Київський національний торговельно-економічний університет

Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка веб-додатку магазину автозапчастин»

Студента 4 курсу, 7 групи,
спеціальності 121
«Інженерія програмного
забезпечення» _____ Афанасьєва Олександра
(підпис студента) Анатолійовича

Науковий керівник
кандидат технічних наук,
професор _____ Пашорін Валерій
(підпис керівника) Іванович

Гарант освітньої програми
кандидат технічних наук,
доцент _____ Цензура Микола
(підпис керівника) Олександрович

КИЇВ – 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення та кібербезпеки

Освітній ступінь бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Затверджую
Зав. кафедри інженерії
програмного забезпечення
та кібербезпеки
Криворучко О.В.
"20" жовтня 2020 р.

Завдання
на випускну кваліфікаційну роботу студентові

Афанасьєву Олександру Анатолійовичу

(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи «Розробка веб-додатку магазину автозапчастин»

Затверджена наказом ректора від "30" жовтня 2020 р. № 3225

2. Строк здачі студентом закінченої роботи

3. Цільова установка та вихідні дані до роботи

Мета роботи дослідження сучасних технологій розробки інтерактивних HTML застосувань та вибір оптимальних методів та засобів розробки Web-порталу магазину автозапчастин

Об'єкт дослідження інтернет-магазин автозапчастин

Предмет дослідження розробка веб-додатку магазину автозапчастин

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
№		Завдання видав	Завдання прийняв

5. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Напрямки розвитку сучасних web-технологій

1.1.1 Поява DHTML

1.1.2 Технології, що застосовуються Web-клієнтами

1.1.3 Технології створення серверних Web-додатків

1.2 Застосування технологій Single Page Application (SPA)

1.2.1 Загальна архітектура технологій SPA

1.2.2 Порівняння патернів проектування

1.3 Висновки до розділу 1

РОЗДІЛ 2. ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ HTML НА ПРИКЛАДІ WEB-ПОРТАЛУ МАГАЗИНУ АВТОЗАПЧАСТИН

2.1 Визначення загальних вимог до Web-порталу

2.2 Визначення функціоналу Web-порталу

2.3 Ролі та права користувачів

2.4 Архітектура Web-порталу

2.5 Висовки до розділу 2

РОЗДІЛ 3. ПРОЕКТУВАННЯ WEB-ПОРТАЛУ МАГАЗИНУ АВТОЗАПЧАСТИН

3.1 Побудова макету інтерфейсу користувача

3.2 Серверна частина

3.2.1 Панель керування адміністратора

3.3 Розробка бізнес-логіки клієнтського ПЗ Web-порталу

3.4 Проектування програмного забезпечення клієнтської частини ПЗ

3.5 Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	*21.09.2020*	*21.09.2020*
2.	*Вступ та перелік літературних джерел*	*14.12.2020*	*14.12.2020*
3.	*Розділ 1. «Огляд існуючих рішень»*	*19.02.2021*	*19.02.2021*
4.	*Розділ 2. «Застосування технологій html на прикладі web-порталу магазину автозапчастин»*	*05.03.2021*	*05.03.2021*
5.	*Розділ 3. «Проектування web-порталу магазину автозапчастин»*	*10.04.2021*	*10.04.2021*
6.	*Висновки*	*24.04.2021*	*24.04.2021*
7.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*	*14.05.2021*	*14.05.2021*
8.	*Підготовка автореферату та презентації доповіді*	*17.05.2021*	*17.05.2021*
9.	*Попередній захист випускної кваліфікаційної роботи*	*18.05.2021 – 21.05.2021*	*21.05.2021*
10.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*01.06.2021*	*01.06.2021*
11.	*Здача прошитої випускної кваліфікаційної роботи на кафедру*	*02.06.2021*	*02.06.2021*
12.	*Публічний захист випускної кваліфікаційної роботи*		

7. Дата видачі завдання «20» жовтня 2020 р.

8. Науковий керівник випускної кваліфікаційної роботи Пашорін В. І.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Цензура М. О
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Афанасьєв О. А.
(прізвище, ініціали, підпис)

11. Відгук керівника випускної кваліфікаційної роботи

Науковий керівник випускної кваліфікаційної роботи _____
(підпис, дата)

Відмітка про попередній захист _____Пашорін В. І._____
(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускна кваліфікаційна робота студента _____Афанасьєва О. А._____
(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____Цензура М. О._____
(прізвище, ініціали, підпис)

Завідувач кафедри _____Криворучко О. В._____
(підпис, прізвище, ініціали)

«____»_____ 2021 р.

АНОТАЦІЯ

Дипломна робота на тему «Розробка веб-додатку магазину автозапчастин» містить 49 сторінки, 16 рисунків, та 13 додатків. Перелік посилань налічує 11 найменувань.

Метою випускної кваліфікаційної роботи є дослідження сучасних технологій розробки інтерактивних HTML застосувань та вибір оптимальних методів та засобів розробки Web-порталу магазину автозапчастин.

Об'єкт дослідження: веб-додаток магазину автозапчастин

Предмет дослідження: розробка веб-додатку магазину автозапчастин

У першому розділі було проаналізовано таку технологію як DHTML, її переваги та можливості. Описані можливі функції та медоти використання такої технології. Розглянуто технології створення інтернет додатків, можливості сучасних web-браузерів, та як вони можуть змінювати код скриптових мов. Було порівняно основні патерни проектування, описані їх функції та ознаки.

У другому розділі були описані загальні вимоги до веб-порталу для його ефективного, надійного та безпечного функціонування. Визначені основні функції веб-порталу та ролі користувачів на ньому. Також була розписана детальна архітектура веб-порталу.

У третьому розділі було спроектовано веб-портал магазину автозапчастин. Для реалізації проекту, розробка виконувалась із застосуванням html/css/js з підключенням бібліотек bootstrap та JQuerry (для забезпечення інтерактивності веб-додатку), Node.js (для функціонування серверу). Покроково описано процес створення веб-порталу та додання функціоналу до нього. Були визначені дії користувачів на веб-порталі для розробки бізнес-логіки. За допомогою обраних технологій створено серверну частину сайту, додано динамічні подання до БД, а також розроблено адаптивний інтерфейс користувача.

ANNOTATION

The thesis on the topic "Development of a web application for an auto parts store" contains 49 pages, 16 figures and 13 appendices. The list of links includes 11 items.

The purpose of the study: explore modern technologies for developing interactive HTML application and choose the best methods and tools for developing a web-portal of auto parts store.

Object of research: web application of an auto parts store.

The subject of research: development of web application for an auto parts store.

In the first section was analyzed technology such as DHTML, its benefits and features. Possible functions and methods of using such technology are described. Technologies of creating Internet applications, capabilities of modern web-browsers, and how they can change the code of scripting languages are considered. The main design patterns were compared, their functions and features were described.

In the second section were described the general requirements for a web portal for its efficient, reliable and secure operation. The main functions of the web portal and the role of users on it are defined. The detailed architecture of the web portal was also painted.

In the third section the web portal of the auto parts store was designed. To implement the project, the development was performed using html / css / js with the connection of bootstrap and JQuerry libraries (to ensure the interactivity of the web application), Node.js (for server operation). The process of creating a web portal and adding functionality to it is described step by step. User actions have been identified on the web portal to develop business logic. With the help of selected technologies, the server part of the site was created, dynamic views were added to the database, and an adaptive user interface was developed.

ЗМІСТ

ВСТУП

На сьогоднішній день комп'ютерні та мережеві технології відіграють все більшу роль в різних областях людської діяльності. Процеси впровадження нових інформаційних технологій впливають на всі її сфери. Швидкий розвиток комунікаційних технологій трансформує багато процесів в сучасному суспільстві.

Інтернет, як найбільш доступна і зручна система глобального обміну інформацією між користувачами не тільки довела свою життєздатність, але і починає витісняти інші способи і канали комунікацій, що відбувається на основі більш низької вартості послуг, високої швидкості передачі даних, більш широкого спектру представленої і переданої інформації.

Сучасні інформаційні системи і системи міжнародного зв'язку дають можливість відійти від традиційної паперової документації як головного носія інформації. Організація електронного інформаційного обміну між користувачами мережі Інтернет має багато переваг, зокрема, можливість швидкого доступу до потрібної інформації в будь-якому місці і в будь-який час, маючи лише вихід до Інтернету.

На сьогоднішній день мільйони людей у світі використовують сучасні технології, які дозволяють їм мати доступ до Інтернету та використовувати безмежні можливості світового інформаційного простору. Саме тому на зміну паперовій документації приходить електронна, з року в рік зростає кількість електронних книг, з’являються електронні бібліотеки.

Багато людей має чим поділитися зі світом, тому переважна більшість сучасних Інтернет-ресурсів повинні мати функціонал, який не лише забезпечує

для користувачів можливість перегляду його вмісту, але також і можливість розміщення та редагування власної інформації.

Інтернет-технології на сьогоднішній день, також як і промислові та фінансові технології, визначають засоби і форму, в яких реалізується спільна діяльність людей заради досягнення певних цілей. Тобто Інтернет-технології це все, що пов'язано з Глобальною мережею, а зокрема і в основному це Web-сторінки в безлічі різних варіантів: форуми, чати, пошта, Інтернет-магазини, новини, фотобанки і т.д. Вміст цих сторінок побудований і працює у відповідності з певними правилами, за певними методам на базі певних програм і технічних засобів (серверів, мереж та ін.).

Перед розробниками сучасних Інтернет-технологій, зокрема, Web-ресурсів, часто постає непросте питання відповідності певним вимогам, а також продуктивності ресурсу, що розробляється. Адже паралельно з впровадженням нових та вдосконаленням існуючих Web-технологій безупинно зростають вимоги до сучасних Web-ресурсів.

Якщо ще 10 років тому переважна більшість Web-сайтів представляла собою набір статичних HTML-сторінок. Сьогодні подібні сайти зустрічаються вкрай рідко, зазвичай так виконані невеликі персональні Web-сайти, а також сайти невеликих компаній, які не претендують ні на що, крім розміщення відносно невеликого обсягу інформації, яка рідко змінюється.

В процесі перетворення Інтернету з набору інформаційних ресурсів на інструмент ведення бізнесу технології створення сайтів істотно змінилися, і на сьогоднішній день більшість Web-сайтів представляють собою набір додатків, яким властиві інтерактивність, засоби персоналізації, засоби взаємодії з клієнтами (наприклад, прийом замовлень і платежів) та партнерами, а нерідко -

і засоби інтеграції з «внутрішніми» корпоративними додатками власника сайту, наприклад, деякої компанії.

Із зростанням складності та об’ємів інформації, яка передається та обробляється в ході взаємодії користувача з Web-ресурсом, враховуючи вимоги до функціоналу, який повинен мати ресурс, актуальним залишається питання продуктивності сучасних Web-ресурсів.

Не зважаючи на те, що на сьогоднішній день більшість користувачів мають доступ до швидкісного Інтернету, враховуючи складність сучасних Web-ресурсів, швидкість завантаження сайтів та роботи з ними залишається актуальним питанням.

Тому актуальним є і питання створення таких Web-ресурсів, які б поєднували у собі зручний інтерфейс користувача з усіма необхідними функціями, можливість одночасної роботи багатьох користувачів, зберігання досить великих об’ємів інформації, і в той же час забезпечували швидкість та надійність роботи із ними.

Метою роботи є дослідження сучасних технологій розробки інтерактивних HTML застосувань та вибір оптимальних методів та засобів розробки Web-порталу магазину автозапчастин.

Для досягнення мети в рамках ВКР необхідно вирішити такі ***завдання:***

- вивчити і проаналізувати предметну область;
- зробити порівняльний аналіз існуючих рішень та обрати оптимальний варіант;
- визначити основні функції Web-порталу;
- спроектувати архітектуру Web-порталу;
- описати інтерфейс користувача, створити макет інтерфейсу;
- розробити алгоритми роботи логіки системи, описати запити, що використовуватимуться в системі та підсистемах.

РОЗДІЛ 1

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Напрямки розвитку сучасних web-технологій

1.1.1 Поява DHTML

На зміну статичним HTML-сторінкам прийшли нові технології, які забезпечили широкі можливості для Web-сайтів. І на сьогоднішній день HTML-сторінки використовуються не лише для відображення деякої статичної інформації, сьогодні – це цілий простір для дій. На сторінках сайтів робляться замовлення, здійснюються розрахунки, демонструється відео, відображається інтерактивна реклама, запускаються онлайн-ігри, тощо.

Важливим кроком у розвитку web-технологій стала поява такої технології як DHTML (Dynamic HTML - Динамічний HTML). Назва DHTML - це комерційний термін, введений Netscape і Microsoft для опису цілого ряду технологій, які були введені в четвертому поколінні Web-браузерів і призначалися для розширення їх динамічних можливостей.

Ці технології були створені для того, щоб обійти ті обмеження, які, як вважалося, існували при розробці Web-сторінок за допомогою HTML. Інтернет досить непогано працював як з текстом, так і з графікою, але люди, які звикли до мультимедіа, чекали чогось більшого. І поява нової технології перевернула уявлення про можливості Web-сторінок.

Створена на основі DHTML сторінка могла змінюватися, не звертаючись до сервера за додатковими даними. Програмісти називають такий код клієнтським, або кодом на стороні клієнта (client-side code).

Користувач, що сьогодні має справу з Web-додатками (а останнім часом - і з Web-сервісами), спілкується з ними за допомогою Інтернет-клієнтів (найчастіше браузерів, але не тільки їх - існують і інші типи клієнтських додатків, наприклад чат-клієнти).

Однією із беззаперечних переваг використання технологій динамічної зміни HTML сторінок без звернення до сервера є швидкість завантаження, дані технології зручні тим, що дозволяють приховувати, відображати і міняти елементи сторінок, не завантажуючи нові. Це значно збільшує швидкість роботи Web-сайту.

Одним з напрямків розвитку сучасних Web-додатків стало розміщення деякої частини логіки додатка (такий як, наприклад, перевірка коректності даних, що вводяться) в самому Web-клієнті, наприклад, в Web-браузері. Зокрема, сучасні Web-браузери здатні інтерпретувати код на скриптових мовах, виконувати Java-аплети і елементи управління ActiveX, використовувати інші спеціалізовані додатки, такі як Macromedia Flash Player.

В даний час більшість браузерів дозволяють працювати з наступними технологіями:

- Каскадні таблиці стилів (CSS). За допомогою CSS визначаються властивості елементів на сторінці.
- Фрагменти коду на JavaScript та інших скрипкових мовах. За допомогою них здійснюється управління функціонуванням об'єктів Web-сторінки.
- Об'єктна модель документа (DOM).

Можна також відмітити, що при розробці Web-додатків часто постає питання відокремлення завдання Web-дизайну від завдань, пов'язаних з реалізацією функціональності додатків. Адже зазвичай при виконанні певних

дій на сторінці, що призводять до зміни її динамічного вмісту, загальний дизайн сторін повинен залишатись без змін.

DHTML додатки, які цілком автономні в браузері і можуть функціонувати без серверної підтримки, такої як база даних, часто змушені звертатися до технологій Single Page Applications, або SPA.

1.1.2 Технології, що застосовуються Web-клієнтами

Розглянемо технології, які використовуються Інтернет-клієнтами, зокрема, сучасними Web-браузерами.

Одна з основних таких технологій – це скриптові мови. Більшість сучасних Web-браузерів здатні інтерпретувати код, написаний на скриптових мовах, таких як VBScript, JavaScript, тощо. Код на цих мовах вбудовується в Web-сторінку і інтерпретується браузером. Типовий приклад застосування скриптових мов - перевірка коректності даних, що вводяться користувачем у відповідні поля HTML-форми, безпосередньо в процесі введення або після нього, без звернення до Web-серверу. Такі приклади застосування скриптових мов можна побачити при заповненні деяких анкет і отриманні повідомлень про те, що не заповнені обов'язкові поля (варто відзначити, що далеко не всі анкети реалізовані подібним чином).

Однак є й інші приклади застосування скриптових мов, що реалізують як чисто дизайнерські ідеї, наприклад кнопки, що змінюють свій вигляд при наведенні на них курсора, «біжучі рядки», так і іншу функціональність, наприклад вбудовані у Web-сторінки засоби звернення до пошукових систем, відображення діалогових панелей, управління іншими об'єктами, вбудованими в Web-сторінку (наприклад, Java-апплетами або елементами управління ActiveX).

Варто зазначити, що код, створений за допомогою скриптових мов, не може працювати самостійно - він виконується в адресному просторі браузера. Крім того, скриптові мови містять обмежений набір засобів (наприклад, вони не володіють засобами доступу до файлової системи).

Наступною технологією, яка використовується з боку клієнта є Java-аплети, які здатні відображати і виконувати майже всі сучасні браузери. Java-аплети – це спеціальні Java-додатки, які користувач отримує у складі Web-сторінки. Ці додатки нерідко включаються до складу Web-сторінок з метою додавання функціональності, яку складно або неможливо реалізувати за допомогою скриптових мов. Аплети можуть виконуватися на всіх платформах, для яких доступна віртуальна Java-машина.

Аплети зазвичай створюються відповідно до правил, що обумовлюють період їхнього існування та способи взаємодії зі своїм оточенням. Найчастіше ці способи дуже обмежені (наприклад, такі операції, як зчитування і запис файлів, за замовчуванням для аплетів заборонені; якщо ж подібні операції необхідні, дозволу на їх виконання для конкретних аплетів і конкретних файлів описуються на клієнтському комп'ютері; мережевий доступ з аплету можливий тільки до того комп'ютера, з якого він був завантажений; запуск інших програм на комп'ютері користувача з аплетів неможливий). Однак аплет здатний зчитувати значення параметрів (наприклад, кольору, шрифтів, файлів з графічними зображеннями, які використовуються при виконанні аплету) з його Web-сторінки, на якій він міститься, і відповідно до цих параметрів змінювати свою поведінку. Крім того, параметри аплету можна змінювати динамічно за допомогою коду на скриптових мовах, що міститься в складі тієї ж сторінки.

Оскільки аплети реалізують виконання коду на комп'ютері клієнта, вони певною мірою є потенційно небезпечним вмістом. Саме тому всі сучасні

браузери мають доступні для користувача засоби обмеження можливостей виконання аплетів.

Деякі з сучасних браузерів (зокрема, Microsoft Internet Explorer) можуть служити контейнерами для елементів управління ActiveX - спеціальних COM-серверів, що виконуються в адресному просторі браузера і також містяться у складі Web-сторінки. Це ще одна важлива можливість браузера як Інтернет-клієнта.

За допомогою елементів управління ActiveX, як і за допомогою Java-аплетів, можна реалізувати будь-яку функціональність, в тому числі і несприятливу для комп'ютера користувача, при цьому, на відміну від Java-аплетів, при виконанні елементів управління ActiveX в загальному випадку немає ніяких обмежень на доступ до файлів і інших ресурсів операційної системи та мережі, а код, що міститься в них, виконується від імені користувача, який їх завантажив. Як і Java-аплети, елементи управління ActiveX можуть зчитувати свої властивості зі сторінки, на якій вони містяться; крім того, властивості елемента управління ActiveX можна міняти динамічно з коду на скриптовій мові, що містяться в складі тієї ж сторінки; в тому ж коді можна обробляти події, що виникають в таких елементах управління.

Звісно, Microsoft Internet Explorer володіє засобами обмеження можливостей виконання елементів управління ActiveX, в тому числі можна обмежити управління ними з коду на скриптових мовах. Проте для контролю безпеки їх виконання є ще один засіб - так званий електронний цифровий підпис. Цифровий підпис поміщається всередину елемента управління ActiveX, для чого потрібна наявність відповідного електронного сертифіката. Електронний підпис, крім відомостей про фірму-виробника, містить і іншу корисну інформацію. Так, наприклад, якщо файл з елементом управління ActiveX після додавання електронного підпису був змінений, то про це буде

негайно повідомлено перед запуском такого елемента управління - при додаванні підпису до елементу управління ActiveX відбувається обчислення контрольної суми відповідного файлу. Проте навіть наявність електронного сертифікату не гарантує відсутності потенційно небезпечного вмісту, а лише дозволяє клієнту встановити його джерело.

Слід пам’ятати про те, що при роботі з елементами управління ActiveX і Java-апплетами користувач може не покладатися на антивірусне програмне забезпечення (неважливо, клієнтське воно чи серверне): ознак, характерних для вірусів (таких як здатність впроваджуватися всередину виконуваних файлів і документів), подібні додатки, як правило, не містять. Щоб запобігти шкідливому впливу подібних елементів можна лише заборонити завантаження або виконання відповідного коду або на рівні налаштувань браузера, або на рівні корпоративних або персональних брандмауерів.

І ще однією поширеною технологією такого роду є додатки Macromedia Flash, котрі на сьогодні являють собою найбільш популярне розширення функціональності Web-браузерів - з їх допомогою багато Web-дизайнерів надають своїм сайтам інтерактивності та оригінальності.

Модель безпеки додатків Flash заснована на тому, що Macromedia Flash Player, як і віртуальна Java-машина, виконує програми в обмеженому адресному просторі, при цьому виконувані програми не мають доступу до файлової системи (крім одного конкретного каталогу, використовуваного Macromedia Flash Player для службових цілей) та інших ресурсів комп'ютера користувача; виключення робиться для мікрофонів і відеокамер, проте користувач повинен дати дозвіл на передачу даних, отриманих з цих пристроїв. Доступ до мережевих ресурсів обмежується доменом, з якого було отримано додаток. Важливою особливістю Flash додатків є те, що вони також можуть управлятися за допомогою коду JavaScript, присутнього на тій же сторінці, де

вони розміщені. Проте сам Macromedia Flash Player для Microsoft Internet Explorer є елементом управління ActiveX і використовує можливості елементів управління ActiveX для доступу до властивостей додатків Flash з скриптових мов.

Крім перерахованих вище найбільш популярних засобів розширення функціональності браузерів є і ряд інших засобів, реалізованих зазвичай у вигляді так званих модулів розширення (plug-in). Оскільки модулі розширення також являють собою виконуваний код, сучасні браузери також мають засоби обмеження можливостей, пов'язаних з їх завантаженням та виконанням.

Усі перераховані засоби розширення функціональності HTML-сторінок можуть бути використані в динамічних сторінках, що генеруються серверними Web-додатками. Так, останнім часом широкого поширення набули засоби створення Web-додатків, що виконуються під управлінням Web-серверів і генеруючих динамічні HTML-сторінки з вбудованим у них кодом на скриптових мовах, призначеним для інтерпретації браузером.

1.1.3 Технології створення серверних Web-додатків

З описаного вище можна зробити висновки, що можливості, пов'язані з виконанням коду в Web-клієнтах, можуть бути істотно обмежені як технологічно, так і за допомогою адміністрування та налаштувань. Це в цілому відповідає повністю виправданим вимогам безпеки. Саме тому, поряд з розвитком засобів розширення функціональності браузерів, розвивалися і технології, пов'язані з виконанням коду додатків не в браузерах, а на самих Web-серверах. Розглянемо найбільш поширені з цих технологій.

Common Gateway Interface (CGI) - це стандартний інтерфейс, що дозволяє виконувати серверні додатки, що викликаються через URL. Вхідною

інформацією для таких додатків служить вміст HTTP-заголовка або тіло запиту, залежно від застосовуваного протоколу. CGI-додатки генерують HTML-код, який повертається браузеру. Свого часу широко використовувався термін «CGI-скрипт», походження якого пояснюється тим, що подібні додатки писалися на скриптових мовах типу Perl, що виконуються, тим не менш, не в браузері, а на сервері. CGI-програми можна створювати за допомогою практично будь-якого засобу розробки, генеруючого консольні додатки для операційної системи, під управлінням якої функціонує Web-сервер.

Основна проблема всіх CGI-додатків полягає в тому, що при кожному клієнтському запиті сервер завантажує цей додаток в окремий адресний простір, а потім ініціює його виконання і вивантаження. Ця особливість обмежує продуктивність додатків і можливість одночасної обробки великої кількості клієнтських запитів.

Проблему обмеженої продуктивності Web-додатків, які виконуються в окремому адресному просторі, можна вирішити, створивши додаток у вигляді бібліотеки, що завантажується в адресний простір Web-сервера і при необхідності залишається там для обробки наступних запитів від інших клієнтів; звісно, в цьому випадку Web-сервер повинен підтримувати завантаження таких бібліотек. Подібні додатки для Microsoft Internet Information Servise носять назву ISAPI (Internet Server Application Program Interface), а для вельми популярного Web-сервера Apache такі бібліотеки називаються Apache DSO (Dynamic Shared Objects).

Недоліком такого підходу є те, що при створенні як CGI-, так і ISAPI-додатків досить складно відокремити завдання Web-дизайну від завдань, пов'язаних з реалізацією функціональності і логіки додатків, - подібні додатки генерують Web-сторінки повністю, тому всі дані, пов'язані з дизайном цих

сторінок, повинні в загальному випадку міститися всередині виконуваного файлу.

Черговим кроком у розвитку технологій створення Інтернет-додатків стала поява засобів, що дозволяють відокремити завдання Web-дизайну від завдань, пов'язаних з реалізацією функціональності додатків. Першою з таких технологій стала Active Server Pages (ASP), побудована на основі ISAPI-фільтра. Основна ідея ASP полягає у створенні Web-сторінок з вбудованими в них фрагментами коду на скриптових мовах. Однак, на відміну від розглянутих вище засобів застосування скриптових мов для розширення функціональності браузерів, зазначені фрагменти коду інтерпретуються не браузером, а сервером (точніше, спеціальною призначеною для цього ISAPI-бібліотекою), і результат виконання цих фрагментів коду виконує заміну самого фрагменту коду в тій версії сторінки, яка передається в користувальницький браузер.

Незабаром після ASP з'явилися й інші технології, що реалізують ідею розміщення всередині Web-сторінки коду, виконуваного Web-сервером. Найбільш відомою з них сьогодні є технологія JSP (Java Server Pages), основна ідея якої - одноразова компіляція Java-коду (сервлета) при першому зверненні до нього, виконання методів цього сервлета і приміщення результатів виконання цих методів в набір даних, що відправляються в браузер. Також ще однією популярною технологією подібного типу є PHP (Personal Home Pages), яка використовує CGI-додатки, що інтерпретують впроваджений в HTML-сторінку код на скриптовій мові.

Новітньою версією технології Active Server Pages є ASP .NET, ключова в архітектурі Microsoft .NET Framework. Основна відмінність цієї технології від ASP з точки зору архітектури додатків полягає в тому, що код, присутній на Web-сторінці, не інтерпретується, а компілюється і кешується, а це сприяє підвищенню продуктивності додатків. За допомогою ASP.NET можна

створювати Web-додатки і Web-сервіси, які не тільки дозволяють реалізувати динамічну генерацію HTML-сторінок, але і інтегруються з серверними компонентами і можуть використовуватися для вирішення широкого кола бізнес-завдань, що виникають перед розробниками сучасних Web-додатків.

У загальному випадку клієнтом Web-сервера може бути не тільки персональний комп'ютер, оснащений звичайними Web-клієнтами (наприклад, Web-браузером), але і мобільні пристрої, що відрізняються обмеженим розміром екрану, малим обсягом пам'яті, а нерідко і неможливістю відображення графіки. Для цих пристроїв існують свої протоколи передачі даних (Wireless Access Protocol, WAP) і відповідні мови розмітки (WML, Wireless MarkupLanguage, CHTML, Compact HTML і т.п.). При цьому необхідно передавати дані на мобільний пристрій у відповідному форматі, для чого нерідко створюються спеціальні сайти (наприклад, з підтримкою WAP та WML). Більш зручним є створення додатків, які здатні генерувати той чи інший код залежно від типу клієнта. Саме такий підхід і реалізований в Microsoft ASP.NET.

З ростом обсягу використовуваних даних і числа відвідувачів Web-сайтів зростають вимоги до надійності, продуктивності і масштабованості Web-додатків. Для задоволення цих вимог бізнес-логіка, що реалізується в Web-додатку, а також сервіси обробки даних і реалізації транзакцій, відокремлюються від інтерфейсу додатків і переносяться на сервер додатків у вигляді бізнес-об'єктів. Сервери додатків і відповідні бізнес-об'єкти можуть бути різного типу (найбільш поширеними з них сьогодні є сервери, що підтримують специфікацію Java2 Enterprise Edition, і сервери, що базуються на технологіях COM і Microsoft.NET). Бізнес-об'єкти часто надають доступ до даних корпоративних інформаційних систем або реалізують яку-небудь

частину їх функціональності, здійснюючи функції інтеграції Web-додатків з іншими додатками, що використовуються на підприємстві.

1.2 Застосування технологій Single Page Application (SPA)

1.2.1 Загальна архітектура технологій SPA

Модель SPA орієнтована на створення інтерактивної Web-сторінки,що не перезавантажується під час сеансу роботи додатка. Всі взаємодії користувача та зміни стану додатка обробляються в контексті одного Web-документа. Архітектура SPA-застосунку є модульною. Модульна архітектура складається з наступних компонентів:

- Базова бібліотека - це основа, на якій будується додаток.
- Модулі. Модуль є незалежною одиницею функціональності на сторінці, яка складається з бізнес-логіки й певних елементів інтерфейсу.
- AMD - API для найбільш частих дій, які може виконувати модуль. У число цих дій входять наступні: взаємодія з іншими модулями, створення запитів Ajax, приєднання і від'єднання обробників подій.
- Ядро. Ядро керує життєвим циклом модуля (запуск і зупинка), взаємодією між модулями та обробкою помилок.

Головними складовими модуля є бізнес-логіка та елементи інтерфейсу. На сьогоднішній день в модулях SPA-застосунків часто використовується бібліотека jQuery, яка фокусується на взаємодії JavaScript та HTML. Бібліотека jQuery допомагає легко отримувати доступ, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними, а також надає зручний API по роботі з AJAX.

Модульна архітектура має наступні переваги:

- пакети визначаються одним спільним публічним інтерфейсом;

- імена є локальними всередині одного пакета;
- деталі реалізації недоступні поза межами пакету;
- порядок завантаження не має значення;
- залежності задаються явно;
- кожен файл представляє один модуль;
- запуск з командної строки без браузера.

В моделі SPA сервер виконує наступні функції:

- віддає статичну частину клієнту;
- віддає дані (XML, JSON і т.п.).

Клієнт в свою чергу:

- агрегує дані;
- будує об'єкти;
- шаблонізує;
- створює DOM;
- контролює View.

Таким чином, вся обробка даних відбувається саме на стороні клієнта.

Варто відмітити, що іноді при реалізації SPA архітектури використовується ще хмарний сервіс. В даному випадку сервер виконує функцію передачі та зберігання статичних даних, а також перенаправлення запитів. А зберігання даних, їх обробка та обчислення виконується на хмарі.

З усього вищезазначеного можна зробити висновок, що додаток, побудований з використанням технологій SPA - це web-додаток, розміщений на одній сторінці, яка для забезпечення роботи завантажує всі javascript-файли (модулі, віджити, контроль і т.д.), а також файли CSS разом із завантаженням самої сторінки.

Архітектура SPA ефективна і для досить складних додатків, які містять багатий функціонал, наприклад, система електронного документообігу, і в яких

кількість файлів зі скриптами може досягати декількох сотень, а іноді навіть тисяч. Досягається це за рахунок наявності інтерфейсу AMD. При завантаженні сайту всі сотні й тисячі файлів зі скриптами не завантажуються одразу, AMD реалізує можливість завантаження скриптів на вимогу. Наприклад, якщо для «головної сторінки» Web-сайту, написаного за допомогою SPA знадобилося 3 скрипта, вони будуть завантажені одразу перед стартом програми. А якщо користувач перейшов на іншу сторінку Web-сайту, наприклад, «Контакти», то принцип AMD завантажить модуль (скрипт + розмітка) тільки перед переходом на цю сторінку.

Головними причинами, чому зручно застосовувати технології SPA при розробці Web-ресурсів, таких як Web-портал, який повинен бути розроблений в ході роботи, є:

- Додатки написані з використанням технологій SPA відмінно працюють на пристроях як стаціонарних, так і мобільних. Персональні комп'ютери, планшети, смартфони, і навіть прості телефони (деякі) можуть безперешкодно працювати з сайтами побудованими за принципом SPA. Можливість роботи на великій кількості пристроїв дозволяє значно розширити аудиторію користувачів у порівнянні з використанням інших підходів.
- Можливість розробки багатого користувальницького інтерфейсу. Оскільки web-сторінка одна, побудувати багатий, насичений користувальницький інтерфейс набагато простіше. Простіше зберігати інформацію про сеанс, управляти станами представлень (views), керувати анімацією (в деяких випадках), тощо.
- Технології SPA дозволяють істотно (у рази) скоротити повторні завантаження одного і того ж контенту. Якщо сайт (або портал) використовує шаблон, то разом з основним змістом якої-небудь сторінки відвідувач сайту обов'язково завантажує розмітку шаблону. Таким чином досягається

скорочення витрат часу і ресурсів на кешування у порівнянні з іншими підходами.

Існує велика кількість базових бібліотек (фреймворків - від англійського слова framework - "основа, структура, каркас"), які реалізують принцип Single Page Application. Їх призначення: забезпечення базових принципів для SPA розробки, мінімізуючи трудовитрати на рішення універсальних задач; використання досвіду створення сайтів багатьох програмістів.

Принципи будь-якого фреймворку, який реалізує парадигму SPA повинні дотримуватися наступних понять і визначень:

- SPA підтримує клієнтську навігацію. Всі переходи користувача між модулям-сторінкам однозначно фіксуються в історії навігації, причому навігація є «глибокою», тобто, якщо користувач скопіює і відкриє посилання на внутрішню модуль-сторінку в іншому браузері або вікні, він потрапить на відповідну сторінку.
- SPA розміщується на одній web-сторінці, отже все необхідне для роботи сайту (порталу): скрипти і стилі повинні бути визначені в одному місці проекту - на єдиній web-сторінці.
- SPA зберігає постійний стан (важливі змінні) роботи клієнта (клієнтського скрипта) в кеші браузера або в Web Storage.
- SPA завантажує всі скрипти, що вимагаються для старту додатку при ініціалізації web-сторінки.
- SPA поступово підвантажує модулі за вимогою.

Фреймворки, реалізують SPA, використовують різні патерни проектування. Для того, щоб визначитись, який з них краще підійде для вирішення поставленої задачі, варто більш детально розглянути кожен з них.

Сучасні SPA-додатки зазвичай структуровані наступним чином:

- Жодний стан / дані не зчитуються з DOM. Додатки повертають HTML та операції над елементами, але нічого не зчитують з DOM. При зберіганні стану в DOM складно дуже швидко отримати доступ до елементу: набагато зручніше мати одне місце де зберігаються дані і надавати інтерфейс доступу до даних, особливо коли ті ж самі дані повинні бути представлені в декількох місцях в інтерфейсі.
- Замість того щоб зберігати дані в DOM або у деяких об'єктах, є велика кількість моделей в пам'яті, які представляють всі стани / дані у додатку.
- Подання відображають зміст моделей. Коли кілька подань залежать від однієї моделі, не потрібно вручну відстежувати кожне залежне від моделі подання. Замість цього є система подій змін, через які подання отримують повідомлення про зміни від моделей і змінюються відповідно до змін моделі.
- Замість того, щоб робити речі глобальними, створюються невеликі підсистеми, що не залежать одна від одної.
- Для забезпечення багатоплатформенності мінімізується та ізолюється DOM-залежний код.

1.2.2 Порівняння патернів проектування

Model-View-Controller (MVC) - це фундаментальний патерн, який знайшов застосування в багатьох технологіях, дав розвиток нових технологій і кожен день полегшує життя програмістам. MVC складається з трьох компонентів: подання (View, користувальницький інтерфейс), модель (Model, бізнес-логіка) і контролер (Controller, містить логіку на зміну моделі при певних діях користувача). Схема патерну MVC наведена на рисунку 1.1.

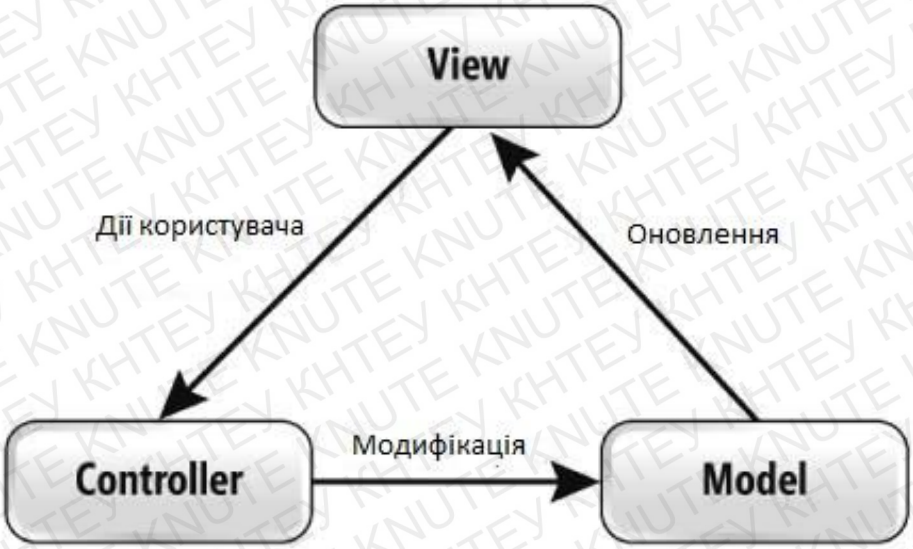

Рис. 1.1. Схема патерну проектування MVC

Основна ідея цього патерну полягає в тому, що і контролер і подання залежать від моделі, але модель ніяк не залежить від цих двох компонент. Саме це дозволяє розробляти і тестувати модель, нічого не знаючи про подання і контролер. В ідеалі контролер так само нічого не повинен знати про подання (хоча на практиці це не завжди так), і в ідеалі для одного подання можна перемикати контролери, а так само один і той же контролер можна використовувати для різних подань (так, наприклад, контролер може залежати від користувача, який увійшов в систему). Користувач бачить уявлення, на ньому ж виконує якісь дії, ці дії подання перенаправляє контролеру і підписується на зміну даних моделі, контролер в свою чергу виконує певні дії над моделлю даних, подання отримує останній стан моделі і відображає її користувачеві.

Ознаки контролера:

- контролер визначає, яке подання має бути відображено в даний момент;
- події подання можуть вплинути тільки на контролер, контролер може вплинути на модель і визначити інше подання.

Контролер перехоплює подію ззовні (викликану діями користувача) і відповідно до закладеної в нього логіки реагує на цю подію, змінюючи модель, за допомогою виклику відповідного методу. Після зміни модель використовує подію щоб повідомити про зміни, і всі підписані на цю подію подання, отримавши повідомлення, звертаються до моделі за оновленими даними, після чого їх і відображають.

Даний патерн використовується, наприклад, в MVC ASP.NET, Angular JS.

Model-View-Presenter (MVP) патерн також складається з трьох компонент. Але в даному випадку поданням немає потреби підписуватися на зміни моделі, тепер контролер, названий в даному підході Presenter (презентер) дає знати поданням про зміни.

Даний підхід дозволяє створювати абстракцію подання. Реалізувати даний патерн можна за допомогою винесення інтерфейсів подання. У кожного подання буде інтерфейс з певним набором методів і властивостей, потрібних презентеру, який в свою чергу ініціалізується даним інтерфейсом, підписується на події подання і за необхідності віддає дані. Даний підхід дозволяє розробляти програми з використанням методології TDD (Test-driven development).

Ознаки презентера:

- двостороння комунікація з поданням;
- представлення взаємодіє безпосередньо з презентером, шляхом виклику відповідних функцій або подій презентера;

- презентер взаємодіє з поданням шляхом використання спеціального інтерфейсу, реалізованого поданням;
- один примірник презентера пов'язаний з одним поданням (на відміну від MVC).

Кожне подання має реалізовувати відповідний інтерфейс. Інтерфейс подання визначає набір функцій і подій, необхідних для взаємодії з користувачем. Презентер повинен мати посилання на реалізацію відповідного інтерфейсу, яке зазвичай передають в конструкторі.

Логіка подання повинна мати посилання на екземпляр презентера. Всі події подання передаються для обробки в презентер і практично ніколи не обробляються логікою подання (в т.ч. створення інших подань).

Приклад використання цього патерну Windows Forms.

Патерн Model-View-ViewModel (MVVM) використовується в WPF і Silverlight. Тут також присутні три компоненти: модель (як і в MVC представляє дані предметної області), подання (відображає модель подання і посилає дії користувача моделі подання) і третій компонент - додаткова модель під назвою ViewModel (модель подання, представляє абстрактне відображення подання). Схема патерну MVVM наведена на рисунку 1.2.

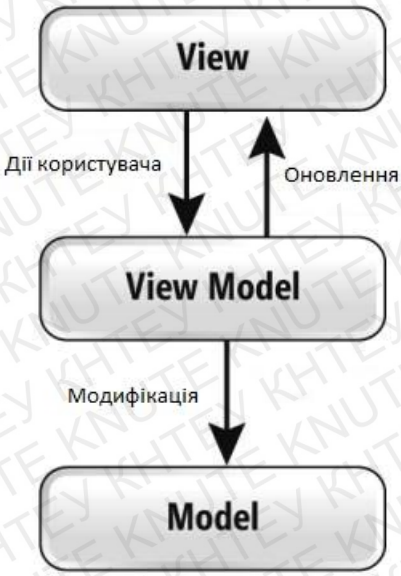

Рис. 1.2. Схема патерну проектування MVVM

Даний патерн підходить до таких технологій, де присутня двостороння синхронізація елементів управління на модель, як в WPF. Відмінність від MVP та MVC патернів полягає в тому, що властивості подання повинні знаходитись не в поданні, а в моделі подання (ViewModel), і вони повинні синхронізуватися з необхідними полями в поданні. В цьому і є основна ідея WPF. Модель подання - це деякий суперконвертор, який перетворює дані моделі в подання, в ньому описуються основні властивості подання, а також логіка взаємодії з моделлю.

Ознаки моделі подання:

- двостороння комунікація з поданням;
- модель подання - це абстракція подання, зазвичай це означає, що властивості подання збігаються з властивостями моделі подання/моделі;

- модель подання не має посилання на інтерфейс подання, зміна стану моделі подання автоматично змінює подання і навпаки, оскільки використовується механізм скріплення даних.
- один екземпляр моделі подання пов'язаний з одним відображенням.

При використанні цього патерну, подання не реалізує відповідний інтерфейс. Подання має мати посилання на джерело даних, яким у даному випадку є модель подання. Елементи подання пов'язані з відповідними властивостями і подіями моделі подання. У свою чергу, модель подання реалізує спеціальний інтерфейс, який використовується для автоматичного оновлення елементів подання.

В JavaScript-інфраструктурі MVVM поданням є розмітка, а моделлю подання - код.

1.3 Висновки до розділу 1

Функціональність браузерів набрала досить значного розвитку. Для нового користувача зараз важливо, щоб браузер показував чітку, красиву картинку та швидко надавав доступ до інформації. У розділі 1 було проаналізовано таку технологію як DHTML, її переваги та можливості. Описані можливі функції та медоти використання такої технології. Розглянуто технології створення інтернет додатків, можливості сучасних web-браузерів, та як вони можуть змінювати код скриптових мов. Було порівняно основні патерни проектування, описані їх функції та ознаки.

РОЗДІЛ 2
ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ HTML НА ПРИКЛАДІ WEB-ПОРТАЛУ МАГАЗИНУ АВТОЗАПЧАСТИН

2.1 Визначення загальних вимог до Web-порталу

Для того, щоб будь-який портал працював ефективно, надійно і гнучко, а також мав досить тривалий життєвий цикл, він повинен задовольняти наступним загальним вимогам:

- інтегрованість, наявність можливості інтеграції з іншими прикладними системами та базами даних;
- багатоплатформеність (різні операційні системи та апаратні засоби);
- масштабованість за кількістю користувачів, обсягом збережених даних, інтенсивності обміну даними, швидкості обробки запитів і даних, способам забезпечення доступу і т.п .;
- надійність (постійна працездатність, мінімізація часу простою і часу відновлення, забезпечення засобами збереження і відновлення даних, резервування);
- безпека - захист даних, додатків і транзакцій, включаючи внутрішню і зовнішню захист для запобігання несанкціонованого доступу до мережі; унікальність реєстрації при санкціонуванні доступу до захищених і персоналізованих портальних додатків на рольовій основі;
- забезпечення швидкої реакції порталу на запити користувачів.

Враховуючи переваги технологій SPA, задля задоволення таких вимог як багатоплатформеність, масштабованість та забезпечення швидкої реакції

порталу на запити користувачів, при розробці доцільно використовувати саме ці технології.

Для забезпечення виконання вимог до безпеки порталу, повинна бути створена система авторизації користувачів, можливість підтримки різних ролей користувачів.

Оскільки для розробки використовуватиметься одна із сучасних платформ, які зазвичай містять засоби інтеграції з іншими продуктами, наприклад, з БД, на цьому питанні можна детально не зупинятись.

Для забезпечення надійності роботи з ресурсом з програмних засобів можна відмітити наявність перевірку помилок, чітко визначити режими доступу до ресурсів (щоб уникнути такої ситуації, коли, наприклад, два користувача одночасно вносять зміни до однієї статті), а також можливість періодичного авто збереження статті, що редагується.

2.2 Визначення функціоналу Web-порталу

Після того як визначено загальні вимоги, які повинен задовольняти будь-який Web-портал, можна перейти до визначення функціональних вимог до порталу, що розробляється.

Портал повинен виконувати наступні функції:

- зберігання великих об’ємів інформації, з часом додаватимуться нові статті, відповідно об’єми інформації, розміщеної на порталі, зростатимуть;
- підтримка пошуку – забезпечує користувачам та гостям доступ до широкого спектру пошуку всередині порталу;
- застосування засобів авторизації доступу до інформації для різних груп користувачів;
- організація інтерактивних діалогів користувачів порталу (чати, дискусії, голосування, коментарі і т.д.);

- персоніфікація доступної інформації у відповідності з правами та привілегіями користувачів, ідентифікація користувача відбувається при аутентифікації;
- перегляд інформації (яка інформація доступна до перегляду, визначається групою користувача);
- розміщення, видалення та редагування елементів (статей);
- інтерактивне редагування інформації, в режимі редагування статті користувачу повинні бути доступні перевірка тексту на помилки, додавання формул, спец символів, графіків.

2.3 Ролі та права користувачів

Засоби персоналізації дозволяють вибирати інформаційне наповнення сторінок для різних користувачів на основі профайлів цих користувачів або на основі бізнес-логіки.

Web-портал призначений для магазину автозапчастин, тому важливим є питання захисту інформації, яку користувачі розміщуватимуть на порталі. Авторизація користувачів та чітке визначення ролей і відповідного набору привілегій для кожної ролі - важлива складова комплексу засобів безпеки.

Нижче наведено список ролей користувачів порталу зі списком доступних для них прав:

1. Гість (незареєстрований користувач):
- перегляд опублікованих статей;
- пошук інформації (серед опублікованих статей).

2. Зареєстрований користувач:
- перегляд опублікованих статей;
- додавання коментаріїв до опублікованих статей;
- пошук інформації (серед опублікованих статей та своїх статей);

- участь в дискусіях та чатах на порталі;
- доступ до особистого кабінету.

2.4 Архітектура Web-порталу

Сайт розроблений із застосуванням html/css/js з підключенням бібліотек bootstrap та jQuery на стороні клієнта, та Node.js на стороні сервера.

Початок проекту розпочався з вибору технології, за допомогою якої буде відображатись візуальна складова проекту.

Для реалізації був вибраний фреймворк bootstrap версії 4.1.1, та бібліотека jQuery 3.6.0 для забезпечення інтерактивності веб-додатку на стороні клієнта.

Структура папок проекту була сформована наступним чином:

Головна сторінка – index.html.

Головний файл запуску проекту на сервері на платформі Node.js - App.js .

На сервері усі файли, які будуть доступні для браузера, розташовані у папці public:

папка «Css» - стилі застосовані у проекті

папка «Img» - картинки товарів головної сторінки

папка «Items» - html-файли категорій товарів

«Js» – javascript код

Поруч з папкою «public» розташована папка «orders», у якій будуть зберігатися усі замовлення, які надішли до сервера. Кожне замовлення окремим файлом у форматі json.

Запуск проекту виконується шляхом запуску команди “node app” у консолі, у директорії проекту.

Запускається процес, який створює локальний сервер на системі користувача, доступний у браузері за адресою http://localhost:5000/.

На реальному сервері треба налаштувати перенаправлення з реального ip сервера на порт 5000.

Для гарного візуального сприйняття контенту на головній сторінці було обрано згрупувати контент сайту групами та виділити у окремі тематичні блоки, що відображено на головній сторінці сайту, де користувач має змогу обрати інтересуючу його категорію товару, та перейти до неї за посиланням. Також на кожній сторінці сайту зверху присутня окрема область, яка містить кнопку «Кошик», за допомогою якої користувач має змогу переглянути список обраних товарів, виконати редагування цього списку.

Початок проекту розпочинається зі створення стандартного шаблону для html-сторінки, та підключенню до проекту бібліотек bootstrap, jquery та файлів style.css та script.js (Додаток А).

Наступний крок, додаємо код, який відповідає за відображення кнопки «Кошик» (Додаток Б).

Результат у браузері рис. 2.1

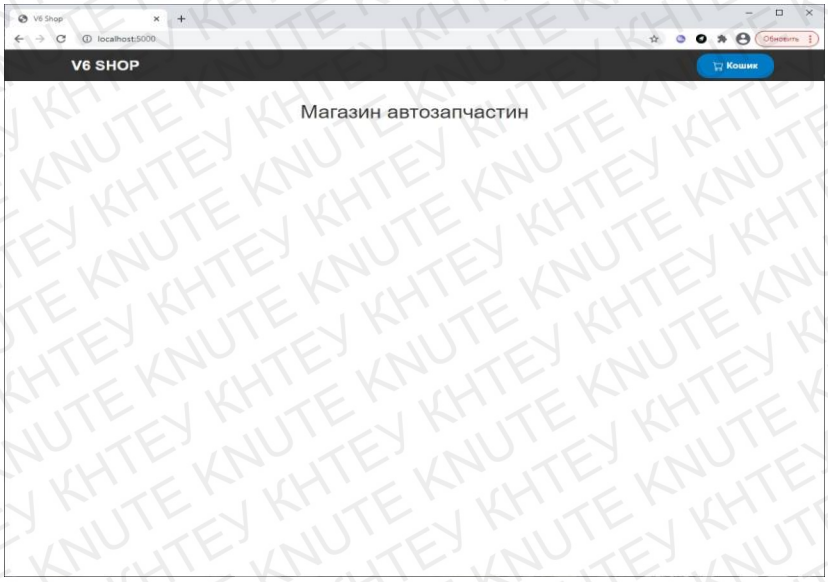

Рис. 2.1. Кнопка кошик

Оформлення структури у html для додавання карток-категорій (Додаток В).

Структура html-коду для кожної картинки товару головної сторінки див. Додаток Г

Результат у браузері рис. 2.2

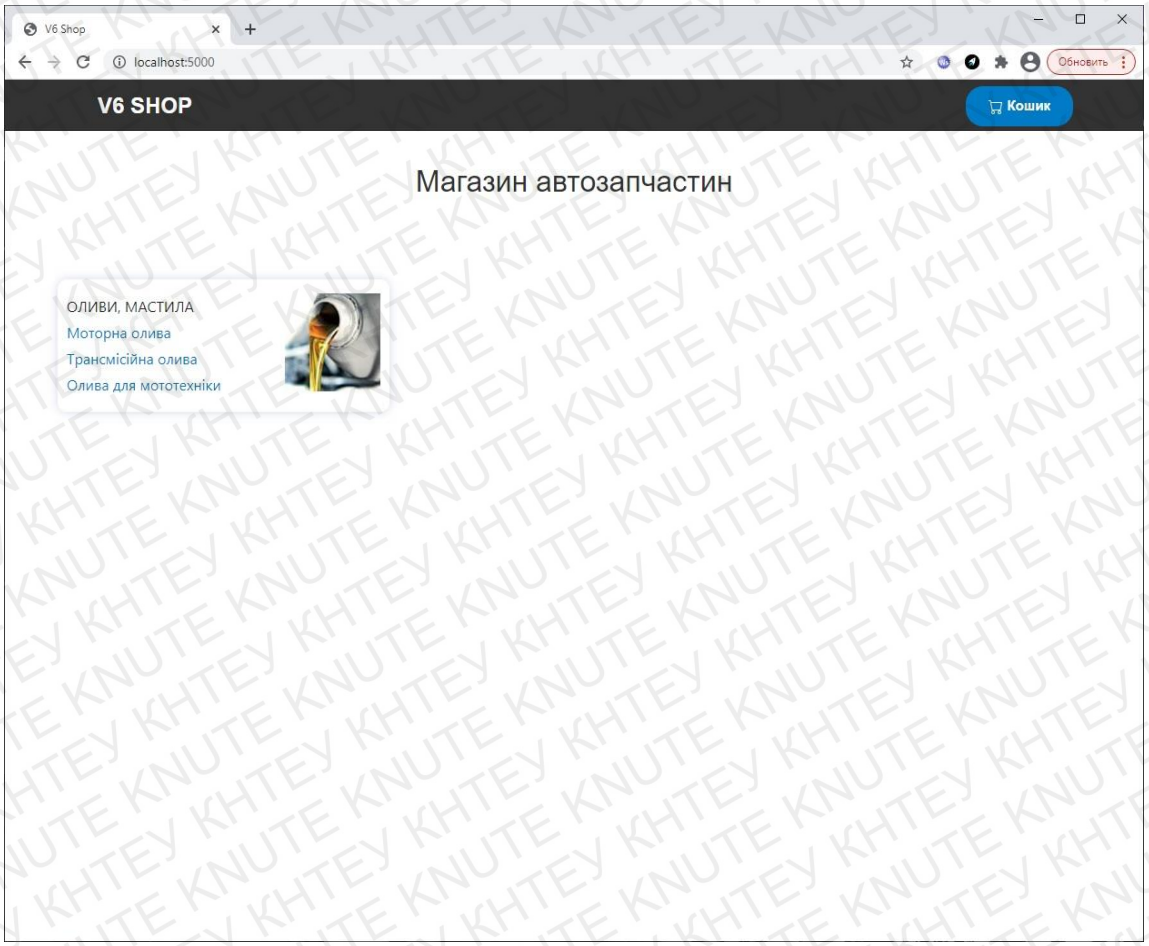

Рис. 2.2. Категорія товарів

Після додавання усіх груп-категорій товарів до html-файлу головної сторінки, результуючий код сторінки див. Додаток Д.

Головна сторінка сайту відображає комплектуючі, які згруповані за категоріями, для кращого візуального сприйняття та швидкого пошуку необхідного товару.

Так виглядає головна сторінка веб-сайту у вікні браузера

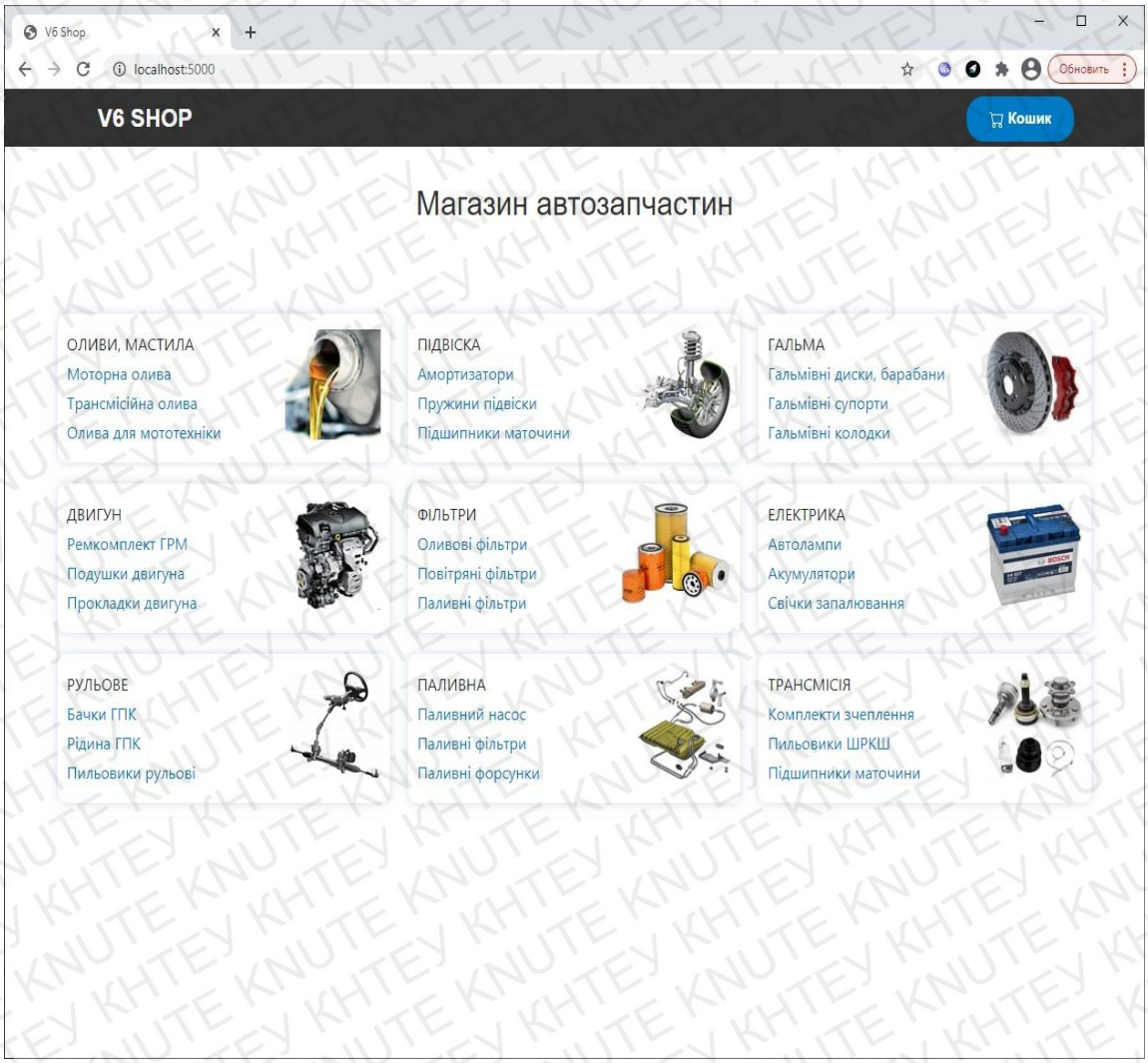

Рис. 2.3. Головна сторінка веб-сайту

На цьому етапі створення головної сторінки завершено.

При переході за посиланням з головної сторінки переходимо до «внутрішніх» сторінок з картками категорії товару.

Формуємо картку товару

Результат у браузері рис. 2.4

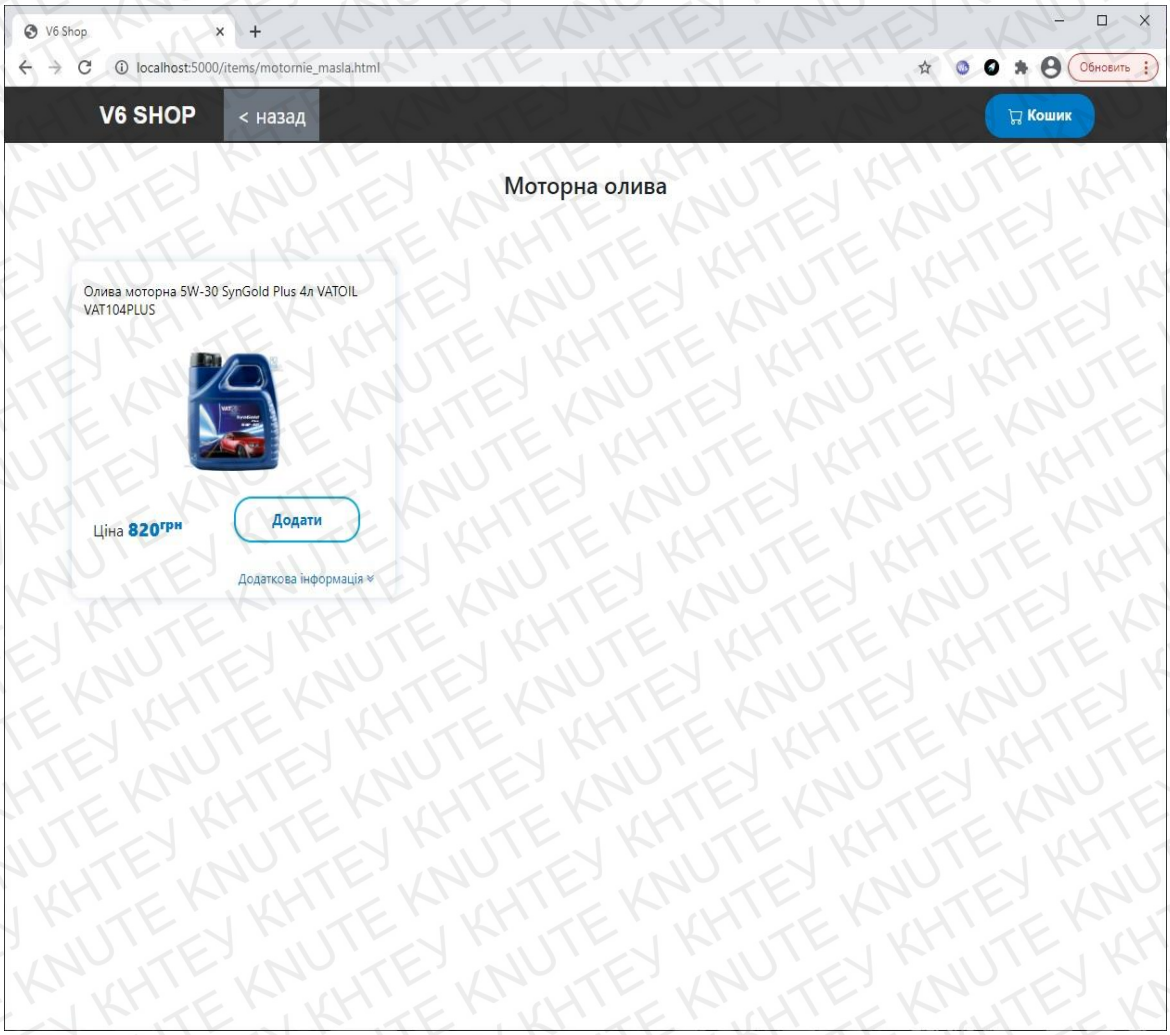

Рис. 2.4. Картка товару

Html-код картки товарів знаходиться у Додатку Е.

Додаємо інші картки до коду сторінки, результат рис. 3.5

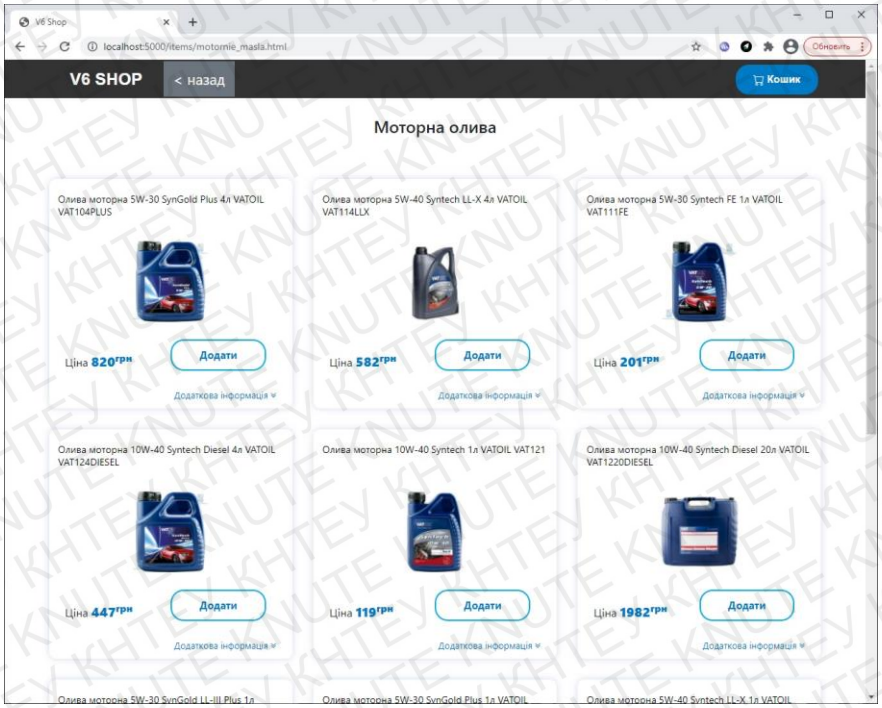

Рис. 2.5. Товари в категорії

На кожній картці товару присутнє фото, опис, ціна. Для отримання додаткової інформації треба у нижньому правому кутку натиснути «Додаткова інформація», якщо вона доступна. Для того, щоб сховати додаткову інформацію, натиснути ще раз. Результат у браузері рис. 2.6, рис. 2.7.

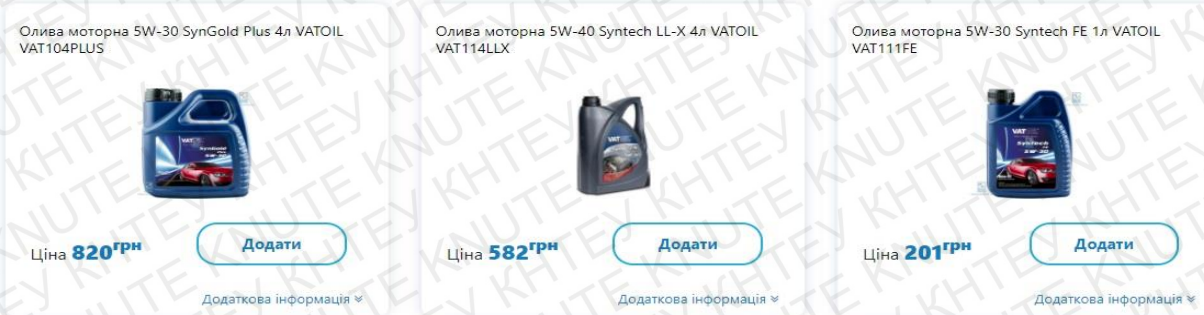

Рис. 2.6. Кнопка «додаткова інформація»

Рис. 2.7. Кнопка «додаткова інформація» в розгорнутому вигляді

Код картки товару у Додатку Е

Аналогічним чином додаємо інформацію до всіх груп товарів за відповідними категоріями.

Для завершення візуального оформлення, додаємо назву магазину «V6 SHOP» на усі сторінки та кнопку «Назад» на усі внутрішні сторінки сайту для зручної навігації.

2.5 Висовки до розділу 2

У другому розділі були описані загальні вимоги до веб-порталу для його ефективного, надійного та безпечного функціонування. Визначені основні функції веб-порталу та ролі користувачів на ньому. Також була розписана детальна архітектура веб-порталу.

РОЗДІЛ 3

ПРОЕКТУВАННЯ WEB-ПОРТАЛУ МАГАЗИНУ АВТОЗАПЧАСТИН

3.1 Побудова макету інтерфейсу користувача

Опис роботи та функціонала спливаючого вікна «Кошик»:

За допомогою кнопки «Додати» інформація з картки товару передається до кошика (до масиву де зберігаються відомості об обраних товарах). Оновлюється данні у «Кошику». З'являється спливаюче вікно «Кошик» з обраними товарами, тут можна обрати бажану кількість кожної позиції та видалити зайві з цього списку.

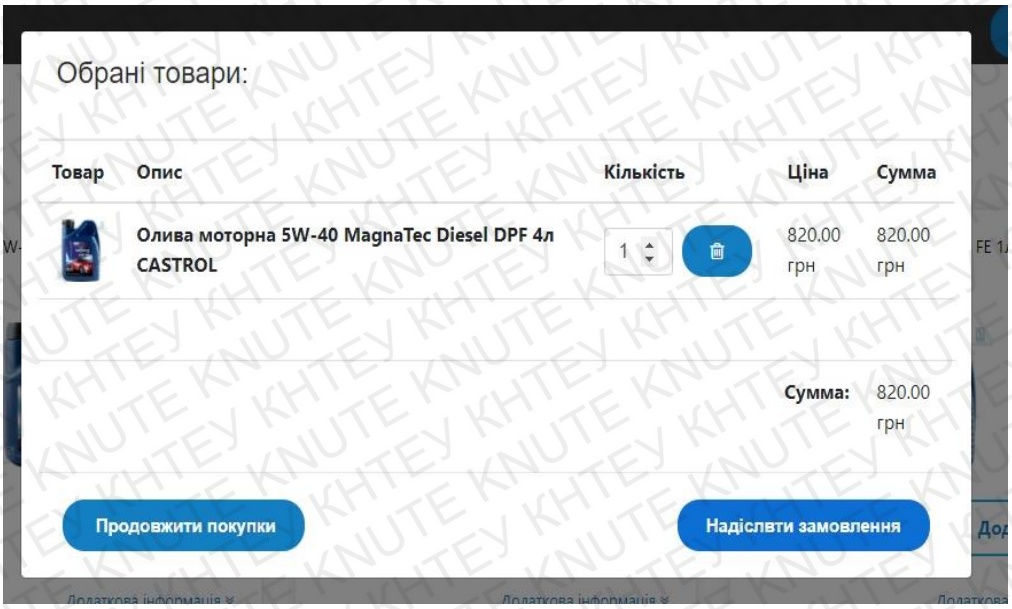

Рис. 3.1. Кошик

Html-код який відповідає за відображення вікна кошика знаходиться у Додатку З.

					КНТЕУ 121 07-02.БР		
Зм.	Аркуш	№ докум	Підпис	Дата			
Зав. кафедри		Криворучко О.В.			Розробка веб-додатку магазину автозапчастин	Стадія	Аркуш	Аркушів
Керівник		Пашорін В.І.				РЗ	37	49
Гарант		Цензура М.О.				Факультет інформаційних технологій, 4 курс, 7 група		
Розробник		Афанасьєв О.А.			Проектування Web–порталу магазину автозапчастин			

При будь яких змінах у кошику дані заносяться до локального сховища браузера LocalStorage, що дозволяє зберігати інформацію про товари при переході з однієї сторінки на іншу та при перезавантаженні браузера.

Щоб повернутись до вибору товарів, можна натиснути кнопку «Продовжити покупки» або клікнути мишкою за межами спливаючого вікна.

Також при зміні кількості товару по кожній позиції автоматично підраховується нова сума замовлення.

При натисненні кнопки «Видалити» (значок) - товар вилучається зі списку, якщо, це була остання позиція, кнопка «Надіслати замовлення» стає неактивною, з’являється напис «Кошик порожній».

При натисканні кнопки «Надіслати замовлення» користувачу пропонується заповнити контактну інформацію, та натисканням кнопки «Відправити» інформація відправляється на сервер NodeJS, звідки на почту власника веб-ресурсу.

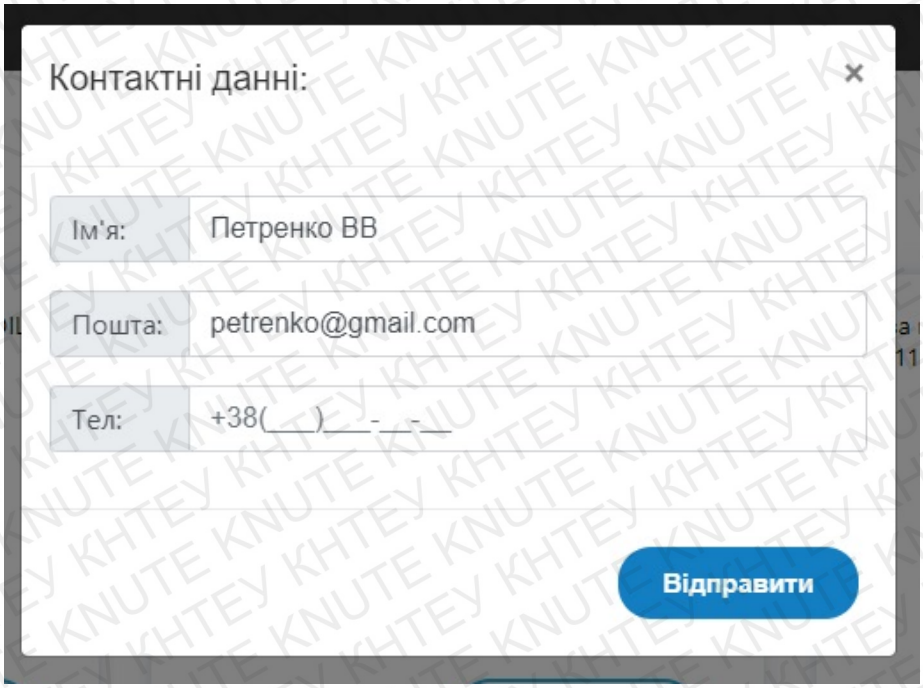

Рис. 3.2. Контакти покупця

Якщо поля форми збору контактних даних не заповнені, при натисканні кнопки «Відправити» ті поля, які порожні, будуть підсвічені червоним кольором, відправлення не відбудеться доки користувач їх не заповнить даними.

Html-код який відповідає за відображення вікна вводу контактних даних знаходиться у Додатку І.

Якщо данні успішно відправлені, користувач отримує відповідно повідомлення.

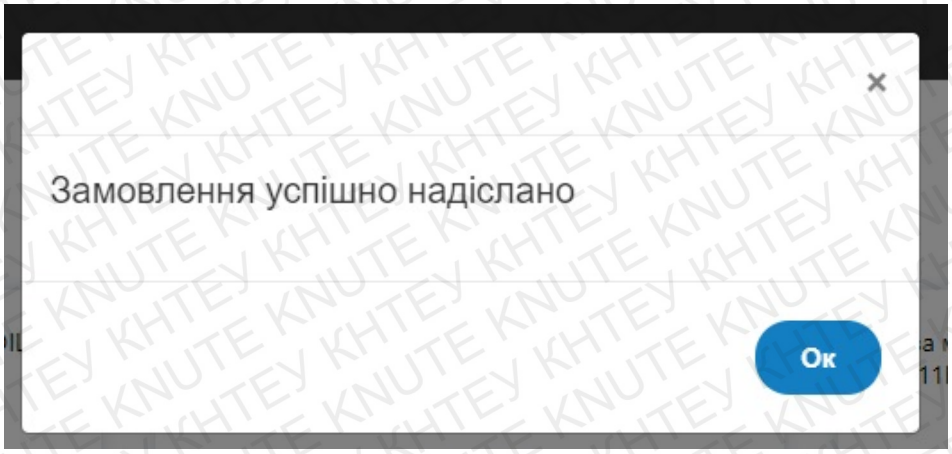

Рис. 3.3. Повідомлення про надсилання замовлення

Код файлу script.js, який відповідає за інтерактивність елементів на веб-сторінці, виконує калькуляцію сум у кошику та забезпечує відправлення даних до серверу знаходиться у Додатку Л.

3.2 Серверна частина

Для реалізації функціоналу сервера була обрана платформа Node.js, та мінімалістичний фреймворк «express», який виконує функцію роздачі файлів, та отримання даних від замовлення клієнта.

Для обробки даних підключена бібліотека «body-parser».

Для відправлення даних на почту обрана бібліотека «nodemailer», яка забезпечує відправлення листів па пошту gmail.

Приклад отриманого замовлення на пошту, рис. 3.4:

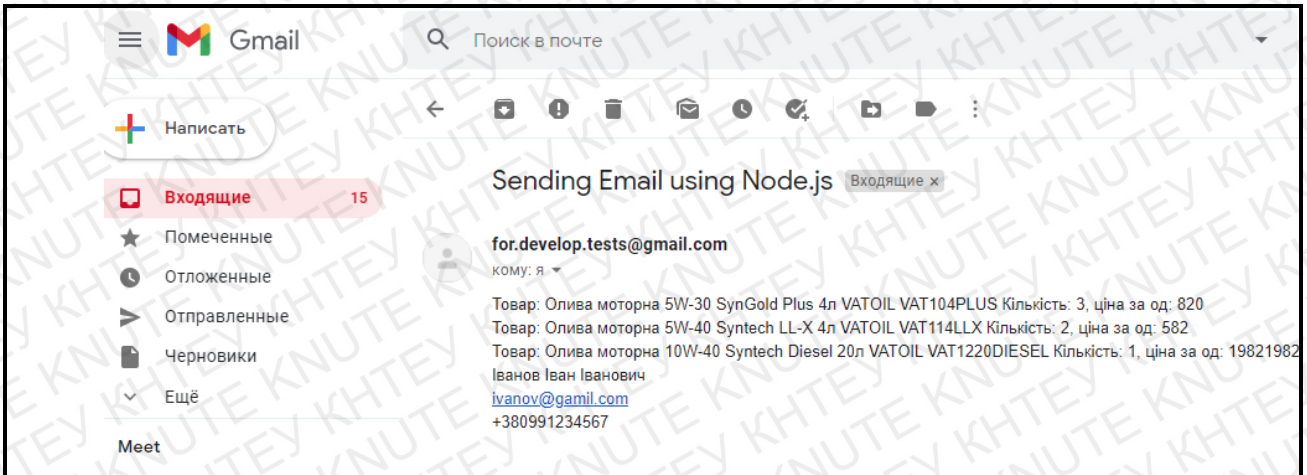

Рис. 3.4. Лист замовлення товару

Код серверної частини, файл app.js, знаходиться у Додатку К.

Також при отриманні даних від клієнта кожне замовлення конвертується у формат json та зберігається в папку «orders» на сервері. Ім'ям файла виступає час, коли він був створений у вигляді timestamp (приклад: 1620568193820). Таким чином унеможливлюються конфлікти з іменами файлів

3.2.1 Панель керування адміністратора

Панель керування Адміністратора знаходиться за адресою http://localhost:5000/admin.html.

При завантаженні сторінки шляхом javascript та технології ajax відбувається запит до серверу, який формує масив даних з файлів, які знаходяться у папці «orders», та відправляє їх користувачу-адміністратору.

З цих даних шляхом javascript формуються окремі «картки» по кожному замовленню, та відображається на сторінці у вигляді рядків.

Кожен рядок інтерактивний та має такий функціонал:

- Кнопка замовлення, яка при натисканні відображає детальну інформацію по замовленню.

Вона має напис: «Замовлення від…» та час, коли було зроблене замовлення.

Детальна інформація відображає ім’я користувача, пошту, телефон користувача, а також перелік усіх позицій, які були замовлені з відображенням картинок товару, найменування, кількості, ціни за одиницю, загальну суму по кожній позиції та загальну суму усього замовлення.

- Кнопку для зміни статусу замовлення. Можливі статуси «Нове замовлення», «Обробляється», «Виконано». При зміні статусу, він оновлюється на сторінці та відправляється запит на сервер, де також здійснюється відповідно зміна у файлі у папці «orders».

- Кнопку для видалення замовлення у вигляді піктограми кошика для сміття. При натисненні кнопки видаляється «картка» замовлення на сторінці та відправляється запит на сервер для видалення запису. Сервер видаляє файл замовлення.

На рис 3.5 зображено панель адміністратора, на якій присутні 4 замовлення:

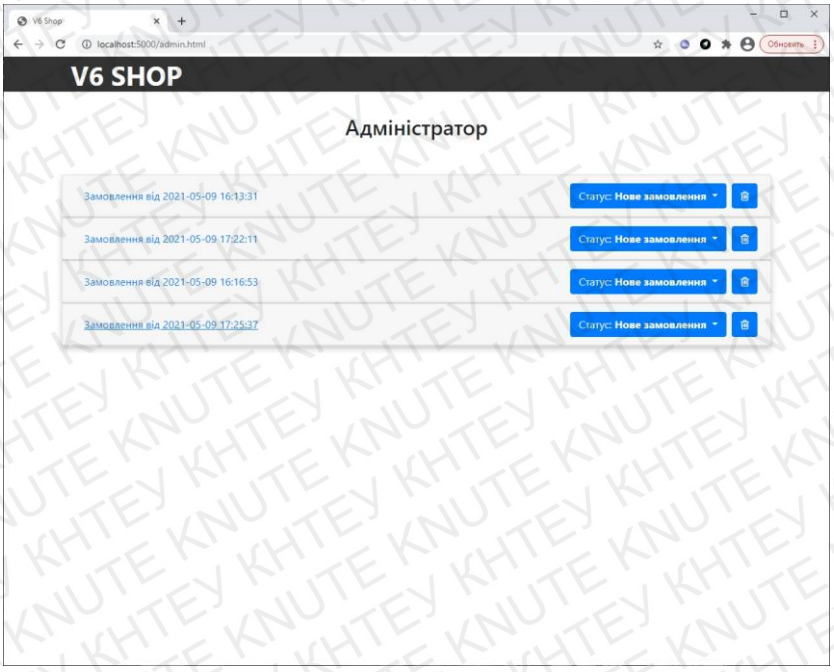

Рис. 3.5. Панель адміністратора

На рис. 3.6 Зображено відкрите меню для зміни статусу замовлення:

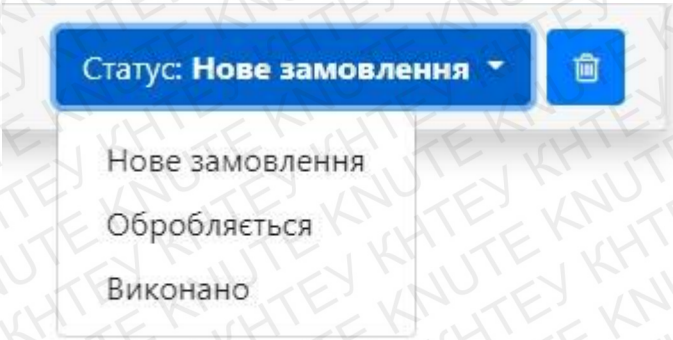

Рис. 3.6. Статус замовлення

На рис. 3.7 зображена «картка» замовлення з детальною інформацією про замовника та переліком товарів, кількість, загальна сума.

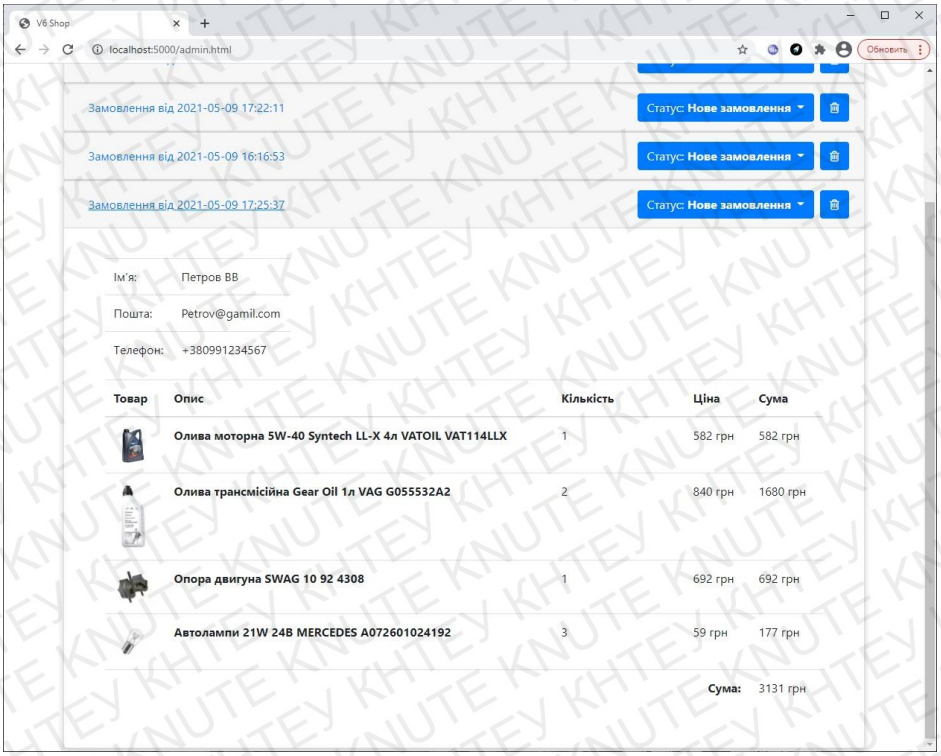

Рис. 3.7. Замовлення клієнта

Код файлу admin.js, який забезпечує інтерактивність та функціонування сторінки адміністратора магазину знаходиться у Додатку М

Код файла admin.html сторінки адміністратора , знаходиться у Додатку Н.

3.3 Розробка бізнес-логіки клієнтського ПЗ Web-порталу

Для розробки бізнес-логіки роботи Web-порталу потрібно чітко визначити дії користувачів, які вони виконуватимуть на порталі, а також реакцію програмного забезпечення на це.

Оскільки для різних ролей користувачів порталу будуть розроблені різні контролери, описуючи діяльність, варто враховувати різних користувачів.
На рисунку 3.8 зображена діаграма діяльності для авторизації та реєстрації.

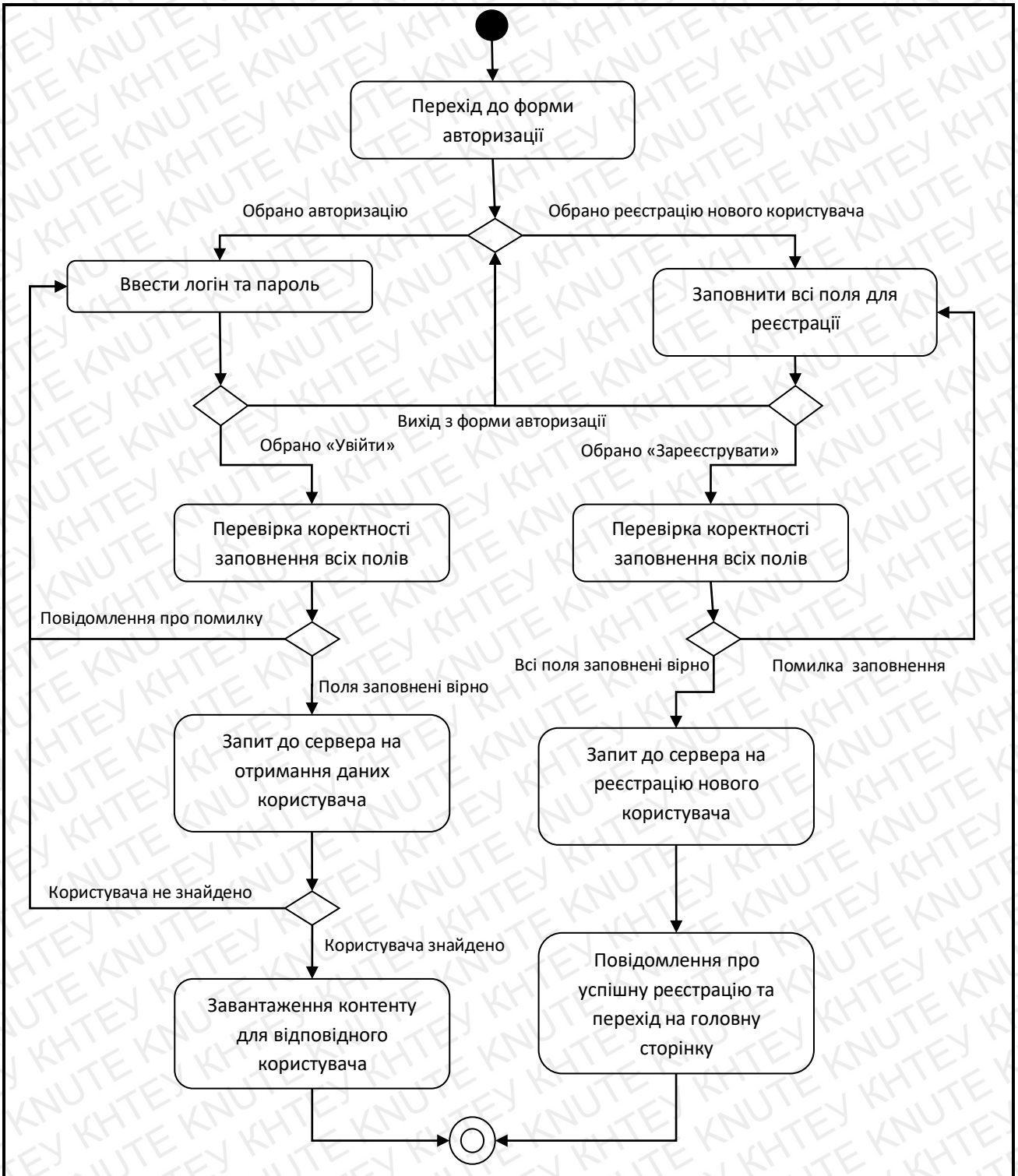

Рис. 3.8. Діаграма діяльності для переходу користувача до форми авторизації

Після того, як новий користувач зареєструвався і отримав підтвердження про успішну реєстрацію, йому потрібно перейти ще раз до форми авторизації, обрати авторизацію та ввести логін та пароль, що були задані при реєстрації.

3.4 Проектування програмного забезпечення клієнтської частини ПЗ

Для реакції на дії користувачів повинен бути написаний ряд контролерів, що реагують на відповідні дії та викликають ті чи інші методи моделей. Відповідні контролери повинні завантажуватись в залежності від ролі поточного користувача, а також відкритої сторінки порталу. Кожен контролер представлятиме собою програмний модуль. Повинні бути передбачені такі програмні модулі для перехоплення дій користувача:

- модуль, який завантажується при відкритті порталу, відповідає за дії з переміщення сайтом, цей самий контролер діятиме для гостей сайту (неавторизованих користувачів);
- модуль для авторизованих користувачів;
- модуль для редакторів (завантажуватиметься після авторизації користувачів з роллю редактора).

Оскільки для архітектури за шаблоном MVC, методи, що відповідають за зміни, прописуються в самій моделі, тому буде достатньо по одному контролеру (модулю з логікою реакції на відповідні дії) для кожного типу користувачів.

Варто детальніше розглянути кожну з моделей, які повинні бути реалізовані.

Сторінки сайту, такі як «Головна», «Корисна інформація», «Список товарів», «Особистий кабінет» і т. п. - різні представлення однією моделі. Для даної моделі повинні бути розроблені наступні методи:

- Відобразити / приховати форму пошуку.

- Відобразити / приховати форму авторизації.
- Відобразити / приховати текст (інформаційне наповнення сторінки).
- Запит на отримання списку товарів (для відображення, якщо користувач перейшов до списку). Для даного методу можна реалізувати різні його варіанти, що приймають різні параметри (задані формою пошуку, або відповідним авторизованим користувачем). Відповідно, при завантаженні сторінки, може бути одразу завантажений лише той варіант методу, що відповідає поточному користувачу.
- Запити на завантаження / зміну / видалення / створення товару (для кожної дії повинен бути окремий метод).
- Запити на підтвердження / відхилення публікації товару (методи, що викликатимуть дані запити, завантажуватимуться лише для редактора).

Форми для заповнення полів для авторизації та реєстрації доцільно реалізувати у вигляді окремих моделей, в яких будуть реалізовані методи, орієнтовані на коректність заповнення кожного поля.

3.5 Висновки до розділу 3

У 3 розділі було спроектовано веб-портал магазину автозапчастин. Для реалізації проекту, розробка виконувалась із застосуванням html/css/js з підключенням бібліотек bootstrap та JQuerry (для забезпечення інтерактивності веб-додатку), Node.js (для функціонування серверу). Покроково описано процес створення веб-порталу та додання функціоналу до нього. Були визначені дії користувачів на веб-порталі для розробки бізнес-логіки. За допомогою обраних технологій створено серверну частину сайту, додано динамічні подання до БД, а також розроблено адаптивний інтерфейс користувача.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В процесі виконання випускної кваліфікаційної роботи було проведено аналіз напрямків розвитку розробки сучасних Web-додатків, зокрема було приділено особливу увагу сучасним тенденціям до створення Web-ресурсів з інтерактивною поведінкою, наповнення сторінок котрих здатне динамічно змінюватись в залежності від дій користувача. Було розглянуто різні підходи до реалізації програмного забезпечення таких ресурсів, як клієнтської, так і серверної частини.

В ході дослідження особлива увага була приділена технологіям Single Page Application (SPA), які на сьогодні отримали широке розповсюдження. Важливими перевагами даного підходу в розробці Web-ресурсів є підвищення швидкості їх завантаження та реакції на дії користувача, зменшення споживання трафіку, а також підтримка різних платформ.

Також в ході роботи був проведений аналіз різних патернів проектування, які використовуються при розробці Web-ресурсів з застосуванням технологій SPA.

Послідовно була проведена робота з проектування Web-порталу для магазину автозапчастин із застосуванням технологій SPA. На відповідних етапах проектування були побудовані діаграми активності для ресурсу, бізнес-логіка роботи порталу, визначені типи користувачів та їх права, на основі чого був проектований інтерфейс користувача та визначені основні моделі та їм методи, а також контролери (модулі логіки ресурсу), що викликатимуть ті чи інші методи моделей.

Також в ході роботи було проведено аналіз робочого місця розробника Web-порталу з точки зору охорони праці. В ході аналізу робочого місця були

виявлені деякі невідповідності державним нормам, що можуть негативно впливати на здоров’я розробника.

Створений в результаті роботи Web-ресурс володіє прекрасними можливостями, що дозволяють в режимі онлайн створювати, зберігати, редагувати та публікувати товари. В сучасному світі, де комп’ютерні технології посідають значне місце в житті людини, подібні ресурси є досить актуальними, адже користувачі ресурсу у будь-якому місці та в будь-який час зможуть отримати доступ до потрібної їм інформації.

Розроблений портал можна вдосконалювати новими функціями, наприклад, додати пошук подібної інформації не лише в межах самого порталу, але й в Інтренеті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тиге Дж. К. DHTML и CSS: Пер. с англ. – М.: ДМК Пресс, 2003. – 506с.

2. Д. Гудман JavaScript и DHTML. Сборник рецептов для профессионалов: Пер. с англ. – Питер, 2004. – 523с.

3. Глушаков С.В., Жакие И.А., Хачиров Т.С. Программирование Web-страниц. – Харьков: Фолио, 2005. – 390с.

4. Полянский А.А. Программирование на CGI. – М.: Мойор, 2003. – 176с.

5. Рассел Джонс А. Active Server Pages 3: полное руководство: Пер. с англ. – М.: Энтроп2001. – 704 с.

6. С.І. Шаповалова, І.Я. Скорська Оптимізація моделі представлення Web-системи на основі SPA-архітектури // Вісник НТУ «ХПІ», 2012, №68. − С. 71−75.

7. Особенности архитектуры Single Page Application [Електронний ресурс]. URL: https://events.yandex.ru/lib/talks/2557/

8. Michael S. Mikowski and Josh C. Powell Single Page Web Applications. – Manning, 2013. – 433p.

9. Single-page vs. Multi-page. Особенности автоматизации тестирования. [Електронний ресурс]. URL: http://seleniumcamp.com/archive/selenium-camp-2013/materials/single-page/

10. Model-View-Controller [Електронний ресурс].
URL: https://msdn.microsoft.com/ru-ru/library/ms978748(en-us).aspx

11. MVC vs. MVP vs. MVVM [Електронний ресурс].
URL: https://nirajrules.wordpress.com/2009/07/18/mvc-vs-mvp-vs-mvvm/

ДОДАТКИ

Додаток А

Шаблон html-сторінки:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Магазин автозапчастин</title>
  <link
href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css" />
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <script type="text/javascript" src="/js/jquery-3.6.0.min.js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
integrity="sha384-
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEU
LTy" crossorigin="anonymous"></script>
  <script type="text/javascript" src="/js/script.js"></script>
</body>
</html>
```

Додаток Б

Код, який відповідає за відображення кнопки «Кошик» на веб-сторінці:

```
<div class="topnav">
  <div class="container">
    <a class=" btn btn-success " href="#cart">
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-cart" viewBox="0 0 16 16">
        <path d="M0 1.5A.5.5 0 0 1 .5 1H2a.5.5 0 0 1 .485.379L2.89
3H14.5a.5.5 0 0 1 .491.592l-1.5 8A.5.5 0 0 1 13 12H4a.5.5 0 0 1-.491-.408L2.01
3.607 1.61 2H.5a.5.5 0 0 1-.5-.5zM3.102 4l1.313 7h8.17l1.313-7H3.102zM5 12a2 2
0 1 0 0 4 2 2 0 0 0 0-4zm7 0a2 2 0 1 0 0 4 2 2 0 0 0 0-4zm-7 1a1 1 0 1 1 0 2 1 1 0 0 1
0-2zm7 0a1 1 0 1 1 0 2 1 1 0 0 1 0-2z" />
      </svg> Кошик</a>
  </div>
</div>
<div class="container">
  <h1 class="h1">Магазин автозапчастин</h1>
  </div>
</div>
```

Додаток В

Структура html-сторінки у яку будуть додаватись картки категоій товарів:

```
<div class="container">
  <h1 class="h1">Магазин автозапчастин</h1>
</div>
<div class="uk-section o-section-discounts" id="home_popular_parts">
  <div class="uk-container">
    <!-- Картки категоій товарів -->
  </div>
</div>
```

Додаток Г

Структура html-коду для кожної картки товару головної сторінки:

```
<div class="o-discount-item uk-width-1-1 uk-width-1-3@l">
  <div class="o-descount-inner"><span>ОЛИВИ, МАСТИЛА<img alt="Олива моторна, мастило" src="/img/oil.jpg" style="float: right;" ></span>
    <div><a href="/items/motornie_masla.html">Моторна олива</a><br>
      <a href="/items/transmissionnie_masla.html ">Трансмісійна олива</a><br>
      <a href="/items/oil_moto_engine.html ">Олива для мототехніки</a></div>
  </div>
</div>
```

Додаток Д

Результуючий html-код головної сторінки:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>V6 Shop</title>
  <link
href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css" />
  <link rel="stylesheet" href="/css/style-main.css">
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
</head>
<body>
  <div class="topnav">
    <div class="container">
      <h1 class="first-page-shop-name pull-left">V6 Shop</h1>
      <button type="button" class="btn btn-success btn-lg pull-right" data-
toggle="modal" data-target="#cartModal" onclick="checkModal()">
        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" class="bi bi-cart" viewBox="0 0 16 16">
```

Додаток Е

Html-код картки товару:

```
<div class="o-product o-product-special ">
  <span class="o-title" style="height: 54px;">Олива моторна 5W-30 SynGold Plus 4л VATOIL VAT104PLUS</span>
  <div class="o-img">
    <img alt="Олива моторна 5W-30 SynGold Plus 4л VATOIL VAT104PLUS"
src="https://www.autoklad.ua/imgbank/Image/pic/vatoil_vat104plus.jpg" data-src="https://www.autoklad.ua/imgbank/Image/pic/vatoil_vat104plus.jpg" data-lzled="true">
  </div>
    <div class="o-price">
      <span>Ціна
        <strong id="tile_price_10138650"><b class="cost">820</b><sup>грн</sup></strong>
      </span>
  </div>
   <div class="o-buy" id="add_tile_VAT_VAT104PLUS_">
      <a style="display: block;" href="javascript:void(0);" class="uk-button o-button-darkblue uk-width-1-1" onclick="addToCart(this);return false;"> Додати </a>
    </div>
    <div class="o-text-primary o-product-article" style="width: 100%">
      <strong>Артикул:</strong> VAT104PLUS
    </div>
```

Додаток Ж

Javascript код, який відповідає за відображення додаткової інформації на картці товару,

додавання інформації до кошика та відображення спливаюче вікна Кошика:

```
function showInfo(el) {
    $(el).parent().parent().find(".o-product-description").toggle()
}
function addToCart(el) {
    let card = $(el).parentsUntil(".uk-grid-margin");
    let itemName = $(card).find(".o-title").text();
    let itemCost = $(card).find(".cost").text();
    addItem(card,itemName,itemCost);
    showCart();
}
```

Додаток З

Html-код який відповідає за відображення вікна кошика:

```
<div id="cartModal" class="modal fade" role="dialog" class="container bootstrap snippets bootdey">
  <div class="content modal-dialog modal-lg">
    <div class="row modal-content panel-shadow" style="width: auto;">
      <div class="col-md-12">
        <div class="panel panel-info ">
          <div class="modal-header">
            <div class="panel-heading">
<h3>
                Обрані товари:
              </h3>
            </div>
          </div>
          <div class="panel-body">
            <div class="">
              <table class="table">
                <thead>
                  <tr>
                    <th>Товар</th>
                    <th>Опис</th>
                    <th>Кількість</th>
                    <th>Ціна</th>
                    <th>Сумма</th>
                  </tr>
                </thead>
```

Додаток І

Html-код який відповідає за відображення вікна вводу контактних даних:

```
<div class="modal fade" id="sendOkModal" role="dialog">
  <div class="modal-dialog">
    <!-- Modal content-->
    <div class="modal-content">
      <div class="modal-header">
        <h4>Контактні данні:</h4>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>
      <div class="modal-body">
        <div id="sended_ok" class="hidden">
          <h4>Замовлення успішно надіслано</h4>
      </div>
    <div id="client_info">
          <div class="input-group mb-3">
            <div class="input-group-prepend ">
              <span class="input-group-text get-contacts-predend" id="basic-addon1">Ім'я:</span>
            </div>
            <input type="text" class="form-control" id="name" aria-label="Username" aria-describedby="basic-addon1">
          </div>
    <div class="input-group mb-3">
          <div class="input-group-prepend ">
            <span class="input-group-text get-contacts-predend" id="basic-addon1">Пошта:</span>
```

Додаток К

Код серверної частини, файл app.js:

```
const fs = require('fs');
const path = require('path');
const PORT = process.env.PORT || 5000;
var nodemailer = require('nodemailer');
var express = require('express')
var app = express();
var bodyParser = require('body-parser');
app.use(express.static(path.join(__dirname, 'public')))
app.listen(PORT, () => console.log(`Listening on ${ PORT }`));
app.use(bodyParser.json()); // to support JSON-encoded bodies
app.use(bodyParser.urlencoded({ // to support URL-encoded bodies
  extended: true
}));
// app.use(express.static('public'));
var ordersPath = __dirname + "/orders/";
app.post('/senddata', function(req, res) {
  console.log("senddata");
  console.log(req.body);
  let data = "";
  for (var i = 0; i < req.body[0].length; i++) {
    data += "Товар: " + req.body[0][i][1] + " Кількість: " + req.body[0][i][2]
+ ", ціна за од: " + req.body[0][i][3] + "грн.\r\n"
  }
```

Додаток Л

Код файлу script.js, який відповідає за інтерактивність елементів на веб-сторінці, виконує калькуляцію сум у кошику та забезпечує відправлення даних до серверу:

```
let items;
function showInfo(el) {
  $(el).parent().parent().find(".o-product-description").toggle()
}
function addToCart(el) {
  $('#cartModal').find('.cart-modal-item').remove()
  let card = $(el).parentsUntil(".uk-grid-margin");
  let foto = $(card).find(".o-img").find("img").attr('src');;
  let itemName = $(card).find(".o-title").text();
  let itemCost = $(card).find(".cost").text();
  addItem(foto, itemName, itemCost);
  showCart();
}
function sendOrderBtn() {
  $("#sended_ok").hide();
  $("#client_info").show();
  $("#cartModal").modal('hide')
  $("#sendOkModal").modal('show')
}
function showCart() {
  checkModal();
  $("#cartModal").modal('show')
}
function checkModal() {
  let list = getItemsList();
```

Додаток М

Код файлу admin.js, який забезпечує функційювання сторінки адміністратора магазину:

```
//шаблон строки
let itemTemplate = `<tr class="cart-modal-item">
  <td><img style="max-width: 50px;" src="" class="img-cart"></td>
  <td><strong class="item-name"> </strong>
  </td>
  <td style="min-width: 160px;">
      <span class="item-value" style="width: 60px;"></span>
  </td>
  <td><span class="item-cost"></span><span> грн</span></td>
  <td><span class="item-cost-total"></span><span> грн</span></td>
</tr>`;
let orderTemplate = `<div class="card">
  <div class="card-header" id="heading">
    <button class="mb-0 btn btn-link collapsed" data-toggle="collapse" data-target="#collapseTarget" aria-expanded="false" aria-controls="collapseTarget">
      Замовлення від <span class="date"></span>
    </button>
    <button type="button" class="btn btn-primary ml-2 pull-right" onclick="removeItem(this, timestamp)"><i class="fa fa-trash-o"></i></button>
    <div class="dropdown pull-right">
   <button class="btn btn-primary dropdown-toggle" type="button" id="dropdownMenu2" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
<span>Статус: </span><b><span class="status"></span></b>
    </button>
```

Додаток Н

Код файла admin.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>V6 Shop</title>
  <link
href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css" />
  <link rel="stylesheet" type="text/css" href="css/style.css">
</head>
</head>
<body>
  <div class="topnav">
    <div class="container">
      <h1 class="first-page-shop-name pull-left">V6 Shop</h1>
    </div>
  </div>
  <div class="container">
    <h2 class="first-page-title"> Адміністратор </h2>
  </div>
  <div class="uk-section o-section-discounts" id="home_popular_parts">
    <div class="uk-container">
      <h1 style="padding-left: 100px;"></h1>
```