомендують новачкам в програмуванні, тому що вона досить

проста, але містить всі базові речі: об'єктно-орієнтовану модель і структури даних. [21]

JavaScript використовують, щоб оживити HTML-сторінки. Він дає змогу додати на сторінку візуально зручну обробку дій користувача або певні візуальні ефекти. Можливості застосування JavaScript обмежуються тільки креативністю розробників і підтримкою браузерів.

PHP

PHP - серверна мова програмування, на якій пишуть back-end розробники. Вона була розроблена спеціально для веб-розробки - написаний на ній код можна впровадити в HTML. Якщо ми виконуємо дію, де відкриваємо свою сторінку в соціальній мережі і вводимо логін та пароль, комп'ютер формує запит з нашими даними і відправляє на сервер. На стороні сервера PHP отримує інформацію з бази даних, яка теж лежить на сервері і формує вашу сторінку: передає на фронтенд ім'я, фотографію, заповнює поля. Результат формується у вигляді готового HTML і з’являється у нас в браузері. PHP не є складною мовою, але щоб побачити всі його можливості, потрібно вивчати і інші технології. Наприклад, роботу з базами даних: мова SQL, а також MySQL, PostgreSQL, SQLite або MongoDB.

База даних - це місце, де зберігаються дані сайту. Це можуть бути тексти сторінок, списки користувачів з їх логінами і паролями, каталоги продукції і багато іншого. На PHP пишеться серверна логіка для інтернет-магазинів, сервісів бронювання, а також великих проектів, якими користуються більшість людей. Наприклад: «Вікіпедія», «ВКонтакте» і «Фейсбук».

React

Це бібліотека на мові JavaScript, розроблена програмістами «Фейсбуку». ЇЇ використовують для розробки веб-додатків. Бібліотеки потрібні, щоб оптимізувати написання коду і не витрачати час на пошук помилок. У бібліотеці зберігаються готові рішення, які програмісти використовують для типових задач. React – є популярною та зручною технологією для розробників.

Для того щоб написати програму використовуючи React, потрібно знати не тільки HTML, CSS і JavaScript, а й вивчити бібліотеку окремо.

Важливим етапом запуску сайту електронної комерції - вибір платформи для інтернет-магазину. На вибір впливає безліч факторів: яка буде вартість будівництва сайту, наскільки сучасним буде магазин, чи буде він актуальним через п'ять років, чи буде хороший трафік з пошуку, і чи стане інтернет-магазин успішним в кінці кінців.

Зараз на ринку існує безліч платформ для створення інтернет-магазину. Різні компанії пропонують свої рішення - хмарні, коробкові, платні, безкоштовні. CMS - це система управління контентом. Правильно вибрана CMS допомагає власнику бізнесу спростити організацію і наповнення сайту. [15]

Одні з найпопулярніших CMS:

- безкоштовні - Magento, OpenCart, WordPress с WooCommerce, Joomla;
- платні - платна версія Magento, 1С-Бітрікс, BigCommerce, Shopify, 3DCart, Volusion;
- для невеликого бізнесу - WordPress с WooCommerce, CS-cart, OpenCart;
- для великих проектів - 1С-Бітрікс, BigCommerce, Magento.

Magento

Входить в список найпопулярніших cms в усьому світі, належить Adobe, має відкритий вихідний код, доступна в автономному та хмарному варіантах. Основний функціонал:

- відмінні аналітичні можливості (середня сума замовлень, аналіз хітів продажів, нові клієнти, оборот за весь період);
- управління наявністю товарів на складі;
- контроль замовлень, рахунків, доставок, транзакцій, статусу замовлення;
- можливість додавання плагінів (близько 1300 безкоштовних і 2700 платних);
- наявність візуального редактора з можливістю попереднього перегляду;
- захист від автоматичного заповнення форм.

OpenCart

Система управління контентом з відкритим кодом. Орієнтована виключно на електронну комерцію, тому ідеально підходить для ведення інтернет-магазину. CMS вийде скачати за кілька хвилин і відразу ж приступити до створення сайту.

Основний функціонал:

- розподіл прав доступу та модерація публікованих матеріалів;
- велика кількість опцій в розділі продажів (доставка, облік замовлень, повернення, виробники, запобігання фінансовій шахрайства);
- більше 3 тис. безкоштовних і 9 тис. платних плагінів;
- інструменти статистики (аналітика продажів, статистика відвідувань);
- резервне копіювання і відновлення даних;
- управління декількома магазинами з одного інтерфейсу адміністратора.

1С-Бітрікс

Популярна CMS, ідеального підходить для середнього та великого бізнесу.

Основний функціонал:

- управління структурою сайту, візуальний редактор;
- організація поетапної публікації матеріалів;
- пошук по всьому сайту і розділах, автоматична індексація контенту;
- управління замовленнями клієнтів, валютами і курсами валют, товаром на складах;
- автоматизована система розсилки новин, новинок;
- збір відгуків та пропозицій в каталог ідей. [7]

WordPress с Woocommerce

WooCommerce - це потужний і популярний плагін для електронної комерції, побудований на WordPress. Система має відкритий вихідний код, вона здатна перетворити веб-сайт WordPress в інтернет-магазин.

Основний функціонал:

- додавання необмеженої кількості товарів і фото;
- розширена сортування і фільтрація товарів;
- автоматизація податкових розрахунків;

- інтеграція з Google Analytics, MailChimp і Facebook;
- мобільний додаток для iOS і Android з управління магазином;
- підтримка багатомовності та цін в різних валютах;
- цілодобовий контроль робочого процесу з постійним доступом до всіх даних;
- поділ прав користувачів, створення бази контактних даних;
- відображення найбільш популярних продуктів, новинок, товарів з найбільш високим рейтингом.

Joomla

Безкоштовна платформа з відкритим вихідним кодом і можливістю модернізації ядра системи.

Основний функціонал:

- візуальний редактор для додавання нової інформації на сайт і відкладена публікація записів;
- вбудована система кешування, яка прискорює завантаження;
- настроюються форми зворотного зв'язку, голосування, пошук по сайту;
- захист від автоматичного заповнення форм;
- модерація публікованих матеріалів;
- відновлення видалених об'єктів;
- багатомовність.

BigCommerce

Сервіс для середнього та великого бізнесу, популярний в зарубіжному сегменті. Його часто порівнюють з Shopify, але даний конструктор для створення сайту пропонує трохи більше функцій за ті ж гроші.

Основний функціонал:

- системи управління запасами, доставкою і поверненням;
- поділ клієнтів за цінами, доступу до продуктів, рекламних акцій;
- панель Analytics Dashboard для збору аналітики на замовлення, клієнтам, маркетингу і т. д .;
- адаптація під мобільні пристрої;

- більше 70 видів знижок і акцій;
- настройка оптових цін для окремих клієнтів або груп;
- вбудована інтеграція з ShipperHQ для точного розрахунку вартості транскордонних перевезень.

Shopify

CMS для інтернет-магазинів, підходить для малого і середнього бізнесу. Для великих проектів є Shopify Plus.

Основний функціонал:

- імпорт клієнтської бази;
- статистика з продажу, платежів, податкових ставок;
- інвентаризація, групи товарів;
- подарункові купони, промокодом;
- відновлення перерваної покупки;
- шаблони email-розсилок;
- щоденне резервне копіювання.

2.2 Огляд баз даних для використання у веб-додатках

Робота з базою даних - невід'ємна складова будь-якого інтернет ресурсу. Цей інструмент дозволяє створювати, редагувати та керувати великими обсягами даних. На основі бази даних ми створюємо сайти використовуючи такі мови програмування як: PHP, HTML, CSS і ін., а також спеціальні системи управління – CMS: WordPress, OpenCart, Joomla та ін. Сьогодні неможливо уявити великий інтернет-магазин без підтримки бази даних.

База даних – являє собою сукупність елементів, систематизованих таким чином, аби ми могли легко знайти і обробити їх за допомогою електронної обчислювальної машини. Система управління базами даних (СКБД) – являє собою сукупність програмних засобів, які використовують для управління створенням і використанням баз даних. Тобто, база даних – це наша інформація, дані, з якими взаємодіє система управління базами даних (СУБД) - програмний продукт, що забезпечує такі можливості як: створення, наповнення, модифікацію і пошук по базах даних. [11]

Вимоги до СУБД:

- надійність: безпека даних, мінімізація збоїв і втрат даних, захищеність бази даних від несанкціонованого доступу, можливість шифрування даних, регулярне резервне копіювання баз даних і можливість відновлення з архіву за необхідності;
- продуктивність: СУБД, як рішення покладених на неї завдань;
- впевненість в підтримці виробника в разі якогось серйозного збою або складній ситуації.

Способи доступу до БД:

- Клієнт-серверні СУБД;
- Файл-серверні СУБД;
- Вбудовувані СУБД.

У клієнт-серверних СУБД (Microsoft SQL Server, Oracle, Firebird, PostgreSQL, InterBase, MySQL та ін.) вся обробка даних ведеться в одному місці, на сервері, в тому ж місці, де зберігаються (зазвичай) дані. До файлів даних має доступ тільки один сервер, одна система - це сама СУБД. Додатки-клієнти посилають запити на обробку та отримання даних з СУБД і отримують відповіді. Додатки-клієнти не мають безпосереднього доступу до файлів даних. Всі промислові СУБД на даний момент є саме клієнт-серверними. У файл-серверних СУБД (Paradox, Microsoft Access, FoxPro, dBase і ін.), навпаки, додатки мають загальний доступ до всіх файлів бази даних (що зберігається зазвичай у якомусь файловому сховищі) і спільно обробляють ці дані. Кожна програма самостійно обробляє дані.

На даний момент файл-серверна технологія вважається застарілою, а її використання у великих інформаційних системах - недоліком. Проблема в тому, що файл-серверні СУБД не мають багатьох переваг клієнт-серверних, таких як кешування даних, паралелізм запитів, висока продуктивність і мають ряд недоліків (складності з підтриманням цілісності бази, відновленням, блокуваннями і т.д.), що призводить в свою чергу до певного падіння надійності і продуктивності. Стан бази в файлових СУБД необхідно постійно моніторити і

проводити операції по її «лікування» за допомогою вбудованих або сторонніх утиліт.

Вбудовувані СУБД (SQLite, Firebird Embedded, Microsoft SQL Server Compact і ін.) Поставляються в складі готового програмного продукту, не вимагаючи процедури самостійної установки. Призначені для локального зберігання даних програми, але не розраховані на колективне використання в мережі. Вбудована безкоштовна СУБД SQLite широко використовується в відомої мобільної ОС Android, яка була розроблена компанією Google, і в багатьох мобільних додатках.

Популярні системи управління базами даних:

Oracle

Oracle RDBMS є на першому місці серед СУБД. Система є популярною у розробників, адже вона легка у використанні, у неї проста документація, підтримка довгих найменувань, JSON, покращений тег списку і Oracle Cloud. До особливостей входить: обробка великих даних, підтримкп SQL, доступ з реляційних БД Oracle, Oracle NoSQL Database з Java / C API для читання і запису даних.

MySQL

MySQL падтримує Linux, Windows, OSX, FreeBSD і Solaris. На початку розробники можуть працювати з безкоштовним сервером, а після перейти на комерційну версію. Ліцензія GPL з відкритим вихідним кодом дає можливість модифікувати ПЗ MySQL. В основі системи управління базами даних MySQL лежить звичайна форму SQL. Програми для проектування таблиць побудовані на інтуїтивно зрозумілому інтерфейсі. MySQL підтримує до 50 мільйонів рядків в таблиці. Розмір файлу заданий за замовчуванням 4 ГБ, але його можна збільшити. Підтримує секціонування і реплікацію, а також Xpath і процедури, тригери та подання. До особливостей входить: масштабованість, легкість використання, безпека, підтримка Novell Cluster, швидкість, підтримка багатьох операційних систем.

Microsoft SQL Server

Одна з найпопулярніших комерційних СУБД. Вона працює на Windows, це перевага, якщо ви використовуєте продукти Microsoft. І графічний інтерфейс, і програмне забезпечення побудовані на командах. Підтримує SQL, як мову маніпулювання даними. До особливостей входить: хороша продуктивність, прив’язаність до платформи, можна встановити різні версії на одному комп'ютері, генерація скриптів для переміщення даних.

PostgreSQL

Масштабна об'єктно-реляційна база даних, яка працює на Linux, Windows, OSX і деяких інших системах. До особливостей входить: підтримка табличних просторів, а також збережених процедур, об'єднань, уявлень і тригерів, відновлення на момент часу (PITR), асинхронна реплікація.

MongoDB

Найпопулярніша NoSQL система управління базами даних. Найкраще підходить для динамічних запитів і визначення індексів. Гнучка структура, яку можна модифікувати і розширювати. Підтримує Linux, OSX і Windows, але розмір БД обмежений 2,5 ГБ в 32-бітних системах. Використовує платформи зберігання MMAPv1 і WiredTiger. До особливостей входить: висока продуктивність, автоматична фрагментація, робота на декількох серверах, підтримка реплікації Master-Slave, дані зберігаються в формі документів JSON, можливість індексувати всі поля в документі, підтримка пошуку за регулярними виразами.

2.3 Огляд платформи Business automation framework

Business Automation Software це програмний продукт, сучасне рішення, яке забезпечує автоматизацію всіх процесів на підприємстві, яка є абсолютно новою програмою, яка розроблена виключно для українського ринку. Вона є заміною вже застарілого софта. Зараз повільно здійснюється перехід з 1С: Підприємство на BAS. Платформа Business Automation Framework (BAF) - це предметно-орієнтоване середовище розробки для вирішення завдань, пов'язаних з процесом автоматизації підприємства. Простою мовою платформу можна назвати основою і серцем системи. Саме на ній розробляються і запускаються конфігурації. Зараз

популярні такі конфігурації, як BAS, наприклад, BAS ERP або BAS Комплексне управління підприємством. Проте потрібно знати, всі конфігурації працюють на платформі BAF. Розробник встановлює платформу один раз, а кількість установок конфігурацій є не обмежена.

Область застосування

Завдяки гнучкості платформи ми можемо використовувати рішення BAS для автоматизації обліку і управління компанією на підприємствах будь-якої спрямованості - виробничих холдингах, бюджетних організаціях, підприємствах сфери обслуговування, оптової та роздрібної торгівлі і багатьох інших. [25]

Рішення з лінійки BAS забезпечують:

- ведення бухгалтерського, податкового та управлінського обліку;
- можливість оперативного управління компанією;
- формування аналітичної звітності та аналіз роботи підприємства;
- вирішення фінансових завдань, а саме планування, бюджетування і руху коштів;
- управління персоналом та розрахунок заробітної плати;
- управління взаємовідносинами з клієнтами та партнерами;
- автоматизацію господарської діяльності;
- і інші завдання.

Гнучкість платформи дозволяє продуктам BAS легко допрацьовуватися під специфічні завдання будь-якого бізнесу, завдяки чому ми можемо отримати ідеальне рішення для комплексної автоматизації бізнесу.

До основних переваг платформи можна віднести:

- кроссплатформенность і роботу з різними СУБД;
- гнучкість платформи і простоту настройки;
- масштабованість і продуктивність системи;
- просту інтеграцію з зовнішніми програмами і обладнанням;
- зручний призначений для користувача інтерфейс;
- засоби швидкої розробки додатків;
- безпеку системи;

- зовнішнє джерело даних;
- можливість роботи через інтернет.

Сучасний і зрозумілий дизайн інтерфейсу забезпечує високу швидкість роботи і легке освоєння системи для початківців користувачів. Існує можливість масового введення інформації, оформлення робочого простору під потреби користувача, повнотекстовий пошук в даних і створення звітів будь-якої складності. Вбудовані інструменти формування звітів і друкованих форм дозволяють формувати різні звіти, деталізувати і групувати в них інформацію, створювати різні види діаграм та зведені таблиці для аналізу даних. Масштабованість системи дозволяє працювати в розрахованому на багато користувачів режимі, забезпечуючи паралельну роботу в програмі великої групи користувачів.

Система включає в себе зручні інструменти для адміністрування:

- конфигуратор;
- механізми розпізнавання користувачів;
- списки інформаційних баз і користувачів;
- журнал реєстрації;
- робота з інформаційними базами (настройка параметрів, вивантаження, завантаження, тестування, виправлення помилок);
- оновлення конфігурації;
- і багато іншого.

2.4 Платформа для створення back-end систем сайтів Node.js

Node.js - це серверна платформа, розроблена на JavaScript Engine Google Chrome (V8 Engine). В її основі, крім інших рішень, лежить принцип відкритого коду. Офіційне визначення Node.js - це платформа, створена на основі виконання JavaScript на Chrome для легкої розробки швидких та масштабованих мережевих додатків.

Node.js передбачає модель неблокуючого вводу-виводу, з управлінням подій, що робить її простою та ефективною, вона є ідеальною основою для

додатків у режимі реального часу, які працюють на розподілених пристроях. В основі додатків Node.js лежить JavaScript і вони виконуються в середовищі Node.js в OS X, Microsoft Windows та Linux. Node.js також забезпечує велику бібліотеку різних модулів JavaScript, що значно спрощує розробку веб-додатків за допомогою готових рішень.

Особливості платформи Node.js

Завдяки деяким важливим функціям, Node.js є популярним вибором для системних архітекторів програмного забезпечення. Наприклад:

- Відкритий вихідний код;

  Node.js - це платформа з відкритим вихідним кодом. Це означає, що правовласник надав різні права на вивчення, редагування і поширення програмного забезпечення будь-кому будь-якою мовою.
- Висока масштабованість;

  Оскільки він використовує механізм подій, Node.js має високу масштабованість і допомагає серверу в неблокуючій відповіді.
- Швидкість;

  Оскільки Node.js побудований на движку JavaScript Google Chrome V8, його бібліотеки дуже просунуті а, отже, здатні виконувати код з більшою швидкістю.
- Немає буферизації;

  Node.js наділений спеціальною функцією, тобто він не буферизує ніякі дані.
- Однопотоковий;

  Оскільки він використовує процес зациклення подій, Node.js може слідувати однопоточній моделі. Це допомагає одному користувачеві обробляти більше одного запиту.
- Асинхронний;

  Node.js має асинхронні бібліотеки. Це дуже корисно, оскільки серверам Node.js не потрібно чекати, поки API відправить відповідь, і перейде до наступного API.

Є популярні області для використання Node.js. Наприклад: програми з введенням / виведенням, програми потокового передавання даних, інтенсивні програми в режимі реального часу (DIRT), додатки на основі JSON API, односторінкові програми. Для цього найчастіше використовуються такі фреймворки на Node.js, як: Hapi.js, Express.js, Nest.js, Koa.js, Socket.io, Meteor.js, Adonis.js, Sails.js та інші. Більш детально про деякі з них:

Hapi.js

Hapi.js - один з найпростіших, безпечних і надійних фреймворків, з яким працюють багато розробників. Ви можете використовувати Hapi.js для створення масштабованих і надійних додатків з мінімальними накладними витратами і готової функціональністю. Hapi.js ідеально підходить для розробки безпечних, real-time, масштабованих і соціальних медіа-додатків. Більшість розробників мобільних додатків воліють Hapi.js для створення проксі-серверів і API-серверів.

Express.js

Express.js є гнучким і мінімалістичним фреймворком додатків. Він не побудований навколо конкретних компонентів, що дає розробникам можливість експериментувати. Вони отримують миттєве налаштування і чистий досвід роботи з JS, що робить Express.js сильним суперником в ніші швидкого прототипування і гнучкої розробки. Express.js ідеально підходить для швидкого створення веб-додатків і сервісів, так як має легкодоступні інструменти генерації API. Це частина заснованої на JavaScript технології MEAN software stack, яка дозволяє використовувати Express.js для створення будь-якого корпоративного браузерного додатку. [8]

Nest.js

Nest.js - серверна платформа, створена для підтримки продуктивності розробників і полегшення їхнього життя. Програмісти здебільшого використовують фреймворк для структурування коду. Nest.js використовується для написання масштабованих, тестових і слабо пов'язаних додатків. Він забезпечує правильний баланс структури і гнучкості, щоб ефективно управляти кодом у великих проектах.

Koa.js

Koa.js - відкритий веб-фреймворк, написаний розробниками Express.js. За допомогою Koa вони прагнули створити меншу і більш надійну платформу для веб-додатків і API. Він надає широкий спектр ефективних методів для прискорення процесу створення серверів. Використовується для створення серверів, маршрутів, обробки відповідей і помилок. Даний фреймворк досить зручний для командної розробки.

Socket.io

Socket.io використовується для установки двонаправленого з'єднання у реальному часі між клієнтом і сервером. Щоб все працювало, на клієнті в браузері і на сервері повинен бути Socket.io. Це дозволяє обмінюватися даними в мільйонах форм, але найкращим методом залишається JSON. Це один з кращих двонапрямлених інструментів комунікації в реальному часі на основі подій. Він повинен використовуватися, коли необхідно додати в додаток функцію аналізу в реальному часі. Socket.io корисний для ігор в реальному часі. Використання базових протоколів HTTP або HTTPS для ігор недоцільно, тому що ці файли громіздкі і вимагають певного часу для встановлення з'єднання.

Висновок до розділу 2

У другому розділі ми проаналізували які є архітектурні рішеня і вибір програмних засобів для реалізації інтернет-магазинів.

Ми дослідили мови програмування та платформи CMS для реалізації веб-ресурсів, оглянули бази даних, які використовують у веб-додатках, проаналізували платформи Business automation framework та платформу для створення back-end систем сайтів Node.js

Провівши аналіз програмних засобів для програмної реалізації back-end системи для інтернет-магазину, можна зробити висновок, що платформа Node.js є хорошим і популярним рішенням для такої реалізації. Для реалізації клієнтської та серверної частини – мова програмування JavaScript. Для обміну даними між частинами розподіленої back-end системи хорошим рішеням є веб-

сервіс SOA. Робота з такими веб-сервісами гарно підтримується як в Node.js так і в платформі BAF.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ BACK-END СИСТЕМИ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ

3.1 Вибір програмної реалізації архітектури інтернет-магазину

Можлива програмна архітектура веб-додатку інтернет магазину буває двох видів :

Одно-ядерна (монолітна) архітектура. Це найбільш популярна та найбільш розповсюджена на даний час схема побудови веб-додатку. Вона характерна для більшості систем керування контентом (CMS) для інтернет магазинів таких, як shop-script, phpshop, joomla, opencart, wucommerce. У цьому рішенні присутня одна база даних та один набір скриптів.

Двох-ядерна (розподілена) архітектура. Менш відома широкому загалу розробників. Відмінність її полягає у тому, що фактично існує дві бази: одна так як і у одноядерній архітектурі розміщена на сервері разом з сайтом і служить лише для ослуговування покупців. інша розміщена на іншому сервері і з нею працює персонал підприємства, в тому числі менеджери, що ведуть асортимент та ціни інтернет-магазину. Ця база асинхронно реплікує підмножину своїх даних у першу базу, забезпечуючи інформацію про статуси замовлень, товари, залишки та ціни, наприклад, з іншого боку перша база теж у асинхронному режимі віддає інформацію про замовлення у другу. Варіантів цієї реплікації може бути кілька, залежно від обраної технології передачі даних. Для другої бази, для роботи персоналу, зазвичай, існує окремий додаток, який працює не у браузері, а у віконному інтерфейсі, що дозволяє значно підвищити зурічність та продуктивність роботи менеджерів. Таким чином, у цій архітектурі, окрім звичного веб-додатку на CMS, існує ще й додаток для персоналу підприємства. Очевидним висновком, чому така архітектура менш розповсюджена – значно вища вартість впровадження та володіння для двох програмних систем, замість однієї. Однак через переваги, які дуже значні, а саме, - вища продуктивність роботи персоналу, краща захищеність основних даних, значний запас для масштабованості, це рішення використовується і набирає популярності.

Переваги та недоліки кожної з архітектур :

Одно-ядерна архітектура.

- Перевагами є відносно недорога розробка такого комплексу ПО і як наслідок, низька ціна, враховуючи наявність цілком робочих безкоштовних варіантів. Нормальний багатокористувацький режим роботи, можливість швидко і недорого доопрацювати систему.
- Недоліками є низька продуктивність управління товарною базою, низька зручність роботи менеджерів, велике навантаження на сервер. Навіть в досить вдалих реалізаціях, де широко використовуються Java, Ajax та інше - все одно інтерфейс більш скутий і повільніший у порівнянні з класичними обліковими системами. Групові операції з товарами, часто вимагають отримання великого обсягу даних з сервера і тому дуже проблематичні. Передавати мегабайти (десятки мегабайт) інформації не завжди найкраще рішення. При таких ресурсних серверних операціях можуть навіть виникати проблеми і з доступом покупців до сайту.

Двох-ядерна архітектура.

- Перевагами є захищеність і продуктивність роботи з базою товарів. При збережені даних на сервері у локальній мережі або прямо на комп'ютері власника магазину, можна виконувати будь-які, найбільш ресурсоємні операції над усіма товарами, не боячись проблем продуктивності, що призводять до втрати замовлень. Можна отримати необмежену швидкість роботи інтерфейсу з базою і всі супутні переваги. Є можливість приймати та обробляти замовлення в разі відсутності мережі інтернет. Можна займатися наповненням магазину за відсутності зв'язку з інтернетом. Друга база, що не пов'язана з інтернетом, фактично недоступна для хакерів.
- Недоліками є більша складність розгортання та більша вартість як рішення, так і володіння.

Як очевидно з вищесказаного двох-ядерна архітектура є переважною у реалізації у технічному плані. Якщо ж зважати на недоліки складності та більшої вартості, то слід врахувати не досить абстрактну описану ситуацію, що

передбачає впровадження обох частин системи «з нуля», а реальну ситуацію на ринку. Такий аналіз дозволить запропонувати найбільш релевантне до ситуації та ефективне архітектурне рішення.

Щодо ситуації на ринку, то вона вже визначена природою самої діяльності торгових підприємств і обліковою політикою прийнятою до вжитку на ринку. А саме, роботи з обробки замовлень, ведення товарів, цін та товарних запасів обліку відвантажень, закупівлі, тощо, вже ведуться у умовно «других» базах даних, оскільки є лише частиною усіх облікових процесів на підприємстві. Таким чином впроваджувати їх з нуля сенсу немає.

Панівна більшість облікових систем є впровадженнями рішень на базі BAF/1C. За деякими даними частка таких рішень на ринку України сягає 98%. Ігнорувати цей факт нерозумно, ба більше того - економічно та технологічно невиправдано.

Економічно – тому що, якщо спертися на вже існуючу інсталяцію рішення на BAF, як на другу базу двох-ядерної архітектури, то це знімає необхідність створення і впровадження складної системи з нуля, а отже знижує вартість впровадження на декілька порядків. Такі рішення є найбільш продуктивними і з точки зору організації праці менеджерів. Менеджери вже працюють у цій системі і скорше за все система була адаптована під їхні вимоги спеціалістами з BAF. Що може бути краще за налагоджено роботу, яка не потребує кардинальних змін ?

Окрім того при роботі в одно-ядерній схемі і наявній впровадженій обліковій системі менеджер повинен заводити інформацію про асортимет у двох системах, а саме на сайт, використовуючи адміністративний інтерфейс і у облікову ситему само собою. Отже роботи додається, що принципово знижує ефективність роботи менеджерів порівняно до двох-ядерної архітектури, що має налаштований обмін між базами .

Технологічно – тому що платформа BAF містить у собі ряд переваг, що дозволяє технологічно нескладно вбудувати її у двох-ядерну архітектуру веб-додатку, а саме :

- вбудований стек технологій для роботи в мережі, для прикладу - веб-сервіси та веб-посилання SOA, HTTP-сервіси, підтримка роботи з даними у форматах XML та JSON;
- клієнт-серверна архітектура роботи, що дозволяє підтримувати постійну доступність та працездатність веб-сервісу.

При роботі у двох-ядерній архідектурі навантаження на базу і back-end систему CMS сайту значно падає, оскільки у ній не працюють менеджери не вводять і не закачують товари, то більшість ресурсів системи задіяні лише у процесах обслуговування клієнтів і це, таким чином справляє позитивний економічний ефект через покращення обслуговування клієнтів та теоретичне збільшення продажів.

Слід також сказати, що двох-ядерна архітектура побудови інтернет магазину гарно масштабується, таким чином, що зміна CMS для сайту чи додавання ще одного сайту майже не зачіпає функціонал другої бази, а саме рішення BAF. Тобто це відповідає схемі роботи «Центральна база – Декілька торгових площадок – Обмін між ними», яка є історично зрозумілою і досить відлагодженою для торгових підприємств.

Така система є ефективною з точки зору роботи підприємства :

- Прибирає подвійну роботу, оскільки товари, залишки та ціни вже введені і вводяться зазвичай у першу чергу у обліковій системі, а вже потім автоматично чи по команді автоматизовано передаються на у базу сайту. Додатковим бонусом є зниження людських помилок, які можуть виникнути під час повторного вводу товарів на сайт;
- Знімає необхідність задіяння у впровадженні сайту персоналу, оскільки дані передаються автоматично. Відпадає необхідність навчання менеджерів роботі з адміністративною частиною сайту і початкового наповнення сайту менеджерами;
- Дозволяє дуже оперативно отримувати замовлення з сайту на стороні облікової системи, для подальшого комплектування та відвантаження;

- Прибирає необхідність повторного ручного вводу замовлень у обліковій системі, що економить робочий час та прибирає помилки через людський фактор;
- Збільшує захищеність даних, оскільки основні дані у обліковій системі не є доступними у інтернет, натомість дані сайту можуть бути знищені у результаті хакерської атаки;
- Збільшує продуктивність роботи сайту з точки зору обслуговування покупців, оскільки ресурси серверу на сайті не використовуються для введення товарів менеджерами чи масових операцій з завантаження/оновлення товарів;
- Створює значну перспективу масштабованості рішення, оскільки облікова система існує і працює роками, веб-сервіс для неї пишеться один раз і може бути використаний для отримання даними різними сайтами створеними на базі різних CMS для інтернет торгівлі. Таким чином заміна платфоми інтернет-магазину чи навіть додавання ще одного інтернет магазину значно полегшуються.

Отже обрана архітектура для реалізації інтернет-магазину – двох-ядерна з другою базою на основі рішення BAF. Очевидно, що back-end система для такого рішення буде розподіленою і складатиметься з таких частин :

- Веб-сервіс SOA і супутні програмні об’єкти та модулі у складі рішення на платформі BAF. Після реалізації цей сервіс опублікований на веб-сервері підприємства і доступний у мережі;
- Back-end частина сайту та додатковий HTTP-сервіс виконані на Node.js. Сервіс служить для трансляції HTTP GET/POST запитів у SOA сервіс BAF для підтримки систем, що не мають у своєму складі підтримки роботи з сервісами SOA.Back-end частина у складі CMS сайту інтернет магазину містить модулі для зв’язку з другою базою для асинхронного наповнення асортименту, поновлення інформації про товари, залишки та ціни, а також передачі замовлень у другу базу для обробки.

3.2 Реалізація HTTP-сервісу на JavaScript для Node.js

Для масштабування двох-ядерного рішення варто забезпечити можливість обміну не лише через сервіси SOA, а й через сервіси HTTP/REST. Дуже багато CMS систем орієнтовані на роботу саме з такими типами сервісів та не мають у своєму складі достатньої підтримки протоколу SOAP. Тому в рамках роботи було розроблено HTTP сервіс для Node.JS, який транслює виклики GET/POST запитів у протокол SOAP і повертає дані з основного веб-сервісу BAF. Такий сервіс може бути встановлений при необхіності на сервері хостері сайту інтернет-магазину. Таким чином бекенд система сайту може вести синхронний обмін з BAF через цей сервіс. Для CMS, які підтримують JavaScript є можливим вбудувати код у складі самої бекенд системи сайту.

Структура методів HTTP сервісу відповідає структурі методів веб-сервісу BAF. У таблиці 3.1 показана відповідність викликів методів HTTP сервісу методам SOAP. Як видно передача вхідних параметрів для GET методів HTTP сервісу адаптована для передачі у заголовку (рядку адреси) і перетворюється у формат JSON у коді самих методів. Інша справа щодо методу putorder . Оскільки розмір вхідного параметру для методу putorder може бути значним, то був обраний тип методу POST, таким чином вхідний параметер значного розміру може бути переданий у тілі запиту, а не у заголовку.

Методи SOAP сервісу	**Методи HTTP сервісу**		
	Тип запиту	**Виклик**	**Призначення**
GetInfo()	GET	/GetInfo	Перевірка працездатності
GetAllGoods(JSONData)	GET	/getgoods?ModelCode=code1,code2,code3	Отримання інформації про товари
GetAllAmountAndPrices(JSONData)	GET	/getprices?ModelCode=code1,code2,code3	Отримання інформації про залишки та ціни
PutDocument(JSONData)	POST	/putorder	Занесення замовлення у базу BAF

Таблиця 3.1 — Відповідність методів HTTP сервісу методам SOAP сервісу

Для забезпечення роботи HTTP-сервісу було передбачено ряд налаштувань. Всі вони знаходяться у файлі config.json (рис. 3.1).

```
{
  "host": "0.0.0.0",
  "port": 8181,
  "soapwdsl": "http://135.181.165.36/IKOS/ws/IkosConnection.1cws?wsdl",
  "soapname": "http://135.181.165.36/IkosConnection",
  "soapuser": "mrobot",
  "soappass": "Pbt754321"
}
```

Рисунок 3.1 – Настройки HTTP сервісу у файлі config.json

Перші дві настройки стосуються привязки сервісу до адреси та порту для роботи. Наступні чотири служать для підключення до сервісу BAF. Вони включають у себе адресу опису веб-сервісу, а також аутентифікаційні дані для підключення.

Робота сервісу організована у спосіб, показаний на рисунку 3.2. При старті сервісу запускається файл index.js, який зчитує конфігурацію з файлу config.json, створю об’єкт додатку, я кий відповідальний за реалізацію всієї логіки додатку, та запускає на прослохувування веб-сервер express, після чого сервіс очікує запитів на ресурси. Перелік ресурсів та назви методів обробників можна побачити у методі «attachRoutes» класу додатку Application. Повний початковий код для класу Application (файл app.js) наведений у додатку А. Ось ділянка коду, що визначає ресурси та їх обробники (Рис. 3.2):

```
app.get('/getinfoloc', this.getInfoHandler.bind(this));
app.get('/getinfo', this.getInfoSoapHandler.bind(this));
app.get('/getgoods', this.getGoodsHandler.bind(this));
app.get('/getprices', this.getRemainsHandler.bind(this));
app.post('/putorder', jsonParser, this.putOrderHandler.bind(this));

app.get('/api', this.test_api.bind(this));
```

Рисунок 3.2 – Частина коду, що визначає ресурси та їх обробники

Як видно з блок-схеми більшість методів працює за однією схемою, тобто створює підключення до SOA сервісу BAF готує і транслює виклик, шляхом виклику методу віддаленого сервісу, після чого повертає результат як є, або створює повідомлення про помилку у разі недоступності віддаленого сервісу.

Але є спеціальний ресурс «/api», який запускає сторінку тестування роботи сервісу. Ця сторінка імітує роботу back-end системи сайту і дозволяє перевірити

працездатність сервісу після встановлення і запуску без написання додаткового коду чи використання відповідних інструментів. Звичайно всі GET методи можна протестувати прямо у браузері шляхом прописування методів та їхніх параметрів у стрічці адреси. Наприклад, тестування методу getgoods може бути виконано таким викликом такого ресурсу:

http://127.0.0.1:8181/getgoods?ModelCode=1605141,1605140,1605142

У відповідь у вікні браузеру буде видано відповідь у форматі JSON з інформацією про товари з бази BAF, як показано на рисунку 3.3.

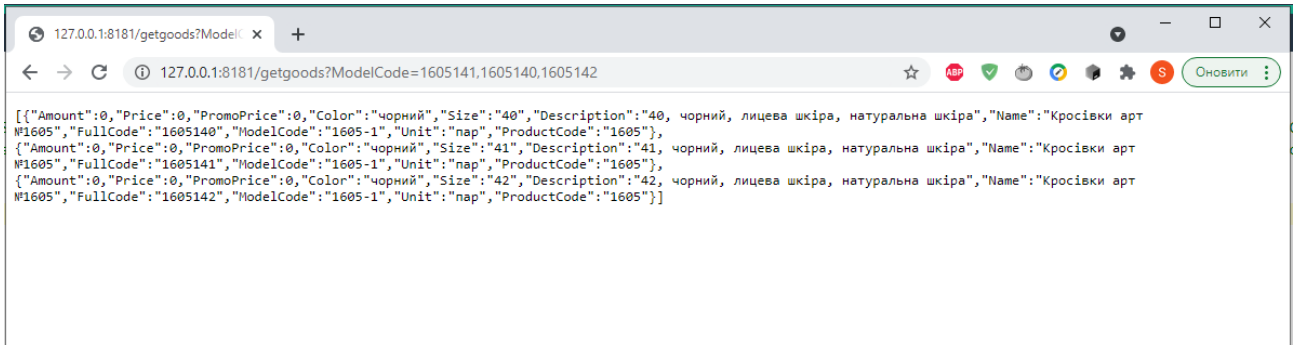

Рисунок 3.3 – Тестування роботи HTTP сервісу у браузері

Одначе, для POST методів тестування у вікні браузера неможливе у принципі. Звичайно можна використати ряд інфтрументів, таких як SoapUI, Boomerang, Advanced REST client, але щоб спростити роботу розробникам back-end частини сайту тестова сторінка просто була влаштована у сам сервіс.

Таким чином, після виклику кореневого ресурсу HTTP сервісу у вигляді - http://127.0.0.1:8181, у вікні браузера з’явиться сторінка для тестування роботи сервісу (рис. 3.5).

У верхній частині сторінки розміщене текстове поле для вводу вхідних параметрів, зліва, а також кнопки для викликів методів сервісу, справа у стовпчик. У нижній частині сторінки розміщене текстове поле для відображення результатів викликів методів сервісу або ж повідомлень про помилку.

Вхідні параметри вводяться у «сирому» вигляді, тобто так, як їх треба передавати у коді викликів. Відповіді відображаються так само, це добре для аналізу формату при написанні коду, що використовує сервіс.

Сторінка має вбудований скрипт написаний на JavaScript і поновлюється за принципом AJAX. Текст html сторінки наведений у додатку Б.

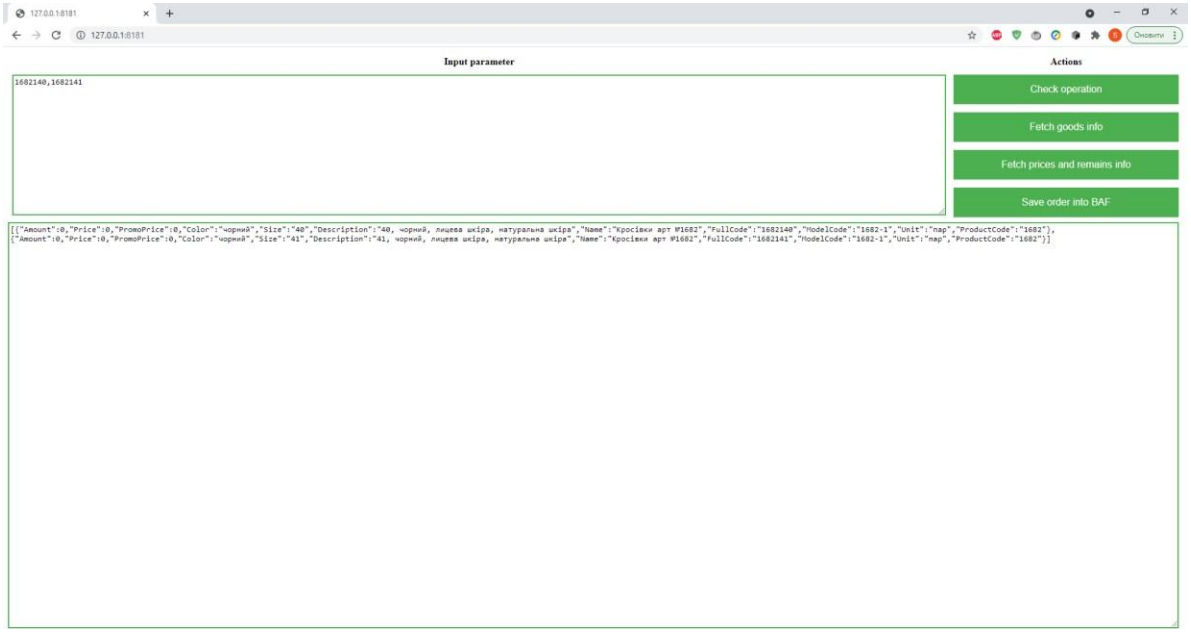

Рисунок 3.4 – Тестова сторінка HTTP сервісу, вигляд у браузері

3.3 Реалізація тестового інтернет магазину на базі коду HTTP-сервісу

Окрім технологічного тестування роботи веб-сервісу у цій роботі було поставлено завдання створити простий інтернет-магазин використовуючи код HTTP сервісу. Такий приклад може служити добрим посібником для програмістів CMS back-end частини сайту, для написання власних модулів підключення до веб-сервісу BAF.

Найпростішим рішенням для реалізації цього завдання було вбудувати тестовий сайт у проект Node.js. Таким чином вже написаний код зв’язку з SOA веб-сервісом може бути використаний у якості backend системи такого тестового сайту.

Варто сказати декілька слів про архітектуру такого рішення. Справа у тому, що використання повного коду трансляції HTTP запитів до веб-сервісу BAF у якості завершеної backend системи сайту не передбачає використання локальної БД сайту. Оскільки звернення до локальної БД замінюються на виклики сервісу. При запиті наприклад інформації про товар на сторінці сайту

дані зчитуються не з БД сайту, а через запит до веб-сервісу BAF. При оформленні замовлення (купівлі одного товару) воно записується знов же таки не у базу сайту, а одразу передається у базу BAF через виклик методу веб-сервісу. Такий дещо ультимативний підхід дозволяє окрім демонстрації концепції ще й оцінити реальну швидкодію обміну через мережу. У реальних же впровадженнях веб-сервіс використовується для наповнення локальної бази даних сайту у фоновому режимі, так само як і запис замовлень у базу даних BAF відбувається у фоновому обміні за розкладом чи за подіями даних у БД сайту. Відображення ж інформації про товари так само як і запис замовлень відбувається через звернення до локальної БД сайту. Що на порядок збільшує швидкодію сторінок проти такої у тестовому сайті.

Одначе автор вважає, що для тестових систем стрес навантаження є пріоритетом для визначення надійності системи загалом.

Зважаючи на те, що тестовий сайт не має на меті демонструвати дизайн чи технології написання front-end модулів, він має декілька обмежень :

- Наповнення його зображеннями товарів не проводиться, тільки вербальна інформація про назву товару, код моделі, ціни та залишки;
- Не реалізовані кошик та підтримка платіжних систем, оскільки реальна реалізація товарів не передбачається;
- Не реалізований повноцінний перегляд списку товарів. Отримати інформацію про товар можна прямо на сторінці деталізації змінивши параметр GET запиту у адресному рядку.

Менше з тим скрипт модулі влаштовані у HTML код сторінок забезпечують повноцінні

Отже back-end частина сайту описана у файлі app.js і транслює виклики front-end частини через інтернет у сервіс BAF. Повна таблиця маршрутизації викликів основного об’єкта додатку виглядає нступним чином (рис.3.5):

```
app.get('/getinfoloc', this.getInfoHandler.bind(this));
app.get('/getinfo', this.getInfoSoapHandler.bind(this));
app.get('/getgoods', this.getGoodsHandler.bind(this));
app.get('/getprices', this.getRemainsHandler.bind(this));
app.post('/putorder', jsonParser, this.putOrderHandler.bind(this));


app.get('/api', this.test_api.bind(this));
app.get('/', this.index.bind(this));
app.get('/contact', this.contact.bind(this));
app.get('/details/:id', this.details.bind(this));
app.get('/create_order/:id', this.createOrder.bind(this));
```

Рисунок 3.5 – маршрутизація викликів основного об’єкта

Перші п’ять обробників викликів стосуються методів самого HTTP сервісу.

Решта – обслуговування front-end частини сайту :

- Виклик «/api» – стосується виклику тестової сторінки HTTP сервісу;
- Виклик «/» – стосується запуску основної сторінки сайту;
- Виклик «/contact» призначений для відображення сторінки контактів;
- Виклик «/details» призначений для відображення сторінки докладної інформації про товар;
- Виклик «/create_order» призначений для введенян даних про покупця товару та запису замовлення у базу.

Програмний код файлу з тестами всіх обробників наведено у додатку А

Код викликів методів back-end частини сайту для запису замовлення у файлі base.html наведений у додатку В

Інтерфейс сайту складається по суті з трьох сторінок :

- Основна з переліком товарів (рис. 3.6)

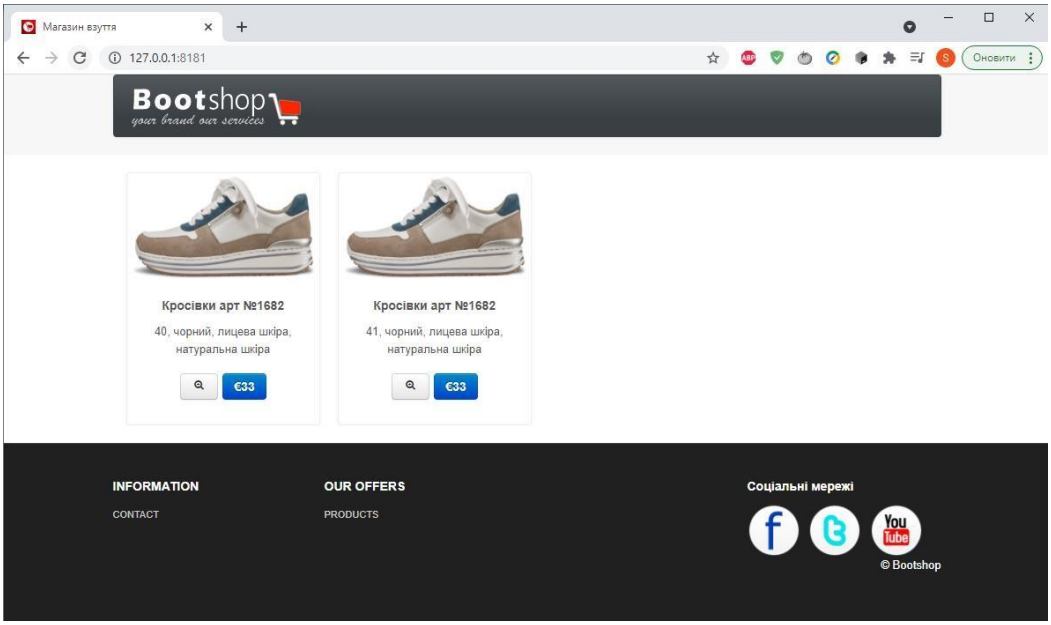

Рисунок 3.6 –.Основна сторінка сайту у браузері

- Сторінка з інформацією про товар (рис. 3.7)

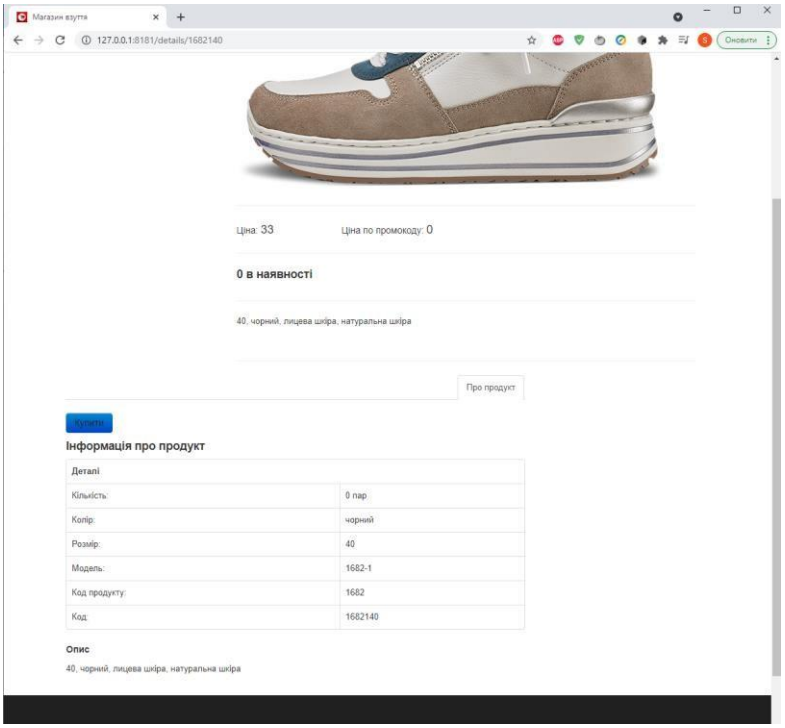

Рисунок 3.7 – Сторінка з інформацією про товар у браузері

- Сторінка оформлення замовлення (рис. 3.8)

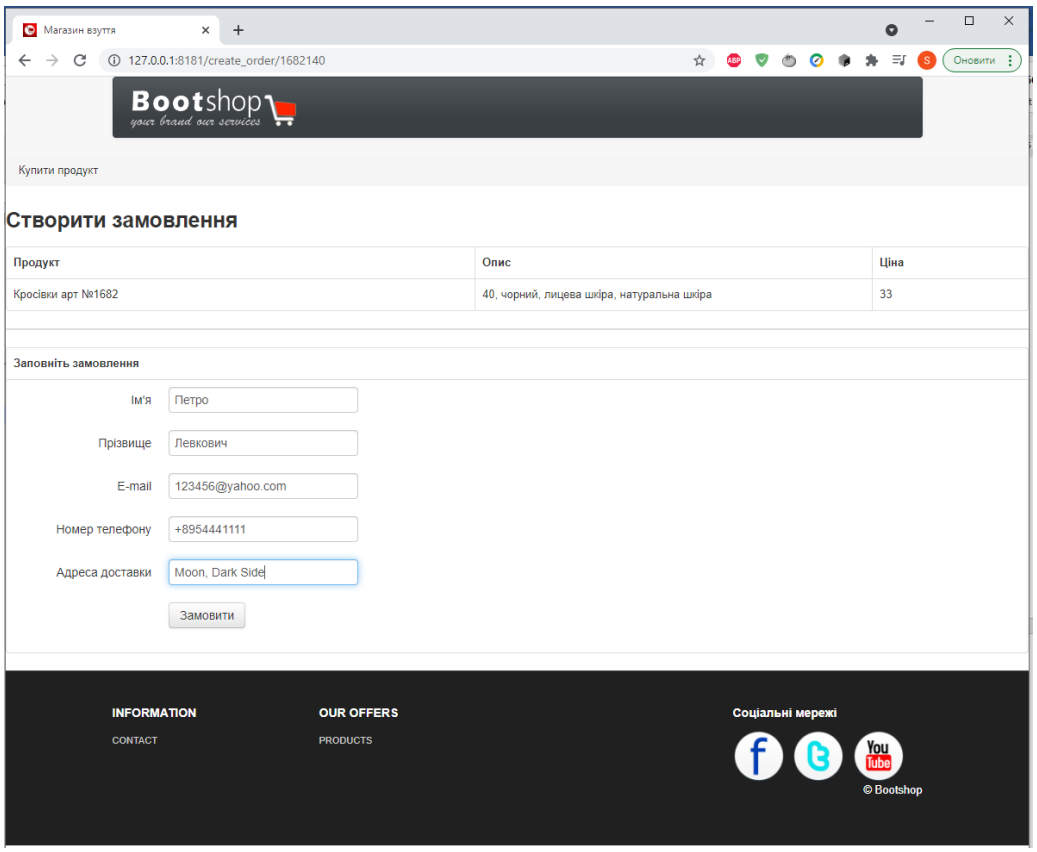

Рисунок 3.8 – Сторінка оформлення замовлення на товар у браузері

3.4 Реалізація частини back-end системи у складі функціоналу BAF

Програмні об’єкти реалізовані у складі функціоналу BAF для забезпечення роботи інтернет-магазину :

- Об'єкт метаданих - план обміну;
- Програмний модуль;
- Об'єкт метаданих - веб-сервіс;
- Об'єкт метаданих - підписка на подію;
- Зовнішня обробка для тестування.

План обміну BAF - це спеціальний об’єкт метаданих, що призначений виконувати наступні функції :

- Надати механізми для умовної чи безумовної реєстрації змін даних для обміну, які відносяться до цього плану обміну;
- Надати механізми для вибірки зареєстрованих змінених об’єктів для подальшої обробки та передачі;

- Надати механізми для видалення зареєстрованих змінених об’єктів після передавання;
- Надати механізми збереження налаштувань, що використовуються під час процесу обміну та інтерфейс для їх введення/зміни.

Для об’єкту метаданих (класу) у базі даних зазвичай створюється один або декілька реальних об’єктів - вузлів, які відповідають іншим зовнішнім системам з якими буде провадитись обмін даними, але один об’єкт, той що відвідає цій базі буде створений автоматично і видалити його неможливо. Тож для кожного плану обміну завжди існує як мінімум один спеціальний вузол і один або декілька вузлів для інших систем. Якщо вузлів для інших систем не існує, то вважається що обмін за цим планом обміну не провадиться.

В рамках функціонування плану обміну працюють два процеси. Перший відслідковує зміни даних і проводить реєстрацію змінених об’єктів у таблиці змін вузла обміну. Другий приймає запити і виклики від веб-сервісу, вибирає зареєстровані об’єкти з тиблиці реєстрації змін вузла і передає їх ініціатору обміну, після успішної передачі видаляє об’єкт обміну з таблиці реєстрації змін. Далеко не всі об’єкти підлягають передачі. Те які об’єкти передаються, а які ні вирішується логікою програмного модулю обміну на базі налаштувань вузла обміну.

Для реєстрації змін об’єктів по умовах треба задати реакцію на подію таких об’єктів. Для цього у платформі BAF передбачено можливість створювати реакцію на подію - об’єкт метаданних “Підписки на подію” і задавати процедуру обробник такої події у одному з загальних програмних модулів.

Отже з метою реєстрації змінених об’єктів було створено загальний програмний модуль під проект обміну та підписку на подію.

Ця підписка реагує на подію “При запису” для об’єктів метаданих “Номенклатура”, “Контрагенти” та “Замовлення покупця”. Основна програмна процедура для виконання обробки події була створена у відповідному спільному модулі.

Функціонал необхідний для вибірки даних по запиту веб-сервісу розміщений у модулі менеджера плану обміну.

Розміщення сервісних процедуру модулі менеджера плану обміну обрано з міркувань логічності призначення, а також, як вказувалось вище, щоб ізолювати програмний код веб-сервісу лише зовнішніми викликами з ехнологічних міркувань.

Веб-сервіс BAF - це спеціальний об’єкт метаданих, що може бути опублікований на веб-сервері, наприклад Apache 2.2, і призначений виконувати запити з мережі по протоколу SOAP.

Публікація і робота веб-сервісу відбувається через спеціальну бібліотеку динамічного компонування, що поставляється у складі платформи BAF. Ця бібліотека використовує файл опису веб-сервісу WSDL для ініціалізації RPC-подібної взаємодії з зовнішніми клієнтами. Цей же опис використовується і самими клієнтами для настроювання провадження обміну з веб-сервісом.

Методи, які реалізує веб-сервіс, забезпечують вимоги до обміну даними з сайтом. Всі методи повертають рядок, що являє структуру відповіді у форматі JSON. Ця структура містить два поля :

- MessageType - Тип повідомлення. Можливі значення :
  - Error - помилка. У полі MessageData опис помилки;
  - Data - дані. У полі MessageData вихідні дані - результат виклику методу;
  - Info - інформація. У полі MessageData - повідомлення.
- MessageData – Дані у форматі JSON, інформаційне повідомлення або опис помилки.

Кожен метод сервісу може мати, а може не мати вхідних параметрів. Якщо параметри присутні, то це насправді один єдиний параметр, що являє собою рядок - структуру у форматі JSON. Таким чином всі реальні параметри інкапсульовані у цю структуру і будуть розібрані та отримані самим методом у процесі обробки.

Методи веб-сервісу:

- GetInfo - Сервісний метод без вхідних параметрів, служить дял перевірки працездатності сервісу у мережі. Повертає структуру у вигляді {“MessageType” : “Info”, “MessageData ” : “Ok !”} ;
- GetGoods - Метод служить для отримання масив з даними товарів, цін та залишків, що зареєстровані як змінені у вузлові плану-обміну;
- GetAmountAndPrices - Метод служить для отримання масиву з цін та залишків для товарів, що зареєстровані як змінені у вузлові плану-обміну;
- GetAllGoods - Метод служить для отримання масив з даними усіх товарів, цін та залишків, що не підпадають під заборону передачі на сайт. Метод має один параметр, у який можна передати перелік кодів товарів, щоб отримати дані не всіх, а лише певних товарів;
- GetAllAmountAndPrices - Метод служить для отримання масив з даними цін та залишків усіх товарів, що не підпадають під заборону передачі на сайт. Метод має один параметр, у який можна передати перелік кодів товарів, щоб отримати дані не про всі, а лише про певні товари;
- PutDocument - Метод служить для занесення даних про замовлення та клієнтів у базу BAF. Метод має один параметр, у який можна передати массив у форматі JSON з даними замовлень у форматі JSON.

Висновок до розділу 3

Провівши аналіз програмної реалізації архітектури інтернет-магазину була обрана розприділена архітектура , яка значно підвищує продуктивність роботи менеджерів з базою товарів та захищеність даних. Розроблений тестовий сайт, back-end система сайту(HTTP сервіс) для Node.JS, яка забезпечує обмін даними між сайтом та веб-сервіс BAF. Веб-сервіс виконує роль служби доступу до актуальної бази товарів, цін та залишків, таким чином ізолюючи клієнтів з інтернет від прямого доступу до даних. Веб-сервіс та тестовий сайт інтернет-магазину демонструють роботу всіх частин розприділеної back-end системи в повному об’ємі.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була розроблена back-end система для інтернет-магазину товарів широкого вжитку.

1. Дослідити сучасні підходи до розроблення і впровадження системи електронної торгівлі (інтернет-магазин);
2. Провести аналіз архітектурних рішень і вибір програмних засобів для реалізації інтернет-магазинів;
3. Програмно реалізувати back-end систему для інтернет-магазину.

В результаті виконання роботи були отримані наступні результати:

1. Дослідили сучасні підходи до розроблення і впровадження системи електронної торгівлі (інтернет-магазин). Проаналізувавши ринок електронної торгівлі, виявили найбільш поширені способи реалізації веб-сервісів на сьогодні, проаналізували сучасні етапи розробки сайтів.

2. Також було проаналізовано які є архітектурні рішеня і вибір програмних засобів для реалізації інтернет-магазинів.

Провівши аналіз програмних засобів для програмної реалізації back-end системи для інтернет-магазину, зробили висновок, що платформа Node.js є хорошим і популярним рішенням для такої реалізації. Для реалізації клієнтської та серверної частини – мова програмування JavaScript. Для обміну даними між частинами розподіленої back-end системи хорошим рішеням є веб-сервіс SOA. Робота з такими веб-сервісами гарно підтримується як в Node.js так і в платформі BAF.

3. Наведено програмно-архітектурну концепцію розробки веб-ресурсу. Обрано розподілену (двох-ядерну) архітектуру створення інтернет магазину.

Створену розподілену back-end систему, як елемент системи інтернет торгівлі товарами широкого вжитку, що включає в себе back-end систему сайту для Node.js, веб-сервіс BAF. Розроблений тестовий сайт інтернет-магазину для демонстрації роботи back-end системи.

Досліджено можливості використання SOA-сервісів та підтримка SOA-інтерфейсу в BAF із застосуванням формату XML/JSON, проаналізовано

платформи та засоби програмування для реалізації розробленої back-end системи.

Back-end частина сайту з таблицею маршрутів та сторінка тестування (API) написані на JavaScript під node.js з використанням фреймворку Express. Для того, щоб динамічно формувати та передавати HTML-сторінки на front-end було використано процесор шаблонів - nunjuks.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. HTTP-сервисы URL: https://expert.chistov.pro/public/302876 (дата звернення 05.05.2021).
2. Node.js URL: https://uk.wikipedia.org/wiki/Node.js. (дата звернення 04.05.2021).
3. SOAP веб-сервіс URL: https://www.quality-assurance-group.com/soap-web-servis-api-testing-interview/ (дата звернення 02.05.2021).
4. Soap-сервисы URL: https://academy.terrasoft.ru/docs/user/kastomizacija_no_code/web_servisy/nastroit_integraciyu_s_web_servisom_soap (дата звернення 01.05.2021).
5. Азарян Д. А., Прончев Г. Б. Современные интернет-технологии и безопасность личности // Юный ученый. — 2016. — №3. — С. 61-63.
6. Анісімов А.В. Інформаційні системи та бази даних: Навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. / Анісімов А.В., Кулябко П.П. –Київ. – 2017. – 110 с.
7. Богачова Т. Г. 1С: Підприємство 8. Управління торговими операціями в питаннях і відповідях. Вид-во: 1С-Паблішинг, 2014. 210 с.
8. Выбор лучшего Node.js фреймворка: Express, Koa или Sails URL: https://umbrellait.com/ru/blog/choosing-the-best-nodejs-framework (дата звернення 12.05.2021).
9. Гарнаев Андрей WEB-программирование на Java и JavaScript / Андрей Гарнаев , Сергей Гарнаев. М.: БХВ-Петербург, 2005. 822 с.
10. Глобальна електронна комерція URL: https://www.emarketer.com/content/global-ecommerce-2019 (дата звернення 07.05.2021).
11. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч.посібник. –Електронне видання, 2018. – 118 с.
12. Обмен с сайтом через веб-сервис URL: https://stimul.kiev.ua/materialy.htm?a=obmen-s-saytom-cherez-veb- servis (дата звернення 28.04.2021).

13. Підручниз з HTML URL: https://hackit-ukraine.com/1326-the-html-handbook (дата звернення 24.04.2021).
14. Плескач В. Л., Затонацька Т. Г. Електронна комерція: навч. посіб. К.: Знання, 2007. 535 с.
15. Програмні системи створення веб-сайтів, CMS: http://www.znannya.org/?view=WebDev (дата звернення 18.05.2021).
16. Пученкин Е. М. Создание web-сайта коммерческой организации // Гаудеамус. 2011. № 18.
17. Радченко М. Г. 1С: Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приёмы: учебное пособие [Текст] / М.Г. Радченко. М.: ООО «1С-Софт», 2015. 965 с.
18. Радченко М. Г. Профессиональная разработка в системе «1С: Предприятие 8». Том 2. М.: Изд-во: 1С - Паблишинг, 2012. 230 с.
19. Системи керування вмістом URL: https://www.victoria.lviv.ua/library/students/wd4/work10.html (дата звернення 03.05.2021).
20. Системи управління сайтом URL: https://recommerce.com.ua/cms-dlya-stvorennya-saitu-internet-magazin (дата звернення 09.05.2021).
21. Современный учебник JavaScript URL: https://learn.javascript.ru (дата звернення 18.04.2021).
22. Тенденції електронної комерції URL: https://www.oberlo.com/blog/ecommerce-trends (дата звернення 12.05.2021).
23. Учебник по JavaScript URL: https://itchief.ru/javascript/introduction (дата звернення 22.04.2021).
24. Шалева О. І. Електронна комерція: навч. посіб. К.: Центр учбової літератури, 2011. 216 с.
25. Що таке платформа Business Automation Framework (BAF): https://kamala-soft.com/uk/blog/chto-takoe-platforma-business-automation-framework-baf/ (дата звернення 22.05.2021).

ДОДАТКИ

Додаток А

Початковий код основного класу Application HTTP сервісу. Файл app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const models = require('./models');
const soap = require('soap');
const config = require('./config.json');
const path = require('path');

class Application {

    constructor (config) {
        this.expressApp = express();
               this.config = config;

        this.attachRoutes();

               this.expressApp.use(express.static('public'));
    }

    attachRoutes () {
        let app = this.expressApp;
        let jsonParser = bodyParser.json();

        app.get('/getinfoloc', this.getInfoHandler.bind(this));
               app.get('/getinfo', this.getInfoSoapHandler.bind(this));
        app.get('/getgoods', this.getGoodsHandler.bind(this));
               app.get('/getprices', this.getRemainsHandler.bind(this));
               app.post('/putorder', jsonParser, this.putOrderHandler.bind(this));


               app.get('/api', this.test_api.bind(this));
               app.get('/', this.index.bind(this));
               app.get('/contact', this.contact.bind(this));
               app.get('/details/:id', this.details.bind(this));
               app.get('/create_order/:id', this.createOrder.bind(this));
    }


       createOrder (req, res) {

               let modelCode = req.params.id;
               let jsonData = [{"ModelCode": modelCode}];
               let params = {JSONData : JSON.stringify(jsonData)};

               soap.createClient('./IkosConnection.wsdl', function(err, client) {
                       if (err) {
                               console.log(`SOAP ERROR : ${err}`);
                               res.send(err);
                               return;
                       };

                       client.setSecurity(new soap.NTLMSecurity(config.soapuser, config.soappass, '',
''));

                       client.GetAllGoods( params, function(err, result) {
                               if (err) {
                                       console.log(`GetAllGoods error : ${err}`);
                                       res.send(err);
                                       return;
                               };
                               let message = JSON.parse(result.return);

                               let item = message.MessageData[0];

                               let data = {
                                       "item": item
                               }

                               res.render('order.njk', data);
                       })
               });
       }
```

```
	details (req, res) {
		console.log(req.params.id);

		let modelCode = req.params.id;
		let jsonData = [{"ModelCode": modelCode}];
		let params = {JSONData : JSON.stringify(jsonData)};

		console.log(params);

		soap.createClient('./IkosConnection.wsdl', function(err, client) {
			if (err) {
				console.log(`SOAP ERROR : ${err}`);
				res.send(err);
				return;
			};

			client.setSecurity(new soap.NTLMSecurity(config.soapuser, config.soappass, '',
''));

			client.GetAllGoods( params, function(err, result) {
				if (err) {
					console.log(`GetAllGoods error : ${err}`);
					res.send(err);
					return;
				};
				let message = JSON.parse(result.return);
				console.log(message);

				let item = message.MessageData[0];

				let data = {
					"stock": item.stock,
					"Description": item.Description,
					"Name":  item.Name,
					"PromoPrice": item.PromoPrice,
					"Price": item.Price,
					"Color":  item.Color,
					"Size": item.Size,
					"ModelCode": item.ModelCode,
					"ProductCode": item.ProductCode,
					"FullCode": item.FullCode,
					"Unit": item.Unit,
					"Amount": item.Amount
				}

				res.render('product_details.njk', data);
			})
		});
	}


	contact (req, res) {
		res.render('contact.njk');
	}


	test_api (req, res) {
		res.sendFile(path.join(__dirname + '/old_index.html'));
	}

	index (req, res) {
		let modelCode = '1682140,1682141';
		let jsonData = [];

		for (let mCode of modelCode.split(",")) {
			jsonData.push({ModelCode : mCode});
		}

		let params = {JSONData : JSON.stringify(jsonData)};

		console.log(params);

		soap.createClient('./IkosConnection.wsdl', function(err, client) {
			if (err) {
				console.log(`SOAP ERROR : ${err}`);
				res.send(err);
				return;
			};
```

```
                        client.setSecurity(new soap.NTLMSecurity(config.soapuser, config.soappass, '',
''));

                        client.GetAllGoods( params, function(err, result) {
                                if (err) {
                                        console.log(`GetAllGoods error : ${err}`);
                                        // res.send(err);
                                        return;
                                };
                                let message = JSON.parse(result.return);
                                console.log(message);

                                let data = {
                                        items: message.MessageData
                                }

                                res.render('index.njk', data);
                        })
                });
        }


    getInfoHandler (req, res) {
        let response = new models.Message("info", "Local mode - Ok!") ;
                console.log(response);
                res.set('Access-Control-Allow-Origin', '*');
        res.json(response);
    }

    putOrderHandler (req, res) {

                //let orderList = req.query.JSONData || '[]';
                let orderList = req.body || [];




                if (!orderList.length) {
                        let err = 'Input list is empty';
                        console.log(`SOAP ERROR : ${err}`);
                        console.log("LIST ERROR");
                        res.send(err);
                        return;
                }

                //let jsonData = JSON.parse(orderList) ;
                let jsonData = orderList ;

                soap.createClient('./IkosConnection.wsdl', function(err, client) {
                        if (err) {
                                console.log(`SOAP ERROR : ${err}`);
                                res.send(err);
                                return;
                        };

                        let params = {JSONData : JSON.stringify(jsonData)};

                        console.log(params);

                        client.setSecurity(new soap.NTLMSecurity(config.soapuser, config.soappass, '',
''));

                        client.PutDocument(params, function(err, result) {
                                if (err) {
                                        console.log(`GetInfo error : ${err}`);
                                        res.send(err);
                                        return;
                                };
                                let message = JSON.parse(result.return);
                                console.log(message);
                                res.json(message);
                        })
                });
    }

    getInfoSoapHandler (req, res) {
                res.set('Access-Control-Allow-Origin', '*');
```

```
		soap.createClient('./IkosConnection.wsdl', function(err, client) {
			if (err) {
				console.log(`SOAP ERROR : ${err}`);
				res.send(err);
				return;
			};

			client.setSecurity(new soap.NTLMSecurity(config.soapuser, config.soappass, '',
''));

			client.GetInfo(function(err, result) {
				if (err) {
					console.log(`GetInfo error : ${err}`);
					res.send(err);
					return;
				};
				let message = JSON.parse(result.return);
				console.log(message);
				res.json(message);
			})
		});
	}

	getGoodsHandler (req, res) {

		let modelCode = req.query.ModelCode || '';
		let jsonData = [];

		for (let mCode of modelCode.split(",")) {
			jsonData.push({ModelCode : mCode});
		}

		let params = {JSONData : JSON.stringify(jsonData)};


		soap.createClient('./IkosConnection.wsdl', function(err, client) {
			if (err) {
				console.log(`SOAP ERROR : ${err}`);
				res.send(err);
				return;
			};

			client.setSecurity(new soap.NTLMSecurity(config.soapuser, config.soappass, '',
''));

			client.GetAllGoods( params, function(err, result) {
				if (err) {
					console.log(`GetAllGoods error : ${err}`);
					res.send(err);
					return;
				};
				let message = JSON.parse(result.return);
				console.log(message);
				if (message.MessageType == 'Data') {
					res.send(message.MessageData);
				} else {
					res.json(message);
				};
			})
		});
	}

	getRemainsHandler (req, res) {

		let modelCode = req.query.ModelCode || '*';
		let jsonData = [];

				for (let mCode of modelCode.split(",")) {
			jsonData.push({ModelCode : mCode});
		}

		let params = {JSONData : JSON.stringify(jsonData)};

		console.log(params);

		soap.createClient('./IkosConnection.wsdl', function(err, client) {
			if (err) {
				console.log(`SOAP ERROR : ${err}`);
				res.send(err);
				return;
```

```
                        };

                        client.setSecurity(new soap.NTLMSecurity(config.soapuser, config.soappass, '',
''));

                        client.GetAllAmountAndPrices( params, function(err, result) {
                                if (err) {
                                        console.log(`GetAllAmountAndPrices error : ${err}`);
                                        res.send(err);
                                        return;
                                };
                                let message = JSON.parse(result.return);
                                console.log(message);
                                if (message.MessageType == 'Data') {
                                        res.send(message.MessageData);
                                } else {
                                        res.json(message);
                                };
                        })
                });
    }
}

module.exports = Application;
```

Додаток Б

Код HTML сторінки тестування HTTP сервісу. Файл index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ua" lang="ua">
<head>
	<style>
		td {
		  padding: 5px;
		  text-align: left;
		  height: 50;
		  vertical-align: top;
		}
		th {
		  padding: 5px;
		  text-align: center;
		}
		button {
			width: 100%;
			height: 100%;
			background-color: #4CAF50;
			border: none;
			color: white;
			padding: 15px 32px;
			text-align: center;
			text-decoration: none;
			display: inline-block;
		  font-size: 16px;
		}
		.InputData {
			width: 100%;
			height: 100%;
			border: 2px solid;
			border-color: #4CAF50;
		}
		.OutputData {
			width: 100%;
			height: 100%;
			border: 2px solid;
			border-color: #4CAF50;
		}
		td > textarea
		{
			width: 100%;
			height: 100%;
		}

	</style>
</head>
<meta charset="utf-8">
<script>
	function getInfo() {

		document.getElementById('OutputDataText').value = "get info ...";

		var xhr = new XMLHttpRequest();
		xhr.open('GET', 'http://127.0.0.1:8181/getinfo', true);

		xhr.setRequestHeader('Access-Control-Allow-Origin', '*');
		xhr.setRequestHeader('Accept', 'application/json');
		xhr.setRequestHeader('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE,OPTIONS');
		xhr.setRequestHeader('Access-Control-Allow-Headers', 'Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With');

		xhr.onload  = function() {
			document.getElementById('OutputDataText').value = xhr.responseText;
		};

		xhr.send();

	}

	function getGoods() {
		let info = document.getElementById('InputDataText').value.trim();
		document.getElementById('OutputDataText').value = "Get goods ...";

		var xhr = new XMLHttpRequest();
```

```
                xhr.open('GET', 'http://127.0.0.1:8181/getGoods?ModelCode=' + info, true);

                xhr.setRequestHeader('Access-Control-Allow-Origin', '*');
                xhr.setRequestHeader('Accept', 'application/json');
                xhr.setRequestHeader('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE,OPTIONS');
                xhr.setRequestHeader('Access-Control-Allow-Headers', 'Content-Type, Access-Control-
Allow-Headers, Authorization, X-Requested-With');

                xhr.onload  = function() {
                        document.getElementById('OutputDataText').value = xhr.responseText;
                };

                xhr.send();

        }

        function GetPrices() {
                let info = document.getElementById('InputDataText').value.trim();
                document.getElementById('OutputDataText').value = "Get prices .... ";

                var xhr = new XMLHttpRequest();
                xhr.open('GET', 'http://127.0.0.1:8181/getPrices?ModelCode=' + info, true);

                xhr.setRequestHeader('Access-Control-Allow-Origin', '*');
                xhr.setRequestHeader('Accept', 'application/json');
                xhr.setRequestHeader('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE,OPTIONS');
                xhr.setRequestHeader('Access-Control-Allow-Headers', 'Content-Type, Access-Control-
Allow-Headers, Authorization, X-Requested-With');

                xhr.onload  = function() {
                        document.getElementById('OutputDataText').value = xhr.responseText;
                };

                xhr.send();

        }

        function PutOrder() {
                let info = document.getElementById('InputDataText').value.trim();

                let jsonData = info;

                //alert(jsonData);

                document.getElementById('OutputDataText').value = "Put order ...";

                var xhr = new XMLHttpRequest();
                xhr.open('POST', 'http://127.0.0.1:8181/putorder', true);

                xhr.setRequestHeader('Access-Control-Allow-Origin', '*');
                xhr.setRequestHeader('Accept', 'application/json');
                xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
                xhr.setRequestHeader('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE,OPTIONS');
                xhr.setRequestHeader('Access-Control-Allow-Headers', 'Content-Type, Access-Control-
Allow-Headers, Authorization, X-Requested-With');

                xhr.onload  = function() {
                        document.getElementById('OutputDataText').value = xhr.responseText;
                };

                xhr.send(jsonData);

        }
</script>

<body>

        <table style="width:100%">
                <tr>
                        <th style="width:80%">Input parameter</th>
                        <th style="width:20%">Actions</th>
                </tr>
                <tr>
                        <td rowspan="4">
                                <textarea type="textarea" class="InputData" name="InputDataText"
access="false" id="InputDataText">1682140,1682141</textarea>
                        </td>
                        <td>
                                <button type="button" class="btn-GetInfo" name="btnGetInfo" value="1"
access="false" style="default" id="btnGetInfo"  onclick="getInfo()">Check operation</button>
```

```
                        </td>
                </tr>
                <tr>
                        <td>
                                <button type="button" class="btn-GetGoods" name="btnGetGoods"
value="2" access="false" style="default" id="btnGetGoods" onclick="getGoods()">Fetch goods
info</button>
                        </td>
                </tr>
                <tr>
                        <td>
                                <button type="button" class="btn-GetPrices" name="btnGetPrices"
value="3" access="false" style="default" id="btnGetPrices" onclick="GetPrices()">Fetch prices and
remains info</button>
                        </td>
                </tr>
                <tr>
                        <td>
                                <button type="button" class="btn-PutOrder" name="btnPutOrder"
value="4" access="false" style="default" id="btnPutOrder" onclick="PutOrder()">Save order into
BAF</button>
                        </td>
                </tr>
        </table>
        <textarea type="textarea" class="OutputData" name="OutputDataText" access="false"
id="OutputDataText"></textarea>
</body>
</html>
```

Додаток В

Текст сторінки base.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Магазин взуття</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">
<!-- Bootstrap style -->
    <link id="callCss" rel="stylesheet" href="/themes/bootshop/bootstrap.min.css" media="screen"/>
    <link href="/themes/css/base.css" rel="stylesheet" media="screen"/>
<!-- Bootstrap style responsive -->
       <link href="/themes/css/bootstrap-responsive.min.css" rel="stylesheet"/>
       <link href="/themes/css/font-awesome.css" rel="stylesheet" type="text/css">
<!-- Google-code-prettify -->
       <link href="/themes/js/google-code-prettify/prettify.css" rel="stylesheet"/>
<!-- fav and touch icons -->
    <link rel="shortcut icon" href="/themes/images/ico/favicon.ico">
    <link rel="apple-touch-icon-precomposed" sizes="144x144" href="/themes/images/ico/apple-touch-
icon-144-precomposed.png">
    <link rel="apple-touch-icon-precomposed" sizes="114x114" href="/themes/images/ico/apple-touch-
icon-114-precomposed.png">
    <link rel="apple-touch-icon-precomposed" sizes="72x72" href="/themes/images/ico/apple-touch-
icon-72-precomposed.png">
    <link rel="apple-touch-icon-precomposed" href="/themes/images/ico/apple-touch-icon-57-
precomposed.png">
       <style type="text/css" id="enject"></style>
  </head>
<body>
<div id="header">
<div class="container">
<!-- Navbar ================================================== -->
<div id="logoArea" class="navbar">
<a id="smallScreen" data-target="#topMenu" data-toggle="collapse" class="btn btn-navbar">
       <span class="icon-bar"></span>
       <span class="icon-bar"></span>
       <span class="icon-bar"></span>
</a>
  <div class="navbar-inner">
    <a class="brand" href="/"><img src="/themes/images/logo.png" alt="Bootsshop"/></a>
              <form class="form-inline navbar-search" method="post" action="products.html" >

       </form>
    <ul id="topMenu" class="nav pull-right">
       <div id="login" class="modal hide fade in" tabindex="-1" role="dialog" aria-
labelledby="login" aria-hidden="false" >
                <div class="modal-header">
                     <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&#65533;</button>
                     <h3>Login Block</h3>
                </div>
                <div class="modal-body">
                     <form class="form-horizontal loginFrm">
                       <div class="control-group">

                            <input type="text" id="inputEmail" placeholder="Email">
                       </div>
                       <div class="control-group">
                            <input type="password" id="inputPassword" placeholder="Password">
                       </div>
                       <div class="control-group">
                            <label class="checkbox">
                            <input type="checkbox"> Remember me
                            </label>
                       </div>
                     </form>
                     <button type="submit" class="btn btn-success">Sign in</button>
                     <button class="btn" data-dismiss="modal" aria-hidden="true">Close</button>
                </div>
       </div>
       </li>
    </ul>
  </div>
</div>
```

```
</div>
</div>
<!-- Header End======================================================================== -->

{% block main %}
{% endblock %}
<script>
	const random = (min, max) => Math.floor(Math.random() * (max - min)) + min;


	function PutOrder() {
		var amount = document.getElementById("price").innerHTML;
		var firstName = document.getElementById('inputName').value;
		var lastName = document.getElementById('inputSurname').value;
		var email = document.getElementById('inputEmail').value;
		var phone  =  document.getElementById('inputPhone').value;
		var address = document.getElementById('inputAddress').value;
		var product_id = document.getElementById("product_id").value.toString();
		var model_id = document.getElementById("model_id").value.toString();
		var service_fee = 2;
		var price = parseInt(amount) + service_fee;
		console.log(price);

		var date = new Date();
		let year = date.getUTCFullYear();
		let month = date.getUTCMonth();
		let day = date.getUTCDate();

		if (month < 10){
			month = "0" + month;
		}
		if (day < 10){
			day = "0" + day;
		}

		date = year + "-" + month + "-" + day;
		console.log(date);

		let jsonData = [{
			"order_id": random(2000, 2999).toString(),
			"order_date": date,
			"order_amount": amount,
			"client_id": random(2000, 2999).toString(),
			"client_firstname": firstName,
			"client_lastname": lastName,
			"client_email": email,
			"client_phone": phone,
			"client_address": address,
			"delivery_type": "3",
			"payment_type": "4",
			"payment_amount": amount,
			"rows": [
				{
					"id": product_id,
					"model_id": model_id,
					"quantity": "1",
					"price": price,
					"discount": "0"
				}
			]
		}]

		jsonData = JSON.stringify(jsonData);

		// console.log(jsonData);

		var xhr = new XMLHttpRequest();
		xhr.open('POST', 'http://127.0.0.1:8181/putorder', true);

		xhr.setRequestHeader('Access-Control-Allow-Origin', '*');
		xhr.setRequestHeader('Accept', 'application/json');
		xhr.setRequestHeader('Content-type', 'application/json; charset=utf-8');
		xhr.setRequestHeader('Access-Control-Allow-Methods', 'GET,POST,PUT,DELETE,OPTIONS');
		xhr.setRequestHeader('Access-Control-Allow-Headers', 'Content-Type, Access-Control-
Allow-Headers, Authorization, X-Requested-With');

		xhr.onload  = function() {
			alert("Ваше замовлення прийнято!");
			window.location.href = "/";
		};
```

```
                xhr.send(jsonData);

        }
</script>
<!-- Footer ================================================================== -->
        <div  id="footerSection">
        <div class="container">
                <div class="row">
                        <div class="span3">
                                <h5>INFORMATION</h5>
                                <a href="/contact">CONTACT</a>
                         </div>
                        <div class="span3">
                                <h5>OUR OFFERS</h5>
                                <a href="#">PRODUCTS</a>

                         </div>
                        <div id="socialMedia" class="span3 pull-right">
                                <h5>Соціальні мережі </h5>
                                <a href="https://www.facebook.com/"><img width="60" height="60"
src="/themes/images/facebook.png" title="facebook" alt="facebook"/></a>
                                <a href="https://twitter.com/?lang=uk"><img width="60" height="60"
src="/themes/images/twitter.png" title="twitter" alt="twitter"/></a>
                                <a href="https://www.youtube.com/"><img width="60" height="60"
src="/themes/images/youtube.png" title="youtube" alt="youtube"/></a>
                         </div>
                 </div>
                <p class="pull-right">&copy; Bootshop</p>
        </div><!-- Container End -->
        </div>
<!-- Placed at the end of the document so the pages load faster
============================================= -->
        <script src="/themes/js/jquery.js" type="text/javascript"></script>
        <script src="/themes/js/bootstrap.min.js" type="text/javascript"></script>
        <script src="/themes/js/google-code-prettify/prettify.js"></script>

        <script src="/themes/js/bootshop.js"></script>
    <script src="/themes/js/jquery.lightbox-0.5.js"></script>
</body>
</html>
```