МІНІСТЕРСТОВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп‘ютерної та програмної інженерії
Кафедра комп’ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____А.С. Савченко

“____”_____2021 р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
“БАКАЛАВР”

Тема: «Інтернет-магазин цифрової техніки»

Виконавець: студентка УС-412 Бойко Єва Олександрівна
(студент, група, прізвище, ім'я, по батькові)

Керівник: к. т. н., доцент Климова Асія Сабирівна
(науковій ступень, вчене звання, прізвище, ім'я, по батькові)

Нормоконтролер: ст. викл. Шевченко О.П.
(П.І.Б.) (підпис)

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп‘ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Освітній ступінь: Бакалавр

Галузь знань, спеціальність, спеціалізація: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Савченко А.С.*
"___" _____ 2021 р

ЗАВДАННЯ

на виконання дипломного проекту студентки

Бойко Єви Олександрівни

1. Тема проекту: «Інтернет-магазин цифрової техніки» затверджена наказом ректора № 636/ст. від 22.04.2021р.
2. Термін виконання роботи: з 10.05.2021 по 14.06.2021р.
3. Вихідні дані до роботи: розробка інтернет-магазину на основі WEB.
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка завдання, розробка та опис програмного рішення, висновки.
5. Перелік обов’язкового графічного матеріалу: постановка завдання на дипломну роботу, результати експериментальної частини роботи.

КАЛЕНДАРНИЙ ПЛАН

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Ознайомлення з постановкою задачі та вивчення літератури. Розробка та затвердження плану дипломного проекту.	11.05.21 – 14.05.21	
2.	Написання 1 розділу. Аналітичний огляд задачі.	15.05.21	
3.	Написання 2 розділу. Порівняльній аналіз.	18.05.21 – 21.05.21	
4.	Написання 3 розділу. Опис можливостей та підходів до створення систем.	22.05.21 – 25.05.21	
5.	Написання 4 розділу, представлення керівнику.	26.05.21 – 31.05.21	
6.	Загальне редагування та друк пояснювальної записки, графічного матеріалу.	01.06.21 – 03.06.21	
7.	Проходження нормо-контролю, перепліт пояснювальної записки.	04.06.21	
8.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	05.06.21 – 08.06.21	
9.	Отримання відгуку керівника, рецензії.	07.06.21	
10.	Підписання необхідних документів у встановленому порядку.	08.06.2021– 11.06.21	
11.	Захист дипломного проекту	15.06.21 – 18.06.21	

Студент **(*Бойко Є.О.)***

Керівник дипломної роботи (***Климова А.С.)***

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Інтернет-магазин цифрової техніки»: 53 с., 23 рис., 10 літературних джерел.

Об’єкт дослідження: Алгоритми створення сайтів на основі WEB-розробки.

Розробка інтернет-магазину для продажу цифрової техніки на основі WEB.

Мета роботи – Розробка інтернет-магазину для продажу цифрової техніки на основі WEB.

Методи дослідження: розробка та реалізація сайта інтернет-магазину, порівняльний аналіз методів реализації, обробка літературних джерел.

ІНТЕРНЕТ-СЕРЕДОВИЩЕ, НАДАННЯ ІНФОРМАЦІЇ, ВЕБ-РЕСУРС, ІНФОРМАЦІЙНО-ПОШУКОВИЙ САЙТ, ІНТЕРНЕТ-МАГАЗИН, JAVASCRIPT, WEB, HTML.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API (англ. application programming interface) - програмний інтерфейс програми, інтерфейс прикладного програмування;

CSRF (англ. Cross-Site Request Forgery, CSRF) - міжсайтова підробка запиту;

CORS (англ. Cross-Origin Resource Sharing) - спільне використання ресурсів з різних джерел;

CSS (англ. Cascading Style Sheets) - каскадні таблиці стилів;

EJS (англ. Embedded JavaScript) - каркас веб-додатків для Node.js;

ES6 (англ. ECMAScript 6) - вбудована розширювана мова програмування яка не має засобів введення-виведення для побудови інших скриптових мов;

HTML (англ. Hyper Text Markup Language) – мова розмітки гіпертексту;

HTTP (англ. Hyper Text Transfer Protocol) - протокол передачі даних, що використовується в комп'ютерних мережах

HTTPS (англ. Hyper Text Transfer Protocol over Secure Socket Layer) - схема URI, що синтаксично ідентична http: схемі, яка зазвичай використовується для доступу до ресурсів Інтернет;

IDE (англ. Integrated Development Environment) - паралельний інтерфейс підключення накопичувачів (жорстких дисків і оптичних приводів) до комп'ютера;

JSON (англ. JavaScript Object Notation) - текстовий формат обміну даними, заснований на JavaScript;

LAMP (англ. Linux Apache MySQL and PHP) - набір відкритого програмного забезпечення, який використовується для створення веб-серверів;

LTS (англ. Long Term Support) - частина життєвого циклу програмного забезпечення (ПЗ);

MVC (англ. Model View Controller) - шаблон програмування, який дозволяє розділити логіку програми на три частини;

MVP (англ. Minimum Viable Product) - мінімально життєздатний продукт;

PDF (англ. Portable Document Format) - міжплатформений відкритий формат електронних документів.

ВСТУП

Веб-розробка пройшла довгий шлях з тих пір, як була створена глобальна комп'ютерна мережа Internet. Розробники повинні були розробити окремі програми для кожної з операційних систем і встановити їх локально на комп 'ютер для зручного використання. Ці програми називалися декстоп – додатками. На противагу цьому, веб- додатки неважко достати з веб-браузера незалежно від операційної системи користувача. Веб-додатки досягли величезного прогресу після того, як десктоп - додатки та статичні веб-сайти перейшли до сучасних, інтерактивних та інтелектуальних веб-додатків. Користувачі могли читати тільки вміст статичних веб-сайтів. Однак через появу веб – додатків користувачі отримують можливість взаємодіяти з сервером, слухати аудіо і дивитися відео, а також малювати на екрані.

Розробка інтернет – магазину - це поєднання фронтенд та бекенд розробки. Існує низка мов програмування та фреймворків для створення веб - додатка. Раніше не існувало мов програмування, які могли б виконувати як фронтенд, так і бекенд-розробку додатку. LAMP stack: Linux, Apache, MySQL і PHP раніше були стандартом для повностекової веб-розробки. PHP в основному використовувався для бекенду, а HTML, CSS і JavaScript були основними стеками для розробки фронтенду, в той час як MySQL використовувався для баз даних. Apache виступає як веб-сервер в операційній системі Linux у стеку LAMP. Однак для розробки веб – додатку з використанням стека LAMP розробникам потрібно було вивчити кілька мов, всі з яких використовують різний синтаксис і мають різну природу. Крім того, з появою Node.js в якості серверної платформи JavaScript зміг забезпечити розробку повностекового програмного забезпечення в рамках єдиної мови програмування. Незважаючи на те, що Node.js є найбільш поширеною технологією для розробки бекенд-систем, існує безліч фреймворків, таких як Angular, React, Vue.js або Knockout.js, а також шаблонні двигуни (шаблонизатор) , такі як EJS, Handlebars або Pug.

Метою дипломної роботи було вивчення різних аспектів веб-розробки за допомогою Full Stack JavaScript та розробка прототипу програми на її основі. Його

метою було вивчити використання Node.js, Express, MongoDB та EJS при розробці веб-додатків Full Stack. Більше того, він також зосереджений на вивченні різних версій JavaScript та їх використання в платформі Node.js.

Різні версії ECMAScript були детально вивчені, особливо ES6. Потім були вивчені особливості та реалізація платформ на основі JavaScript, таких як Node.js, Express та MongoDB, та порівняно з іншими технологіями. Крім того, проілюстровано потенційні загрози безпеці для програми Node.js та різні способи їх пом'якшення. Також, за допомогою цієї платформи було розроблено мінімально життєздатний продукт програми.

Як результат, за допомогою Node.js та Express було розроблено інтернет-магазин цифрової техніки з використанням основних пакетів і модулів Node.js, а також проміжного програмного забезпечення Express. Інші сторонні проміжні програми також використовувались для розробки працюючого додатка-прототипу.

Повнотекстовий JavaScript вважається найкращою технологією для розробки сучасних, масштабованих та безпечних веб-додатків. Однак нездатність Node.js обробляти складні обчислення даних та алгоритми робить непридатною розробку великих корпоративних додатків.

РОЗДІЛ 1
РОЗВИТОК СФЕРИ ІНТЕРНЕТ-МАГАЗИНІВ В УКРАЇНІ І СВІТІ

1.1. Світовий ринок інтернет-магазинів.

Доходи інтернет-ринку зростають в усіх регіонах світу, і найшвидше — в Східній Азії. В цій частині світу генерується більш ніж 50% доходів, а Китай за рівнем продаж в інтернет-комерції перевершив США.

Лідери світового ринку — компанії з Китаю, США, Великобританії. Найбільші компанії заробляють мільярди на своїх сайтах. До кінця 2020 року загальний об'єм продаж інтернет-магазинів у всьому світі досяг відмітки у 2 трильйони долларів США. (рис. 1.1)

Рис. 1.1. Діаграма загальних продаж у світі

Серед великих ніш найбільший середній чек - в ніші комп'ютерів, телевізорів і мультимедійних пристроїв, тут він становить $ 212. Найменша середня сума покупки - в ніші квітів і подарунків.

Якщо подивитись на рейтинг продаж світових торгових мереж, можна легко побачити, що найбільшу виручку мають компанії, які працюють и інтернет-комерції.(рис 1.2)

Топ-10 торгових мереж за темпами зростання виручки

Місце	Бренд	Країна	Зростання обороту за 2014-19 роки, %
1	Alibaba	Китай	44
2	JD.com	Китай	35
3	Albertsons Companies	США	25
4	Amazon.com	США	22
5	Apple	США	13
6	China Resources Enterprise	Китай	13
7	X5 Retail Group	Росія	10
8	Suning	Китай	9
9	A.S. Watson	Гонконг	8
10	Ikea	Швеція	7

Рис. 1.2. Рейтинг торгових мереж

1.2. Українська індустрія інтернет-магазинів

На даний момент, сфера інтернет-магазинів України розвивається, й досить швидко. Вже існує український маркетплейс rozetka.com.ua, на якому може продавати свої товари будь-який підприємець. Зараз сайт посідає 5 місце за відвідуваністю серед сайтів в Україні.(рис. 1.3)

	Сайт	Охоплення *
1	google.com	22 123 040
2	youtube.com	16 894 210
3	facebook.com	15 357 240
4	wikipedia.org	12 899 700
5	rozetka.com.ua	10 845 820
6	olx.ua	10 554 310
7	prom.ua	9 970 980
8	ukr.net	9 895 100
9	privatbank.ua	9 463 500
10	sinoptik.ua	7 239 820

Рис.1.3. Відвідуваність сайтів в Україні

Але, нажаль, на даний час можна побачити тенденцію розвитку лише вже існуючих гігантів сфери, та відсутність великої конкуренції через різні показники – наприклад, відсутність адаптивного дизайну, його незручність, або взагалі, деякі компанії не мають власного інтернет-магазину та зовсім не отримують прибутку в Інтернеті, через страх або невпевненість у надійності такої торгівлі. Такі висновки можна зробити, проаналізувавши відвідуваність українських інтернет-магазинів. (рис.1.4)

Багато компаній або зовсім не створюють власний інтернет-магазин, вважаючи, що цей ринок вже переповнений(що зовсім не так), або створюють власну платформу, але не витримують конкуренції з вже існуючими сайтами, через їх розрекламованість.

Рис.1.4. Відвідуваність українських інтернет-магазинів

Висновки до розділу 1

Узагальнюючи вищезгадане, можна прийти до висновків, що інтернет-магазин – це потужний інструмент для торгівлі. Особливо у нинішніх умовах, пов’язаних з пандемією коронавірусу, коли турбота про безпеку та здоров’я людини є дуже важливими. Інтернет-магазин – це ідеальне спасіння для підприємця, який хоче покращити та прискорити сервіс, отримати більше прибутку, та піклуватися про своїх клієнтів. Український ринок має місце для розвитку нових сайтів, не засмічений багатьма платформами, тож є непоганим початком розвитку.

Щоб задовольнити ці бажання, не обов`язково створювати величезну компанію із сотнями співробітників. Достатньо менеджера, кількох дизайнерів, кількох розробників та одного інвестора. Це оптимізує багато бізнес-процесів та безумовно збільшить виручку

РОЗДІЛ 2
ВИБІР, ОБГРУНТУВАННЯ ТА ОПИС ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Етапи створення та структура сайтів

До початку створення інтернет-магазину цифрової техніки потрібно розуміти класифікацію веб-сайтів та як виглядає структура HTML-сторінки. Після цього варто зазначити основні етапи створення веб-сайту, поглибити знання в обраній мові розробці JavaScript, вивчити основи роботи з Node.js, Express, MondoDB та Mongoose.

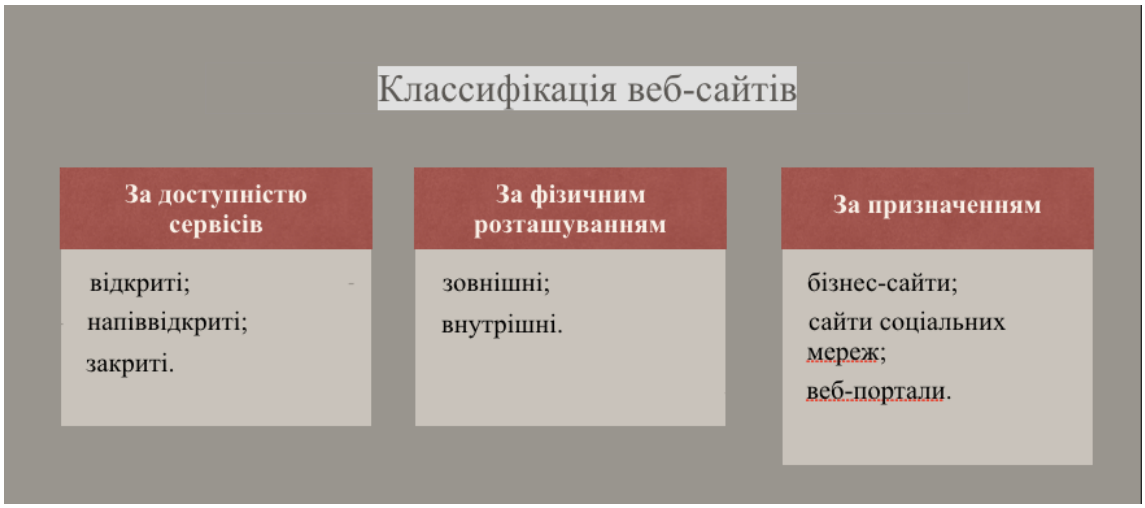

Рис. 2.1. Класифікація веб-сайтів

Перший етап створення сайту – **проектування.**

Обговорюються мета, бюджет проекту і виходячи з цього формується технічне завдання.

Наступний етап розробки – **дизайн.** Результатом є дизайн-макет сайту: графічний файл, виконаний в одній з графічних програм .

Також не слід забувати про **маркування сторінок.** На основі дизайн макету створюється макет шаблоних сторінок написаних на мові HTML.

Далі потрібно **запрограмувати сервіс**. Наприклад, якщо є форма форма для реєстрації клієнтів.

Та останній крок **– публікація та наповнення сайту інформацією.** Створюється каркас, який згодом заповнюється текстовою інформацією.

Головні теги, які застосовуються для створення сайту:

<header> - тег <header> у HTML використовується для визначення заголовка документа або розробки.

Тег заголовка містить інформацію, що відноситься до заголовку та заголовку пов'язаного вмісту. Тег <header> нельзя розміщати всередині <footer>, <address> або іншого елемента <header>.

<nav> – тег <nav> використовується для створення блоків навігації, посилань, що ведуть на інші веб-сторінки. Інколи включає в себе тег <li>. Одна сторінка HTML може містити кілька тегів <nav>.

<ul> – використовується для створення маркованого списку

Дуже важливо, кожний рядок списку <ul> повинен буду загорнутий в тег <li>.

** ** Перший елемент списку **
**

** ** Другий елемент списку **
**

<ol> – використовується для створення нумерованого списку .

Дуже важливо, кожний рядок списку <ol> повинен буду загорнутий в тег <li>.

** ** Перший елемент списку **
**

** ** Другий елемент списку **
**

<li> – використовується в тегах <ul> і <ol> для того, щоб відокремити кожний елемент списку.

<h> – даний тег включає в себе заголовки шести рівнів <h1> … <h6>. Заголовок першого рівня <h1> – це самий головний тег і він повинен буди лише один на сайті, тегів <h2> … <h6> може буди безліч.

<p> – застосовується для визначення абзацу.

** ** – в даний тег можна розміщати інші теги для зміни їх виду або вмісту (за допомогою CSS).

<form> – форма в HTML є цілою частиною документа, оскільки дозволяє користувачеві надавати інформацію, яку ми можемо отримувати та обробляти на стороні сервера.

<section> – даний тег застосовується для розділу сторінки на логічні частини або відокремлення певних секцій/частин на сайті.

<footer> – тег footer зберігає інформацію про логотип компанії, ,контакти компанії та ім’я автора.

<img> – даний тег використовується для того, щоб додати зображення на html сторінку.

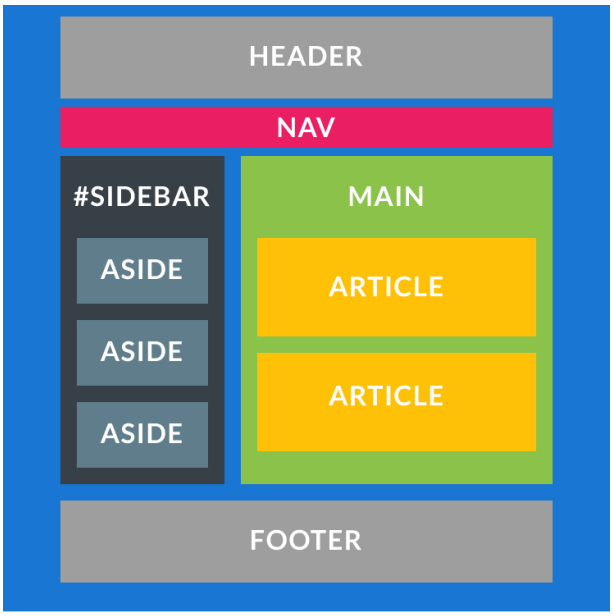

Рис. 2.2. Приклад структури сторінки

2.2. JAVASCRIPT. Різні версії та ES6

JavaScript - це динамічна об'єктно-орієнтована мова з першокласними функціями на основі об'єктів, яка може повертати значення, передаючи саму функцію іншим функціям як аргумент. JavaScript був вперше розроблений в 1995 році Бренданом Ейхом в компанії під назвою Netscape для того щоб вона служила скриптовою мовою для Java. Незважаючи на те, що JavaScript в основному використовувався як клієнтська програма для створення скриптів, він також значно розвинувся для підтримки серверних скриптів. Зараз JavaScript розвивається як мова програмування, яка використовується для розробки фронтенду і бекенду веб-додатку, баз даних, а також робототехніки.

Full-stack JavaScript Development має кілька переваг порівняно з використанням різних мов для розробки бекенду і фронтенду. Це відмінний технологічний стек для розробки динамічних і високопродуктивних додатків. Спільне використання коду і можливість багаторазового використання допомагають скоротити кількість рядків коду до 40% за рахунок спільного використання частин коду або навіть шаблонів, бібліотек і моделей. Це робить технічне обслуговування або рефакторинг дуже простим і економічним. Більш того, використання однієї мови допомагає підвищити ефективність команди і скоротити розрив між ними, так як всі команди, що працюють над додатком, будуть краще розуміти, що роблять інші. Крім того, розробникам не потрібно турбуватися про синтаксичні відмінності, як це відбувається коли на проекті користуються різними мовами.

Крім того, JavaScript підвищує швидкість і продуктивність веб – додатку. PayPal перейшов від Java до використання Node.js в якості серверної платформи для свого додатку і опублікував звіт про те, як JavaScript і Node.js допомогли їм підвищити продуктивність свого додатку і значно скоротити кількість написаного коду. Згідно зі звітом, додаток було побудовано в два рази швидше з меншою кількістю людей в команді, ніж коли він був написаний на Java. Вони також заявили, що у додатку, який був створений на JavaScript код був на 33% менше, та на 40%

менше було файлів з обома додатками, що складаються з одних і тих же функцій. Аналогічно, запит в секунду був подвоєний у додатку на JavaScript, а середній час відгуку було зведено до мінімуму на 35 %.

Незважаючи на всі переваги розробки повного стека на JavaScript є також і декілька недоліків. Node.js не може обробляти важкі обчислення, обробку даних, машинне навчання, алгоритми і важкі математичні обчислення, оскільки такі операції можуть блокувати вхідні запити через асинхронний характер моделі програмування. Крім того, Node.js є відносно новою технологією і поки не має надійної і сильної підтримки бібліотечної системи.

Компанія Stack Overflow опитала понад 1 00 000 професійних розробників програмного забезпечення та студентів, які сподіваються в майбутньому розповісти про свої улюблені технології для веб-розробки в 2018.

Рис. 2.3. Результати дослідження Stack Overflow

Як показано на рис. 2.3 69,8% від загальної кількості опитаних вказали JavaScript як технологію, яку вони часто використовують. Це ясно говорить про те, що JavaScript є однією з найпопулярніших мов.

JavaScript був стандартизований в Netscape в 1996 році з ECMA (Ecma International - Європейська асоціація стандартизації інформаційно - комунікаційних систем), яка встановлює стандарти в секторах IT та ICT в якості хосту. Міжнародні роботи ECMA роблять JavaScript сучасним і відповідним чинному технологічному стандарту. Відтоді випускаються різні версії ECMAScript з яких значущою версією є EcmaScript6 (ES6). Після наймасштабнішого випуску в 2015 році ECMA прагне випускати нові версії щороку. Отже, ECMA змінила назву ECMAScript з номера випуску на рік випуску. Однак система обчислення користується більшою популярністю серед JavaScript-спільноти. Отже, система обчислення буде використовуватися в цьому документ

Перша версія JavaScript була випущена в червні 1997 року компанією ECMA International з назвою ECMAScript 1. Наступного року в червні 1998 року був випущений ECMAScript 2, який містив незначні редакційні зміни відповідно до стандарту ISO для JavaScript. Тим не менш, третє видання JavaScript, ECMAScript 3 було випущено в грудні 1999 року і представило ряд нових можливостей для мови. Додавання таких функцій, як потужні формальні вирази, цикл do-while, такі методи обробки рядків як: concat, replace, match, split і splice за допомогою регулярних виразів, обробка помилок за допомогою try/catch, звужені визначення помилок, числове форматування вихідних даних і кілька інших незначних змін зробили мову більш потужною.

ECMAScript 4 (ES 4) - це версія JavaScript, яка так і не з 'явилася. Для цієї модернізації були заплановані різні нові функції, які включають класи, простори імен, інтерфейси, визначення типів, засоби отримання і установки та багато іншого. Основна мета цього випуску полягала в тому, щоб зробити мову відповідною для розробки великомасштабного програмного забезпечення і підкреслити розвиток ES4 бібліотек, а також зробити її сумісною з ES3. Однак через розбіжності між двома командами, що працюють над вдосконаленням мови, від ES4 відмовилися в липні 2008 року. Замість цього вони уклали угоду про випуск інкрементного оновлення ECMAScript 5.

П 'яте видання ECMAScript вийшло в грудні 2009 року після того, як від ES4 відмовилися. Це інкрементне оновлення попереднього ES3. Перше видання, яке пізніше було перейменовано в ES5. Деякі з нових функцій, включених в ES5, - це підтримка суворого режиму, яка допомагає писати більш чистий код, різні методи пошуку і управління масивами, такі як map, reduce, filter, indexOf and forEach властивості і атрибути, а також JSON.

ES6, також офіційно відомий як EcmaScript2015 - назва, яку використовують для JavaScript-версії, оскільки термін "JavaScript" має торгову марку Oracle. ES6, який випустили в червні 2015 року, є основним оновленням для JavaScript після того, як його стандартизували 1996 року. ES6 був випущений з метою підтримки написання складних додатків, бібліотек, а також генераторів коду. Крім того, основними цілями ES6 були підвищення експлуатаційної сумісності та збереження кращої і простої системи управління версіями. У результаті в новій версії пропонується кілька нових функцій, таких як модулі, класи, функції - стрілки і проксі-сервери ES6.

Хоча ES6 випустили у 2015 року, більшість браузерів не підтримували нових функцій ES6. Отже, транспілери (програми, що виконують транспіляцію програми. Транспіляція - перетворення програми, при якому використовується вихідний код програми, написаної на одній мові програмування в якості вихідних даних, і проводиться еквівалентний вихідний код на іншій мові програмування.) використовувалися для перетворення коду JavaScript, написаного ES6 мовою, на ES5. Babel і Google Traceur були популярними транспілерами, використовуваними розробниками. Однак всі сучасні браузери підтримують нові функції ES6 тому використання транспілерів не актуально. Деякі з нових функцій ES6 будуть розглянуті нижче.

ES6 ввів нові ключові слова: let та const для того щоб визначити змінну, яка має область видимості на додаток до області функції var в ES5. Змінні let, визначені у виразі функції, не можуть бути доступні поза цією конкретною функцією. На відміну від let, змінні, оголошені з const, не можуть бути оголошені знову з тією ж назвою. Значення, визначені за допомогою змінної const, не є незмінними, але значення можна змінити, якщо значення є об 'єктом.

Функції-стрілки є новим доповненням до ES6. Вони корисні для коротких викликів функцій через скорочений синтаксис, що використовує стрілку. У ньому опущено необхідність написання функції через ключові слова, а також через фігурні дужки. Вона не створює власну область і просто наслідує від функції, в якій він знаходиться. Це допомагає вирішити існуючу проблему з ключовим словом у JavaScript.

Отже, типові значення можуть передаватися функцією кожного разу, коли параметри не надаються у виклику функції, ініціалізуючи параметр за замовчуванням. Ця функція вже була доступна в PHP і Java. Крім того, у ES6 також додаються rest параметр і spread оператор. Параметр rest позначається трьома точками, за яким йде параметр (...this), що зберігає всі аргументи функції в масиві. Він дозволяє оголошувати змінну кількість аргументів з одним ідентифікатором. Оператор spread може бути взятий протилежно параметру rest з тим же синтаксисом, що і параметр rest, який приймає масив або будь-який ітерабельний об 'єкт і присвоює різним змінним.

Крім того, ES6 внесла багато інших змін до ES6 версії JavaScript. ES6 більше підходить для розробки передових і динамічних додатків з введенням шаблонних літералів, модулів, класів, проксі-серверів, проміси, деструктуризації, гетерів і сетерів. Більш того, додавання інших функцій і методів підвищило продуктивність і швидкість додатків JavaScript.

ES6 був випущений після паузи в чотири роки, через це з 'явилось багато нових доповнень до мови. Технічний комітет ECMA з розробки ECMAScript, TC-39, зрозумів, що випуск нової версії після такой паузи є нестійким. Тому нову версію вирішили випускати з кожним роком та з меншою кількістю доповнень до мови.

ES7 внесла три зміни до версії ECMAScript 2016 року. По-перше, новий оператор з синтаксисом (* *) є ярликом для отримання експоненційного значення. Він виконує операцію, що і Math.pow, але з меншою кількістю коду. По-друге, Array.prototype.includes повертає логічне значення при перевірці наявності або відсутності елемента в масиві. Нарешті, також було змінено суворий режим з функціональним діапазоном. Використання strict у функціях з деструктивними

параметрами або функцією, що має типові параметри, викликає синтаксичну помилку, оскільки значення параметрів за замовчуванням також можуть бути функцією.

ES8, також формально відомий як ECMAScript 2017, вийшов у червні 2017 року. Він включав асинхронні функції, загальну пам 'ять і атоміки, кінцеві коми в перелічуваних параметрах функцій і викликах, визначення послідовностей і методи об’єкта такі як object.values, object.entries, string.prototype.padStart, string.prototot.

ES9 або ECMAScript 2018 - остання версія ECMAScript, випущена в червні 2018 року. Він підтримує асинхронну ітерацію ітеративного об'єкта, такого як масиви і послідовності. Promises and generator functions тепер можна виконувати ітерацію за допомогою асинхронного ітератора. Аналогічним чином, ES9 також додав більше функцій регулярних виразів, таких як dotAll flag, затвердження look-background, look-behind assertions, Unicode capture groups та named capture groups. Rest оператори та spread оператори тепер також працюють для властивостей об 'єкта. Крім того, ES9 також додав метод .finally () в проміси. Це функція зворотного виклику, яка виконується щоразу при дозволі або відхиленні промісів.

2.3. Node.js

Node.js – це крос-платформенна серверна платформа програмування з відкритим вихідним кодом, побудована на середовищі Google Chrome V8 JavaScript, яка запускає код JavaScript поза браузером. Була розроблений Райаном Далем в 2009 році. Node.js є неблокуючою, асинхронною і керованою подіями системою введення-виводу, що означає, що вона не очікує завершення одного виклику API для виклику наступного. Замість цього вона повертається до попередньої події з функцією зворотнього виклику, одразу після виконання наступної події, яка була вказана раніше. Неблокуюче походження Node.js робить її масштабованою, легкою та ефективною і найкраще підходить для розробки веб-додатків в реальному часі. Node.js створює сервер самостійно за допомогою основного HTTP-модуля, слідкує за вхідними запитами, обробляє ці запити, перевіряє правильність вхідних даних,

підключається до бази даних і повертає відповідь просто повертаючи JSON-файл або шляхом візуалізації HTML-сторінки.

Рис. 2.4 ілюструє однонаправлений неблокуючий потік виконання подій вводу-виводу Node.js. Платформа створює цикл подій на основі одного потоку даних, для виконання всіх запитів, що надходять на сервер. Це не блокує виконання вводу-виводу і призводить до збільшення швидкості і продуктивності таких додатків.

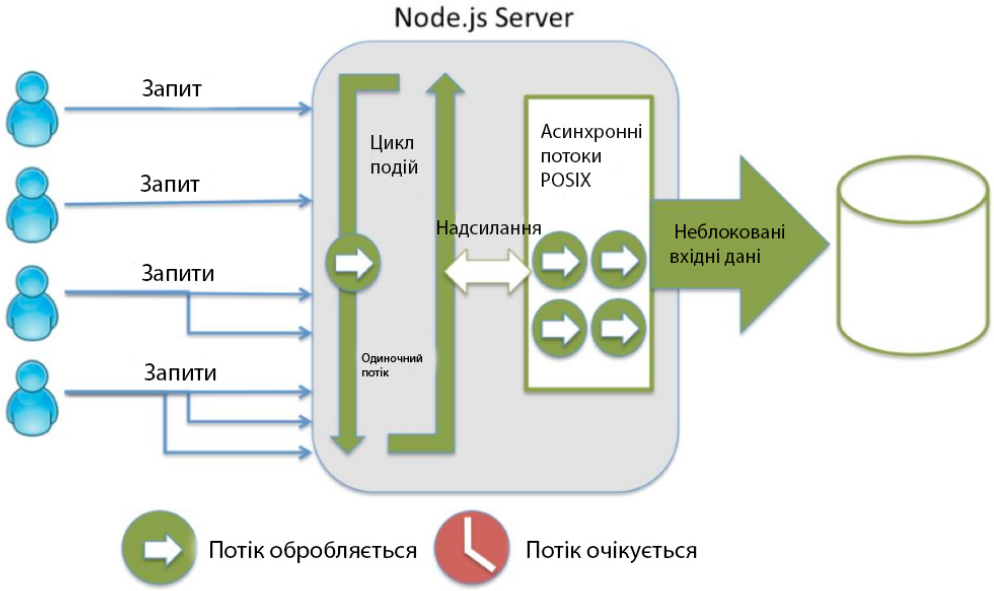

Рис. 2.4. Виконання однопотокового вводу-виводу Node.js

Декілька запитів одночасно обробляються на сервері Node.js одночасно. Ці запити обробляються одним потоком і утворюють цикл подій, який делегується як асинхронний потік для неблокуючого виконання вводу-виводу.

Продуктивність і стабільність Node.js регулярно поліпшуються, а нові можливості JavaScript зі щорічних випусків ECMAScript інтегруються в Node.js в його нових випусках.

2.4. Express

Express - це фреймворк Node.js, який прискорює та полегшує веб-розробку, пропонуючи різні функції на додаток до основних модульних вузлів. Express є невеликим та гнучким фреймворком Node.js, він пропонує кілька сторонніх проміжних програм, які розширюють функціональність експрес-програми та додають функції за потреби.

Express - це комбінація проміжного програмного забезпечення, яке виконується зверху вниз в циклі запиту-відповіді. Кожне проміжне програмне забезпечення має доступ до запиту та об'єкта відповіді, а також наступну функцію, які передаються з одного проміжного програмного забезпечення в інше. Проміжне програмне забезпечення приймає запит, виконує код всередині, змінює об'єкти запиту та відповіді і викликає наступну функцію, яка активізує наступне проміжне програмне забезпечення в черзі. Експрес-додаток може мати доступ до проміжного програмного забезпечення рівня додатка, проміжного програмного забезпечення рівня маршрутизатора, проміжного програмного забезпечення обробки помилок, вбудованого проміжного програмного забезпечення та проміжного стороннього програмного забезпечення.

Проміжне програмне забезпечення програмного рівня прив'язане до об'єкта програми за допомогою функцій app.use () і app.method (), де "method" є виконуваною командою HTTP. Проміжне програмне забезпечення рівня маршрутизатора відрізняється тільки від проміжного програмного забезпечення рівня додатка, оскільки воно прив'язане до екземпляра експрес-маршрутизатора. Проміжне програмне забезпечення для обробки помилок при 'язане до об' єкта програми і зазвичай приймає чотири аргументи: запит, відповідь, помилку, і наступні об'єкти. Вбудовані проміжні програми, такі як express.static і express.json, статично обслуговують файли і аналізують вхідний запит в JSON-файл відповідно. Крім того, інші проміжні програми сторонніх виробників розширюють функціональні можливості експрес-додатком шляхом аналізу файлів cookie або шляхом аналізу файлів cookie.

Крім того, маршрутизатори в експрес-режимі ділять додаток на кілька міні-експрес-додатків, які об'єднуються для формування експрес-додатку. Експрес-маршрутизатор складається з команди HTTP (GET, POST, PUT, DELETE) та шляху, або розташування ресурсу. Крім того, функції зворотного виклику також задаються в методі маршрутизації, де в якості аргументів може використовуватися ряд функцій зворотнього виклику. Такі функції використовують функцію next (), переходячи до іншої функції зворотнього виклику, викликаючи наступний метод.

2.5. MongoDB та Mongoose

MongoDB - це гнучка і масштабована база даних файлів. База даних складається з набору документів. Він скидає традиційний метод зберігання даних у реляційних базах даних і зберігає дані гнучким способом без будь-яких обмежень формату і структури. Дані зберігаються у форматі BSON, який є двійковим JSON. Документ може містити послідовності, числа, що плавають і навіть масиви та об 'єкти. Це робить роботу бази даних більш швидкою і простою. MongoDB вбудовує інші піддокументи в документ, надаючи посилання на цей документ замість приєднання до колекцій. База даних MongoDB підтримує різні операції з базою даних, такі як запит документів, вставлення нових документів, а також редагування та видалення існуючих документів.

Основною особливістю бази даних MongoDB є її здатність зберігати динамічні дані. Документи в одній колекції можуть мати різні властивості та пари ключових значень. Це усуває вимогу збереження аналогічно структурованих даних, яке є обов 'язковим в інших реляційних базах даних. Ця функція забезпечує велику гнучкість для зберігання непротиворечивых даних в одній колекції. Крім того, збереження всіх даних для об 'єкта в одному файлі підвищує швидкість роботи бази даних порівняно з реляційною базою даних. Це дозволяє уникнути необхідності об 'єднання багатьох таблиць для отримання даних з різних рядків і стовпчиків.

Хоча MongoDB є базою даних без схеми і спирається тільки на дисципліну розробника для підтримки структури документа, але для правильної роботи більшість програм потребує структуровані дані. Mongoose розроблявся саме для того, щоб вирішити це питання. Він забезпечує застосування стандартної структури до всіх файлів у колекції за допомогою схеми. Крім того, можна перевірити дані, що зберігаються в файлах, а також дозволити збереження в базі даних тільки допустимих даних. На додаток до цього Mongoose надає всі можливості, що існують в MongoDB, з додаванням функцій побудови запитів і бізнес-логіки даних. Крім того, він може з 'єднувати базу даних з сервером і виконувати аналогічні операції для читання, записи, оновлення і видалення даних.

Висновки до розділу 2

Сфера розробки веб-сайтів швидко розвивається, створюються нові мови та методи для їх розробки. Розробка веб-додатків - це поєднання інтерфейсної та внутрішньої розробки. Для розробки веб- додатків можна вибрати ряд мов програмування та фреймворків. Раніше не існувало мов програмування, які могли б виконувати як інтерфейсну, так і внутрішню розробку програми. використовувався як база даних. Однак з появою NodeJS як серверної платформи, JavaScript зміг забезпечити повноцінну розробку програмного забезпечення в рамках однієї мови програмування. Саме тому для виконання даного проекту було обрано JavaScript та Node.js.

РОЗДІЛ 3
РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ ЦИФРОВОЇ ТЕХНІКИ

3.1. Введення до проекту

Прототип програми Prydbay.com був розроблений для демонстрації використання повного стека JavaScript для розробки веб-додатку. Розроблений додаток - це мінімальний життєздатний продукт (MVP) для купівлі та продажу товарів через Інтернет. Пізніше MVP може бути розроблений для охоплення різних інших аспектів програми електронної торгівлі в Інтернеті.

Prydbay.com - це веб - додаток, який полегшує купівлю та продаж товарів через Інтернет. У ньому беруть участь два користувача, а саме адміністратор і користувачі. Адміністратори несуть відповідальність за додавання продуктів, опису і ціни продукту в базу даних. Аналогічно, користувачі можуть переглядати всі продукти, перераховані в додатку, додавати продукт в кошик, замовляти і оплачувати куплені продукти.

Після вивчення вимог власника програми і потенційних клієнтів були перераховані наступні вимоги, щоб додаток був готовий в якості MVP:

• Користувачі можуть створити обліковий запис для себе;

• Користувачі можуть змінювати інформацію про свій обліковий запис;

• Користувачі можуть переглядати всі доступні продукти;

• Користувачі можуть перевірити ціну та опис продукту, а також побачити картину продукту;

• Користувачі можуть додати один або кілька продуктів до кошика;

• Користувачі можуть переглядати продукти в кошику;

• Користувачі можуть видаляти продукти з кошика;

• Користувачі можуть замовити і оплатити куплену продукцію;

• Користувачі можуть завантажити рахунок за замовлення.

• Адміністратори можуть додати продукт в базу даних;

• Адміністратори можуть редагувати продукт у базі даних.

На основі перелічених вище вимог було розроблено додаток MVC (Model View Controller). Серверна частина складається з model і controller, в той час як дизайн реалізований на клієнтській стороні. Програму, розроблену за допомогою зразка MVC, розділено на три окремі частини: model, contoller і view, всі з яких взаємодіють один з одним. Model - це об 'єкт, що містить дані програми. Model може отримувати дані від controller, а також відправляти дані в view. Contoller є основною магістраллю програми, яка містить всю логіку, необхідну для роботи додатку. Такі дії, як додавання, редагування, видалення та вилучення даних з model, обробляються в controller. View - це те, з чим користувач бачить і взаємодіє. Користувачі передають дані через view.

Рис. 3.1. ілюструє область моделі, інтерфейсу і контролера програми MVC, а також те, як вони взаємодіють один з одним.

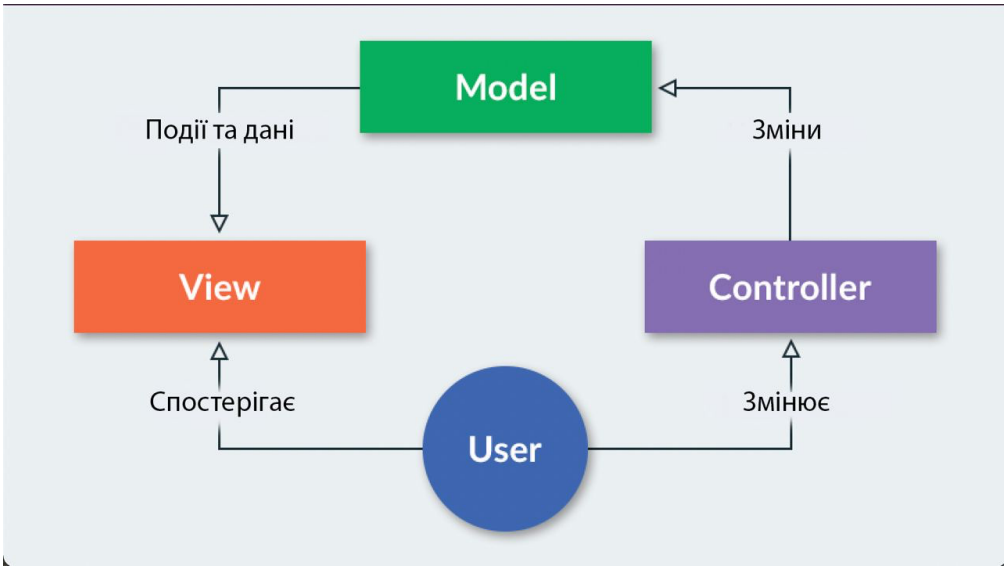

Рис.3.1. Область моделі, інтерфейсу і контролера програми MVC

Як показано на рис.3.1., view приймає вхідні дані від користувача і надсилає їх contoller. Потім contoller приймає дані і модифікує model. Нарешті, view отримує оновлені дані з modeli і відображає змінені дані в view.

Параметри середовища розробки життєво важливі для вибору правильних інструментів і середовища для виконання будь-якої роботи. Правильне поєднання таких інструментів підвищує ефективність роботи, економить час, оптимізує продуктивність, а також підвищує надійність виконуваної роботи. Використання правильних інструментів і мов програмування значно спрощує розробку програм.

Prybay.com повністю розроблений на MacBook Air 2020 з MacOS Big Sur в якості операційної системи. Крім того, для розробки повністю функціонального додатку для електронної торгівлі були використані ще кілька програмних засобів та інструментів.

PhpStorm - кроссплатформенна, легка і потужна IDE, розроблена JetBrains з вбудованими інструментами для розробників, інтелектуальною підтримкою кодування, інтелектуальною навігацією за кодом, надійним рефакторингом і тестуванням. Версія PhpStorm, яка використовується для розробки цієї програми - 2018.2. Крім того, для спрощення розробки програми Node встановлюється пакет Node.js і NPM.

Керування версіями - це засіб програмного забезпечення, що забезпечує трек змін, внесених у документ або вихідний код у спеціальній базі даних, щоразу при зміні джерела. Керування версіями спрощує спільну роботу команди або команди над проектом. Є численні системи управління версіями Git, Sub version, Team Foundation Version Control тощо. Git - одна з найбільш поширених систем управління версіями, яка була обрана для цього проекту, оскільки це розподілена система, яка робить акцент на швидкості, цілісності даних і підтримці розподілених нелінійних робочих процесів. Згідно з опитуванням, проведеним Stack Overflow серед професійних розробників програмного забезпечення в 2021 році, колосальний 88,4 відсотка респондентів використовували Git як систему контролю версій.

Весь код і історія коду для цієї програми були розміщені на GitHub, який є популярним сервісом управління версіями хостингу з 31 млн зареєстрованих користувачів і понад 96 млн репозиторіїв станом на листопад 2018 року .

3.2. NPM, Node.js, MongoDB, Express and EJS

Node Package Manager (NPM) - це менеджер пакетів з відкритим вихідним кодом, який допомагає встановити кілька пакетів JavaScript, доступних в Інтернеті. Він також використовується для спільного використання і розповсюдження коду, управління залежностями додатків, а також надання та отримання відгуків від інших користувачей. NPM є типовим менеджером пакунків Node.js.

Версія Node.js Long-Term Support (LTS) була завантажена і встановлена з офіційного сайту Node.js. Версія LTS була обрана так як це стабільна версія Node.js із запланованим циклом випуску і розширеною системою підтримки і патчем безпеки. Програму було ініціалізовано за допомогою команди npm init. Під час ініціалізації було вказано назву програми, версії, опис проекту та назву автора. Створено новий файл "package.json" з інформацією, наданою під час ініціалізації.

Після встановлення Node.js, Express framework версії 4.16.0 був встановлен за допомогою наступної команди.

sudo npm install –express –save -g

Крім того, було використано експрес-генератор для генерації стандартної структури тек програми з найбільш часто використовуваним проміжним програмним забезпеченням, вже включеним у додаток. Для встановлення експрес-генератора в програмі використовувалася наступна команда.

sudo npm install express-generator -g

Після цього новий експрес-додаток згортається за допомогою команди express <shopping_cart>. При цьому завантажуються мінімально необхідні файли та теки для програми. Після переходу до нещодавно створеної папки < shopping_cart > всі необхідні модулі, вже включені до файлу package.json, встановлюються за допомогою команди npm install.

Різні версії npm, Node.js і Express, які використовуються при розробці програми, показано на Рис 3.2.

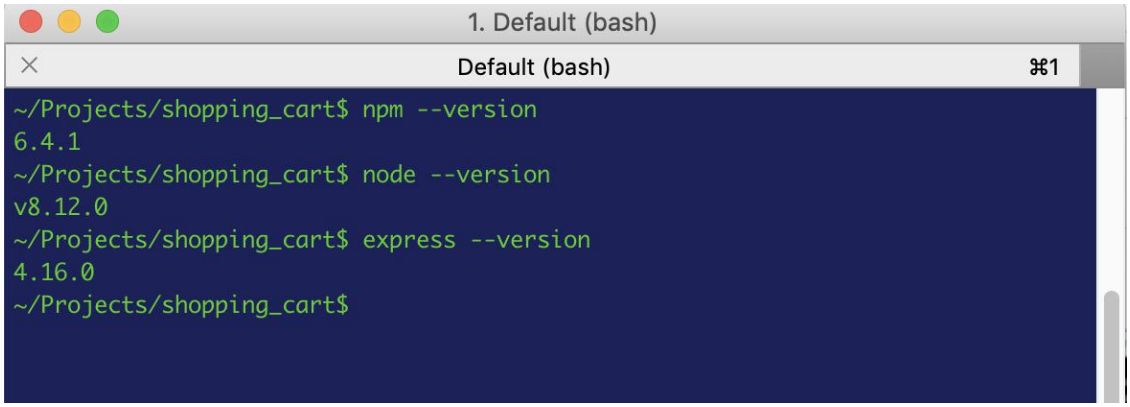

Рис. 3.2. Перевірка версій NPM, NODE та EXPRESS

Для розробки програми використовується версія Node.js версії 8.12.0, як показано на рис. 3.2., яка була останньою версією довгострокової підтримки (LTS) на момент розробки цієї програми. Крім того, при розробці проекту використовувалися npm версії 6.4.1 і Express версії 4.16.0.

MongoDB може бути встановлений локально на машині, на якій розробляється додаток або безпосередньо на MongoDB Atlas в якості хмарного сховища. Обліковий запис було створено на веб-сайті MongoDB. Після створення нового проекту було створено новий безкоштовний кластер MongoDB. Крім того, був створений користувач з доступом для читання і запису до бази даних, яка використовувалася в додатку Node.js для читання і запису даних. Крім того, поточну IP-адресу комп 'ютера було додано в список IP-адрес, щоб програма могла взаємодіяти з сервером. Нарешті, драйвер MongoDB було встановлено у програмі за допомогою команди "npm install mongodb - save".

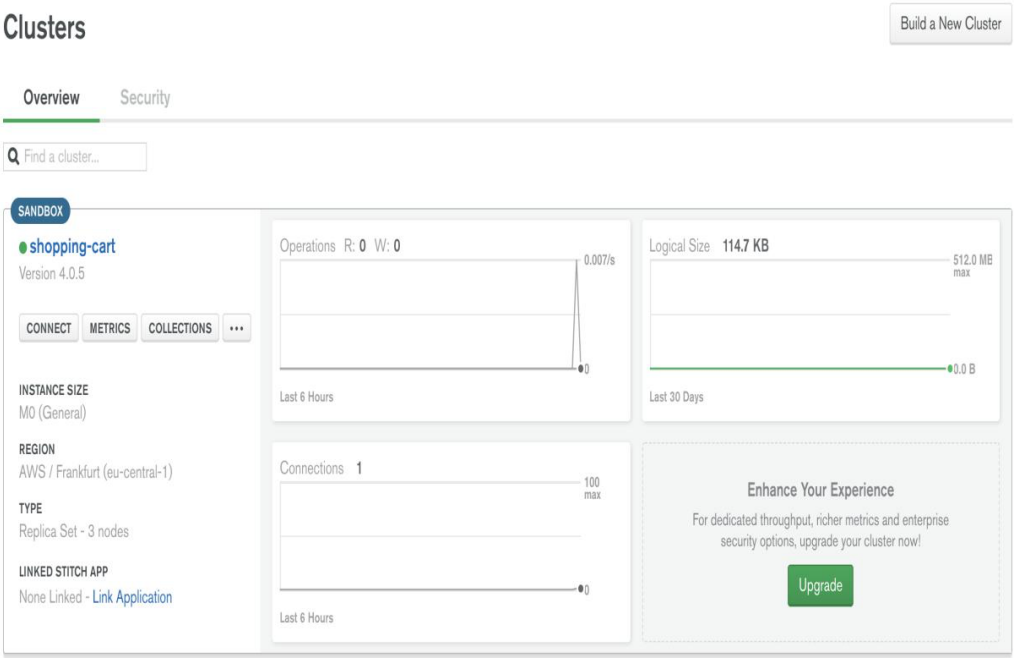

Рис. 3.3. Огляд кластера MongoDB, який використовується для програми.

Кластер MongoDB показує географічну область, де розташовано сервер, операції читання і записи, кількість активних з 'єднань на сервері, а також розмір бази даних, як показано на рис.3.3.

Наступним кроком є використання Mongoose для спрощення зв 'язку з сервером і базою даних. Остання версія mongoose, доступна на момент установки, версія 5.3.10, була встановлена за допомогою команди ‘npm install mongoose -save’ та з 'єднання бази даних було створено між додатком і сервером за допомогою URL, імені користувача і пароля, створеного при налаштуванні MongoDB Atlas. Крім того, були створені необхідні схеми і моделі, які будуть розглянуті нижче.

Для відображення динамічного вмісту на сторінці HTML вибрано вбудований механізм шаблонів JavaScript (EJS). Крім цього, розглядалися й інші шаблонні двигуни, такі як Pug і Handlebars. Ці шаблони використовують різний синтаксис та різні набори функцій для виводу вмісту HTML. Оскільки EJS використовує звичайний синтаксис HTML, а також може реалізувати певну логіку у фронтенді за допомогою простого JavaScript, EJS був обраний як механізм шаблону для програми.

EJS можна встановити за допомогою команди npm install ejs -save. Отже, він був встановлений як механізм інтерфейсу для програми.

3.3. Додаткові пакети Node

Крім основних пакетів та інструментів, описаних раніше, в проекті були встановлені численні інші пакети для забезпечення додаткових функцій програми. Деякі з них обговорюються нижче.

Nodemailer - це безпечний пакет, що полегшує надсилання повідомлень електронної пошти з програми. Він був встановлений з командою "npm install -save nodemailer". Крім звичайного тексту та HTML-коду зображення та файли, також можуть відправлятися за допомогою Nodemailer.

У деяких випадках користувачеві необхідно надіслати електронний лист. Типовими прикладами таких випадків є надсилання листа користувачеві для скидання пароля, повідомлення про успішну реєстрацію користувачеві тощо. Створення та обслуговування власного сервера електронної пошти є дуже складним і дорогим процесом. Отже, існувала потреба в сторонніх серверах електронної пошти, які могли б відправляти електронні листи безпечно і надійно. SendGrid є одним з механізмів доставки електронної пошти, що полегшує завдання відправки електронних листів. На сайті SendGrid було створено безкоштовний обліковий запис, який може надсилати 100 електронних листів на день безкоштовно. Транспортний плагін під назвою "nodemailer-sendgrid-transport" був встановлений з командою "npm install nodemailer-sendgrid-transport -save". Цей переносник дозволяє Nodemailer надсилати електронну пошту через API SendGrid.

Рахунок-фактура повинен бути створений, поки користувач замовляє продукти з додатку. PDF (Portable Document Oneat) є найпоширенішим форматом для надсилання рахунків-фактур. Доступні різні пакунки вузлів для друку pdf у Node.js. PDFKit створює друковані, багатосторінкові та складні документи кожного разу, коли користувач хоче побачити або завантажити їх за допомогою чистого JavaScript. Його було встановлено в проект за допомогою команди "npm install pdfkit -save". Рахунок-

фактура був створений з динамічними даними користувача. PDF-рахунок, створений за допомогою "pdfkit", показано на рис.3.4.


Prydbay.com

Invoice

	Product	Quantity	Rate	Amount
1	Oneplus 6	1	€549.99	€549.99
2	Computer	2	€599.99	€1199.98

Total Sum: €1749.97

Рис. 3.4. Рахунок-фактура

Як видно на рис.3.4, рахунок-фактура створюється для закупівлі, виконаної в Prydbay.com. Рахунок-фактура включає захардкодовані дані, а також динамічні дані, кількість і показник продукту, а також загальна сума.

Кожен інтернет-магазин потребує вид платіжних послуг, де клієнт може оплатити товари, які він придбав онлайн. Сторонній провайдер під назвою stripe інтегрований у додаток, який здійснює всі платежі в магазин. Клієнтська сторона програми збирає дані кредитної картки від клієнта за допомогою stripe і відправляється в stipe для перевірки. Як тільки дані кредитної картки є дійсними, stripe відправляє токен на сервер, і сума, яка повинна бути виплачена, відправляється в stripe знову з тим же токеном. Потім Stripe списує кошти з кредитної картки, управляє платежем і відправляє відповідь назад на сервер. Нарешті, після отримання успішної відповіді сервер очищає кошик користувача і створює замовлення. Приклад платіжної форми Stripe представлений на рис. 3.5.

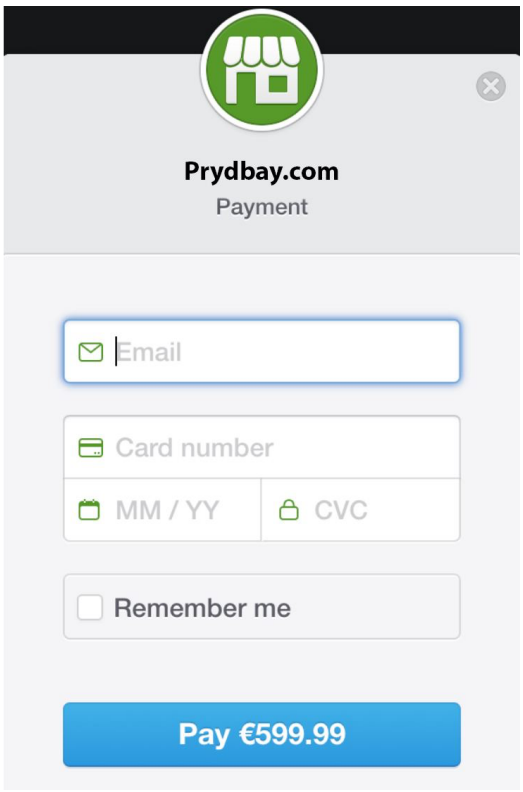

Рис. 3.5. Платіжна форма Stripe

На рис. 3.5 показана платіжна форма компанії Stripe, яка збирає інформацію про кредитну картку клієнта. Форма відправляється в stripe скрипт разом із загальною нараховуваною сумою і ключем API Stripe, наданим додатку.

3.4. Програмна логіка та безпека

Середовище розробки, всі пакети вузлів та інструменти були завантажені та встановлені. Після налаштування цієї базової структури для розробки програми почався процес розробки та фактичного кодування. Остаточна структура файлової системи проекту показана на рис. 3.6.

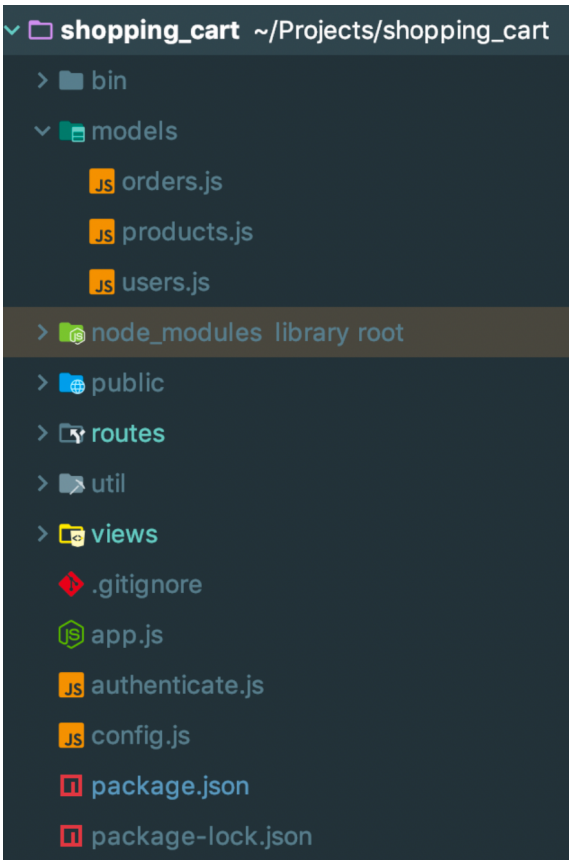

Рис. 3.6. Файлова структура програми

Програма складається з декількох папок і файлів, як показано на рис. 3.6, для реалізації логіки програми. Ці файли поділяються на різні папки, щоб структура залишалася простою та зрозумілою. Моделі, контролери та види програми розміщуються в різних теках, які відповідають архітектурі MVC. Каталог "node_modules" містить всі залежності серверної сторони. Папка "public" містить всі статично обслуговувані файли та клієнтські сценарії, такі як JavaScript-файли, таблиці

стилів, файли jQuery, а також зображення, завантажені на сервер. Подібно, папка "util" складається зі служб і бібліотек, таких як код для видалення файла.

Файл .gitignore містить у собі всі не відстежувані файли та папки, такі як node_modules, файли налаштування бази даних та інші приватні ключі API, які не слід завантажувати в систему управління версіями. Крім того, "authenticate.js" складається з прикладної логіки, яка визначає автентифікацію користувача, тоді як у файлі "config.js" зберігаються конфігураційні дані, необхідні для запуску сервера. Крім того, реалізація інших файлів і тек буде описана нижче.

Папка bin (та файл www) містить різні сценарії для запуску, зупинки або перезапуску проекту. Крім того, вона також встановлює HTTP-сервер із визначеним портом або 3000 за замовчуванням, прослуховує сервер і повідомляє про помилки, якщо такі є. Це основна точка входу в програму, яка спочатку вимагає файлу точки входу програми 'app.js' для обробки решти реалізації.

Папка "models" була створена для визначення стандартної структури для документів, що зберігаються в певній моделі. Додаток в основному складається з трьох моделей, а саме: користувачі, продукти та замовлення. Модель користувача, модель товару та модель замовлення зберігають колекцію документів користувача, документів товару та документів замовлення відповідно. Схема для об'єкта продукту створюється в моделі товару та експортується для використання в інших частинах програми, як показано на рис. 3.8.

Рис. 3.8. Схема моделі товару

Модель товару приймає властивості товару, визначені у схемі, як показано на рис. 3.8. Відповідно до нього, властивість imagePath є обов’язковою і має бути рядком. Подібним чином модель товару також містить назву, опис та ціну товару. Об'єкт «user» зберігає посилання на користувача, який створив цей продукт. Крім того, властивість mongoose 'timestamps' додає створений та змінений об'єкт дати та часу до властивості product. Нарешті, модель товару експортується для використання в інших частинах програми.

Папка «route» містить всю логіку програми на стороні сервера. REST API, а також маршрутизація реалізовані в цьому розділі. Файл «users.js» містить логіку та маршрути для всіх пов'язаних із користувачем дій, таких як відображення сторінки реєстру та входу, реєстрація користувача в базі даних та перевірка інформації про користувача кожного разу, коли користувач намагається увійти до системи. У цьому файлі існує маршрут виходу, а також маршрут скидання пароля. Також інші файли, такі як «orderRouter.js», «cartRouter.js» та «productRouter.js», містять усі маршрути та логіку, пов'язані із замовленням, кошиком та товаром відповідно.

Папка «views» містить клієнтську частину програми. Вона відображає шаблони HTML у браузері та надає інформацію клієнту. На рис 3.9 наведено загальну структуру каталогу "views".

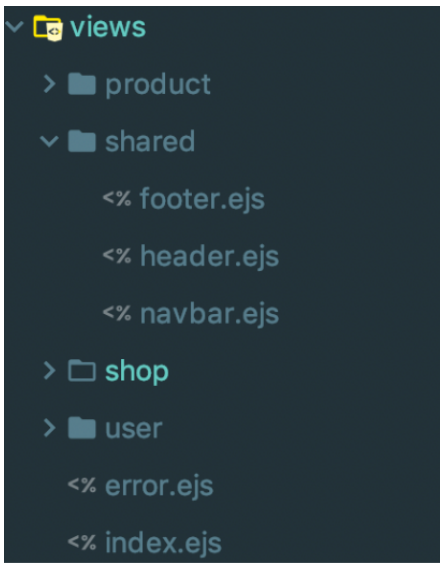

Рис. 3.9. Структура папки Views

Папка розподілена відповідно до структури сторінки, яку вона відображає, як показано на рис. 3.9. Файли «index.ejs» та «error.ejs» - це сторінки за замовчуванням, на яких відображається сторінка та сторінка помилок програми. Крім того, папка 'shared' містить заголовок, нижній колонтитул та панель навігації, яка є спільною для всіх відтворених сторінок. Каталоги товарів, магазинів та користувачів містять шаблони ejs для відображення відповідних HTML-сторінок.

Файл «app.js» є точкою входу до Express-програми. Він імпортує все необхідне проміжне програмне забезпечення, налаштовує програму з необхідними налаштуваннями та конфігурацією, створює об’єкт програми та експортує цей об’єкт із модуля. Фрагмент із файлу «app.js» наведено на рис. 3.10.

```
const path = require('path');

const express = require('express');
const cookieParser = require('cookie-parser');
const bodyParser = require('body-parser');
const logger = require('morgan');
const session = require('express-session');
const csrf = require('csurf');
const MongoDBStore = require('connect-mongodb-session')(session);
const mongoose = require('mongoose');
const flash = require('connect-flash');
```

Рис. 3.10. Фрагмент коду app.js

Як показано на рис. 3.10, у файл «app.js» імпортується та з нього використовується декілька бібліотек вузлів, моделей, маршрутів та об'єктів з інших частин програми, які були експортовані за допомогою функції 'module.exports'. Окрім цього, було встановлено підключення до бази даних, а об’єкт програми було створено за допомогою Express. Цей об’єкт програми був використаний для налаштування механізму перегляду або шаблону, додавання бібліотек, імпортованих як проміжне програмне забезпечення, додавання всіх імпортованих маршрутів для визначення маршрутів програми, обробки помилок та остаточного експорту створеного об’єкта програми.

У файлі «package.json» вказана детальна інформація про проект, а також необхідні залежності проекту. Він може містити ключові слова, домашню сторінку, помилки, ліцензію, автора, співавторів, сховище тощо, на додаток до основної інформації, такої як назва, версія та опис. Файл «package.json», який використовується в проекті, представлений на рис. 3.11.

Рис. 3.11. Файл «package.json»

Як показано на рис. 3.11, файл 'package.json' включає ім'я, опис та номер версії програми, залежності, необхідні для програми, та номер їх версії, а також сценарії для запуску або зупинки програми.

Безпека є життєво важливим елементом у всіх веб-додатках. Уразливість веб-додатків з кожним днем зростає через швидке збільшення кількості кібератак та порушення даних. Хакери використовують різні методи, такі як фішинг та злом для доступу до облікових даних для входу, а також даних кредитної картки. Більше того, незахищені маршрути забезпечують легкий доступ зловмисників до програми. Отже, веб-програми повинні бути надійно захищені. Більше того, оскільки веб-програми для електронної комерції включають велику кількість грошових операцій через купівлю-продаж товарів, такі програми повинні ставити безпеку найвищим пріорітетом. Невелике порушення безпеки може призвести до величезних збитків як для компанії, так і для її клієнтів.

Аутентифікація - це процес верифікації користувача, який отримує доступ до програми. Користувач верифікується на сервері за допомогою простої форми входу, даних онлайн-банкінгу, відбитків пальців, розпізнавання голосу або обличчя, а також шляхом сканування сітківки. Однак найпоширеніший процес аутентифікації користувача - це форма входу. Користувач створює для себе обліковий запис із деякою основною інформацією про них, наприклад електронною поштою та типами пароля, який шифрується та зберігається в базі даних. Отже, під час входу в систему сервер перевіряє інформацію, надану користувачем інформації, що зберігається в базі даних. Після успішної перевірки для користувача створюється сеанс, який зберігається у браузері клієнта як файл cookie та надсилається разом з кожним іншим запитом, надісланим користувачем.

На відміну від цього, авторизація - це процес ідентифікації, якщо аутентифікований користувач має дозвіл на доступ до послуг, що надаються додатком. Prydbay.com має три рівні ролей користувачів, а саме неавторизованих користувачів, автентифікованих користувачів та адміністраторів. Неавторизовані користувачі можуть переглядати всі доступні продукти, бачити деталі кожного продукту та реєструватися в програмі. Крім того, зареєстровані користувачі можуть отримати доступ до таких послуг, як додавання товару до кошика, оплата товарів у кошику та створення замовлення продуктів, що купуються на додаток до послуг, до яких отримують доступ неавторизовані користувачі.Адміністратори можуть додавати продукти до бази даних, редагувати їх та бачити замовлення, розміщені користувачами.

Вхідні дані, що передаються на сервер користувачами, завжди слід розглядати як загрозу безпеці. Користувачі, які використовують програму, не завжди є звичайними користувачами, можливо, це люди, які намагаються завдати шкоди серверу або викрасти конфіденційну інформацію з бази даних. Отже, дані, що надсилаються користувачем на сервер, повинні завжди перевірятися. Перевірка може бути реалізована як на стороні клієнта, так і на стороні сервера. Клієнтська перевірка

насправді не перешкоджає застосунку від шкідливих атак. Отже, перевірка на стороні сервера є обов’язковою для всіх програм, що включають дані користувача.

Express - валідатор - це сторонній пакет, який перевіряє дані, надіслані користувачами. Його можна встановити в програмі за допомогою команди 'npm install –save expressvalidator'. Вхідні дані перевіряються на кожному маршруті, а помилки, якщо такі є, збираються у константу, яка надсилається користувачеві для виправлення та повторної подачі даних. Подібним чином, вхідні дані також можна перевірити за допомогою Express - валідатора, перш ніж зберігати їх у базі даних.

Атаки CSRF виконуються шляхом викрадення сеансів у зареєстрованого користувача та надсилання запитів із підробленими даними на реальний сервер. Цей тип атаки не може викрасти дані; однак він може маніпулювати та змінювати дані користувача, такі як електронна адреса, пароль або навіть переказувати кошти.

Цей шаблон атаки пом'якшено в додатку за допомогою пакета вузлів під назвою «csurf». «Csurf» генерує маркер, який називається маркером «csrf», який вбудовується у всі форми, які при поданні змінюють дані користувача на сервері. Потім цей маркер надсилається на сервер у кожному запиті і перевіряється «csurf», якщо він є дійсним маркером. Таким чином, якщо фальшиві сайти надсилають запит на сервер із вкраденим сеансом, у запитах відсутній маркер і він відхиляється.

Node.js - це поєднання вбудованих основних модулів та інших сторонніх модулів. Такі модулі імпортуються в додаток відповідно до потреб, що допомагають у роботі з різними функціоналами, які недоступні в Node.js за замовчуванням. Реалізація одного модуля в додатку може вимагати ряд інших модулів, від яких він залежить. Чим більше сторонніх модулів у додатку, тим він вразливіший до атак та витоків даних.

Усіма сторонніми пакетами можна ефективно управляти за допомогою npm. «npm» — це менеджер пакетів, що входить до складу Node.js. З npm автоматично перевіряються всі встановлені бібліотеки. Більше того, команда «npm audit» перевіряє шкідливі пакети з встановлених сторонніх бібліотек і перелічує всі такі бібліотеки. Крім того, інший сторонній пакет під назвою «snyk» робить додаток більш безпечним, перевіряючи всі відомі вразливості у списку залежностей. Він був встановлений за

допомогою команди «npm install –g snyk» перевірив проект за допомогою команди «snyk test». Тестовий приклад проекту наведено на малюнку 3.12.

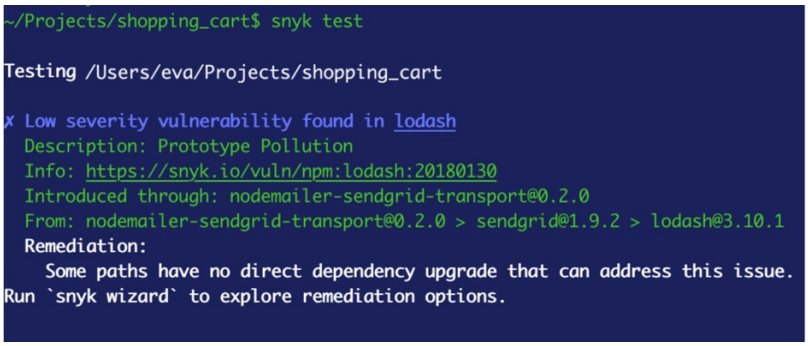

Рис. 3.12. Тестування проекту за допомогою snyk

Команда «snyk test» була запущена в інструменті командного рядка, де перераховані всі вразливості у зовнішніх залежностях програми, як показано на рис. 3.12. Проект виявив уразливість з низьким ризиком у залежності «nodemailer-sendgrid-transport» . Він також надає стратегії виправлення, запускаючи іншу команду «snyk wizard».

Вкладання шкідливого коду в заголовок HTTP може вплинути на вбудовані в браузер механізми безпеки, такі як політика того самого джерела чи фільтр XSS (міжсайтові сценарії). Це може призвести до виявлення конфіденційної інформації, такої як маркер CSRF. Більше того, зловмисник може використовувати вразливості XSS, налаштувавши файли cookie. Наприклад, зловмисник може ввести якийсь код, який деактивує політику браузера з тим самим джерелом і активує CORS (Cross-Origin Resource Sharing), конфіденційна інформація може бути викрадена за допомогою коду JavaScript.

Введення заголовка HTTP можна запобігти в програмі Node.js за допомогою стороннього пакета під назвою «Helmet». «Helmet» - це сукупність невеликих проміжних програм, що встановлюють відповідь заголовка HTTP. Різні модулі в бібліотеці можуть бути активовані в програмі, яка встановлює Політику безпеки вмісту, контролює попереднє завантаження DNS-браузера, запобігає розбиттю кліків, видаляє заголовок X-Powered-By, відключаючи місце для сторонніх клієнтів.

Зловмисники зможуть отримати доступ до програми, викравши сеанс у зареєстрованих користувачів. Сеанси та файли cookie, що використовуються в додатку, повинні бути завжди захищеними, щоб зловмисники не змогли викрасти їх з браузера клієнта. Сеанс може бути викрадений шляхом викрадення сеансу або фіксації сеансу. Під час викрадення сеансу зловмисник намагається викрасти або передбачити дійсний ідентифікатор сеансу та використовувати його для отримання несанкціонованого доступу до веб-сервера. Поширеними методами викрадення сеансу є прогнозування дійсного токена сеансу, слідкуваннння за токеном сеансу, атаки на стороні клієнта, такі як XSS та шкідливі коди JavaScript, а також перехоплення зв'язку між клієнтом та сервером. Однак під час фіксації сеансу зловмисник вже має доступ до дійсного сеансу, який потім вводиться у браузер жертви. Вкладення коду можна зробити, надіславши маркер сеансу в URL-адресу, або надіславши як приховане поле форми, або встановивши ідентифікатор сеансу як файл cookie в браузері жертви. Коли жертва входить до програми, веб-сервер не створює новий ідентифікатор сеансу і призначає той, що надісланий зловмисником, як ідентифікатор сеансу жертви. За допомогою цього ідентифікатора сеансу зловмисник може ввійти в додаток як жертва та виконувати шкідливі дії.

Проміжне програмне забезпечення Express-session використовується в додатку для обробки сеансу та файлів cookie. Він був встановлений у програмі за допомогою команди «npm install –save express-session». Для того, щоб захистити файли cookie, потрібно використовувати підписаний файл cookie з важко вгадуваним секретом. Крім того, при ініціалізації сеансу можна встановити різні інші параметри, що забезпечує безпеку файлів cookie. Параметри «secure» та «httpOnly», якщо встановлено значення true, надсилає файли cookie лише через захищене з'єднання HTTPS. Подібним чином параметри домену та шляху порівнюють домен файлу cookie з доменом сервера та надсилають файл cookie на запит, лише якщо обидва вони збігаються. Термін дії файлів cookie може закінчитися через певний проміжок часу, що мінімізує ризик неправильного використання.

Протокол транспортного рівня (TLS) - це криптографічний протокол, який забезпечує весь зв'язок між браузером та сервером. Це нова версія Secure Sockets

Layer (SSL). Цей протокол запобігає крадіжці даних, зашифровуючи дані ще до їх надсилання з браузера клієнта на веб-сервер. Сертифікат TLS можна легко отримати безкоштовно у програмі «Let's Encrypt». «Let's Encrypt» - це безкоштовний і відкритий центр сертифікації, який надає цифровий сертифікат, щоб увімкнути HTTPS-з'єднання для веб-сайтів.

MongoDB Atlas забезпечує безпечний спосіб зберігання даних у хмарі. Для мінімізації ризику атаки на базу даних застосовуються різні заходи безпеки. Кожен кластер в MongoDB Atlas має користувачів із певними ролями. Цього користувача потрібно налаштувати під час підключення бази даних до сервера. Тільки аутентифікований користувач із доступом до читання та запису до бази даних може виконувати операції читання, запису, оновлення та видалення в базі даних. На додаток до цього, Atlas дозволяє підключатись до бази даних лише з IP-адрес, перелічених у білому списку проекту. Потрібно додати IP-адресу машини, на якій розгортається проект, щоб успішно підключитися до бази даних.

3.5 Демонстрація прототипу інтернет-магазину цифрової техніки

Прототип MVP, програми для демонстрації використання Full Stack JavaScript, був успішно розроблений. Для розробки бекенд-системи використовувалися такі програми як NodeJS разом з Express і кількома іншими бібліотеками, тоді як EJS використовувався як механізм створення шаблонів. Для зберігання даних як документів використовувалася база даних NoSQL, MongoDB.

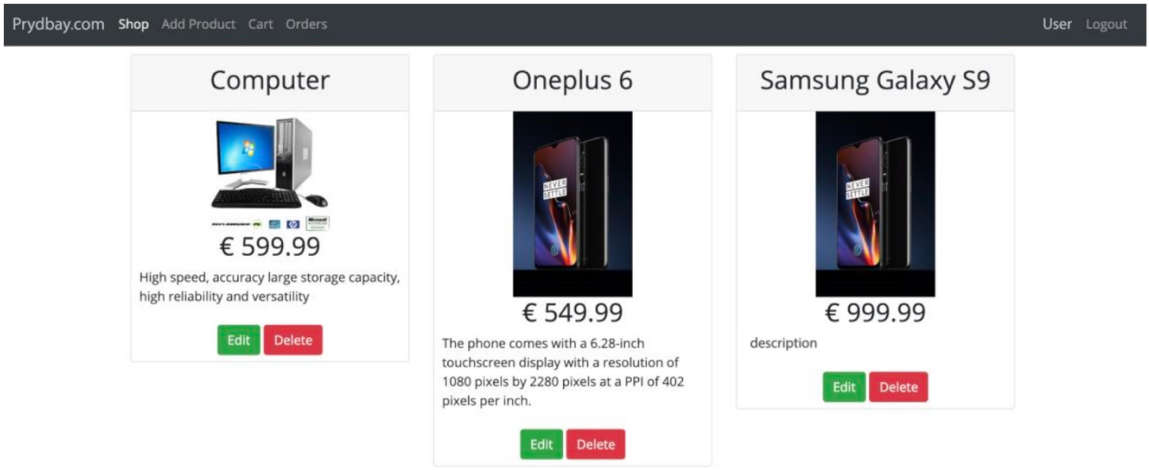

Рис. 3.13. Початкова сторінка веб-сайта

Початкова сторінка програми для користувачів admin показана на рис 3.13.

Вона має панель навігації для переходу до різних сторінок. Однак кнопки "Add Product (Додати продукт)", "Edit (Змінити)" та "Delete (Видалити)" відображаються тільки для користувачів-адміністраторів. Крім того, кожен окремий продукт пов'язаний з кнопкою редагування та видалення. Сторінка з інформацією про продукт доступна при натисканні на нього, який також має кнопку для додавання продукту до кошика. Основний вигляд сторінки кошика показано на рис 3.14.

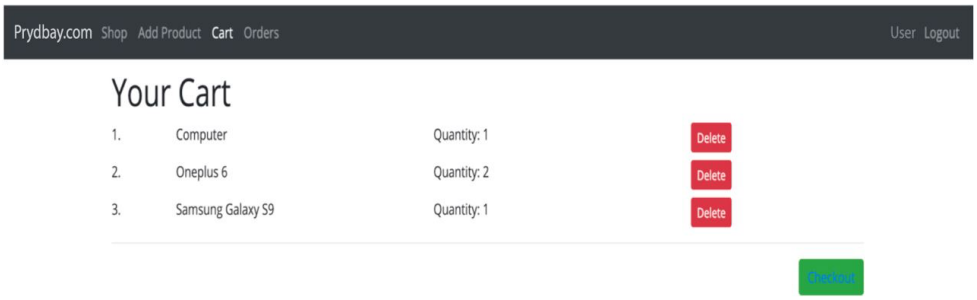

Рис. 3.14. Сторінка кошика

На сторінці "Orders (Кошик)" відображаються продукти, додані користувачем як показано на рис 3.14. Крім того, користувач може видалити продукт з кошика за

допомогою кнопки видалення. Кнопка "Your Orders (Оформлення замовлення)" відкриває форму, яка дозволяє користувачеві ввести інформацію про кредитну картку і оплатити замовлення. Після виконання платежу користувач перенаправляється на сторінку " Orders (Замовлення)" і відображається сторінка, як показано рис. 3.15.

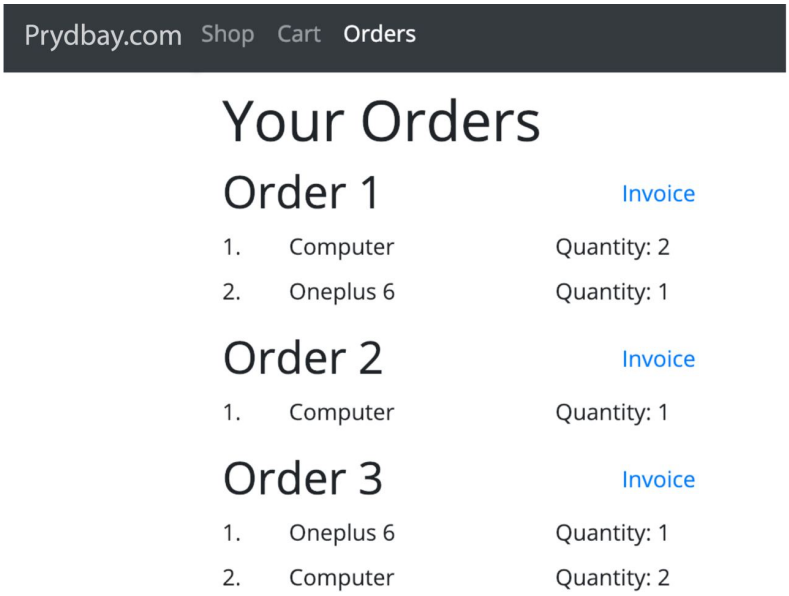

Рис. 3.15. Сторінка «Orders (Замовлення)»

Як показано на рис. 3.15, на сторінці «Orders (Замовлення)» відображаються всі замовлення, розміщені зареєстрованим користувачем. Сторінка також містить посилання для завантаження PDF-рахунка за кожним замовленням.

Клієнтська сторона цього додатку розроблена з використанням механізму шаблонів EJS, щоб зробити додаток готовим як мінімально життєздатний продукт.

Full-stack JavaScript виявився одним з найкращих стеків для розробки програми. Model, view і controller полегшив реалізацію бізнес-логіки в додатку. Більше того, не було необхідності перемикатися між різними мовами програмування та знання JavaScript було достатньо для розробки повної програми. Крім того MongoDB використовує формат JSON для зберігання документів, який є найкращим форматом для розробників JavaScript, оскільки легко трансформувати дані і працювати з ними.

Точно так само документи можуть зберігатися в хмарі навіть у середовищі розробки.Крім того, NodeJS, Express і MongoDB вже є перевіреною технологією і використовуються невеликими компаніями для технологічних гігантів; отже, легко отримати рішення від спільноти розробників у разі будь-яких труднощів.

Незважаючи на всі ці переваги, існує також кілька недоліків, які необхідно обговорити. По-перше, було складно керувати базою даних, оскільки MongoDB заснована на об 'єктній моделі документа, абсолютно відмінній від традиційних реляційних баз даних, які використовувалися. Він не забезпечує такий самий рівень функціональних можливостей і гнучкості, як реляційні бази даних. Висока ймовірність повторення одних і тих же даних у різних коллекціях, оскільки один файл містить інші впроваджені документи. Іноді рішення про збереження посилання на ідентифікатор об 'єкта або повний документ має бути прийнято, наприклад, при збереженні замовлення повний об' єкт продукту повинен бути вбудований в об 'єкт замовлення, оскільки зміни в об' єкті продукту впливають на замовлення.

По-друге, обробка платежу за допомогою сервісу Stripe була проблематичною в проекті, який використовує csurf для захисту від атак CSRF. Оскільки Stripe вже постачається із захистом CSRF, маркер csrf, створений проектом, спричинив помилку під час обробки запиту. Таким чином, маршрут оплати за участю Stripe повинен був бути виведений з каталогу контролера і був поміщений у файл "app.js" до реалізації проміжного програмного забезпечення csurf. Це якимось чином вплинуло на архітектуру MVC програми.

Висновки до розділу 3

У розділі було розглянуто програмну логіку та процес розробки веб застосунку. Створено інтерфейс та прораховано шаблон програмування, який дозволяє розділити логіку програми на три частини: model, view та controller (MVC). Підключено платіжну систему, створено шаблони рахунків-фактур, підключено необхідні додаткові пакети та бібліотеки, підключено БД та кластер MondoDB та Mongoose, використано фреймворк Express. Крім того, було використано експрес-генератор для

генерації стандартної структури тек програми з найбільш часто використовуваним проміжним програмним забезпеченням, вже включеним у додаток. Також були підключені необхідні пакети для забезпечення безпечної роботи додатку.

Основним завданням цього розділу була розробка та демонстрація прототипу інтернет-магазину цифрової техніки.

ВИСНОВКИ

Головною метою дипломної роботи було створення проекту інтернет магазину цифрової техніки, вивчення та звітування про різні аспекти повних стекових фреймворків JavaScript та розробка прототипу програми на його основі. Оскільки авторка роботи новачок у роботі з фреймворками JavaScript, але не новачок у мові програмування JavaScript, значну кількість часу було витрачено на вивчення різних аспектів фреймворків та платформ таких як NodeJS, Express та MongoDB. Переглянуто переваги та недоліки розробки веб-додатків із використанням цих технологій та порівняно з іншими подібними технологіями. Окрім цього, було визнано швидкість та продуктивність програми.

Досвід, накопичений під час дипломної роботи, довів, що JavaScript є найкращою мовою для розробки веб-додатків, оскільки він забезпечує всі компоненти, необхідні для розробки додатків, такі як NodeJS для серверної сторони, Angular, React та Express та інші для клієнтської сторони, а також MongoDB як базу даних. Додаток з використанням NodeJS відзначається великою швидкістю та швидким циклом розробки. Більше того, він найкраще підходить для розробки високопродуктивних та масштабованих веб-додатків. Наявність великої кількості бібліотек і пакетів NodeJS допомагає розширити послуги, що надаються NodeJS, без особливого занепокоєння. Оскільки нові функції JavaScript додаються до мови щороку, і щорічно планується випуск нових версій ECMAScript, безсумнівно, що JavaScript - це технологія майбутнього.

Тим не менше, JavaScript - не найкраще рішення для розробки всіх видів додатків. NodeJS не може підтримувати важку обробку даних та обчислення у серверній системі. Отже, високотехнологічні підприємства, які використовують машинне навчання або алгоритми, не можуть скористатися цією технологією. Незважаючи на це, залучення спільноти розробників JavaScript, його відкритість коду, допомагає NodeJS розвиватися та процвітати.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Иванова, Г.С. Технология программирования: учебник/ Г.С. Иванова – М.: КНОРУС, 2011. – 336 с.
2. HTML5 BOOK. «2.3. Блокові і рядкові елементи» [Електронний ресурс]. Режим доступу: https://html5book.ru/block-inline-elements/ (дата звернення 16.05.2021) – Назва з екрана.
3. Астахова, И.Ф. SQL в примерах и задачах: учебное пособие/ И.Ф. Астахова, А.П. Толстобров, В.М. Мельников – Минск: Новое знание, 2002. – 176 с.
4. Дейт, К. Дж. Введение в системы баз данных/Кристофер Дейт: пер. с англ. – 8-е изд. – М.: Издательский дом «Вильямс», 2006. – 1328 с.
5. Роберт Мартін «Чиста архітектура. Мистецтво розробки програмного забезпечення» / Роберт Мартін; Видавничий дім « Пітер », 2018. – 352с.
6. Поддубный, А. Расчет экономического эффекта от внедрения системы автоматизации [Електронний ресурс]. Режим доступу: http://www.antegra.ru/news/experts/_det-experts/4/(дата звернення 16.05.2021) – Назва з екрана.
7. Орлова, И.В. Экономико-математические методы и модели: компьютерное моделирование: Учебное пособие / И.В. Орлова. – М.: Вузовский учебник, НИЦ ИНФРА-М, 2013. - 389 с.
8. Девід Херрон «Node.js Розробка серверних веб-додатків на JavaScript» / Девід Херрон; Видавець Litres, 2017. – 144с.
9. Markus Egger — MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF [Електронний ресурс]. Режим доступу: https://www.packtpub.com/application-development/mvvm-survival-guide-enterprisearchitectures-silverlight-and-wpf (дата звернення 16.05.2021) – Назва з екрана.
10. Ітан Браун Веб-розробка із застосуванням Node і Express. Повноцінне використання стека JavaScript / Ітан Браун; Видавничий дім "Пітер", 2016. – 336с.