Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Рибалки Ростислава Миколайовича*
(ПІБ)

академічної групи *121-18-2*
(шифр)

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

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

на тему: *Розробка серверної частини інтернет-магазину комп'ютерної техніки на базі технології Node.js*

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

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

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

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

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18-2* *Рибалки Ростислава Миколайовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка серверної частини інтернет-магазину комп'ютерної техніки на базі технології Node.js*

затверджена наказом ректора НТУ «ДП» від *«18» травня 2022 р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

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

Завдання прийняв до виконання _____ *Рибалка Р.М.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 65 с., 11 рис., 3 дод., 28 джерел.

Об'єкт розробки: програмний додаток серверної частини для інформаційної системи інтернет-магазину з продажу комп'ютерів, комплектуючих та техніки.

Мета кваліфікаційної роботи: створення програмного забезпечення серверної частини для інформаційної системи інтернет-магазину для спрощення адміністрування, спрощення механізму оформлення товарів, ведення обліку замовлень і прибутку, оформлення продажу, здійснення аналізу, автоматизація процесу ведення електронної бази даних системи, виконання запитів та отримання звітів для аналізу діяльності підприємства.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

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

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

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

Практичне значення полягає у створенні програмного додатка, що надає можливість електронного зберігання даних про товари, продажі, надходження, ведення бази даних, підбиття підсумків з продажу.

Актуальність даного програмного продукту визначається великим попитом на подібні розробки, що оптимізують та спрощують дії щодо ведення бази даних складу-магазину, скорочують час на оформлення продаж, підвищують ефективність діяльності магазину шляхом електронного ведення документації з можливістю аналізу наявних даних.

Список ключових слів: ПРОГРАМА, БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК, АЛГОРИТМ, ДОДАТОК, ІНТЕРНЕТ-МАГАЗИН, ВЕБ-САЙТ, БЕКЕНД, ЗАПИТИ.

ABSTRACT

Explanatory note: 65 pages, 11 figs., 3 appx., 28 sources.

Object of development: software application of the server part for the information system of the online store for the sale of computers, components and equipment.

The purpose of the qualification work: creation of server software for the online store information system to simplify administration, simplify the mechanism of goods registration, accounting for orders and profits, sales, analysis, automation of the electronic database system, queries and reports for analysis.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the task.

In the first section the subject branch is analyzed, the urgency of the task and the purpose of development are determined, the statement of the task is formulated, the requirements to the software implementation, technologies and software are specified.

The second section analyzes the available solutions, selected platforms for development, designed and developed the program, describes the program, algorithm and structure of its operation, as well as calling and loading the program, determines the input and output data, describes the parameters of hardware.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance lies in the creation of a software application that provides the ability to electronically store data on goods, sales, receipts, database maintenance, summarizing sales.

The relevance of this software product is determined by the high demand for such developments, which optimize and simplify the actions of maintaining the database of the online store, reduce the time for registration of sales, increase the efficiency of the store by electronic documentation with the ability to analyze available data.

List of keywords: PROGRAM, DATABASE, INFORMATION SYSTEM, ACCOUNTING, ALGORITHM, APPLICATION, ONLINE STORE, WEBSITE, BACKAND, REQUESTS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

СКБД – система керування базами даних;

ПК – персональний комп’ютер;

ОС – операційна система;

HTML – (HyperText Markup Language) – мова розмітки гіпертекстових документів;

CSS – (Cascading Style Sheets) – каскадні таблиці стилів;

HTTP – (HyperText Transfer Protocol) – протокол передачі даних;

WAP – (Wireless Application Protocol) – технологія, що використовується для запуску Інтернет-додатків на мобільних терміналах;

HTTPS – (HyperText Transfer Protocol Secure) – розширення протоколу HTTP для підтримки шифрування з метою підвищення безпеки;

API – (Application Programming Interface) – набір чітко визначених методів для взаємодії різних компонентів;

CRUD – (Create, Read, Update, Delete) 4 базові функції управління даними «створення, зчитування, зміна і видалення»;

ID – (Identifier) – унікальна ознака об'єкта, що дозволяє відрізняти його від інших об'єктів;

JSON – (JavaScript Object Notation) – текстовий формат обміну даними;

BSON – (Binary JavaScript Object Notation) –двійкова форма представлення простих структур даних і асоціативних масивів;

NPM – (Node Package Manager) – менеджер пакунків для мови програмування JavaScript;

MEAN – акронім від «MongoDB, Express.js, Angular.js, Node.js» — скорочення, що описує комплекс програмного забезпечення, який, використовується для веб-розробки.

ВСТУП

В сучасному світі всі бізнес-процеси безупинно перетікають до мережі Інтернет і власник бізнесу рано чи пізно має розібратися з проблемою створення інтернет-магазину. І тут можливо декілька підходів.

Перш за все, це відкриття сторінки свого магазину на вже існуючій торгівельній платформі, так названі «маркетплейси». Цей метод є найпростішим, так як дозволяє зекономити на розробці власного інтернет-магазину, так як по-факту, все що вам потрібно, це лише створити сторінку свого магазину на вже готовому сайті, а опрацюванням замовлень і просуванням ваших товарів клієнтам, буде займатися сама платформа. Більшість невеликих роздрібних магазинчиків здебільшого і обирає цей спосіб, так як на початку свого шляху, ще не має потреби в окремому інтернет-магазині, та відповідних ресурсів для його створення та підтримки. Але цей метод має багато недоліків.

Подібний торгівельний майданчик має велику кількість обмежень, від редагування зовнішнього вигляду, до функціональності. До того ж, при розширенні бізнесу, цього може бути вже замало, як з точки зору практичності, так і репутаційно, адже великий бізнес з подібним «інтернет-магазином», буде сприйматися клієнтами та бізнес-партнерами не дуже серйозно. Тому, рано чи пізно, доводиться задуматися про створення інтернет-магазину.

Завдяки тому, що інтернет-магазин буде розроблятися з нуля, його стиль, оформлення та фунціонал можуть бути підігнані з урахуванням ваших потреб. Звісно, доведеться зазнати витрат на розробку, але зазвичай, воно того варте, адже повноцінний інтернет-магазин виконує одразу декілька функцій.

Він містить всю необхідну інформацію про ваш бізнес, контактні дані, посилання на соціальні мережі, відгуки клієнтів і в той же час дає клієнтам можливість ознайомитися з каталогом і одразу ж зробити замовлення, при цьому, покупець має можливість одразу сплатити обрані товари та послуги на сайті, що значно підвищує зручність його використання.

До того ж, інтернет-магазин є більш безпечним, так як не залежить від працездатності сторонніх платформ або ресурсів і прив’язаний лише до хостингу, проблем з яким, у сучасному світі, зазвичай не виникає. Варто відміти, що окрім перерахованого раніше, повноцінний інтернет-магазин, це більш гнучке та універсальне рішення, яке підходить у більшості сфер бізнесу. У будь який момент часу, в залежності від побажань та вподобань власника, або ж у випадку зміни тенденцій на ринку, чи при рішенні зміни напрямку, розширенні бізнесу тощо, зовнішній вигляд і функціонал сайту, може бути змінено.

Процес розробки інтернет-магазину складається з умовно двох частин, клієнтської, та серверної, або ж як їх ще називають фронтенд та бекенд. Фронтенд – це все, що браузер може читати, виводити на екран та/або запускати. Тобто це HTML, CSS та JavaScript. В свою чергу, бекенд – це усе, що працює на сервері, тобто «не в браузері» або «на комп'ютері, підключеному до мережі (зазвичай, до Інтернету), який відповідає на повідомлення від інших комп'ютерів».

Результатом виконання даної роботи є застосунок, який забезпечує працездатність інтернет-магазину, та безпеку і зручність його використання клієнтами.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

Веб-сервер - сервер, який отримує HTTP-запити від клієнтів і надає їм відповіді HTTP, як правило, разом із HTML-сторінками, файлами, потоками даних тощо. Веб-сервер може бути програмним забезпеченням, яке діє як веб-сервер, або комп'ютером, який безпосередньо запускає програмне забезпечення. Веб-браузер надсилає запит на сервер, щоб отримати ресурс, призначений для URL-адреси. Ресурсом може бути сторінка HTML, файл, потік або інші дані, які запитує клієнт. У відповідь веб-сервер надсилає запитані дані клієнту.

Перший в історії веб-сервер, побачив світ у 1990 році. Він був створений та запущений британським вченим Тімом Бернерсом-Лі, як проект, що мав зробити обмін даними між вченими інституту CERN (Європейський центр ядерних досліджень), базуючись на гіпертекстовій системі. В результаті, Бернерс-Лі, в ході виконання проекту, розробив новий браузер під назвою WorldWideWeb та перший в світі веб-сервер, що було запущено на комп'ютері NeXT під керуванням операційної системи NeXTSTEP. Сам же сервер, отримав назву CERN httpd.

Станом на квітень 2022 року за дослідженням Netcraft [25], найпоширенішим веб-сервером, на який припадає понад 22% ринку, є Apache, безкоштовний веб-сервер, який найчастіше використовується в UNIX подібних операційних системах.

Серед інших веб-серверів варто виділити:

- IIS від Microsoft, що поширюється разом із сімейством операційних систем Windows;
- Nginx – безкоштовний веб-сервер, розроблений Ігорем Сисоєвим з 2002 року і дуже популярний серед великих веб-сайтів;
- Lighttpd – безкоштовний веб-сервер;

- Google Web Server – це веб-сервер, від компанії Google;
- Resin – це безкоштовний сервер веб-додатків;
- Cherokee – це безкоштовний веб-сервер, який керується лише через веб-інтерфейс;
- Rootage – це веб-сервер, написаний на Java;
- THTTPD – це простий, невеликий, швидкий і безпечний веб-сервер;
- Open Server – безкоштовна програма з графічним інтерфейсом, яка використовує багато спеціалізованого безкоштовного програмного забезпечення;
- H2O – це безкоштовний, швидкий веб-сервер, написаний на C;
- nghttp2 – це веб-сервер, вбудований у Node.js;
- Go HTTP – це веб-сервер, вбудований у Go;

Як клієнт для доступу до веб-сервера можна використовувати різноманітні пристрої. Найпоширенішим способом є веб-браузер, але, крім того, клієнтом може бути мобільний телефон через протокол WAP або HTTP, спеціальне програмне забезпечення або навіть пристрій, який може запитувати оновлення або іншу інформацію з веб-сервера.

1.2. Призначення розробки та галузь застосування

Веб-сервери використовуються для підтримки веб-додатків. Основною функцією сервера є підтримка файлової активності при перегляді веб-сайту щодня й щохвилини.

Веб-сервер може мати різні додаткові функції, такі як:

- автоматизація веб-сторінки;
- запис запитів користувачів;
- автентифікація та авторизація користувачів;
- підтримка динамічно генерованих сторінок;
- безпечні з'єднання з клієнтами на базі HTTPS.

Подібні додатки часто використовуються для роботи саме додатків у сфері електронної комерції. Цифрова економіка, включаючи всі фінансові та торгові операції через комп'ютерні мережі, а також бізнес-процеси, пов'язані з цими операціями.

Основними компонентами електронної комерції є мобільна комерція, електронні перекази грошей, управління ланцюгом поставок, Інтернет-маркетинг, онлайн-обробка транзакцій, електронний обмін даними, системи управління запасами та системи автоматичного збору даних.

Основними підставами для активного розвитку галузі є технологічний прогрес та зростання кількості активних користувачів мережі Інтернет. Електронна комерція дозволяє клієнтам долати географічні бар'єри та купувати товари в будь-який час і в будь-якому місці. Онлайн та традиційні ринки мають різні бізнес-стратегії. Якщо традиційні роздрібні продавці пропонують менший асортимент товарів через місце на полицях, онлайн-магазини часто не зберігають запаси, а натомість надсилають замовлення клієнтів безпосередньо виробникам. Стратегії ціноутворення традиційних і інтернет-магазинів також відрізняються. Ціни традиційних роздрібних продавців засновані на завантаженості магазинів і витратах на запаси. Ціна в інтернет-магазині залежить від швидкості доставки.

1.3. Підстави для розробки

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

Отже, підставами для розробки «виконання кваліфікаційної роботи» є:

− освітня програма 121 «Інженерія програмного забезпечення»;

− навчальний план та графік навчального процесу;

− наказ ректора Національного технічного університету «Дніпровська

політехніка» №268-с від 18 травня 2022р;

− завдання на кваліфікаційну роботу на тему «Розробка серверної частини інтернет-магазину комп’ютерної техніки на базі технології Node.js».

1.4. Постановка завдання

Метою роботи є розробка серверної частини для інтернет-магазину, що забезпечить коректне функціонування всіх функцій сайту та його логіки, організацію та зв’язок з базою даних, основну логіку та алгоритми роботи API, інтеграцію із необхідними зовнішніми сервісами.

В даній роботі розглядається сервер, розроблений на базі технології Node.js з використанням фреймворку Express, призначеного бля роботи з веб, а також нереляційної СКБД MongoDB разом з бібліотекою Mongoose, яка створює з'єднання між MongoDB та платформою веб-додатків Express. Node.js забезпечує високий рівень швидкодії та більш зручний зв’язок з базою даних та клієнтською частиною.

Розроблений застосунок, може використовуватися як для функціонування конкретного проекту інтернет-магазину для якого він був розроблений, так і ,в разі необхідності, для сторонніх проектів, при цьому підлягаючи незначним маніпуляціям і змінам.

Головним користувачем застосунку буде як адміністратор ресурсу, так і звичайний клієнт інтернет-магазину. Адміністратор має можливість додавати, оновлювати та видаляти товари, опрацьовувати замовлення, редагувати категорії, а також контроль списку клієнтів. Також адміністратор отримує статистку з продаж товарів, прибутку, кількості користувачів.

З іншого ж боку, звичайний користувач не має доступу до функціоналу адміністрації. Він може зареєструватись та авторизуватися на сайті, переглядати категорії, товари і їх опис, додавати їх до кошику та робити замовлення.

Інформація яку надає сервер, в першу чергу використовується для задоволення запитів користувачів. Це відображення категорій, списку товарів та

оформлення замовлень. Для адміністрації, вихідні дані використовуються для підведення статистики та надання загальної інформації про інтернет-магазин, як список товарів та категорій, так і список користувачів та перелік замовлень та їх статусів і деталей.

Для збору та подальшої обробки даних, які сервер отримує від клієнтської частини, використовується JSON. Сервер отримує вхідні дані користувача, після чого опрацьовує їх та зберігає до бази даних. Це можуть бути як дані користувача вказані при реєстрації, так і деталі його замовлення. Якщо дивитися з боку адміністрації інтернет-магазину, то вхідною інформацією будуть дані, вказані при створенні або оновленні товару та категорії. Окрім цього, адміністратор має змогу контролювати статус замовлення і змінювати його, що теж є вихідною інформацією.

Для підтримки працездатності застосунку, достатньо стабільного з’єднання до мережі та комп’ютера або хостинга, що задовольнить рекомендовані технічні вимоги. В ролі адміністратора може виступати як одна людина, так і декілька, в залежності від обсягів продаж та кількості активних користувачів і замовлень.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблений застосунок має забезпечувати стабільне виконання наступних функцій:

- коректне функціонування сайту;
- створення бази даних;
- забезпечення зв’язку з базою даних;
- реєстрація та авторизація користувачів;
- верифікація користувачів за допомогою веб-токенів;
- перевірка рівнів доступу;
- забезпечення фунціоналу для адміністрування сайту;

- шифрування паролів користувачів;
- зберігання даних клієнтів до бази даних;
- надання інформації з бази даних за запитом;
- алгоритми роботи API;
- робота CRUD операцій;
- обробка даних оплати від клієнта.

1.5.2. Вимоги до інформаційної безпеки

Задля забезпечення інформаційної безпеки необхідно в першу чергу подбати про безпеку комп'ютера, або ж хостинга, на якому відбувається запуск сервера. Незалежно від розміщення інтернет-магазину, локально на власному пристрої, чи в хмарі за допомогою спеціалізованого хостинг-сервісу, особливу увагу слід приділити безпеці. Як мінімум, необхідно використовувати антивірусний застосунок і регулярно оновлювати його модулі та бази, створювати резервні копії даних, стежити за конфігурацією веб-сервера з відкритими можливостями та доступом лише в обсязі, необхідному для сайта, використовувати надійні паролі для доступу та часто їх змінювати.

Зокрема не варто забувати про використання актуальних версій програмного забезпечення та технологій, на яких базується інтернет-магазин. У більшості випадків проникнення шкідливого ПЗ можливе через вразливості в програмному коді. Оновлення програмної платформи сайту допомагає усунути потенційні діри в безпеці.

Задля впровадження додаткового рівня безпеки використовується JSON Web Token, що створюється сервером і шифрується, після чого присвоюється користувачу для подальшого підтвердження особи. В середині цього токену якраз і зберігається зашифрована інформація користувача, а саме його ID та чи має він права адміністратора.

Коли користувач реєструється або авторизується на веб-сайті, його пароль негайно шифрується за допомогою бібліотеки bcrypt.js, яка забезпечує безпечне

шифрування пароля за допомогою алгоритму шифрування Blowfish, який реалізує блочно-симетричне шифрування. Оригінальний же пароль користувача в чистому вигляді ніде не зберігається.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для стабільної роботи застосунку необхідно забезпечити відповідні, або подібні їм, технічні характеристики обладнання:

- центральний процесор: Intel x86 або сумісний процесор, 4 ядра, тактова частота 1,4 ГГц і вище;
- оперативна пам’ять: Мінімум 2 Gb;
- відеоадаптер: Вбудований або дискретний;
- відеопам'ять: 64 Mb;
- накопичувач: Від 2 Gb;
- підтримувані протоколи передачі даних: HTTP / HTTPS;
- вимоги до мережі: Стабільний канал зв'язку від 10 Мб/сек;

Серед операційних систем Windows рекомендовано 64-розрядну операційну систему, наприклад, Windows Vista, Windows 7, Windows 8, Windows 10, Windows Server 2008 або Windows Server 2012. У разі використання операційної системи на базі Linux, рекомендується використовувати x86 систему, наприклад, Astra Linux, ALT Linux, CentOS, Debian, Red Hat Enterprise Linux, Ubuntu.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для запуску та роботи серверної частини інтернет-магазину на комп’ютері має бути встановлена версія програмної платформи Node.js не нижче 14.17.3. Перед запуском, необхідно виконати команду «npm i», що встановить всі бібліотеки та пакунки, необхідні для роботи застосунку. Для середовища виконання Node.js, npm (Node Package Manager) є менеджером пакунків за замовчуванням і встановлюється разом, за замовчуванням.

Задля подальшої роботи необхідно створити базу даних MongoDB та колекцію, в якій будуть зберігатися всі необхідні дані та налаштувати змінні оточення, які виконують роль параметрів конфігурації (HTTP-порт, який буде прослуховувати ця програма, рядок підключення до бази даних, ключ шифрування веб-токенів тощо).

Як платформа Node.js, так і MongoDB є сумісними з більшістю операційних систем та браузерів, що відкриває доступ до кросплатформи та високого рівня масштабованості.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ЗАСТОСУНКУ

2.1. Функціональне призначення програми

Програма має виконувати з’єднання з БД, а також приймати та опрацьовувати HTTP запити від користувачів інтернет-магазину та його адміністрації. Ці запити, відповідають конкретним функціям, а саме:

- реєстрація користувачів на сайті;
- авторизація раніше зареєстрованих користувачів;
- редагування та оновлення даних користувача;
- видалення користувача;
- створення нових категорій та їх подальше редагування та видалення;
- додавання нових товарів;
- завантаження зображень товарів та можливість їх заміни в подальшому;
- редагування та видалення вже існуючих товарів;
- отримання загальної статистики з кількості товарів;
- збереження та обробка замовлень;
- відображення всіх замовлень;
- відображення замовлень конкретного користувача;
- оновлення статусу замовлення;
- видалення замовлень;
- підрахунок загальної кількості замовлень;
- підрахунок прибутку;

2.2. Опис застосованих математичних методів

Оскільки особливості предметної галузі розв'язуваної задачі не передбачають застосування математичних методів, при розробці серверної

частини інтернет-магазину комп’ютерної техніки на базі технології Node.js математичні методи не використовувалися.

2.3. Опис використаної архітектури та шаблонів проектування

В ході розробки застосовується архітектура клієнт-сервер. Вона може бути визначена як концепція інформаційної мережі, де більшість ресурсів зосереджена на серверах, що обслуговують своїх клієнтів.

Модель взаємодії клієнт-сервер насамперед визначається розподілом відповідальності між клієнтом і сервером. Операції можна розділити на 3 рівні:

- рівень представлення даних, тобто інтерфейс користувача, що відповідає за представлення даних користувачеві та отриманні від нього даних та запитів;
- прикладний рівень, що відповідає за реалізацію основної функціональної логіки застосунку та обробку вхідної та вихідної інформації;
- рівень керування даними для зберігання та доступу до даних.

Серед переваг архітектури клієнт-сервер, можна виділити те, що більша частина всіх обчислень опрацьовується на серверній частині, зменшуючи навантаження на комп’ютер користувача. Завдяки обмеженню даних, що надсилаються сервером користувачеві до рівня лише необхідних, можна значно зменшити використання трафіку.

При створенні програми з нуля важливо використовувати узгоджений стандартизований стек програмного забезпечення. Створення серверної частини із набором інструментів, призначених для спільної роботи, скорочує час розробки та оптимізує ресурси.

MEAN — це стек з відкритим вихідним кодом, який в основному використовується для створення веб-додатків. Програми стека MEAN є гнучкими, масштабованими та розширюваними, що робить їх ідеальним кандидатом для хмарного хостингу. Стек містить власний веб-сервер, тому його можна легко розгорнути, а базу даних можна масштабувати на вимогу для

компенсації тимчасових сплесків використання. MEAN виходить у світ оптимізованим, щоб скористатися всіма перевагами економії коштів і підвищення продуктивності хмари.

Схема роботи додатку на базі MEAN стеку, у повній мірі співпрацює з клієнт-серверною архітектурою. Як можна побачити з рис. 2.1 робота застосунку відбувається в декілька етапів.

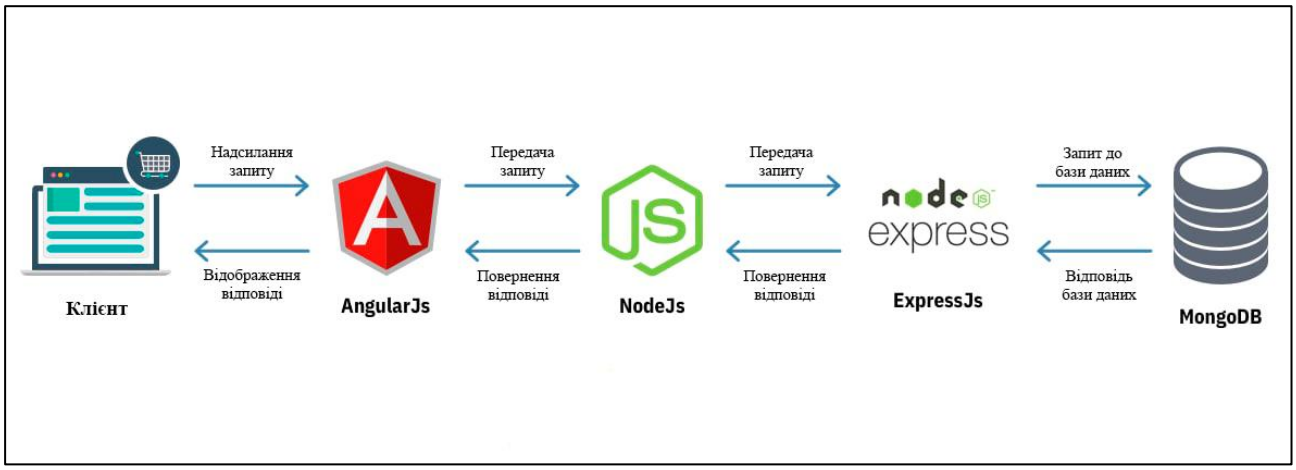

Рис. 2.1. Схема роботи MEAN стеку

Першим етапом є AngularJS. Це інтерфейсний фреймворк системи середнього стеку. Його мета – відправити запит клієнта вперед, і коли запит буде оброблений, він відобразить результати. Він починається з того, що клієнт робить запит, який потім обробляється AngularJS.

Далі NodeJS обробить запит, надісланий AngularJS. Після того, як AngularJS обробить запит, він перейде до наступного етапу, де NodeJS дозволить запит.

Після цього ExpressJS, що використовується для створення API інтерфейсів веб-додатків, перенаправить початковий запит до бази даних для отримання результатів. Залежно від типу запиту, ExpressJS встановить з'єднання з MongoDB, яка потім викличе або встановить дані.

MongoDB – це база даних усієї системи. MongoDB зберігає всі дані та витягує їх у разі потреби. MongoDB виконує роботу, а потім передає її ExpressJS, яка потім перенаправляє її в NodeJS.

Нарешті, запит клієнта відображається AngularJS, коли він отримує результат від NodeJS. Весь цей процес можна узагальнити, сказавши, що AngularJS відображає результати, отримані MongoDB.

2.4. Опис використаних технологій та мов програмування

Основа проекту створюється на базі технології Node.js та спеціалізованого фреймворку для роботи веб-додатків Express. Основна ідея Node.js полягає у використанні неблокуючого, керованого подіями вводу-виводу, щоб зробити все простим і ефективним під час роботи з додатками. Node.js дозволяє одночасно обробляти велику кількість з'єднань, підтримуючи при цьому високу пропускну здатність, тим самим забезпечуючи великий рівень масштабованості.

Середовище виконання Node.js базується на двигуні Chrome V8 JavaScript. Використовуючи Node.js як основу, необхідно враховувати всі переваги повнофункціональної розробки на JavaScript, а це, як велика кількість безкоштовних інструментів, так і оптимальна швидкість роботи застосунку з можливістю подальшого переходу на кросплатформу.

Ще однією важливою перевагою є асинхронна обробка запитів. У контексті серверної частини синхронна обробка передбачає, що код виконується послідовно. Таким чином, кожен новий запит блокує потік інших запитів, а інші команди виконуються лише після виконання попередньої команди. Асинхронний підхід, який використовується в Node.js, у свою чергу, максимізує однопотокову обробку, тим самим скорочуючи час відповіді в кілька разів.

Багато провідних компаній після переходу змогли досягти значних покращень. Міжнародна система електронних платежів PayPal, наприклад, змогла знизити час відгуку на 35% після відмови від Java на користь Node.js.

Як просте у використанні середовище програмування, Node.js став ідеальним рішенням для так званих мікросервісних архітектур. Цей підхід передбачає розробку єдиної програми як набору невеликих служб, кожна з яких використовує свій власний процес і взаємодіє з тривіальними механізмами, часто з API ресурсів HTTP.

У той час як інші серверні технології, такі як PHP і Ruby on Rails, можуть забезпечувати зв'язок за допомогою JSON, Node.js робить це без використання JavaScript для перетворення між двійковими моделями. Це особливо корисно при створенні RESTful API для підтримки баз даних NoSQL. Це «безперебійне» підключення до одного з основних стандартів передачі даних є ще однією перевагою екосистеми JavaScript.

Лише кілька мов програмування пропонують багату екосистему пакунків, як-от NodeJS. При встановленні Node.js, автоматично встановлюється програма NPM (Node Package Manager). Будь-який розробник Node.js може запакувати свої бібліотеки та рішення в модуль, який кожен може встановити за допомогою NPM, офіційного менеджера пакунків Node. На NPM зібрані десятки тисяч бібліотек та інструментів для розробки. Завдяки постійній підтримці з боку спільноти NodeJS, NPM зосереджується на тому, щоб заохочувати користувачів додавати нові пакети, щоб у вас було багато готових рішень для вашої конкретної проблеми.

В свою чергу фреймворк Express є легким та доступним для освоєння, для початку вивчення якого, достатньо базових знань JavaScript, оскільки він не використовує екзотичний синтаксис або структуру програми. До того ж існує зручна власна документація, а також є безліч прикладів та навчальних посібників, які інші користувачі також опублікували в Інтернеті. Зважаючи на потреби гнучкості для всіх видів API, Express є одним з найкращих рішень.

Він підтримує проміжне програмне забезпечення, яке легко створити самостійно або додати вже встановлене для багатьох різних сценаріїв, наприклад, переконатися, що всі маршрути після певної точки в REST API мають доступні дані, або додати перевірки безпеки, які дозволяють або забороняють

доступ до маршрутів, на підставі запиту або будь-яких критеріїв, які вважаються доречними.

Величезна кількість існуючих пакунків на NPM, чудово працюють разом із Express, щоб розширити його можливості та зняти тягар необхідності мати справу з багатьма поширеними ситуаціями, з якими стикається розробка API.

Для роботи на рівні керування даними, використовується нереляційна СКБД MongoDB в поєднанні з спеціалізованою бібліотекою Mongoose. В порівнянні з реляційними базами даних MongoDB пропонує документо-орієнтовану модель даних, завдяки чому вона працює швидше, має кращу масштабованість, її легше використовувати. Якщо традиційний SQL використовує таблиці, то MongoDB, в свою чергу, використовує колекції. Якщо в реляційній базі даних таблиці зберігають жорстко структуровані об’єкти одного типу, колекція може містити різні об’єкти з різною структурою та різними наборами властивостей.

Одним із популярних стандартів для обміну та зберігання даних є JSON. Він ефективно описує складні структури даних. Підхід до зберігання MongoDB у цьому відношенні подібний до JSON, хоча фактично і не використовує JSON у звичному вигляді. MongoDB використовує скорочений формат під назвою BSON, що дозволяє швидше шукати та обробляти дані.

Бібліотека Mongoose, відповідальна за створення зв’язку між БД та платформою веб-додатків Express, дозволяє визначати об'єкти із строго-типізованою схемою, що відповідає документу MongoDB. Mongoose надає величезний набір функціональних можливостей для створення та роботи зі схемами. Відразу після визначення схеми Mongoose дає можливість створити модель, засновану на певній схемі. Потім модель синхронізується із документом БД за допомогою визначення схеми моделі. На даний момент Mongoose містить вісім типів даних схеми, які можуть мати властивість, що зберігається в MongoDB, а саме:

- String;
- Number;
- Date;

- Buffer;
- Boolean;
- Mixed;
- ObjectId (_id);
- Array.

2.5. Опис структури програми та алгоритмів її функціонування

Якщо розглянути окремі складові програми та її компоненти, то можна побачити чітку, логічну і просту структуру (рис.2.2). Основна програма, умовно кажучи, сам сервер, при запуску слухає з'єднання на порті 3000 і запускає з’єднання з базою даних, а також запускає виконання проміжних обробників і чекає на звернення через описані маршрути.

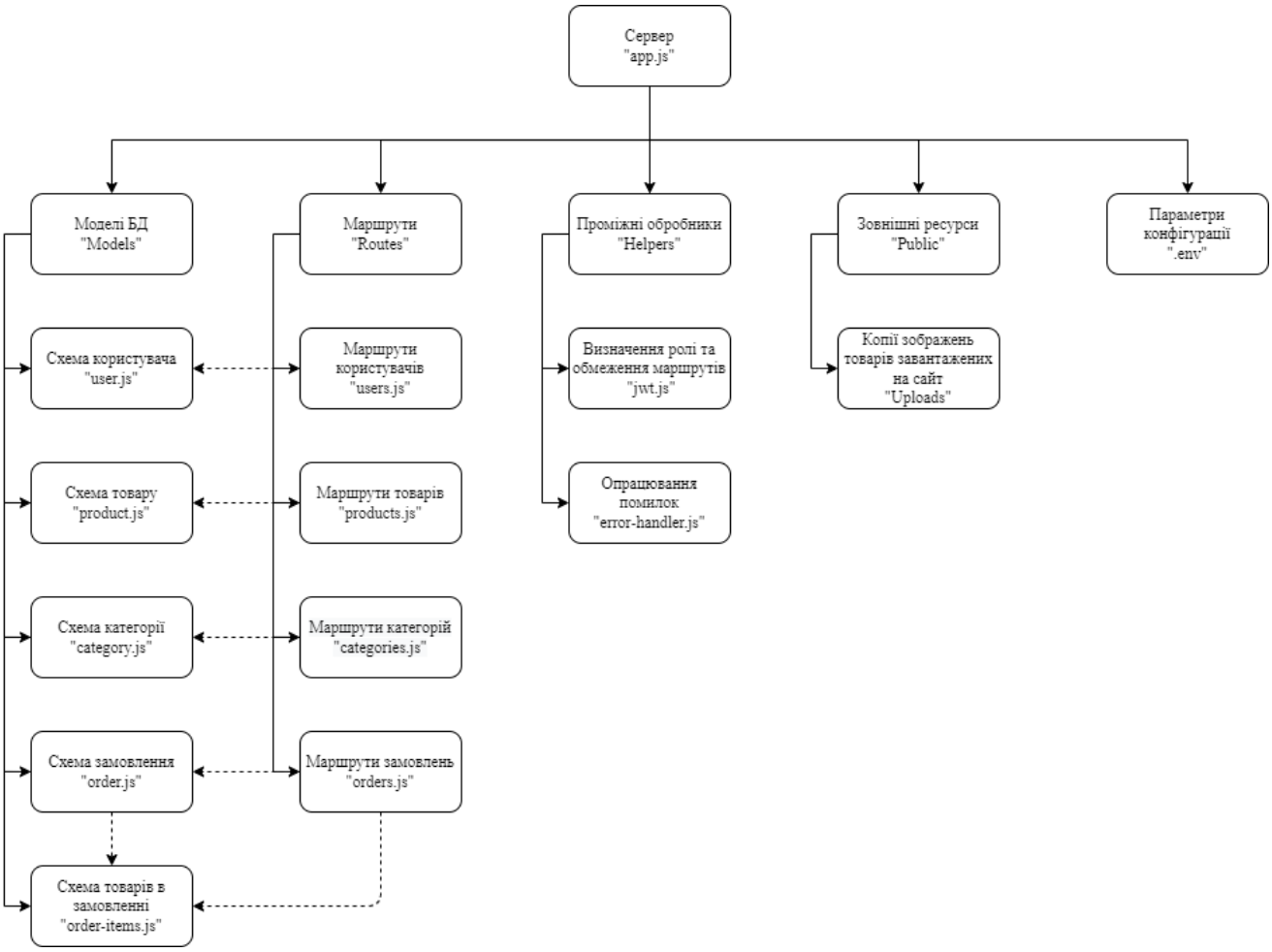

Рис. 2.2. Схема файлової структури програми

В теці «models» зберігаються окремі схеми для кожної з моделей БД. Все в Mongoose починається зі схеми. Кожна схема відображається в колекції MongoDB і визначає форму документів у цій колекції. Вона визначає метадані моделі, такі як типи даних, властивості, значення за замовчуванням тощо. Далі, ці схеми використовуються для роботи відповідних маршрутів, що мають змогу при запиті до БД знаходити, задавати, редагувати або ж видаляти дані.

Всі маршрути знаходяться в окремій теці «routes». Вони визначають, як програма відповість на клієнтський запит до конкретної адреси. Там же, в кожному окремому маршруті, прописано логіку відповіді на конкретний запит. Шляхи маршрутів, у поєднанні з методом запиту, визначають конкретні адреси (кінцеві точки), де можуть бути створені запити. Шляхи маршрутів можуть бути рядками, шаблонами рядків або навіть регулярними виразами. Бібліотека Express підтримує наступні методи маршрутизації, що відповідають методам HTTP: get, post, put, head, delete, options, trace, copy, lock, mkcol, move, purge, propfind, proppatch, unlock, report, mkactivity, checkout, merge, m -search, notify, subscribe, unsubscribe, patch, search і connect, але основними та найчастіше використовуваними є перші шість.

Для проміжних обробників було виділено окрему теку «helpers» для зручності. Там знаходиться обробник «jwt.js», що займається верифікацією рівня доступу користувача на основі веб-токену, в якому зберігається інформація про те, чи є даний користувач адміністратором. Одночасно з цим, саме в цьому файлі визначається перелік адрес та їх запитів, до яких звичайний користувач без прав адміністратора не може мати доступ.

В цій же теці знаходиться обробник помилок, що відповідно до стандартного коду помилки надсилає спеціалізоване повідомлення. Це дозволяє конкретизувати та зробити повідомлення про помилку більш зрозумілим і змістовним.

Звісно при завантаженні зображення продукту на сайт, це зображення має десь зберігатись, тому в теці для зовнішніх ресурсів було створено теку «uploads»

саме для цих цілей. Під час завантаження, зображення копіюється в цю теку, але вже з новою назвою, і саме посилання на цей новий файл використовується далі.

Задля підвищення зручності роботи і конфігурації параметрів застосунку, або ж просто, змінних оточення, ці параметри було винесено в окремий файл «.env». Там зберігаються такі змінні як, ключ з'єднання з БД, номер порту, який прослуховує сервер, секретний ключ шифрування веб-токенів.

При запуску головного файлу, сервер починає прослуховувати, вказаний порт та повідомляє про успішний запуск, повідомленням в терміналі (рис. 2.3). Після цього відбувається з'єднання з базою даних. Як видно на рис. 2.2, якщо з'єднання відбулося без помилок, буде виведено повідомлення з відповідним текстом, а також назва бази даних, яка використовується. Паралельно з цим, відбувається запуск проміжних обробників та ініціалізація моделей маршрутів з папки «routes». На цьому етапі сервер повністю розгорнуто і він готовий приймати запити.

Рис. 2.3. Повідомлення про успішний запуск застосунку

Так як всі маршрути, пов'язані з обробкою даних, їх запитом або ж навпаки наданням користувачеві, вони постійно звертаються та використовують схеми відповідних документів. Схеми дозволять легше працювати з базою даних та визначати загальну структуру документів. Наприклад, в схемі для категорій визначається, що категорія повинна мати назву, іконку та колір. Таким чином, в базі даних, документ категорії буде мати саме таку структуру, як ми задали в схемі (рис.2.4).

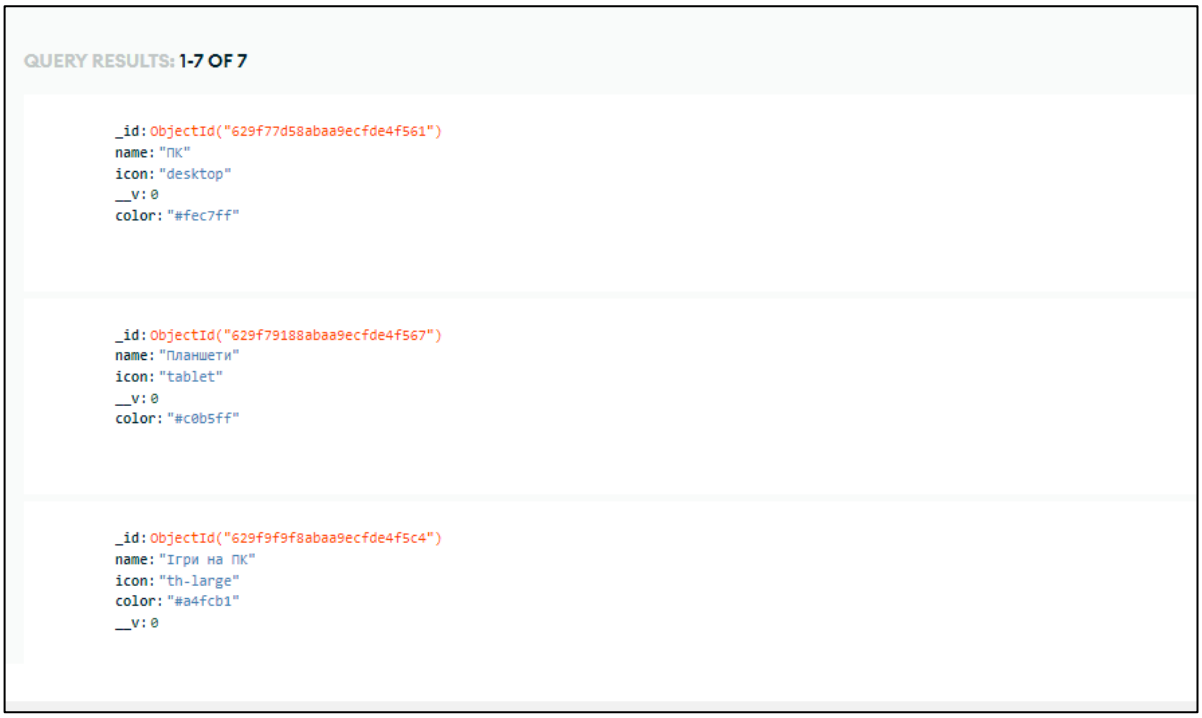

Рис. 2.4. Структура документів категорій в базі даних

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Всі дані, які застосунок приймає, обробляє та надає, зберігаються в форматі JSON. Звісно що користувач не буде робити запити напряму, використовуючи спеціальне програмне забезпечення, тому всі дані, які запитує сервер, він спочатку чекає від фронтенд частини. Та в свою чергу займається тим що, отримані від користувача дані, передає до серверу в потрібному форматі. І навпаки, для надання яких небудь даних користувачу, сервер надсилає їх до фронтенду, а той вже виводить їх на сторінку в належному вигляді. Без обробки даних від фронтенду, вони будуть виводитися у форматі JSON за замовчуванням.

2.7. Опис розробленого програмного продукту

2.7.1. Використані технічні засоби

Розробка та подальше тестування програмного застосунку серверної частини інтернет-магазину відбувалася на ПК з наступними характеристиками:

- процесор: AMD Ryzen 5 3600 6-Core 3.59 GHz;
- оперативна пам'ять: 24,0 ГБ;
- відеоадаптер: NVIDIA GeForce GTX 1650 Super;

2.7.2. Використані програмні засоби

В якості операційної системи було використано 64-розрядну операційну систему Windows 11 Pro. Сам програмний застосунок було розроблено з використанням редактору вихідного коду Visual Studio Code, 1.68.0 версії.

Для роботи з базою даних було використано MongoDB Compass, 1.32.2 версії. MongoDB Compass — це потужний графічний інтерфейс для запитів, агрегації та аналізу даних MongoDB у візуальному середовищі. Compass є безкоштовним у використанні та доступним, він може бути встановлений на операційні системи macOS, Windows і Linux.

Тестування роботи API відбувалося через HTTP-клієнт Postman та спеціальний застосунок для ПК Postman Desktop App, 9.21.2 версії. За допомогою Postman можна надсилати HTTP запити до сервісів та отримувати від них відповіді, тим самим перевіряючи коректність їх роботи.

Як платформу для розміщення коду та контролю версій і спільної роботи, було обрано GitHub, що дозволяє працювати декільком людям над проектами з будь-якого місця. Повнофункціональний графічний інтерфейс Sourcetree, який надає доступ до репозиторіїв Git та Mercurial на базі Windows та Mac та візуалізує процес розробки без доступу до командного рядка, значно полегшив процес розробки.

2.7.3. Виклик та завантаження програми

Для запуску серверної частини необхідно мати встановленою на комп’ютері платформу Node.js. Разом з нею, за замовчуванням буде встановлено NPM. Програма використовує декілька різних пакунків. Тому через менеджер пакунків

їх треба також встановити. Для цього достатньо буде відчинити папку з програмою через термінал, та виконати команду «npm i», яка в автоматичному режимі завантажить і встановить всі необхідні для роботи пакунки.

Наступним кроком потрібно зареєструвати акаунт MongoDB, та створити БД в якій і будуть зберігатися дані. Це можна зробити в ручному режимі, або ж автоматично. Для з'єднання з БД, програмі потрібен спеціальний ключ, в якому вказується ім'я акаунту власника та його пароль, а також назву кластеру. В самій ж програмі, для з'єднання, вказується й назва БД, що буде використовуватися. Якщо бази даних з такою назвою немає, то вона буде створена автоматично.

Далі необхідно налаштувати параметри конфігурації. Серед них головними якраз і є HTTP-порт, який буде прослуховувати ця програма, ключ підключення до бази даних, а також ключ шифрування веб-токенів. По факту, це просто змінні оточення, але в контексті застосунку, назвати їх параметрами конфігурації серверу, буде доцільніше. Вони зберігаються в окремому файлі з розширенням «.env», що робить їх використання і налаштування зручнішим і легшим (рис. 2.5). При використанні хмарних платформ, краще за все буде відмовитись від використання цього файлу. Замість цього визначення змінних оточення буде залежати від особливостей використовуваної платформи, так як більшість подібних хмарних платформ, підтримують оголошення змінних оточення за допомогою спеціальних вбудованих інструментів і налаштувань. Це буде зручніше та безпечніше.

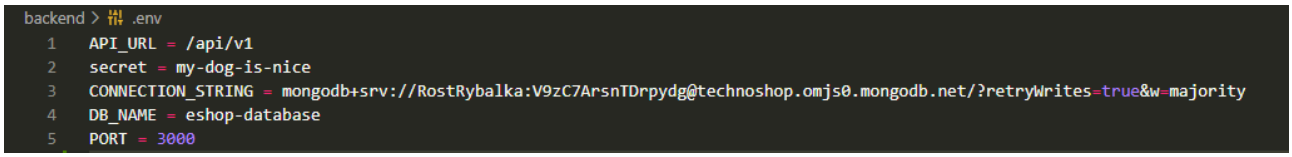

Рис. 2.5. Файл налаштування змінних оточення

Після виконання всіх налаштувань та етапів попередньої підготовки, сервер може бути повноцінно запущений. Для цього в тому ж терміналі можна виконати команду «npm run server». Ця команда запускає спеціальний скрипт під назвою

«server», який зберігає в собі команду «nodemon backend/app.js», яка в свою чергу замість звичайного варіанту запуску, використовує спеціальний пакунок «nodemon». Він дозволяє автоматично перезапускати застосунок у разі внесення змін до його коду. Якщо ж такої необхідності немає, програма може бути запущена за допомогою звичайної команди «node backend/app.js».

2.7.4. Опис інтерфейсу користувача

Так як розроблений додаток є сервером, що забезпечує роботу інтернет-магазину, фактично, графічний інтерфейс в ньому відсутній. Робота сервера полягає в опрацюванні HTTP запитів від користувачів, а за інтерфейс який буде це все відображати відповідає фронтенд. Тому для перевірки працездатності та коректності роботи, скористуємося HTTP-клієнтом Postman.

Наприклад для реєстрації нового користувача існує маршрут «…/api/v1/users/register» який має метод «POST», що надсилає дані до серверу. Перевіримо його роботу. Для цього в Postman вказуємо відповідну адресу запиту та метод, в тілі запиту, пишемо необхідні дані у форматі JSON (рис.2.6). Надсилаємо запит, і як видно на рис. 2.6, отримуємо відповідь, в якій наш пароль одразу зашифровано.

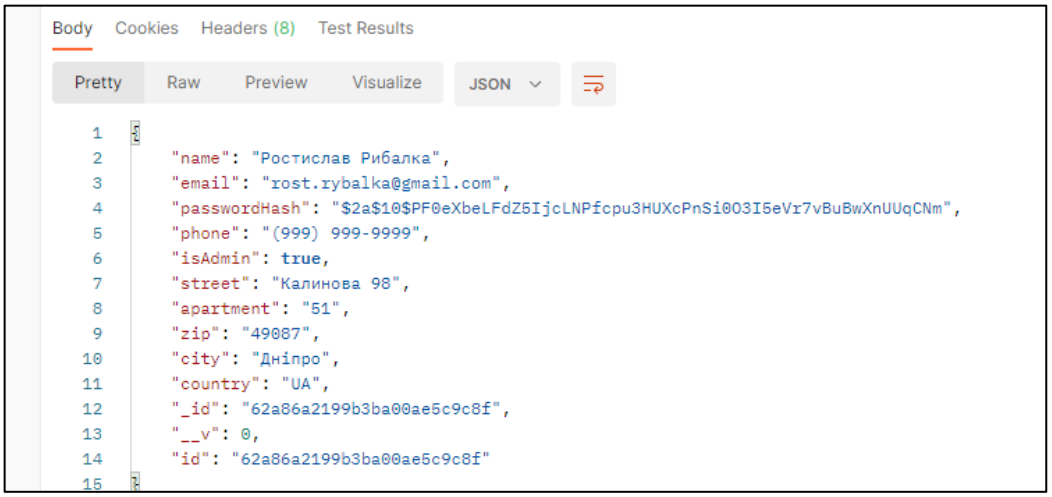

Рис. 2.6. Фрагмент вікна Postman з відповіддю сервера на запит реєстрації користувача

Задля того щоб перевірити, що все було правильно збережено до бази даних, перейдемо до застосунку MongoDB Compass, та переглянемо відповідну колекцію. Як можна побачити на рис. 2.7, нового користувача було успішно створено та додано до БД. Всі необхідні дані записано вірно. Пароль до бази даних надходить одразу зашифрованим.

Рис. 2.7. Дані нового користувача успішно додано до БД

Тепер перевіримо роботу маршруту «…/api/v1/users/login», який відповідає за авторизацію вже зареєстрованих користувачів. Звернемося до сервера знову, але вже з новою адресою запиту. Так як для авторизації, користувачеві потрібно ввести дані та надіслати їх до серверу, знову використовуємо метод «POST». Виконаємо запит, але на цей раз в тілі передаємо лише електронну адресу і пароль (рис. 2.8). В результаті чого, як можна бачити, сервер успішно авторизував користувача та згенерував його веб-токен.

Рис. 2.8. Фрагмент вікна Postman з відповіддю на запит авторизації

Так як створений нами раніше користувач мав рівень доступу адміністрації, перевіримо чи вірно працюють запити доступ до яких має тільки адміністрація. Для цього звернемося за маршрутом, що має підраховувати сумарний прибуток магазину, та надавати його адміністрації «…/api/v1/orders/get/totalsales». Виконаємо запит за відповідною адресою, але так як ми хочемо отримати інформацію від серверу, змінимо метод на «GET».

Я видно на рис. 2.9, сервер не виводить лише повідомлення про те, що ми не були розпізнані як адміністратор.

Рис. 2.9. Фрагмент вікна Postman з повідомленням про відсутність необхідного рівня доступу для запиту

Це відбувається через те, що верифікація рівня доступу відбувається завдяки вет-токену. Сама в ньому закодовано, чи є користувач адміністратором, а також його ім’я. Так як ми жодним чином не надали серверу веб-токен на перевірку, він і не зміг верифікувати наш доступ до запитів рівня адміністрації, і відмовив у виконанні запиту, надіславши відповідну помилку.

Якщо ж перейти в Postman до відповідної вкладки «Authorization» та обрати в ньому пункт «Bearer token» і вказати веб-токен, що ми отримали при авторизації користувача, то при повторному запиті на отримання підсумку прибутку з продаж ми отримаємо відповідь (рис. 2.10).

Рис. 2.10. Фрагмент вікна Postman з відповіддю на запит отримання підсумку прибутку з використаним веб-токеном адміністрації

Так як звичайні запити на отримання та надсилання інформації працюють за схожими принципами, перевіримо завантаження зображень товару. Для цього було створено маршрут «…/api/v1/products/gallery-images/:id». При створенні нового товару можна додати головне зображення, але якщо необхідно додати декілька, це можна зробити як раз завдяки окремому маршруту. Наприкінці адреси для запиту необхідно вказати ID товару та змінити метод на «PUT». При цьому, варто пам'ятати, що цей запит є доступним лише адміністрації, тому не забуваємо вказати веб-токен у відповідному полі, як це було зазначено раніше.

Перейдемо до вкладки «Body» та обираємо пункт «form-data». В полі «KEY» вказуємо «images» обираємо пункт «Files». Таким чином, у полі «VALUE» можна обрати файли до завантаження. Обираємо зображення на носієві, та робимо запит. Як можна бачити на рис. 2.11, запит було опрацьовано вірно і обрані нами зображення з новою назвою були успішно скопійовані до теки з зовнішніми ресурсами та посилання на них було збережено до масиву зображень відповідного товару.

Рис. 2.11. Фрагмент вікна Postman з відповіддю про успішне завантаження декількох зображень товару

У випадку виникнення помилок, за замовчуванням, буде виводитися тільки код помилки. Для того щоб конкретизувати, та зробити повідомлення зрозумілішими було створено проміжний обробник помилок «error-handler.js». В ньому, відповідно до ситуації в ході якої виникла помилка та її коду, записано спеціальні повідомлення. Наприклад, якщо сталася помилка верифікації веб-токена, буде виведено помилку «The user is not authorized», замість звичайного коду помилки «401», як це було продемонстровано на рис. 2.9.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 2100;
2. коефіцієнт складності програми – 1,6;
3. коефіцієнт корекції програми в ході її розробки – 0,06;
4. годинна заробітна плата програміста – 150 грн/год;

Середня годинна зарплатня Junior JavaScript Developer в Україні була вирахувати виходячи з даних «Української спільноти програмістів (DOU)». Станом на грудень 2021 року середня зарплатня Junior Front-End розробника 900$ у місяць. При курсі валют НБУ на початок червня 2022 року один американський долар дорівнює 29,25 грн, тому середня зарплата в гривнях дорівнює 26325 грн. При стандартному графіку (176 годин/місяць) зарплатня за годину буде становити близько 150 грн.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,4;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,4;
7. вартість машино-години ЕОМ – 19 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

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

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

$t_и$ – витрати праці на дослідження алгоритму рішення задачі;

$t_а$ – витрати праці на розробку блок-схеми алгоритму;

$t_п$ – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \qquad (3.2)$$

де q – передбачуване число операторів (2100);

C – коефіцієнт складності програми (1,4);

p – коефіцієнт кореляції програми в ході її розробки (0,06).

$$Q = 2100 \cdot 1{,}6 \cdot (1 + 0{,}06) = 3\,561{,}6;$$

Витрати праці на вивчення опису задачі $t_и$ визначаються з урахуванням уточнення опису і кваліфікації програміста:

$$t_и = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \text{ людино-годин} \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,4);

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1,4);

$$t_и = \frac{3\,561{,}6 \cdot 1{,}4}{85 \cdot 1{,}4} = 41{,}9, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot K}; \quad (3.4)$$

$$t_a = \frac{3\,561{,}6}{20 \cdot 1{,}4} = 127{,}7, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25) \cdot K}; \quad (3.5)$$

$$t_n = \frac{3\,561{,}6}{25 \cdot 1{,}4} = 101{,}8, \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4...5) \cdot K}; \quad (3.6)$$

$$t_{\text{отл}} = \frac{3\,561{,}6}{5 \cdot 1{,}4} = 508{,}8, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{\text{к}} = 1{,}4 \cdot t_{\text{отл}}; \quad (3.7)$$

$$t_{\text{отл}}^{\text{к}} = 1{,}5 \cdot 508{,}8 = 763{,}2, \text{ людино-годин,}$$

Витрати праці на підготовку документації:

$$t_{\partial} = t_{\partial p} + t_{\partial o}; \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису;

$$t_{\partial} = \frac{Q}{(15...20)\cdot K}; \qquad (3.9)$$

$$t_{\partial} = \frac{3\,561{,}6}{20 \cdot 1{,}4} = 127{,}2, \text{ людино-годин,}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{до} = 0{,}75 \cdot t_{др}; \qquad (3.10)$$

$$t_{до} = 0{,}75 \cdot 127{,}2 = 95{,}4, \text{ людино-годин.}$$

$$t_{\partial} = 127{,}2 + 95{,}4 = 222{,}6, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 41,9+ 127,7 + 101,8 + 508,8 + 222,6 = 1052,8, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1 052,8 людино-годин для розробки даного програмного забезпечення.

3.2. Рахунок витрат на створення програми

Витрати на створення ПЗ *Кпо* включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

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

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}\text{, грн,} \qquad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

З урахуванням того, що середня годинна зарплата програміста становить 150 грн/год, то отримаємо:

$$З_{ЗП} = 1052{,}8 \cdot 150 = 157920\text{, грн.}$$

Вартість машинного часу $З_{МВ,}$ необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_{М}\text{, грн,} \qquad (3.13)$$

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

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 508{,}8 \cdot 19 = 9\,667{,}2 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 157920 + 9\,667{,}2 = 167587{,}2 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}\text{, мес.} \qquad (3.14)$$

де B_k – число виконавців;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні Fр=176 годин).

Витрати на створення програмного продукту:

$$T = \frac{1052{,}8}{1 \cdot 176} = 5{,}9 \text{ міс.}$$

Висновки. Програмне забезпечення розроблено для забезпечення доступу користувачів до основних програм та пропозицій компанії продажу комп’ютерної техніки, ефективної взаємодії між потенційними споживачами та компанією, створення позитивного іміджу компанії, підвищення продажу товарів. Вартість даного програмного забезпечення близько 167587,2 грн і не вимагає додаткових витрат при розробці програми. Очікуваний час розробки становить 5,9 місяця. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка програмного забезпечення для організації роботи інтернет-магазину комп'ютерної техніки, а саме для ведення купівлі, продажу, обліку та контролю за замовленнями товарів.

Програма являє собою систему, що реалізує автоматизований збір, обробку і маніпулювання даними і включає технічні засоби обробки даних. База даних, яка забезпечує зберігання інформації і маніпулювання даними, та являє собою сукупність даних, що відповідають об’єктам структури системи. Програма призначена для забезпечення діяльності інтернет-магазину і надає можливість виконувати наступні дії:

- додавання, видалення і редагування інформації про товари, категорії, користувачів, замовлень;
- перегляд інформації про товари, категорії, користувачів, замовлень;
- здійснення операцій продажу;
- можливість входу в систему з різними рівнями доступу до даних: користувач та адміністратор;
- можливість зміни користувача у ході роботи програми;
- здійснення контролю введених даних: перевірка на відповідність типів, на введення даних коректного формату та відповідність до запитуваних полів;
- можливість перегляду інформації в режимі реального часу.

Актуальність поставленої задачі обумовлюється широким попитом на такі програмні продукти, що надають можливість електронного зберігання даних про товари, продажі, замовлення, ведення бази даних, підбиття підсумків з продажу та отримання статистичних даних про кількість товарів та користувачів.

Розроблене програмне забезпечення інформаційної системи призначене для застосування в будь-якій сфері онлайн продаж з подібними функціональними вимогами.

Створений додаток дозволить оптимізувати та спростити дії по веденню бази даних; скоротити час на оформлення продажу; підвищити ефективність

діяльності компанії шляхом електронного ведення статистики з продажу і можливістю аналізу наявних даних і надання необхідних звітів.

Структура програми являє собою клієнтський додаток, написаний на мові програмування JavaScript на базі платформи Node.js та з використанням фреймворку для веб-додатків Express, що взаємодіє з базою даних MongoDB за допомогою бібліотеки Mongoose. База даних та програмний застосунок працюють з використанням формату JSON, що полегшують роботу з даними.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення, що становить 1052,8 людино-годин, підраховані витрати на створення програмного забезпечення склали 167587,2 грн і розрахований період розробки дорівнює 5,9 місяця.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. А. Аллан. Клієнтська розробка для професіоналів. Node.js – СПБ.:2017. – 220с.
2. Хэррон, Д. Node.js Разработка серверных веб-приложений на JavaScript / Д. Хэррон. - М.: ДМК, 2018. - 144 с.
3. Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем. – М.: Интернет–Университет Информационных Технологий «Интуит», 2016. – 570 с.
4. Алекс Янг, Бредлі Мек, Майк Кантелон (2018) Node.js в дії. /Алекс Янг, Бредлі Мек, Майк Кантелон//Видавництво Питер.2-е видання ISBN 978-5-496-03212-4.
5. Ethan Brown/Итан Браун. /Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. /Питер:Издательский дом «Питер»/ 2016. –336 с.//ISBN 978-5-496-02156-2
6. Прамодкумар Дж. Садаладж, Мартин Фаулер. NoSQL: новая методология разработки нереляционных баз данных. Диалектика-Вильямс. 2017. -192 с. ISBN978-5-8459-1920-5
7. Каскиаро М., Маммино Л./Node.js. Design Patterns/Шаблоны проектирования Node.JS. /Москва:ДМК-Пресс/2017 –396 //ISBN: 978-5-97060-485-4
8. Дэвид Хэррон. Node.js Разработка серверных веб-приложений на JavaScript. /ДМК-Пресс/2014//ISBN: 978-5-94074-976-9, 978-1-849515-14-6
9. Пауэрс Ш. Изучаем Node. Переходим на сторону сервера. — 2-е изд... — СПб.: «Питер», 2017. — С. 304. — ISBN 978-5-496-02941-4.
10. Node JS. [Електронний ресурс] - Режим доступу: https://metanit.com/web/nodejs/
11. Брэд Дейли, Брендан Дейли, Калеб Дейли. Разработка веб-приложений с помощью Node.js, MongoDB и Angular: исчерпывающее руководство по использованию стека MEAN = Web Development with Node and

Express. — 2-е изд.. — Санкт-Петербург: «Диалектика-Вильямс», 2020. — 656 с. — ISBN 978-5-6040044-8-7.

12. Холмс С. Стек MEAN. Mongo, Express, Angular, Node. — СПб.: «Питер», 2017. — С. 496. — ISBN 978-5-496-02459-4.

13. What Is The MEAN Stack? Introduction & Examples. [Електронний ресурс] - Режим доступу: https://www.mongodb.com/mean-stack

14. Colin J. Ihrig, Adam Bretz. Full Stack JavaScript Development With MEAN. — SitePoint, 2015. — ISBN 9780992461256.

15. Архитектура JS Back-end: подводные камни, принципы работы, лайфхаки. [Електронний ресурс] - Режим доступу: https://dou.ua/forums/topic/33590/

16. Connecting Angular application with Nodejs backend. [Електронний ресурс] - Режим доступу: https://dev.to/rajesh04159786/connecting-angular-application-with-nodejs-backend-1l81

17. Ланг К., Чоу Дж. Публикация баз данных в Интернете - СПб.: Символ-Плюс, 2019р. - 206 с

18. Эрик Редмонд, Джим. Р. Уилсон. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL = MongoDB in Action. — ДМК Пресс, 2013. — 384 с. — ISBN 978-5-94074-866-3.

19. Steve Hoberman. Data Modeling for MongoDB. — Technics Publications, 2014. — 226 с. — ISBN 978-1-935504-70-2.

20. Использование переменных окружения в Node.js. [Електронний ресурс] режим доступу: https://habr.com/ru/company/ruvds/blog/351254/

21. Работа с JSON – MDN Web Docs. [Електронний ресурс] режим доступу: https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON

22. Офіційний опис та документація Node.js. [Електронний ресурс] режим доступу: https://nodejs.org/uk/docs/

23. Офіційний опис та документація Express.js. [Електронний ресурс] режим доступу: https://expressjs.com/ru/

24. Офіційний опис та документація Mongoose. [Електронний ресурс] режим доступу: https://mongoosejs.com/docs/api.html

25. April 2022 Web Server Survey [Електронний ресурс] режим доступу: https://news.netcraft.com/archives/2022/04/27/april-2022-web-server-survey.html

26. Середня заробітна плата програміста у Дніпрі станом на зиму 2022 року. [Електронний ресурс] – Режим доступу: https://dou.ua/lenta/articles/salary-report-devs-winter-2022/

27. Методичні рекомендації до виконання кваліфікаційних робіт здобувачів першого рівня вищої освіти спеціальності 121 Інженерія програмного забезпечення / О.С. Шевцова, І.М. Удовик; Д: НТУ «Дніпровська політехніка», 2021. – 65 с.

28. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи ” / Уклад. О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 11 с.