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 *Левченко Дмитра Максимовича*
(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 інтернет-магазину комп'ютерної техніки на базі технологій TypeScript та Angular*

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* *Левченко Д.М.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інтернет-магазину комп'ютерної техніки на базі технологій TypeScript та Angular*

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 _____ *Левченко Д.М.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

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

РЕФЕРАТ

Пояснювальна записка: 76 с., 33 рис., 1 табл., 3 дод., 21 джерел.

Об'єкт розробки: інтернет-магазин комп’ютерної техніки на базі технологій TypeScript та Angular.

Мета кваліфікаційної роботи: створення програмного забезпечення для підтримки операцій інтернет складу-магазину для підвищення продуктивності його працівників і для скорочення затрат часу на оформлення товарів, ведення обліку товарів, оформлення продажу, планування та підбірки комплектації комп’ютерів, здійснення аналізу його діяльності, реалізованих за допомогою автоматизації 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, отримання звітів і формування супутньої ділової документації.

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

Список ключових слів: ПРОГРАМА, СКЛАД, КОМП’ЮТЕР, ІНФОРМАЦІЙНА СИСТЕМА, ОБЛІК, АЛГОРИТМ, ПРОЕКТУВАННЯ, МЕНЮ, ВКЛАДКА, ЗАСТОСУНОК, МАГАЗИН, ІНТЕРНЕТ.

ABSTRACT

explanatory note: 76 pages, 33 pictures, 1 table, 3 additions, 21 sources.

Object of development: online store of computer equipment based on TypeScript and Angular technologies.

The purpose of the qualification work: to create software to support the operations of the online store to increase the productivity of its employees and to reduce the time spent on registration of goods, accounting, sales, planning and selection of computers, analysis of its activities by automating the process of maintaining an electronic database of the system, executing requests and receiving reports to analyze the activities of the enterprise.

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.

The first section analyzes the subject area, determines the relevance of the task and the purpose of development, formulates the task, specifies the requirements for software implementation, technology and software.

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 technical means.

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, receiving reports and the formation of related business documents.

The relevance of this software product is determined by the high demand for such developments, which optimize and simplify the actions of maintaining a database of 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 and provide the necessary reports and related business documentation.

List of keywords: PROGRAM, WAREHOUSE, COMPUTER, INFORMATION SYSTEM, ACCOUNTING, ALGORITHM, DESIGN, MENU, INSERT, APP, SHOP, INTERNET.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML – HyperText Markup Language;

CSS – Cascading Style Sheets;

HTTP – Hypertext Transfer Protocol;

WAP – Wireless Application Protocol;

HTTPS – Hypertext Transfer Protocol Secure;

СУБД - набір взаємопов'язаних даних і програм для доступу до цих даних.

API - Application Programming Interface;

CRUD операції - 4 основні функції управління даними «створення, читання, оновлення і вилучення»;

ID – ідентифікаційний номер;

JSON – JavaScript Object Notation;

BSON - двійкова форма представлення простих структур даних і асоціативних масивів;

ГГц - одиниця вимірювання в системі SI частоти періодичних процесів, дорівнює 10^9 Гц;

Gb - кратна одиниця вимірювання кількості інформації, що дорівнює 1024 мегабайтам;

Мб/сек – швидкість передачі даних, що дорівнює 1 мегабайту у секунду.

NPM - Node Package Manager;

JavaScript – динамічна, об’єктно-орієнтована прототипна мова програмування;

Node.js - платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript;

TS – TypeScript;

SCSS – Sassy Cascading Style Sheets;

VSCode – багатофункціональний текстовий редактор коду з можливістю встановлення додатків різного типу;

Front-end розробник – розробник сайту зі сторони інтерфейсу;

Back-end розробник – розробник сайту зі сторони операцій з даними;

DOM – Document Object Model.

ВСТУП

Тематика даної кваліфікаційної роботи присвячена розробці програмного забезпечення для інтернет-магазину комп’ютерної техніки на базі технологій Angular та TypeScript застосовуючи середовище редагування коду VSCode.

Метою даної кваліфікаційної роботи є створення програмного застосунку для реалізації самого інтернет-магазину та панелі адміністраторів. У процесі роботи буде вивчено інструментальні засоби та принципи роботи фреймворку Angular, функціонал якого забезпечує можливість повноцінно створити інформаційну інтерактивну систему інтернет-магазину з використанням мов HTML, SCSS та TypeScript.

Значна частина задач пов’язана з налагодженням роботи компонентів програми, їх взаємодією одне-з-одним та з базою даних за допомогою серверної частини клієнту.

Рано чи пізно власникам бізнесу доведеться подбати про створення інтернет-магазину. Можна знайти величезну кількість інформації про те, як вирішити цю проблему від власного торгівельного майданчика на готовому сервісі до повноцінного інтернет-магазину. Перший варіант все ще має продовжуватись у пошуках справжнього інтернет-магазину. Проблема в тому, що можливості редагування вигляду і функціональності торгового майданчика дуже обмежені, а ваші бізнес-партнери можуть сприйняти такий «інтернет-магазин» не дуже серйозно. Інтернет-магазин є однією з форм електронної комерції, зазвичай типу B2C – бізнес-до-споживача. B2C включає збирання інформації клієнтами; купівлю фізичних речей чи інформаційних/електронних товарів; і, для інформаційних товарів, одержування товару по електронній мережі. Приклади B2C моделей — мережні компанії продажу в роздріб типу Amazon.com, Drugstore.com, Beyond.com. B2C електронна комерція зменшує ціну угод, збільшуючи доступ споживачів до інформації й дозволяючи споживачам знайти найбільш конкурентоспроможну ціну за товар або послугу. Також вона зменшує ринкові бар'єри входу, тому що вартість створення й розкручування сайту набагато менша ніж установка структури фірми.

У випадку інформаційних товарів, B2C електронна комерція є ще більш привабливішою, тому що це зберігає фірми від фактору додаткової вартості фізичної мережі розподілу. Крім того, для країн із зростаючою кількістю користувачів Internet, поставка інформаційних товарів стає усе більш і більш доступною.

Завдання кваліфікаційної роботи полягає в створенні додатку програми інтернет-майданчика для продажу товарів, ведення звітності, організації списку товарів та складу. Також для оперативного приймання та обробки замовлень на купівлю товарів від користувача. До додаткових завдань відноситься встановлення зв’язку з базою даних про товари та можливість призначення користувача з правами адміністратора задля організації робочого процесу та контролю над ним. Також присутня потреба у створенні дизайну макету майбутнього інтернет-магазину.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

В наш час проблема створення інтернет-версії магазину для бізнесів різних категорій та розмірів є дуже гострою. Причиною цьому служить великий розвиток технологій, комп'ютеризація населення та інтеграція у світ смарт-техніки. Не можна також забувати й про світові події останніх років, які змінили пріоритети у веденні бізнесу сфери послуг з офлайн-взаємодії на онлайн-взаємодію як бізнесу з бізнесом, так і бізнесу з клієнтом. Виходячи з цього людству необхідно було підлаштовуватись під новий стан речей та все більше переводити комерцію в інтернет.

Зараз онлайн-простір налічує незчисленну кількість WEB-сторінок, які являють собою цифрові майданчики для купівлі-продажу різних товарів, оформлення послуг з доставки або надають свої сервіси для проходження транзакцій. Також дуже популярною зараз є тенденція для малого бізнесу не робити все з нуля, тобто не створювати свій власний інтернет-майданчик для ведення справ, а використовувати так звані «майданчики-посередники» які являють собою багатофункціональну платформу для розміщення товарних оголошень з простою формою реєстрації клієнтів. Також такі майданчики пропонують зручний спосіб ведення звітності для підприємця-клієнта і відображення у інтерфейсі власного кабінету на акаунті як для сторони, що продає деякий товар, так і для сторони, що купляє цей товар. Гарним прикладом для цього можуть послужити такі майданчики нашої країни, як Rozetka, або ж Prom.ua.

Звісно такий спосіб виведення свого бізнесу в інтернет має ряд переваг для підприємців невеликого масштабу продажів, серед них:

- Швидкість та зручність реєстрації акаунту на майданчику;
- Зрозумілий інтерфейс;
- Зручна форма створення оголошень та їх відображення;
- Офіційна діяльність;
- Можливість контактування з покупцем за допомогою залишання власних контактів у оголошенні або ж форми зворотного зв'язку.

Як бачите, перелік переваг дуже великий для малого бізнесу, але що робити підприємцям по-крупніше – відповідь досить проста: створювати власний індивідуальний майданчик для своєї підприємницької діяльності і її потреб.

Саме у цей час з'являється потреба у розробниках таких майданчиків, а саме у front-end та back-end розробниках програмного забезпечення. Як вже було зазначено у переліку умовних позначень – це розробники сайту зі сторін сервера та інтерфейсу. У рамках цієї дослідницької роботи була проведена робота над front-end складовою інтернет-майданчика.

Якщо розглянути концепцію розробки, то ми побачимо, що для цього процесу нам знадобляться три мови програмування – HTML, CSS та JavaScript. Ці мови складають так звану «базу» програмування WEB-сторінок різних типів складності та різного призначення. Але не справа ще полягає в тому, що у нас час неможливо зробити конкурентоспроможний застосунок використовуючи лише ці три базові елементи. Для того, щоб сайт працював, нам потрібно щось більш точне, що більш детально давало би нам змогу втілення поставленої задачі, а також щось таке, що зробило би більш зручнішим та швидшим процес збірки окремих файлів та структур у велику цілісну систему, яка могла б автономно та незалежно функціонувати без постійної допомоги розробника. Саме для таких завдань існують такі речі, як Фреймворки, або ж англійською – Frameworks.

Framework являє собою каркас, платформу для створення веб-продуктів нового покоління, їх ефективної підтримки. Він призначений для складних, масштабних проектів, дозволяє реалізувати нестандартні рішення.

Він не формує для продукту, що розробляється жорсткі рамки. Він надає базові модулі, на основі яких створюється гнучкий сайт з широкими можливостями модернізації, розширення функціоналу за рахунок приєднання додаткових додатків в майбутньому.

Стандартна архітектура фреймворка ґрунтується на поділі трьох шарів:

1. Модель - відповідає за формування структури, правил бізнес-логіки;
2. Уявлення - його основна функція – графічне відображення даних;
3. Контролер - реалізує зв’язок з користувачем, перетворюючи отриману від нього інформацію в команди для попередніх двох шарів.

Хочеться також зазначити, що крім цього інструменту, розробка веб-сайтів може виконуватися IT-фахівцем самостійно з написанням коду з нуля або з залученням готової CMS. Перший варіант занадто витратний за часом, вимагає маси зусиль в ході реалізації, подальшого тестування. Використання CMS дозволяє швидко створювати інтернет додатки, проте накладає на їх функціонування певні обмеження, тобто вийти за рамки «конструктора» не вийде. Такі способи розробки більше підходять для простих, статичних проектів.

Між ними фреймворк займає проміжну позицію. Робота з ним складніше, ніж з готовим движком, але результат буде отримано в рази швидше, ніж при написанні коду. Розробка сайту з його використанням отримала величезну популярність в останні роки, так як фреймворк має декілька значних переваг перед конструкторами:

- висока швидкість створення додатків, сервісів;
- можливість постійного розвитку продукту за рахунок інтеграції в структуру нових розширень;
- простий зручний супровід;
- максимальна безпека зберігання даних;
- застосування фреймворка для сайту забезпечує йому швидкість реагування, можливість витримувати значні навантаження.

Як і процес розробки, фреймворки поділяють на Back-end, Front-end та Full-stack. У рамках цієї дослідницької роботи буде розглянуто Front-end фреймворки та розробку на них.

Одразу треба зазначити, що Front-end фреймворки не пов’язані з логікою роботи програми, так як функціонують безпосередньо в браузері. Призначені для створення користувацьких інтерфейсів з різноманітною графікою, анімацією. Приклади: React, Vue, Angular.

Далі буде більш детально проведено аналіз аналогів у виборі Front-end фреймворків.

1) React

React – це відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC.

Серед особливостей цього фреймворку можна зазначити:

- Односторонню передачу даних;
- Віртуальний DOM;
- Компоненти React зазвичай написані на JSX;
- React надає змогу не лише для рендерингу HTML в браузері;
- Використовує методи життєвого циклу.

2) Vue

Vue.js – це JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться.

У цього фреймворку ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об'єкти. Це робить керування станами дуже простим та інтуїтивним. Vue надає оптимізований ре-рендеринг з коробки без потреби робити що-небудь додатково. Кожен компонент слідкує за своїми реактивними залежностями під час рендерингу, тому система знає точно коли має відбуватись ре-рендеринг і які компоненти потрібно ре-рендерити.

Серед особливостей цього фреймворку можна зазначити:

- Шаблони;
- Реактивність;
- Переходи. Vue надає різноманітні шляхи для застосування ефектів переходу, коли елемент додають, оновлюють або видаляють з DOM;
- Роутинг.

3) Angular

Angular – це написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який був переосмислений та перероблений тією ж командою розробників.

Також варто відзначити, що Angular пропонує рендеринг на стороні сервера, який прискорює завантаження початкової сторінки і, отже, покращує SEO, спрощуючи сканування динамічних сторінок. Швидке відображення сторінок значно покращує сприйняття веб-застосунків для наступного покоління, написаних в рамках Angular. Також, за допомогою даного фреймворку можна легко тестувати код, легко створювати персоналізовані об’єктні моделі документа (DOM) та моделювати дані обмежено для використання невеликих моделей даних.

Серед особливостей цього фреймворку можна зазначити:

- Angular підтримується Google;
- Angular надає клієнтську MVC-інфраструктуру, яка допомагає у запуску та створенні динамічних додатків із сучасним рівнем якості;
- Програми, написані на Angular, сумісні з різними браузерами. Angular автоматично обробляє код JavaScript, що підходить для кожного браузера;
- Чистий і точний дизайн інтерфейсу користувача;
- Зручна та проста маршрутизація;
- Структура Angular полегшує розширення синтаксису HTML і легко створює повторні компоненти за директивами.

В якості фреймворку було обрано саме Angular. Також великою перевагою є використання мови TypeScript.

TypeScript – мова програмування, яка позиціонується як засіб розробки вебзастосунків, що розширює можливості JavaScript.

Переваги TypeScript над JavaScript:

- можливість явного визначення типів та статична типізація;
- підтримка використання повноцінних класів як в традиційних об'єктно-орієнтованих мовах;
- підтримка підключення модулів.

ці нововведення мають підвищити швидкість розробки, прочитність, рефакторинг і повторне використання коду, здійснювати пошук помилок на етапі розробки та компіляції, а також швидкодію програм.

Для задання стилів будет використана мова SCSS.

SCSS – це скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS). Спроектована Гемптоном Кетліном та розроблена Наталі Вейзенбаум. Scss призначений для підвищення рівня абстракції коду та спрощення файлів CSS.

1.2 Призначення розробки та галузь застосування

Застосунок розробляється з метою поліпшення підприємницької діяльності з продажу товарів та оптимізації процесів звітності та менеджменту замовлень. Застосунок є універсальною програмою, яка підійде для магазинів різних типів, різних тематик та списків товарів для продажу клієнтам.

Цього вдалося досягнути за допомогою гнучкої компонентної системи в якій усі сторінки сайту та моделі його роботи існують незалежно одна від одної, але просто звертаються за даними одна до іншої коли на це виникає певна потреба.

Універсальність товарної бази досягнута за допомогою існування back-end серверу, де зберігаються усі дані. А також із-за обширного використання методів строкової інтерполяції змінних, що означає – що не важливо що буде товаром на продаж, доки воно відповідає спеціальній формі заповнення інформації щодо об’єкту як товарної одиниці – воно буде зрозуміло та чітко відображатись у переліку товарів або категоріях магазину.

Також у програмі були застосовані сучасні методи розробки UI/UX дизайну сторінок, що робить інтерфейс більш інтуїтивно-зрозумілим як для звичайного користувача, так і для адміністраторів магазину і також зводить присутність візуального шуму близько до нуля, але при цьому зберігаючи цікаві та гарні

елементи, які в свою чергу впливають на користування програмою як на дуже позитивний досвід.

1.3 Підстави для розробки

Розробка ведеться на підставі наказу на дипломування 268-с від 18травня2022р.

1.4 Постановка завдання

Завдання дослідницької роботи полягає в розробці front-end частини WEB-додатку, яка виконує функції сучасного інтернет-магазину для продажу товарів будь-якої категорії з можливістю групування їх у замовлення для подальшого виконання та моніторингу їх стану. Також повинна бути присутня система реєстрації користувача як адміністратора задля швидкої, зручної та наочної роботи з товарами, замовленнями, категоріями та іншими користувачами.

Для цієї задачі був обраний фреймворк Angular, що працює на мовах TypeScript та HTML тому що, як зазначалось у попередніх розділах, в нього присутні всі необхідні функції для виконання поставленої задачі а також його компонентний підхід у розробці додатку є більш зручним для виконання.

Додатком буде користуватись людина, що має права адміністратора для затвердження замовлень, групування та редагування, створення/видалення списку товарів, моніторинг замовлень та також редагування та створення/видалення категорій. До функцій адміністратора належить контроль за списком клієнтів та редагування внутрішньої інформації про будь-який товар/категорію та змога зміну статусу замовлення у базі. Також додатком з іншого боку буде користуватись звичайний клієнт, який буде мати змогу переглядати список усіх товарів та інформацію про них і буде мати можливість сортування списку по категоріях. Звичайний користувач буде мати змогу користуватись віртуальним кошиком задля додавання товарів до свого

замовлення, яке потім буде бачити адміністратор у реальному часі за допомогою працюючої back-end частини.

До головної структури магазину належать сервер та клієнтський застосунок, який відіграє роль WEB-застосунку з власним інтерфейсом.

Вихідною інформацією front-end частини буде повністю робочий застосунок з переліком товарів, категорій, користувачів та замовлень, який буде мати дві окремі частини – для адміністратора та для звичайного користувача. Вихідною інформацією адміністратор буде адміністративна панель, яка буде мати в собі відображення списку товарів та категорій, користувачів з можливістю додавання, редагування та видалення. Вихідною інформацією користувача буде застосунок онлайн-магазину зі змогою придбання товарів і оформленням замовлень через кошик.

Вхідною інформацією для адміністратора буде масив даних, який буде поступати з back-end серверу, який пов'язаний з базою даних, а також масиви об'єктів типу продукт, категорія, користувач та замовлення. Замовлення адміністратор буде отримувати від користувача через кошик. Вхідною інформацією для користувача будуть сформовані адміністратором списки товарів та категорій, які він зможе угруповувати у замовлення.

Розв’язання завдання автоматизованим способом припиняється за умови відсутності стабільного інтернет-підключення та персонального комп’ютера чи ноутбука з необхідними для роботи програмами та застосунками.

1.5 Вимоги до програми або програмного виробу

1.5.1 Вимоги до функціональних характеристик

Вхідні дані для адміністратора додатку повинні вводитися за допомогою back-end частини у форматі JSON. Вихідними даними адміністратора є заповнення компонентів типу «форма» задля створення, редагування та зберігання інформації про товари, категорії, користувачів та замовлення.

Здебільшого адміністратор буде заповнювати поля строковими символами або ж цифрами, картинками формату jpg, jpeg, png. Вихідними даними будуть вже готові об’єкти різних класів: товар, категорія, користувач чи замовлення, моделі яких описані як статичні інтерфейси з полями з даними та які заповнюються інформацією з back-end частини за допомого цих інтерфейсів.

Вхідні дані для користувача будуть об’єктами різних класів: товар, категорія. Ці об’єкти будуть включати в себе поля різних типів даних та відображатися за допомогою графічного інтерфейсу застосунку. Вихідними даними користувача буде інформація про створене замовлення за допомогою компоненту кошика, яке буде надсилатись адміністратору через базу даних, яка зв’язана з back-end сервером та буде являти собою об’єкт з полями даних різних типів, який теж описаний у окремому інтерфейсі мовою TypeScript.

1.5.2 Вимоги до інформаційної безпеки

Так як ми маємо справу з інтернет-застосунком, то першим чином що треба зробити – це перевірити стабільність інтернет-з’єднання та хосту, а також його захищеність. Мережа, до якої підключені користувач та адміністратор повинні бути захищені протоколами шифрування та мати стабільний та гарний зв’язок без великої кількості посередників.

Незалежно від того, де розташований проект додатку: локально чи на хмарному-сервері – завжди треба перевіряти стан антивірусу та регулярно його оновлювати а також слідкувати за станом з’єднання з мережею та захищеністю каналів шифрування.

Також треба слідкувати за актуальними версіями програмного забезпечення та різних модулів системи, тому що зловмисники можуть за допомогою так званих «дір» у безпеці старих версій програм отримувати доступ до конфіденційної та захищеної інформації. Оновлення програмної платформи сайту допомагає регулярно закривати різні діри у його безпеці.

Задля впровадженого додаткового рівня безпеки використовується шифрувальний метод, що має назву JSON Web Token, який створює так званий «токен» на сервері и шифрує його щоб потім присвоїти користувачеві для подальшого підтвердження його особи. В середині цього токену якраз і зберігається зашифрована інформація цього користувача, а саме – його ID та чи має він права адміністратора.

Коли користувач заходить до акаунту чи реєструється на сайті його пароль одразу ж шифрується за допомогою бібліотеки bcrypt.js, яка забезпечує безпечне шифрування паролю за допомогою метода BlowFish, який реалізує блочно-симетричне шифрування. Оригінальний пароль користувача в чистому виді ніде не зберігається.

1.5.3 Вимоги до складу та параметрів технічних засобів

Для стабільної роботи з додатком вам необхідні відповідні технічні, або необхідні характеристики обладнання для операційної системи Windows:

- Операційна система (ОС): Windows 10 і вище;
- Центральний процесор (ЦП): intel core i5;
- Відеоадаптер: Intel(R) Iris® Xe Graphics;
- Відеопам’ять: 4 Гб;
- Накопичувач: 520 ГБ;
- Оперативна пам’ять: 32 Гб.

У випадку використання комп’ютерів на системі macOS потрібні наступні технічні характеристики або подібні:

- ОС: macOS Monterey і вище;
- ЦП: Intel Core i7;
- Відеоадаптер: Radeon Pro 555X;
- Відеопам’ять: 4 ГБ;
- Накопичувач: 520 ГБ;
- Оперативна пам’ять: 16 Гб.

1.5.4 Вимоги до інформаційної та програмної сумісності

Код проекту сумісний з усіма існуючими JavaScript та TypeScript фреймворками та може бути легко імпортований з одного фреймворку в інший, але з невеличким редагуванням із-за різних системних функцій фреймворків.

Для успішного запуску проекту потрібен CLI вашого фреймворку, та остання версія пакункового менеджера NPM.

РОЗДІЛ 2

ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Функціональним призначенням даного додатку є функціонування front-end частини інтернет-магазину, а саме: графічне відображення даних, які зберігаються у базі даних та надсилаються програмі через back-end сервер. Дані повинні являти собою об’єкти з полями різних типів, які заздалегідь були зазначені у файлах моделей чи інтерфейсів. Серед даних можна виділити такі об’єкти: категорія товарів, товар, користувач, замовлення. Також інтерфейс повинен надавати змогу не тільки приймати дані з сервера, а і відсилати їх йому. Зокрема повинна бути присутня функція додавання нових товарів, категорій та користувачів зі сторони адміністратора, а також змога їх подальшого редагування через форми разом із замовленнями. Зі сторони покупця функціональне призначення полягає у змозі відображення додатком списку усіх товарів та категорій зі змогою їх сортування для зручного пошуку та наявність сторінки-кошика, куди можна додавати товари, які користувач хоче купити та потім оформлення усіх товарів в одне замовлення.

Експлуатаційне призначення програми для адміністратора магазину полягає в змозі редагування інформації про товари, категорії, користувачів та замовлення. А також у змозі їх додавання до бази даних, або видалення з неї за допомогою зручного графічного інтерфейсу. Зі сторони клієнта експлуатаційне призначення додатку полягає в змозі перегляду списку наявних товарів та категорій, а також подальшого додавання їх до сторінки кошика за допомогою графічного інтерфейсу, де далі можна сформувати власне замовлення, яке надійде до бази даних та відобразиться у адміністратора в його панелі.

2.2 Опис застосованих математичних методів

У виконанні даної дослідницької роботи не було застосовано ніяких математичних методів.

2.3 Опис використаної архітектури та шаблонів проектування

Архітектура програмного забезпечення, що використовується для розробки та функціонування програмного додатку має назву «клієнт-сервер».

В основі клієнт-серверної архітектури лежать два компоненти: клієнт і сервер.

Клієнт – комп'ютер на стороні користувача, який відправляє запит до сервера для надання інформації або виконання певних дій.

Сервер – більш потужний комп'ютер або обладнання, призначене для вирішення певних завдань з виконання програмних кодів, виконання сервісних функцій за запитом клієнтів, надання користувачам доступу до певних ресурсів, зберігання інформації і баз даних.

Модель такої системи полягає в тому, що клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові(Рис. 2.1). Сервер може обслуговувати кілька клієнтів одночасно. Якщо одночасно приходить більше одного запиту, то вони встановлюються в чергу і виконуються сервером послідовно. Іноді запити можуть мати пріоритети. Запити з більш високими пріоритетами повинні виконуватися раніше.

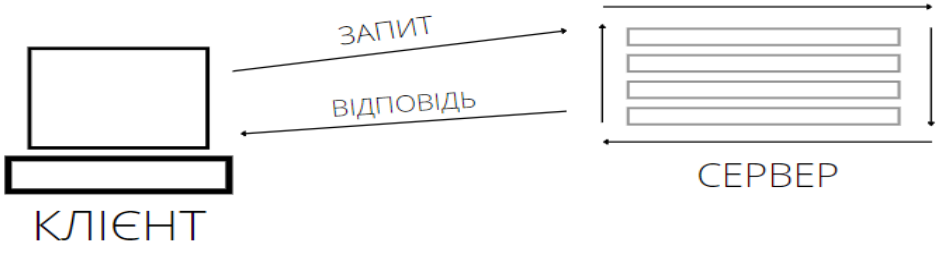

Рис. 2.1. Схема функціонування архітектури Клієнт-сервер

Функції, які реалізуються на сервері:

- зберігання, доступ, захист і резервне копіювання даних;

- обробка клієнтського запиту;

- відправлення результату (відповіді) клієнту.

Функції, які реалізуються на стороні клієнта:

- надання користувальницького інтерфейсу;

- формулювання запиту до сервера і його відправка;

- отримання результатів запиту і відправка додаткових команд (запитів на додавання, оновлення або видалення даних).

Архітектура клієнт-сервер визначає принципи спілкування між комп'ютерами, а правила і взаємодії визначені в протоколі.

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

В даній дослідницькій роботі використовується протокол HTTP – це протокол передачі гіпертексту, на основі якого працюють всі сайти. Він запитує необхідні дані у віддаленій системи (веб-сторінки, файли).

Наша концепція побудови системи клієнт сервер застосовує варіант з сильним клієнтом.

Сильний клієнт – концепція, в якій частина обробки інформації надається клієнтові. У такому випадку сервер виступає сховищем даних, а вся робота по обробці та подання інформації переноситься на комп'ютер клієнта.

Система (застосунок), яка заснована на клієнт-серверній взаємодії, включає три основних компоненти: уявлення даних, прикладний компонент, компонент управління ресурсами і їх зберігання.

Наша архітектура є дворівневою і складається з двох вузлів (Рис. 2.2):

а) сервер, який відповідає за отримання запитів і відправку відповідей клієнту, використовуючи при цьому лише власні ресурси;

б) клієнт, який представляє користувацький інтерфейс.

Принцип роботи полягає в тому, що сервер отримує запит, обробляє його і відповідає безпосередньо, без використання сторонніх ресурсів.

Взаємодія клієнт-сервер дозволяє розділяти функціонал і обчислювальне навантаження між клієнтськими додатками (замовниками послуг) і серверними додатками (постачальниками послуг). Знання архітектури додатка дозволяє тестувальнику більш якісно провести функціональне, крос-браузерне тестування, тестування юзабіліті і швидкодії.

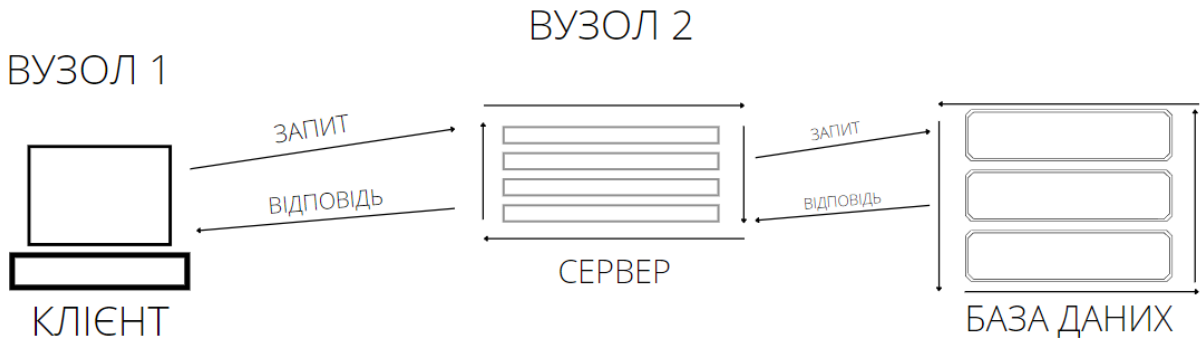

Рис. 2.2. Схема структури з двома вузлами

2.4 Опис використаних технологій та мов програмування

Головною технологією розробки front-end частини є фреймворк під назвою Angular, як вже було зазначено у пункті 1.1 – він має низку переваг серед інших фреймворків, здебільшого це:

- Програми, написані на Angular, сумісні з різними браузерами. Angular автоматично обробляє код JavaScript, що підходить для кожного браузера;

- Чистий і точний дизайн інтерфейсу користувача;

- Зручна та проста маршрутизація.

- Структура Angular полегшує розширення синтаксису HTML і легко створює повторні компоненти за директивами.

Також на вибір фреймворку вплинуло власне бажання студента.

Як можна побачити на рисунку 2.3 – Angular має досить зручну схему роботи з компонентами, розподіляючи функціонал програми на окремі модулі для більш захищеної та точної роботи функцій застосунку. Головними файлами є файл шаблону (Template) де міститься розмітка сторінки у форматі HTML, та файл компонента (Component) написаний мовою TypeScript та відповідаючий за логіку роботи програми, і який і зв'язує шаблон з усіма іншими допоміжними сервісами, такими як моделі компоненті (Component Module) і моделі сервісів (Service Module) – вони, як правило, містять у собі правила та методи зв'язку з back-end частиною застосунку та містять чіткі форми з полями, де описуються потрібні типи даних для представлення об'єктів. Також Angular може використовувати різні допоміжні сервіси та директиви (Directive).

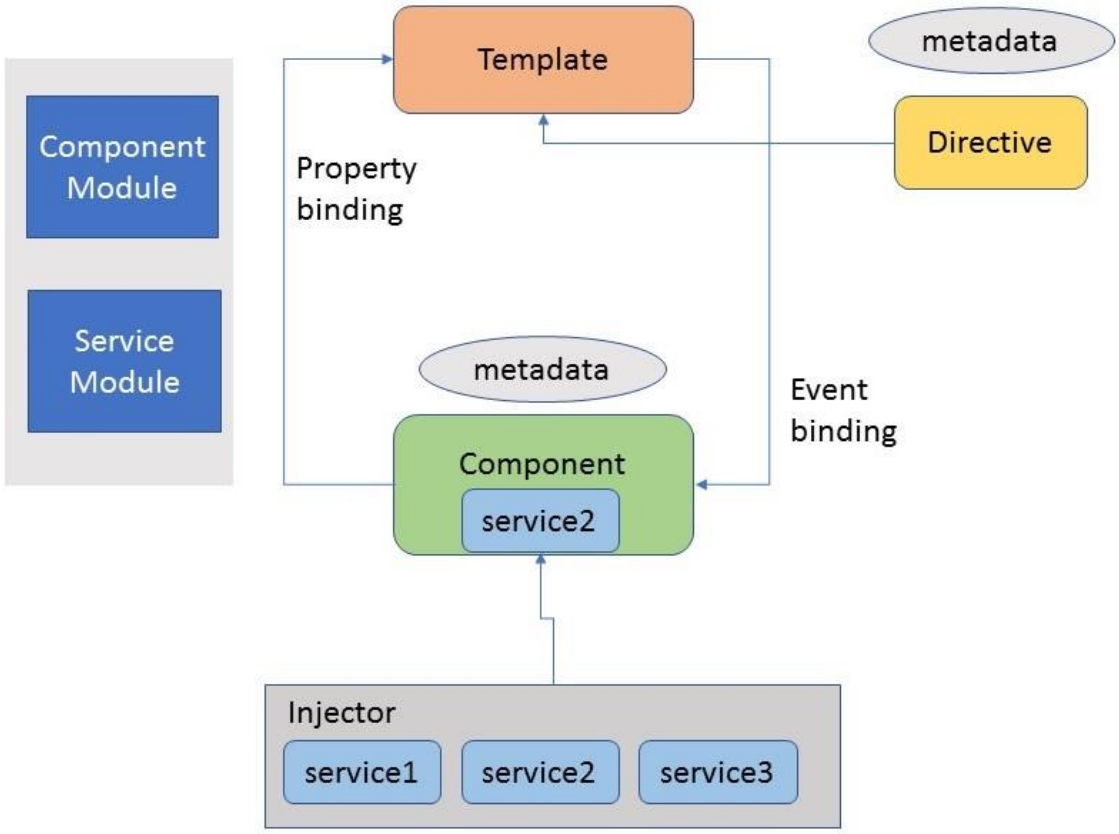

Рис. 2.3. Схема роботи будь-якого компоненту Angular

Мовою програмування логіки front-end частини застосунка була обрана мова TypeScript тому що вона є типізованим аналогом JavaScript і вносить поняття статичної типізації змінних, також дозволяє використання класів, інтерфейсів та модулів у різних структурах проекту, що робить код більш схожим до концепції об’єктно-орієнтовного програмування, а це, в свою чергу, робить проект більш стабільним та захищеним від втручання у код зловмисників.

Мовою гіпертекстової розмітки була обрана мова HTML тому що вона є більш розповсюдженою у світі, підтримується всіма браузерами та фреймворками, а отже є універсальною і простою та ефективною у застосуванні.

Мовою каскадних таблиць та стилізації документа було обрано SCSS тому що вона також як і HTML підтримується більшістю браузерів та є більш зручною, ніж звичайний CCSS, тому що краще реалізує поняття інкапсуляції класів.

Також для передачі даних був застосований формат JSON, тому що він базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передавання структурованої інформації через мережу.

2.5 Опис структури програми та алгоритмів її функціонування

Застосунок поділяється на клієнтську частину, та частину адміністратора.

Клієнтська частина буде мати в собі такі сторінки (представлені на рисунку 2.4):

- Головна сторінка;

Відображає список категорій, який бере з бази даних, містить зручні карточки категорій із посиланнями на сторінку зі списком товарів з цієї категорії.

- Сторінка списку товарів;

Відображає список товарів, який бере з бази даних, містить зручні карточки товарів із посиланнями на сторінку з детальною інформацією.

- Сторінка детальної інформації про товар;

Містить більш детальну інформацію про товар та посилання на додавання товару у кошик.

- Сторінка кошика;

Містить сторінку кошика, куди покупець може додавати товари для подальшого замовлення.

- Сторінка оплати;

Містить поля для вводу інформації користувача, та кнопку переходу до оплати замовлення.

- Сторінка логіну;

Містить форму логіну.

- Сторінка реєстрації;

Містить форму реєстрації.

- Сторінка замовлень користувача.

Містить інформацію про замовлення користувача.

Рис. 2.4. Список сторінок клієнтської частини

Частина адміністратора буде мати в собі такі сторінки (представлено на рисунку 2.5):

- Сторінка списку товарів;

Відображає список товарів, який бере з бази даних, містить зручні карточки товарів із посиланнями на сторінку з редагуванням.

- Форма додавання/редагування товару;

Містить більш детальну інформацію про товар та форму з полями для редагування інформації про товар.

- Сторінка списку категорій;

Відображає список категорій, який бере з бази даних.

- Форма додавання/редагування категорії;

Містить більш детальну інформацію про категорію та форму з полями для редагування інформації про категорію.

- Сторінка логіну;

Містить форму логіну.

- Сторінка реєстрації;

Містить форму реєстрації

- Сторінка зі списком замовлень;

Містить список замовлень користувачів.

- Форма редагування статусу замовлення;

Форма з детальною інформацію про замовлення та можливістю зміни його статусу.

- Сторінка редагування банеру;

Редагує банер.

- Сторінка списку користувачів;

Відображає список користувачів, який бере з бази даних.

- Форма додавання/редагування користувача.

Містить форму з полями детальної інформації про користувача та можливістю надавання йому прав адміністратора.

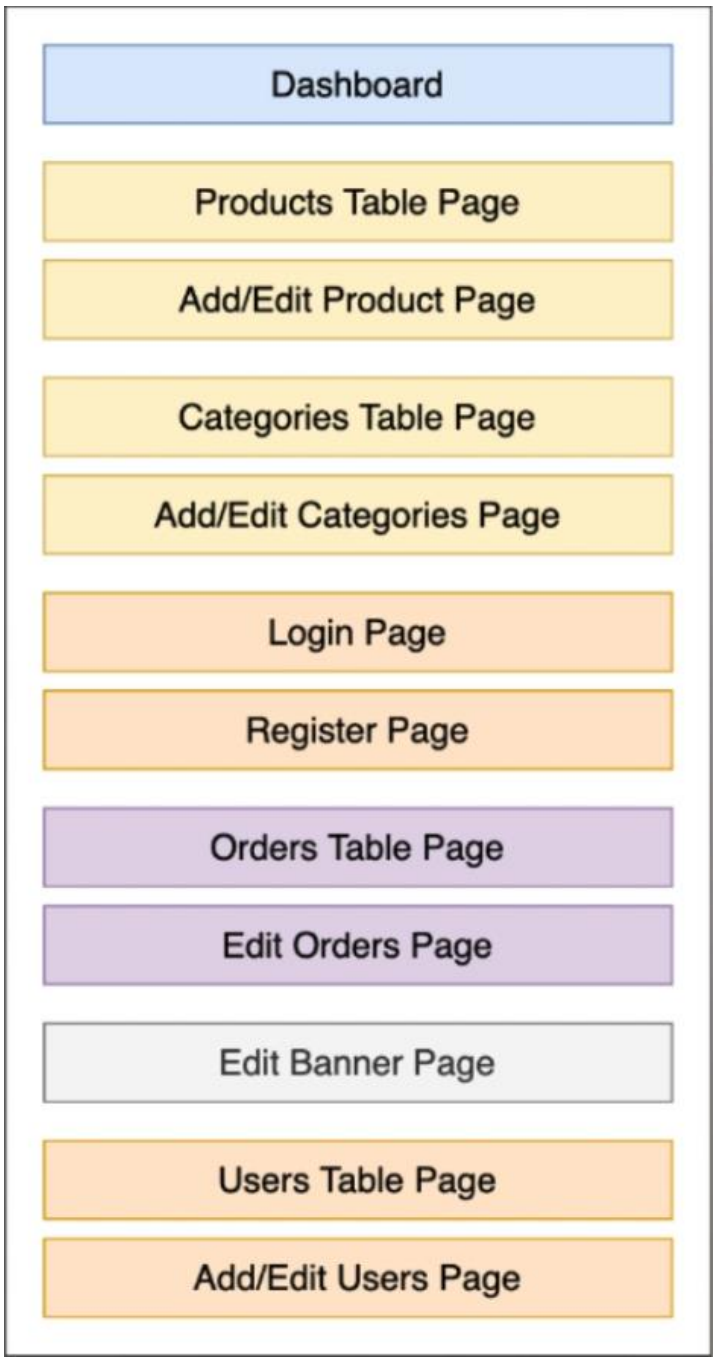

Рис. 2.5. Список сторінок адміністраторської частини

На рисунку 2.6 наведено список маршрутизація між сторінками клієнтської частини:

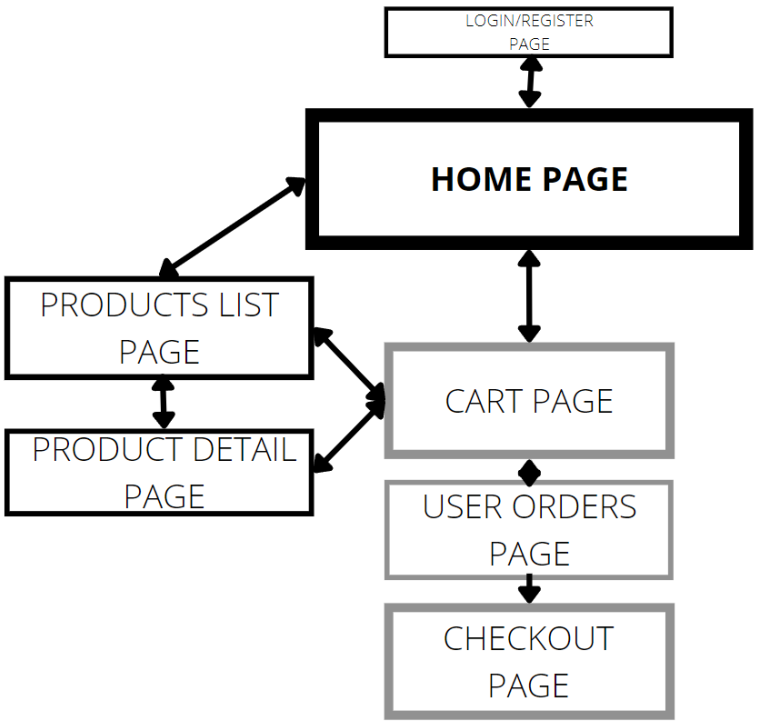

Рис. 2.6. Маршрутизація сторінок клієнтської частини

На рисунку 2.7 наведено список маршрутизація між сторінками панелі адміністратора:

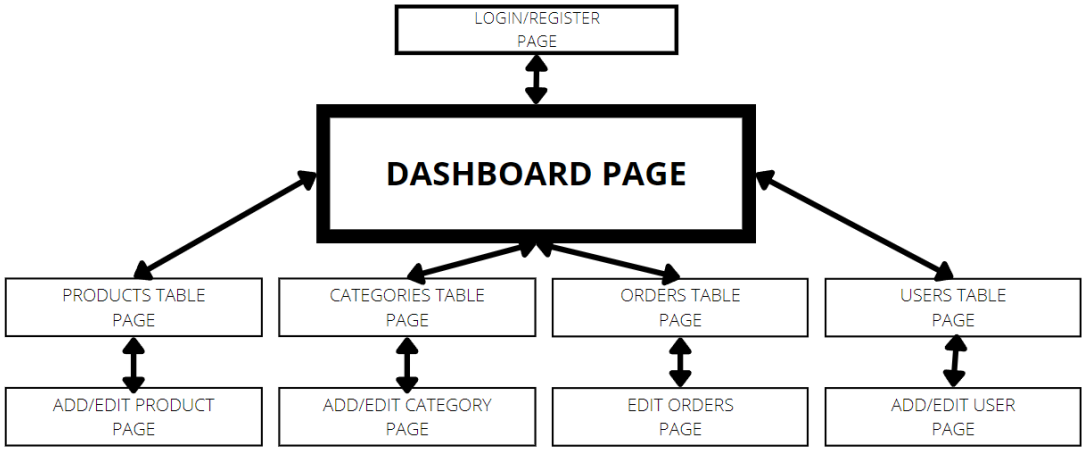

Рис. 2.7. Маршрутизація панелі адміністратора

2.6 Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними у front-end частині застосунку є масив даних у форматі JSON. Як вже було оголошено у пункті 2.4 – цей формат є надійним завдяки шифруванню та легко-читабельним для людини завдяки зрозумілому синтаксису.

Також до вхідних даних можна віднести об’єкти даних з полями різних типів, зокрема це типи: string, number, object, array та boolean. Такі дані надходять до сервера з клієнтської частини за допомогою заповнення різних форм додавання або редагування товарів, де є відповідні поля для їх введення.

На рисунку 2.8 наведено приклад такої форми.

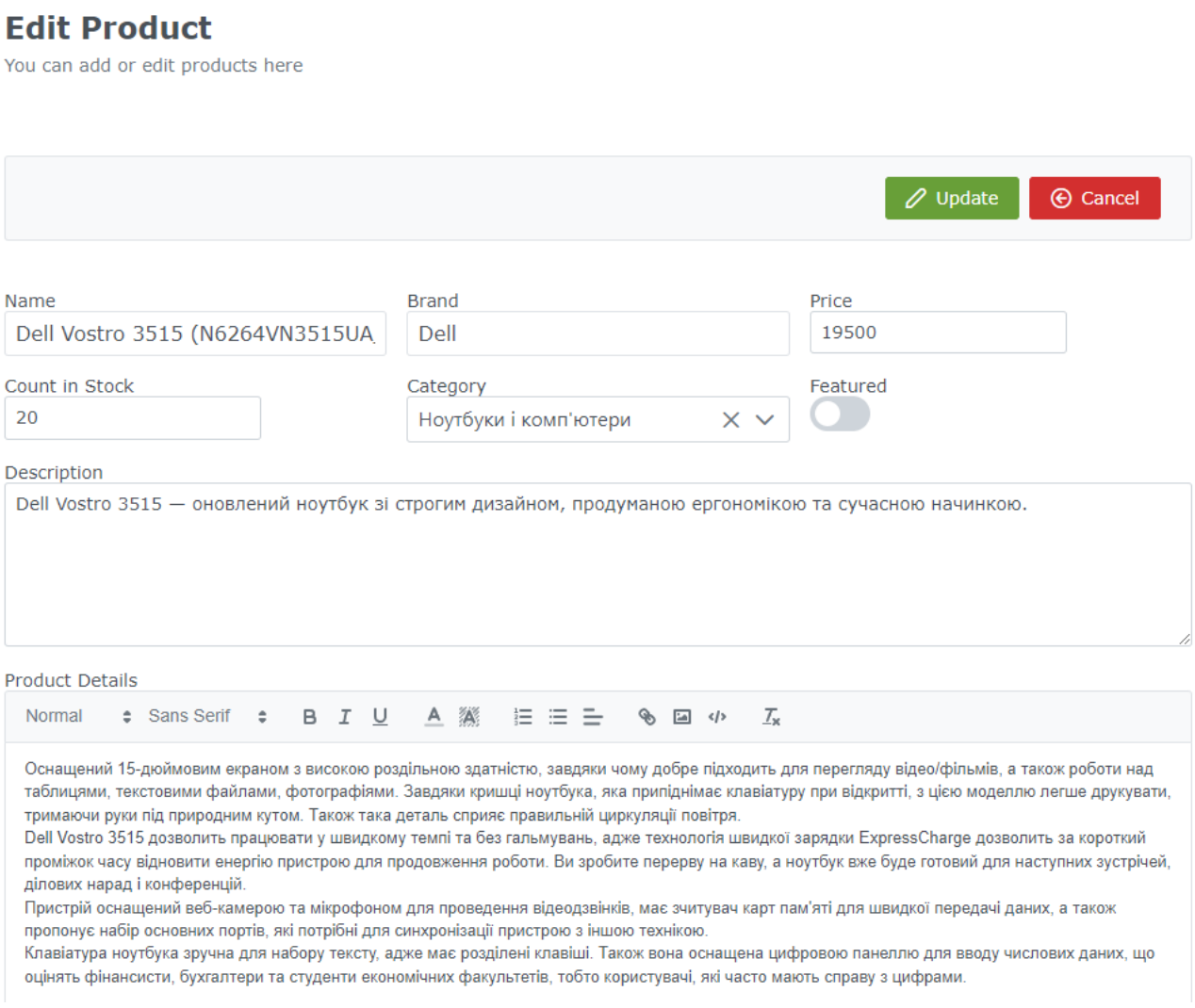

Рис. 2.8. Приклад форми для заповнення даних

Як можемо побачити, форма має різні поля для вводу даних у текстовому та чисельному форматах, також є перемикач стану, вхідними даними якого є логічний тип даних boolean.

Вхідні дані клієнтського сервісу представлятиме все той же текстовий файл у форматі JSON, що передаватиме всю потрібну інформацію про отримане від користувача замовлення до бази даних, звідки вона потраплятиме до панелі адміністраторів. Для передачі вихідних даних використовується клас об’єкту Order, у якому зберігається детальна інформація про замовника, а також про саме замовлення (таблиця 2.1).

Таблиця 2.1

Таблиця «Order»

Назва поля	Тип даних
id	string
orderItems	OrderItem[]
shippingAddress1	string
city	string
zip	string
country	string
phone	string
status	number
totalPrice	string
user	string
dateOrdered	string

Вихідними даними front-end частини додатку є інтерфейс, де відображаються усі дані, надіслані з бази даних у форматі JSON, які перемальовуються за допомогою технологій мови TypeScript, HTML та SCSS

щоб потрапити у сприятливий для користувача вигляд. Це відноситься як до частини адміністратора, так і до частини покупця.

На рисунках 2.9, 2.10 та 2.11 зображені приклади вихідних даних для панелі адміністратора.

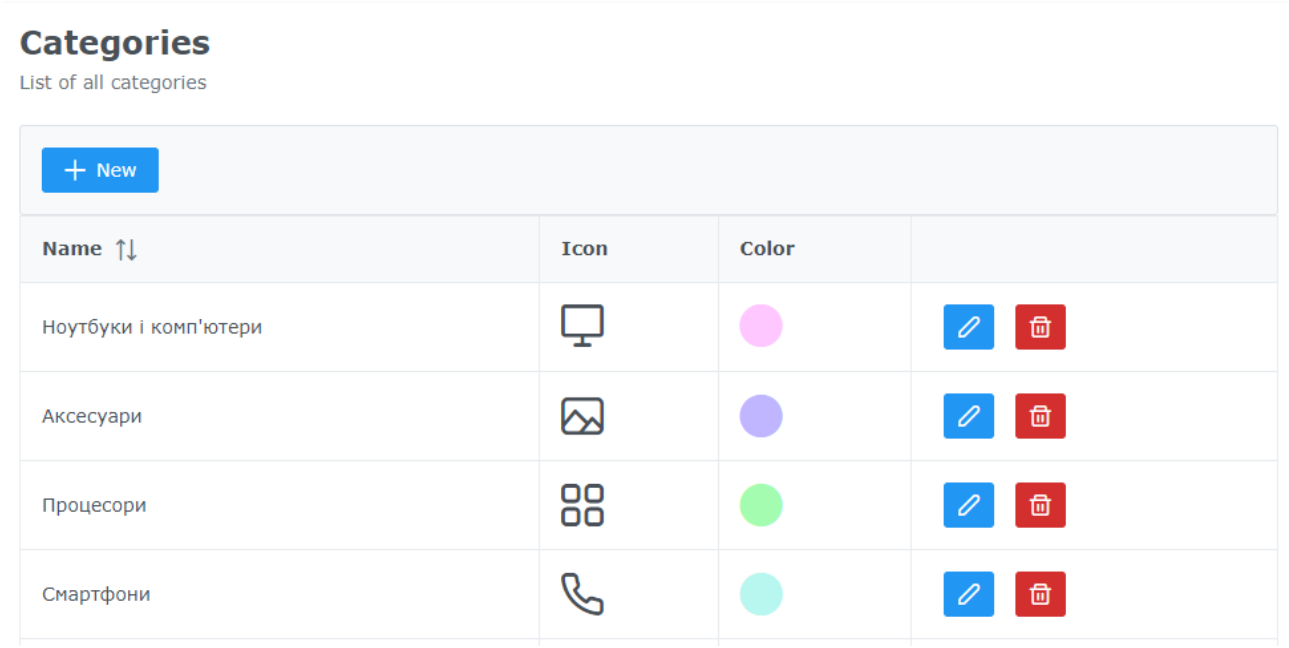

Рис. 2.9. Приклад вихідних даних адміністраторської панелі

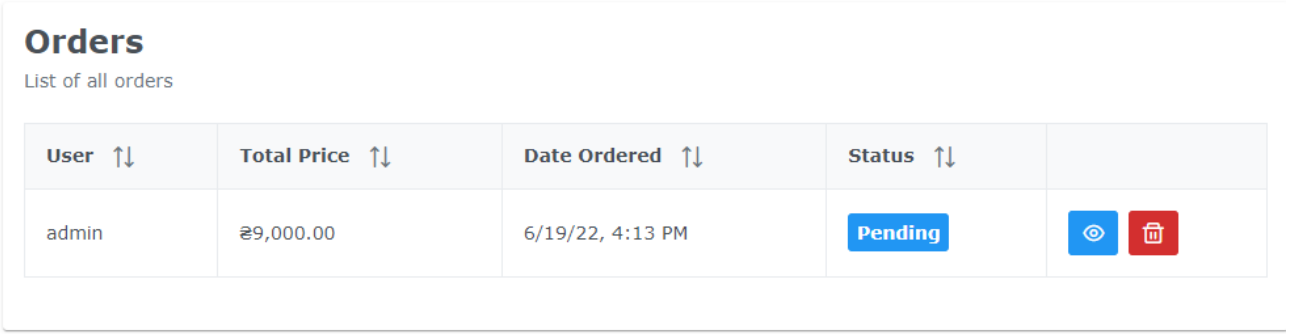

Рис. 2.10. Приклад вихідних даних адміністраторської панелі

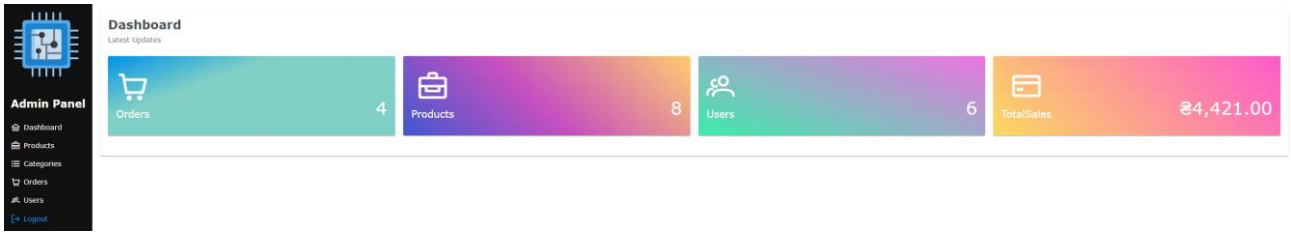

Рис. 2.11. Приклад вихідних даних адміністраторської панелі

На рисунках 2.12, 2.13 та 2.14 зображені приклади вихідних даних для інтерфейсу покупця.

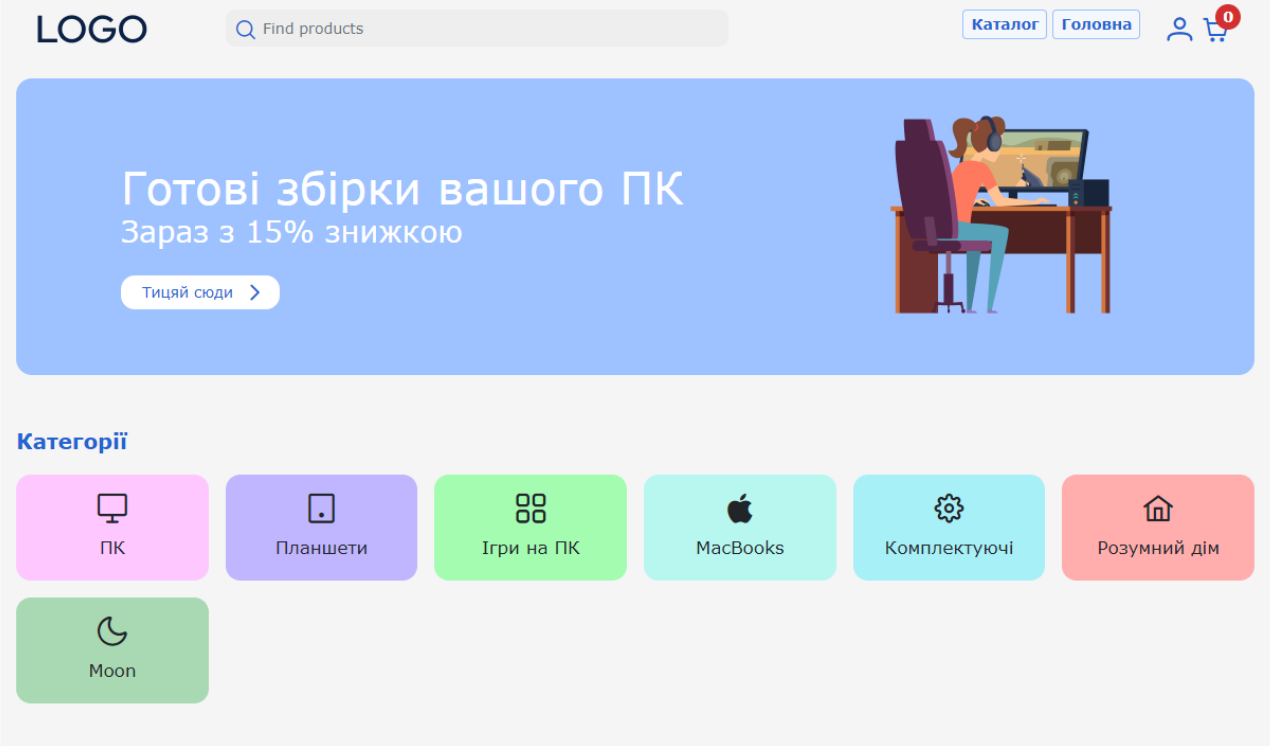

Рис. 2.12. Приклад вихідних даних панелі покупця

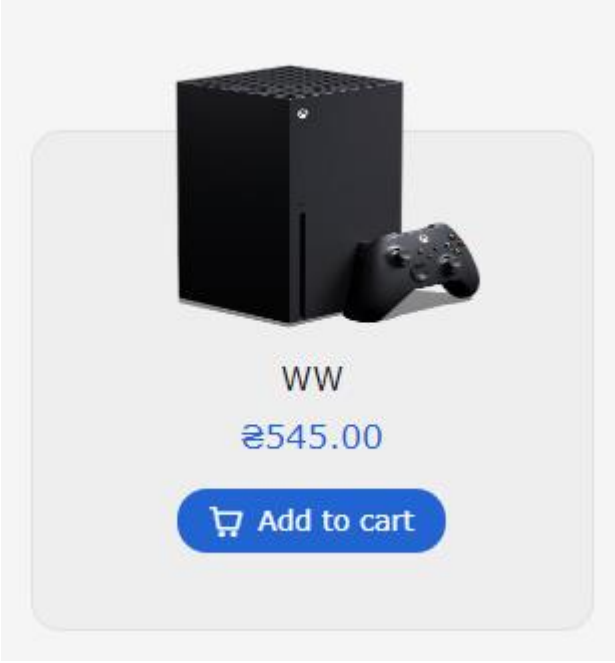

Рис. 2.13. Приклад вихідних даних панелі покупця

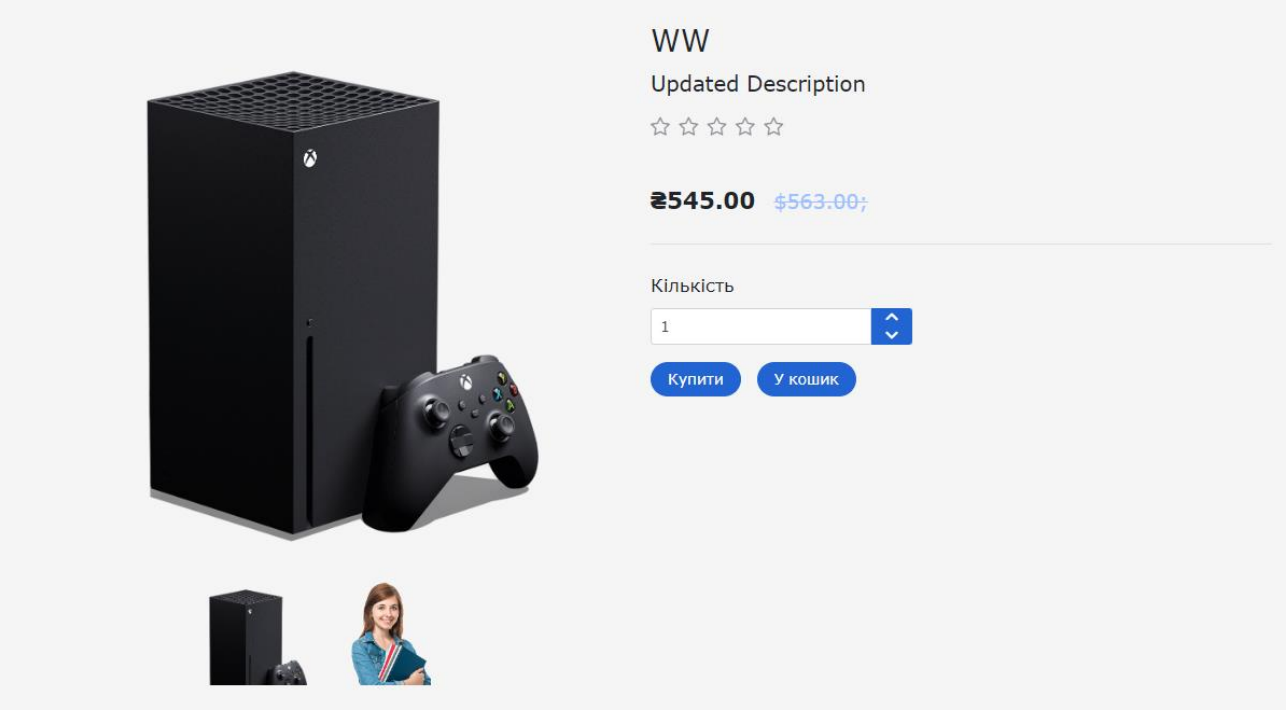

Рис. 2.14. Приклад вихідних даних панелі покупця

2.7 Опис розробленого програмного продукту

2.7.1 Використані технічні засоби

Для стабільної роботи з додатком та його розробки була використана система з даними характеристиками:

- Центральний процесор: AMD Ryzen 7 3700X 8 core 16 threads;
- Відеоадаптер: GeForce RTX 2070 SUPER;
- Відеопам’ять: 8 GB;
- Оперативна пам’ять: 16GB DDR4 3200 Mhz;
- Накопичувач: 1 TB.

Для стабільної розробки чи роботи системи рекомендуються такі, або схожі технічні характеристики.

2.7.2 Використані програмні засоби

Операційна система комп’ютера рекомендується Windows 10 та новіше.

Windows 10 – це операційна система від компанії Microsoft для персональних комп'ютерів, ноутбуків, планшетів, лептопів-трансформерів і смартфонів.

Для роботи застосунку повинна бути встановлена остання версія пакункового менеджера NPM.

NPM – це це менеджер пакунків для мови програмування JavaScript. Для середовища виконання Node.js це менеджер пакунків за замовчуванням. Включає в себе клієнт командного рядка, який також називається npm, а також онлайн-базу даних публічних та приватних пакунків, яка називається реєстром npm. Реєстр доступний через клієнт, а доступні пакунки можна переглядати та шукати через вебсайт npm. Менеджер пакунків та реєстр керуються npm, Inc.

Також робота додатку потребує встановлення останньої версії Angular та Angular CLI.

Angular CLI – це офіційний інтерфейс командного рядка, що використовується в екосистемі Angular. Його мета – спростити створення високоякісних додатків. Простіше кажучи, це набір утиліт для створення нових проектів, оновлення існуючого коду, додавання сторонніх бібліотек.

Для зручного оперування програмою рекомендується встановити застосунок Visual Studio Code останньої версії.

Visual Studio Code – це засіб для створення, редагування та зневадження сучасних вебзастосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Щоб мати змогу зв’язуватись з back-end частиною потрібно мати останню версію платформи Node.js.

Node.js – це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Якщо

раніше JavaScript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

2.7.3 Виклик та завантаження програми

Щоб викликати завантаження програми потрібно спочатку перейти до кореню проекту з додатком та у командному рядку викликати команду «npm run server» для запуску серверної частини додатку, команду можна побачити на рисунку 2.15.

```
\shop final project\Techno-Base\my-team\Techno-Base> npm run server
```

Рис. 2.15. Команда запуску серверної частини додатку

Коли сервер запуститься, у терміналі з’явиться відповідне повідомлення про встановлення успішного зв’язку з базою даних, яке можна побачити на рисунку 2.16.

```
> my-team@0.0.0 server D:\Dmytro\Dima\DIPLOMA\shop final project\Techno-Base\my-team\Techno-Base
> nodemon backend/app.js

[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node backend/app.js`
server is running http://localhost:3000
we are using eshop-database
Database Connection is ready...
```

Рис. 2.16. повідомлення про успішну роботу сервера

Далі, все ще знаходячись у кореневій папці проекту треба викликати консольну команду «ng serve назва_проекта» або «npx nx serve назва_проекта».

Для адміністративної панелі слід вказати назву проекта admin, а для сторінки магазину – techno-base. Команду можна побачити на рисунку 2.17.

```
> Executing task: npx nx serve admin <
```

Рис. 2.17. команда запуску front-end частини застосунка

Після введення команди та натискання клавіші «Enter» можна буде побачити повідомлення про те, що клієнтська часина запущена та працює на локальному порті localhost:4200. Повідомлення можна побачити на рисунку 2.18.

```
> nx run admin:serve:development

✓ Browser application bundle generation complete.

Initial Chunk Files   | Names         |  Raw Size
vendor.js             | vendor        |   4.26 MB |
styles.css, styles.js | styles        | 832.24 kB |
scripts.js            | scripts       | 429.43 kB |
polyfills.js          | polyfills     | 294.87 kB |
main.js               | main          | 265.61 kB |
runtime.js            | runtime       |   6.51 kB |

                      | Initial Total |   6.04 MB

Build at: 2022-06-16T09:51:55.057Z - Hash: 7dd565138537f508 - Time: 6485ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


✓ Compiled successfully.
```

Рис. 2.18. Повідомлення про успішний запуск клієнтської частини

Далі необхідно перейти до будь-якого браузера та ввести у адресний рядок адресу та порт «localhost:4200». Приклад адреси та порта можна побачити на рисунку 2.19.

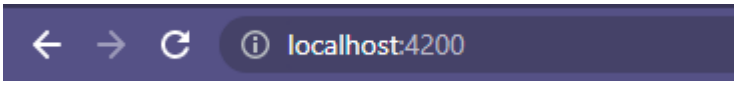

Рис. 2.19. Приклад введення адреси та порта застосунку

Ви попадете на сторінку логіну користувача. Для отримання доступу до сторінки адміністратора треба ввести логін: «admin@gmail.com» та пароль: «admin». Побачити панель логіну можна на рисунку 2.20.

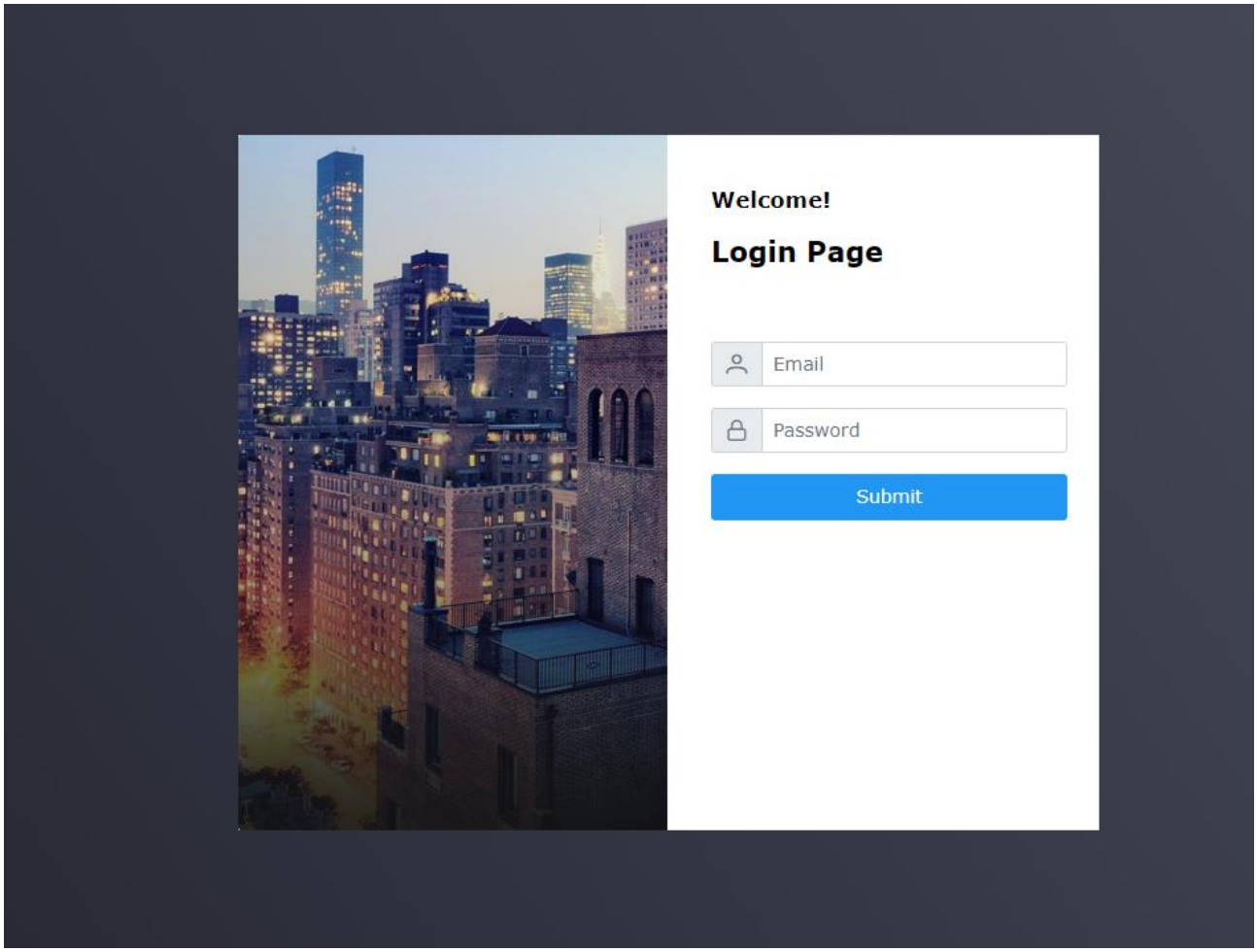

Рис. 2.20. сторінка логіну користувача

Після чого, якщо ви все зробили правильно, вам відкриється головна сторінка адміністраторської панелі якщо ви вказали у назві проекта admin, та головна сторінка покупця, якщо ви вказали techno-base. На рисунку 2.21 можна побачити приклад головної сторінки адміністратора.

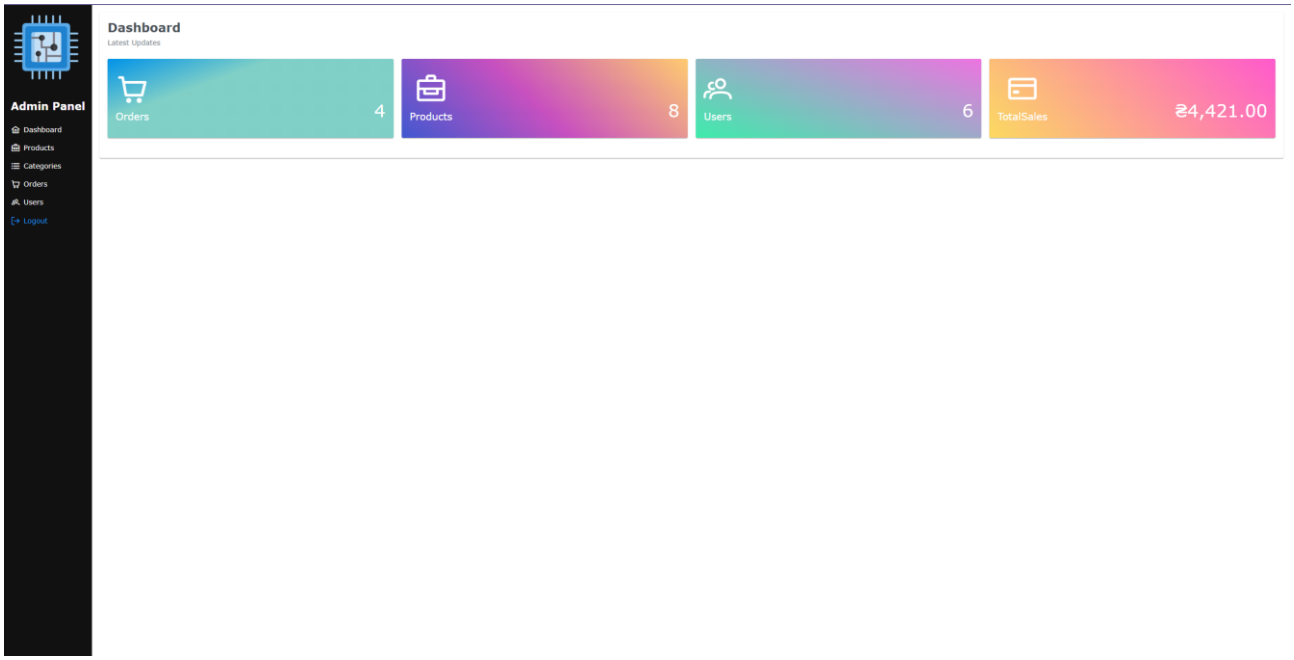

Рис. 2.21. Головна сторінка адміністратора магазину

2.7.4 Опис інтерфейсу користувача

Одразу після переходження по посиланню localhost:4200 вам відкривається головна сторінка адміністраторської панелі, тобто Dashboard. На рисунку 2.22 у розділі 2.7.3 можна побачити як він виглядає. На ньому адміністратор бачить стилсу статистику щодо продажів, кількості користувачів, кількості товарів та замовлень. Ця статистика оновлюється у реальному часі та відображена більш близько на рисунку 2.22.

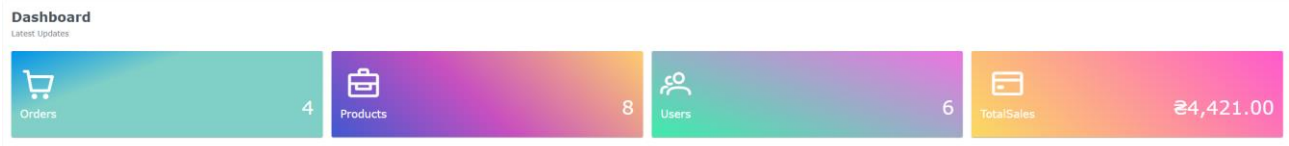

Рис. 2.22. Стисла статистика для адміністратора

Зліва можна побачити компонент бокового меню, або ж Sidebar, у якому знаходяться усі посилання на сторінки панелі адміністратора. Більш детально його можна розглянути на рисунку 2.23.

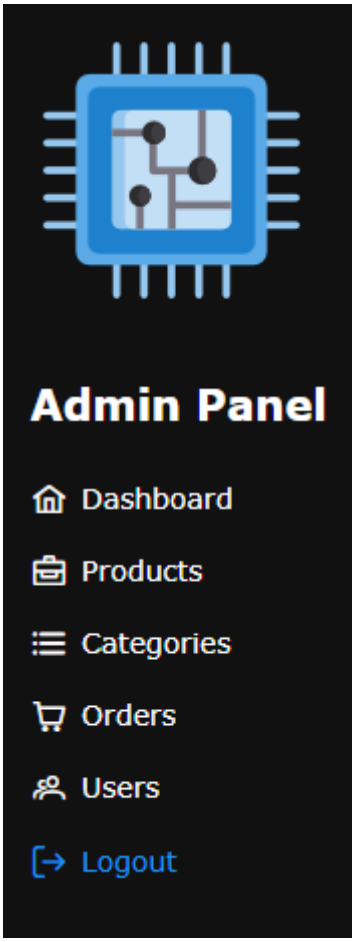

Рис. 2.23. Компонент бокового меню

Якщо натиснути на кнопку «Products» то можна потрапити до списку усіх товарів магазину, список можна побачити на рисунку 2.24. У кожного товару є кнопки видалення та редагування, а також зверху кнопка додавання нового товару.

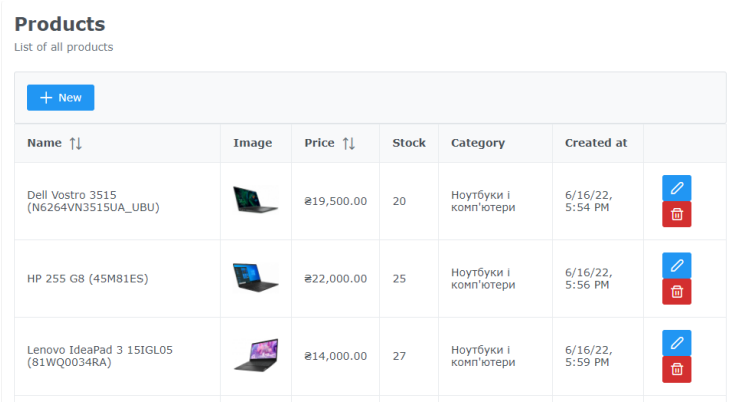

Рис. 2.24. Список товарів панелі адміністратора

Якщо натиснути на червону кнопку видалення, то отримаємо повідомлення з потребою підтвердження операції. Повідомлення можна побачити на рисунку 2.25. При натисканні кнопок Yes або No будуть виконані відповідні операції щодо видалення чи повернення назад до списку.

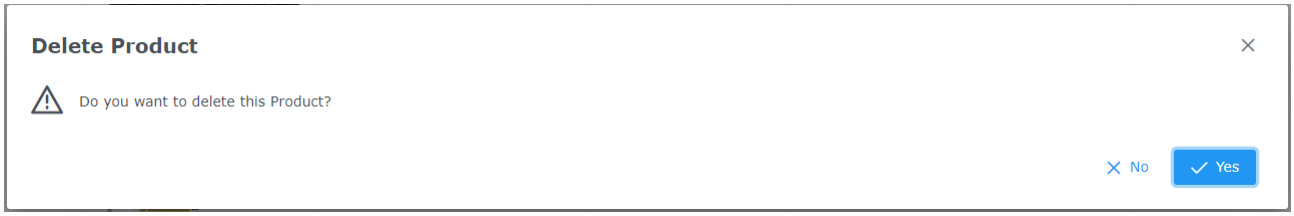

Рис. 2.25. Вікно підтвердження видалення товару

Якщо натиснути кнопку New або на кнопку редагування товару – застосунок перейде до однієї і тієї ж форми додавання/редагування певного товару. Приклад форми можна побачити на рисунку 2.26. У формі є необхідні поля для заповнення інформації про товар, а також можливість додавання фотографії до товару.

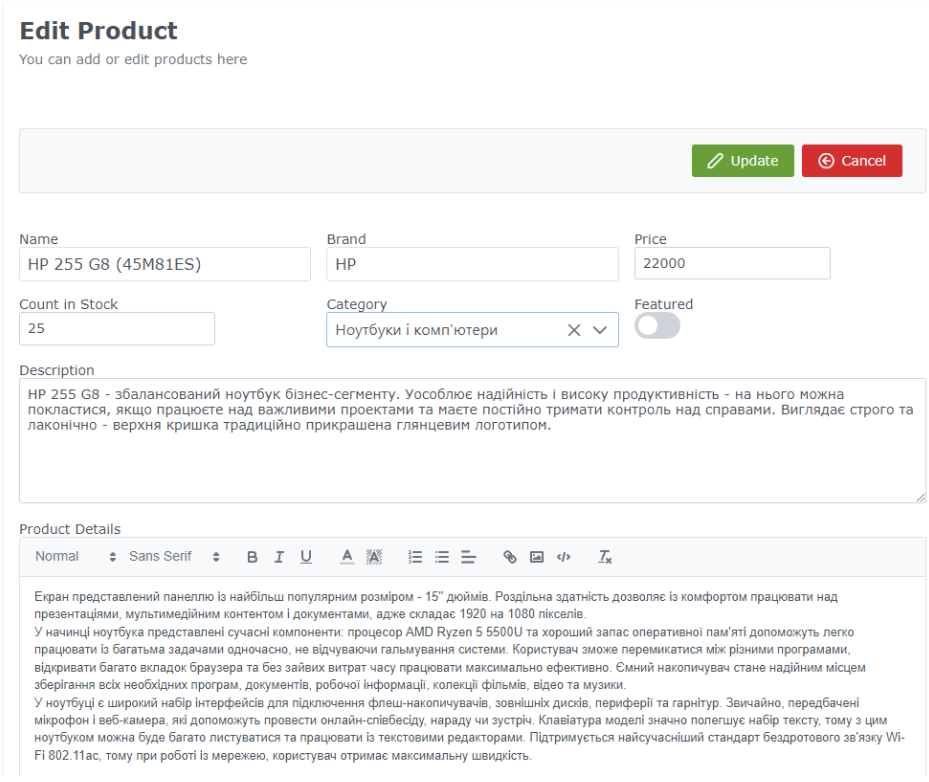

Рис. 2.26. форма додавання/редагування товару

Аналогічно до товарів, якщо натиснути на кнопку Categories на боковій панелі, застосунок перейде до списку категорій, який працює аналогічно з товарами. Тобто має кнопки додавання та редагування категорії, а також кнопки видалення.

Список категорій можна побачити на рисунку 2.27, а форму додавання/редагування категорій на рисунку 2.28.

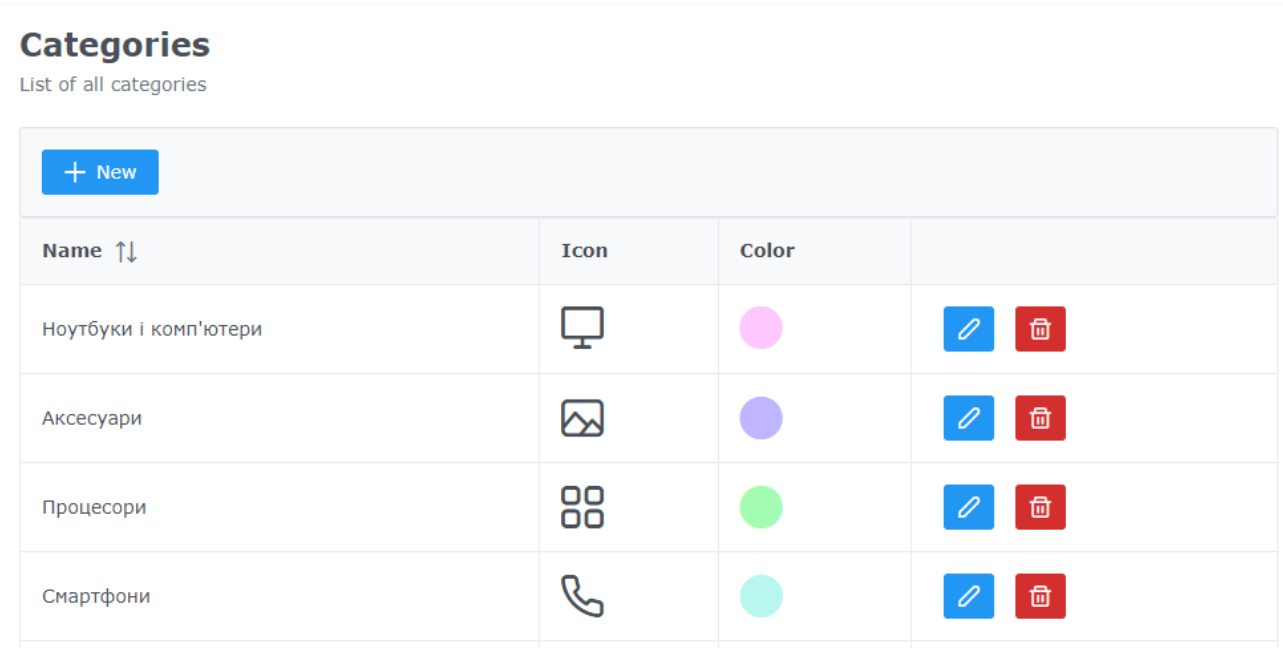

Рис. 2.27. Список категорій

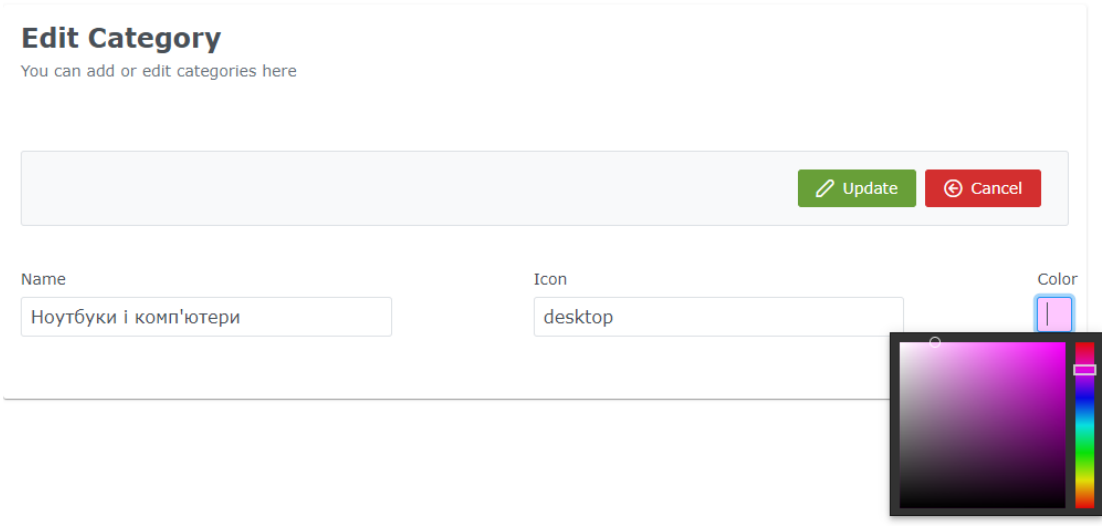

Рис. 2.28. Форма додавання/редагування категорії

Якщо натиснути на кнопку Orders на боковій панелі – застосунок перейде до списку замовлень, які надсилаються від покупців. У кожного замовлення є кнопки редагування та видалення. Список замовлень можна побачити на рисунку 2.29.

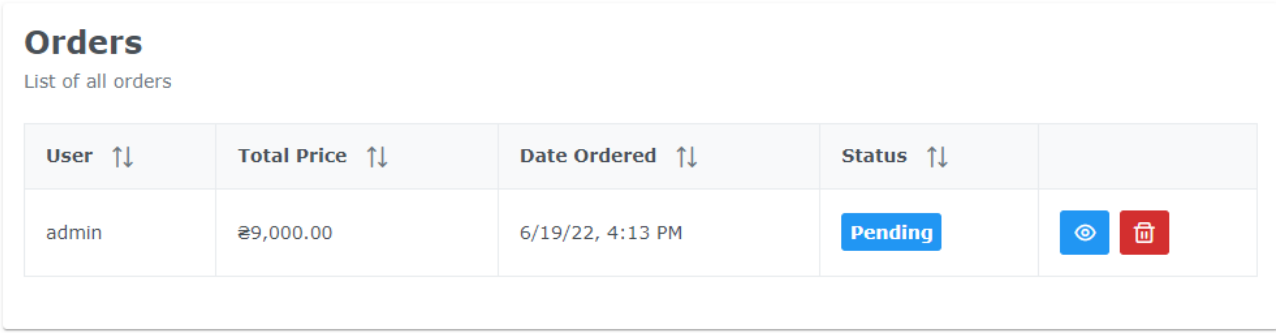

Рис. 2.29. Список замовлень користувачів

Якщо перейти до редагування будь-якого замовлення – адміністратору додатка відкриється форма, в якій буде детально прописана уся інформація щодо замовлення та випадаючий список, у якому можна буде змінити статус замовлення. Форму замовлення можна побачити на рисунку 2.30, а випадаючий список на рисунку 2.31.

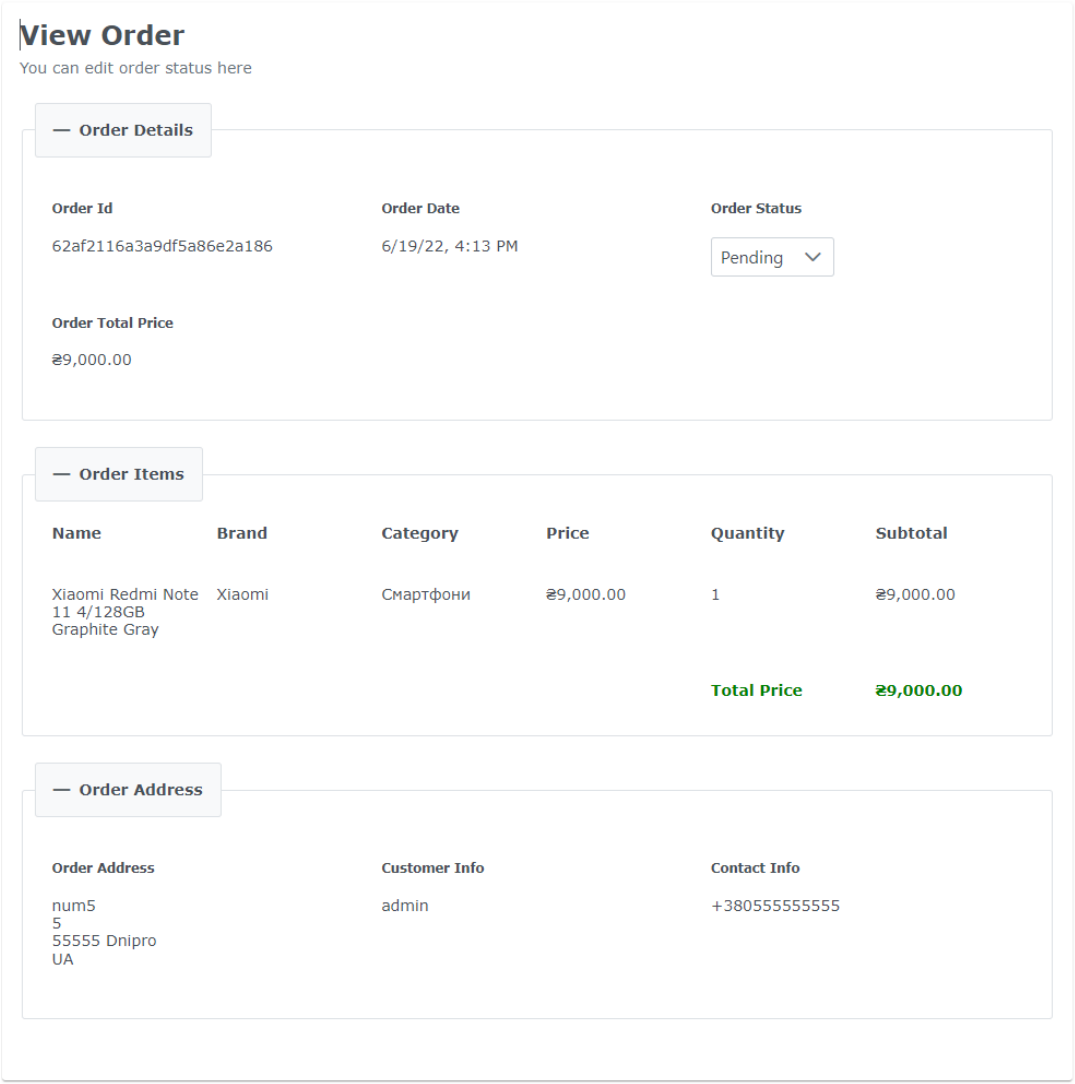

Рис. 2.30. Форма замовлення

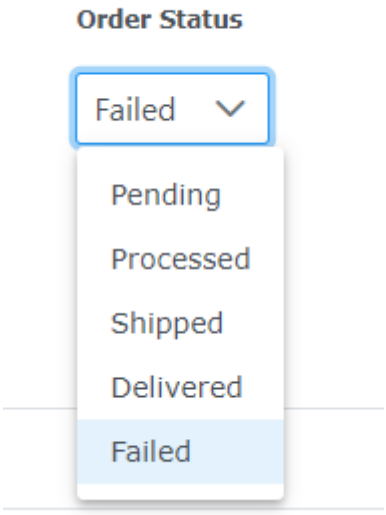

Рис. 2.31. Випадаючий список статусів замовлення

Якщо на боковій панелі натиснути на кнопку Users, то відкриється аналогічний категоріям та товарам список користувачів магазину, які є у системі та відповідні кнопки редагування/додавання користувача та видалення. Кнопки редагування та додавання користувача відсилають адміністратора на відповідну форму. Список можна побачити на рисунку 2.32, а форму на рисунку 2.33.

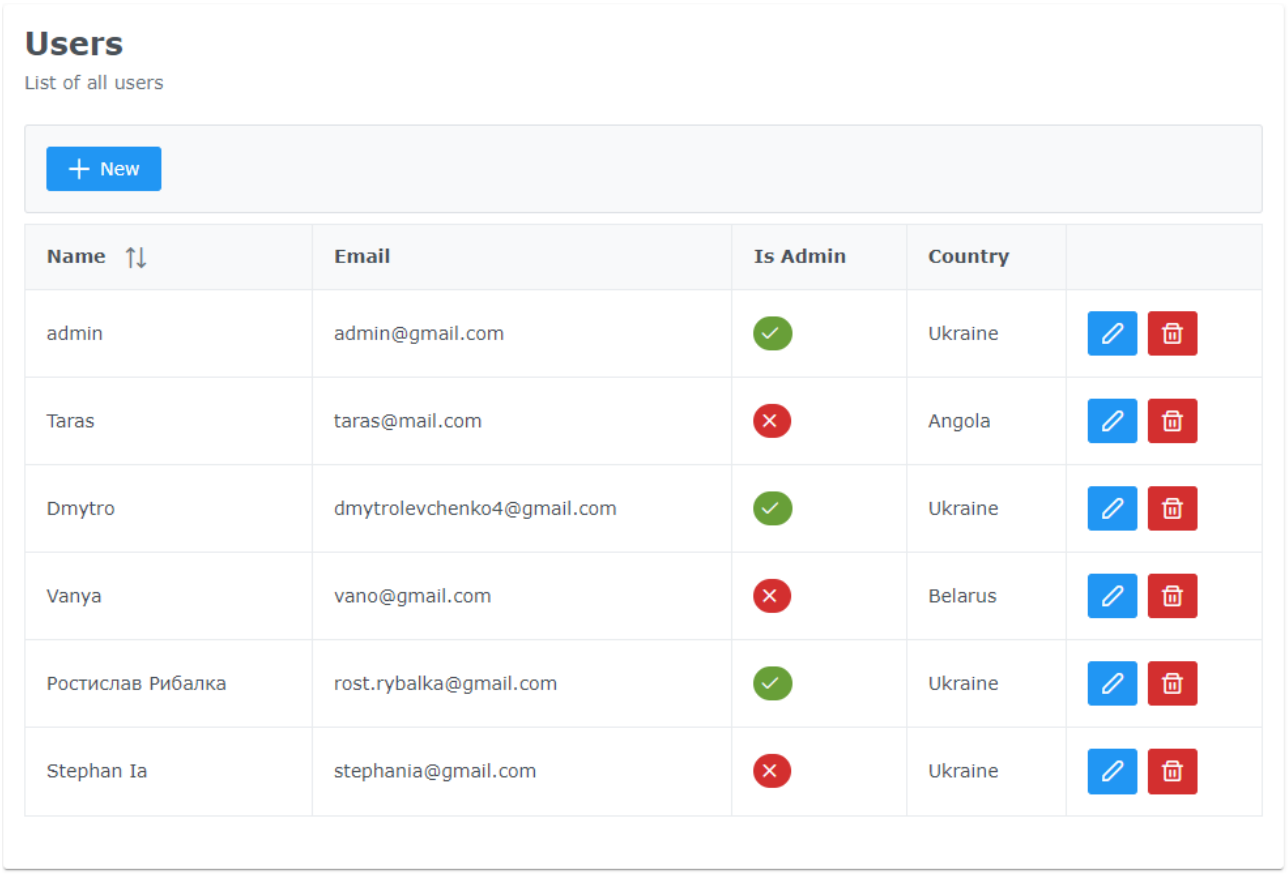

Рис. 2.32. Список користувачів магазину, шо є у системі

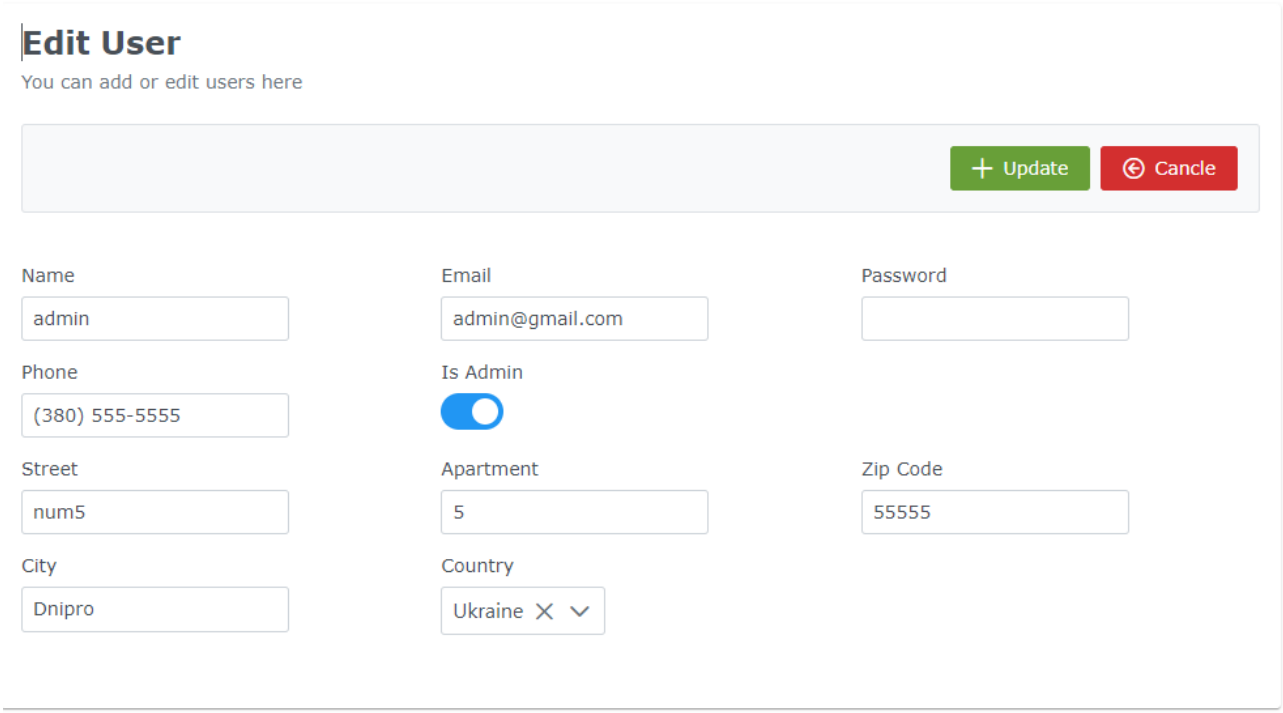

Рис. 2.33. Форма додавання/редагування користувача

Якщо на боковій панелі натиснути Logout, то із локального сховища системи буде видалений токен, який був присвоєний при логіні та користувача викине на панель логіну до застосунку. Панель логіну можна побачити на рисунку 2.21.

РОЗДІЛ 3

ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 2000;
2. коефіцієнт складності програми – 1,2;
3. коефіцієнт корекції програми в ході її розробки – 0,06;
4. годинна заробітна плата програміста – 150 грн/год;

Середня годинна зарплатня розробника була вирахувана використовуючи статистичні данні з сайту dou.ua. Середня заробітна плата Junior Angular розробника становить 900 доларів у місяць. Наразі курс долара за даними НБУ становить 29,25 гривень. Таким чином зарплата у гривнях для розробника у місяць становить 26325. Якщо для працівників встановлений 40-годинний робочий тиждень, як правило, це означає, що вони працюють 5 днів на тиждень по 8 годин на день. Відповідно, на місяць випадає від 21 до 23 трудових днів, якщо на будні дні не припадають свята. Тож у середньому візьмемо 22 робочих дні по 8 годин. З цього виходить, що розробник працює 176 годин на місяць у середньому. Тому це приблизно 150 грн/год (149,57 грн/год).

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,4;
7. вартість машино-години ЕОМ – 18 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:

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

де t_o – витрати праці на підготовку й опис поставленої задачі (в нашому випадку приймається 50).

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – 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):

$$Q = q \cdot C \cdot (1 + p), \text{ (3.2)}$$

де q – передбачуване число операторів (2000);

C – коефіцієнт складності програми (1,2);

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

$$Q = 2000 \cdot 1{,}2 \cdot (1 + 0{,}06) = 2544;$$

Витрати праці на вивчення опису задачі ti визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85) \cdot K}$$, людино-годин, (3.3);

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,2);

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

$$t_u = \frac{2544 \cdot 1{,}2}{85 \cdot 1{,}4} = 25{,}7$$, людино-годин

Витрати праці на розробку алгоритму рішення задачі:

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

$$t_a = \frac{2544}{20 \cdot 1{,}4} = 90{,}9$$, людино-годин.

Витрати на складання програми по готовій блок-схемі:

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

$$t_n = \frac{2544}{25 \cdot 1{,}4} = 72{,}7$$, людино-годин.

Витрати праці на налагодження програми на ЕОМ:

за умови автономного налагодження одного завдання:

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

$$t_{\text{отл}} = \frac{2544}{5 \cdot 1{,}4} = 363{,}4, \text{ людино-годин,}$$

за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{\text{к}} = 1{,}2 \cdot t_{\text{отл}}; \qquad (3.7)$$

$$t_{\text{отл}}^{\text{к}} = 1{,}2 \cdot 363{,}4 = 436{,}08, \text{ людино-годин,}$$

Витрати праці на підготовку документації:

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

де $t_{\partial p}$ – трудомісткість підготовки матеріалів і рукопису;

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

$$t_{\partial} = \frac{2544}{20 \cdot 1{,}4} = 90{,}9, \text{ людино-годин,}$$

де $t_{\partial o}$ – трудомісткість редагування, печатки й оформлення документації;

$$t_{\partial o} = 0{,}75 \cdot t_{\partial \text{р}}; \qquad (3.10)$$

$$t_{\partial o} = 0{,}75 \cdot 90{,}9 = 68{,}2, \text{ людино-годин.}$$

$$t_{\partial} = 90{,}9 + 68{,}175 = 159, \text{ людино-годин.}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 25{,}7 + 90{,}9 + 72{,}7 + 363{,}4 + 159 = 761, \text{ людино-годин.}$$

У результаті ми розрахували, що в загальній складності необхідно 761 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:

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

де t – загальна трудомісткість, людино-годин;

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

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

$$З_{ЗП} = 761 \cdot 150 = 114150, \text{ грн.}$$

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

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

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

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

$$З_{МВ} = 363{,}4 \cdot 18 = 6541{,}2 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 114150 + 6541{,}2 = 120691{,}2 \text{ грн.}$$

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:

$$T = \frac{761}{2 \cdot 176} = 2{,}2 \text{ міс.}$$

Висновки. Front-end інтернет-магазину продажу комп'ютерної техніки має вартість 120691,2грн. Ймовірно очікуваний час розробки за рахунок роботи одразу двох розробників 2.2 місяці при стандартному 40-годинному робочому тижні і 178-годинному робочому місяці. Цей термін пов'язаний з кількістю операторів і включає в себе час для дослідження та розробку алгоритму розв'язання задачі, розробку дизайну і створення документації. На розробку мобільного додатку буде витрачено 761 людино-годин.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка програмного забезпечення для інтернет-магазину комп’ютерної техніки та пристроїв, а саме для ведення продажу, обліку та контролю за товарами, також за користувачами сайту та замовленнями покупців.

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 являє собою проіменовану сукупність даних, організованих за певними правилами, що включає загальні принципи опису, зберігання і маніпулювання даними.

Програма призначена для забезпечення діяльності інтернет-магазину товарів комп’ютерної техніки та застосунків і надає можливість виконувати наступні дії:

- Додавання, видалення і редагування інформації про товари, категорії, замовлення, користувачів і покупців;
- Перегляд інформації про товари, категорії, замовлення, користувачів і покупців;
- Перегляд і друк інформації про продані товари, кількість замовлень і ціну товару;
- Здійснення пошуку необхідної інформації про товари, категорії, замовлення та користувачів;
- Здійснення операцій продажу;
- Можливість входу в систему з різними рівнями доступу до даних: адміністратор, користувач;
- Можливість зміни користувача у ході роботи програми;
- Здійснення контролю введених даних: перевірка на відповідність типів, на введення обов'язкових полів даних, а також, на введення тільки можливих значень, прочитуваних із необхідних таблиць;
- Можливість перегляду інформації з таблиць в режимі реального часу.

Актуальність поставленої задачі обумовлюється широким попитом на такі програмні продукти, що надають можливість електронного зберігання даних про товари, продажу, користувачів, ведення бази даних, підбиття підсумків з продажу, отримання звітів і формування супутньої ділової документації.

Розроблене програмне забезпечення інформаційної системи призначене для застосування в будь-якій мережі інтернет-магазинів з подібним родом діяльності і схожими функціональними вимогами.

Створений застосунок дозволить оптимізувати та спростити дії по веденню бази даних мережі інтернет-магазинів комп'ютерної техніки та застосунків; скоротити час на оформлення продажу; підвищити ефективність діяльності компанії шляхом електронного ведення документації з продажу і можливістю аналізу наявних даних і надання необхідних звітів і супутньої ділової документації.

Структура програми являє собою клієнтський застосунок, написаний на мові програмування високого рівня TypeScript з використанням фреймворку Angular, що взаємодіє з базою даних «MongoDB» шляхом надсилання запитів до серверної частини за допомогою технології Node Express. База даних розроблена мовою JSON та взаємодіє із додатком за допомогою JavaScript.

В «Економічному розділі» визначено трудомісткість розробки програмного забезпечення (761 чол-год), підраховані витрати на створення програмного забезпечення (120691,2 грн.) і гаданий період розробки (2,2 міс.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сучасний підручник JavaScript URL: https://learn.javascript.ru/ (дата звернення: 02.05.2022).
2. Statcounter URL: https://blog.statcounter.com (дата звернення: 01.05.2022).
3. Програмуваня по-українськи URL: programming.in.ua (дата звернення: 25.04.2022).
4. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.
5. Удовик І.М., Коротенко Л.М., Шевцова О.С. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Програмне забезпечення”. – М : НТУ «Дніпровська політехніка», 2013. 16 с.
6. Скорочення слів в українській мові у бібліографічному описі. Загальні правила та вимоги : ДСТУ 3582-97. – Чинний від 1998–07–01. – К. : Держстандарт України, 1998. – 24 с. – (Державний стандарт України).
7. Прохоренок Н.А. HTML, JavaScript, PHP і MySQL. Джентльменський набір Webмайстра - СПб. : БХВ-Петербург, 2010. 912 с.
8. Чіртік А.В. Популярний самовчитель HTML. - СПб. : Пітер, 2012. 56 с.
9. Зандстра М. PHP: Объекты, шаблоны и методики программирования. -М.: Вильямс, 2015. 577 с.
10. Ташков П.А. Веб-мастеринг. HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка. -М. : Питер, 2010. 512 с.
11. Михайлов С., Кулі Л. Основи дизайну. - К .: Нове знання, 1999. 106 с.
12. Печников В.Н. Создание Web-сайтов без посторонней помощи. – М.: Триумф, 2006, 463 с.

13. Вакансія Full Stack Web Developer URL: https://www.work.ua/ru/jobs/2679106/ (дата звернення: 28.05.2022).

14. Дронов В. А. HTML 5, CSS 3 и Web 2.0. Разработка современных Webсайтов / В.А.Дронов - СПб.: БХВ Петербург, 2011. - 416 с.: ил. - (Профессиональное программирование).

15. Десять лучших IDE и редакторов кода для веб-разработчиков [Електронний ресурс]/ URL: https://www.reg.ru/blog/10-luchshih-ide-iredaktorov-koda-dlya-veb-razrabotchikov/ (дата звернення: 06.05.2022).

16. Загрози інформаційної безпеки [Електронний ресурс] / URL: https://uk.wikipedia.org/wiki/Загрози_інформаційної_безпеки (дата звернення: 06.05.2021).

17. Статичні та динамічні web-сайти/ URL: https://armedsoft.com/ua/blog/statychni-ta-dynamichni-web-sayty (дата звернення: 11.05.2022).

18. Botha, J., Bothma, C., Geldenhuys, P. Managing E-commerce in Business. Cape Town: Juta and Company Ltd. – 2005. – p. 3.

19. Todd Fredrich. REST API Tutorial. – 2012. – p. 13-15.

20. JavaScript. Изучение веб-разработки. URL: https://developer.mozilla.org/ru/docs/Learn/JavaScript/ (дата звернення: 25.05.2022).

21. Никсон Робин. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-е изд. : Издательство 25 лет Питер, СанктПетербург, 2016. С. 331 – 668.

Додаток А

ЛІСТИНГ ПРОГРАМИ

Apps/admin/src/main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from 'environments/environment';

if (environment.production) {
    enableProdMode();
}

platformBrowserDynamic()
    .bootstrapModule(AppModule)
    .catch((err) => console.error(err));
```

Apps/admin/src/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { CategoriesService } from '@my-team/products';
import { JwtInterceptor, UsersModule } from '@my-team/users';

import { AppComponent } from './app.component';
import { DashboardComponent } from './pages/dashboard/dashboard.component';
import { ShellComponent } from './shared/shell/shell.component';
import { SidebarComponent } from './shared/sidebar/sidebar.component';
import { CategoriesListComponent } from './pages/categories/categories-list/categories-list.component';
import { CategoriesFormComponent } from './pages/categories/categories-form/categories-form.component';
import { ProductsListComponent } from './pages/products/products-list/products-list.component';
import { ProductsFormComponent } from './pages/products/products-form/products-form.component';
import { UsersListComponent } from './pages/users/users-list/users-list.component';
import { UsersFormComponent } from './pages/users/users-form/users-form.component';
```

```
import { OrdersListComponent } from './pages/orders/orders-list/orders-
list.component';
import { OrdersDetailComponent } from './pages/orders/orders-
detail/orders-detail.component';

import { CardModule } from 'primeng/card';
import { ToolbarModule } from 'primeng/toolbar';
import { ButtonModule } from 'primeng/button';
import { TableModule } from 'primeng/table';
import { InputTextModule } from 'primeng/inputtext';
import { ToastModule } from 'primeng/toast';
import { ConfirmationService, MessageService } from 'primeng/api';
import { ConfirmDialogModule } from 'primeng/confirmdialog';
import { ColorPickerModule } from 'primeng/colorpicker';
import { InputNumberModule } from 'primeng/inputnumber';
import { InputTextareaModule } from 'primeng/inputtextarea';
import { InputSwitchModule } from 'primeng/inputswitch';
import { DropdownModule } from 'primeng/dropdown';
import { EditorModule } from 'primeng/editor';
import { TagModule } from 'primeng/tag';
import { InputMaskModule } from 'primeng/inputmask';
import { FieldsetModule } from 'primeng/fieldset';
import { AppRoutingModule } from './app-routing.module';

const UX_MODULE = [
    CardModule,
    ToolbarModule,
    ButtonModule,
    TableModule,
    InputTextModule,
    ToastModule,
    ConfirmDialogModule,
    ColorPickerModule,
    InputNumberModule,
    InputTextareaModule,
    InputSwitchModule,
    DropdownModule,
    EditorModule,
    TagModule,
    InputMaskModule,
    FieldsetModule
];

@NgModule({
    declarations: [
        AppComponent,
        DashboardComponent,
        ShellComponent,
        SidebarComponent,
        CategoriesListComponent,
        CategoriesFormComponent,
```

```
        ProductsListComponent,
        ProductsFormComponent,
        UsersListComponent,
        UsersFormComponent,
        OrdersListComponent,
        OrdersDetailComponent
    ],
    imports: [BrowserModule, BrowserAnimationsModule, HttpClientModule,
FormsModule,    ReactiveFormsModule,    AppRoutingModule,    UsersModule,
...UX_MODULE],
    providers: [CategoriesService, MessageService, ConfirmationService, {
provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor, multi: true }],
    bootstrap: [AppComponent]
})
export class AppModule {}
```

Apps/admin/src/app.component.ts

```
import { Component } from '@angular/core';

@Component({
    selector: 'admin-root',
    templateUrl: './app.component.html'
})
export class AppComponent {
    title = 'admin';
}
```

Apps/admin/src/app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AuthGuard } from '@my-team/users';
import { CategoriesFormComponent } from './pages/categories/categories-
form/categories-form.component';
import { CategoriesListComponent } from './pages/categories/categories-
list/categories-list.component';
import          {          DashboardComponent          }          from
'./pages/dashboard/dashboard.component';
import   {   OrdersDetailComponent   }   from   './pages/orders/orders-
detail/orders-detail.component';
import { OrdersListComponent } from './pages/orders/orders-list/orders-
list.component';
import   {   ProductsFormComponent   }   from   './pages/products/products-
form/products-form.component';
import   {   ProductsListComponent   }   from   './pages/products/products-
list/products-list.component';
import  {  UsersFormComponent  }  from  './pages/users/users-form/users-
form.component';
import  {  UsersListComponent  }  from  './pages/users/users-list/users-
list.component';
```

```
import { ShellComponent } from './shared/shell/shell.component';

const routes: Routes = [
    {
        path: '',
        component: ShellComponent,
        canActivate: [AuthGuard],
        children: [
            {
                path: '',
                component: DashboardComponent
            },
            {
                path: 'categories',
                component: CategoriesListComponent
            },
            {
                path: 'categories/form',
                component: CategoriesFormComponent
            },
            {
                path: 'categories/form/:id',
                component: CategoriesFormComponent
            },
            {
                path: 'products',
                component: ProductsListComponent
            },
            {
                path: 'products/form',
                component: ProductsFormComponent
            },
            {
                path: 'products/form/:id',
                component: ProductsFormComponent
            },
            {
                path: 'users',
                component: UsersListComponent
            },
            {
                path: 'users/form',
                component: UsersFormComponent
            },
            {
                path: 'users/form/:id',
                component: UsersFormComponent
            },
            {
                path: 'orders',
                component: OrdersListComponent
```

```
            },
            {
                path: 'orders/:id',
                component: OrdersDetailComponent
            }
        ]
    }
];

@NgModule({
    imports: [RouterModule.forRoot(routes, { initialNavigation: 'enabled'
})],
    exports: [RouterModule],
    declarations: [],
    providers: []
})
export class AppRoutingModule {}
```

Apps/admin/src/app/pages/dashboard/dashboard.component.html

```
<div class="admin-page">
    <p-card header="Dashboard" subheader="Latest Updates">
        <div class="grid">
            <div class="col-3">
                <p-card styleClass="d-orders">
                    <div class="d-item">
                        <div>
                            <i class="pi pi-shopping-cart"></i><br />
                            <span class="name">Orders</span>
                        </div>
                        <span class="number">{{ statistics[0] }}</span>
                    </div>
                </p-card>
            </div>
            <div class="col-3">
                <p-card styleClass="d-products">
                    <div class="d-item">
                        <div>
                            <i class="pi pi-briefcase"></i><br />
                            <span class="name">Products</span>
                        </div>
                        <span class="number">{{ statistics[1] }}</span>
                    </div>
                </p-card>
            </div>
            <div class="col-3">
                <p-card styleClass="d-users">
                    <div class="d-item">
                        <div>
                            <i class="pi pi-users"></i><br />
```

```
                        <span class="name">Users</span>
                    </div>
                    <span class="number">{{ statistics[2] }}</span>
                </div>
            </p-card>
        </div>
        <div class="col-3">
            <p-card styleClass="d-totalsales">
                <div class="d-item">
                    <div>
                        <i class="pi pi-credit-card"></i><br />
                        <span class="name">TotalSales</span>
                    </div>
                    <span class="number">{{ statistics[3] | currency:
'UAH':'symbol-narrow' }}</span>
                </div>
            </p-card>
        </div>
    </div>
</p-card>
</div>
```

Apps/admin/src/app/pages/dashboard/dashboard.component.ts

```
import { Component, OnInit } from '@angular/core';
import { OrdersService } from '@my-team/orders';
import { ProductsService } from '@my-team/products';
import { UsersService } from '@my-team/users';
import { combineLatest } from 'rxjs';

@Component({
    selector: 'admin-dashboard',
    templateUrl: './dashboard.component.html'
})
export class DashboardComponent implements OnInit {
    statistics: number[] = [];
    constructor(private usersService: UsersService, private ordersService:
OrdersService, private productsService: ProductsService) {}

    ngOnInit(): void {
        combineLatest([
            this.ordersService.getOrdersCount(),
            this.productsService.getProductsCount(),
            this.usersService.getUsersCount(),
            this.ordersService.getTotalSales()
        ]).subscribe((values) => {
            this.statistics = values;
        });
        console.log('Dashboard has been loaded');
        console.log(this.statistics);
    }
```

```
}
```

Apps/admin/src/app/pages/products/products-list/products-list.component.html

```
<p-toast></p-toast>
<div class="admin-page">
    <p-card header="Products" subheader="List of all products">
        <div class="p-grid">
            <div class="p-col-12">
                <p-toolbar>
                    <div class="p-toolbar-group-left">
                        <p-button    label="New"    icon="pi    pi-plus"
routerLink="form"> </p-button>
                    </div>
                </p-toolbar>
            </div>
        </div>
        <div class="p-grid">
            <div class="p-col-12">
                <p-table      [value]="products"      [paginator]="true"
[rows]="10" styleClass="p-datatable-gridlines" responsiveLayout="scroll">
                    <ng-template pTemplate="header">
                        <tr>
                            <th    pSortableColumn="name">Name<p-sortIcon
field="name"></p-sortIcon></th>
                            <th>Image</th>
                            <th  pSortableColumn="price">Price<p-sortIcon
field="price"></p-sortIcon></th>
                            <th>Stock</th>
                            <th>Category</th>
                            <th>Created at</th>
                            <th></th>
                        </tr>
                    </ng-template>
                    <ng-template pTemplate="body" let-product>
                        <tr>
                            <td>{{ product.name }}</td>
                            <td          style="width:          10%"><img
[src]="product.image" style="width: 100%" alt="img" /></td>
                            <td>{{     product.price     |     currency:
'UAH':'symbol-narrow' }}</td>
                            <td>{{ product.countInStock }}</td>
                            <td>{{ product.category.name }}</td>
                            <td>{{  product.dateCreated  |  date:  'short'
}}</td>
                            <td>
                                <p-button    styleClass="p-button-primary
mx-2" icon="pi pi-pencil" (click)="updateProduct(product.id)"></p-button>
                                <p-button styleClass="p-button-danger mx-
2" icon="pi pi-trash" (click)="deleteProduct(product.id)"></p-button>
```

```
                        </td>
                    </tr>
                </ng-template>
            </p-table>
        </div>
    </div>
</p-card>
</div>
<p-confirmDialog [style]="{ width: '50vw' }" [baseZIndex]="10000"
rejectButtonStyleClass="p-button-text"></p-confirmDialog>
```

Apps/admin/src/app/pages/products/products-list/products-list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { ProductsService } from '@my-team/products';
import { ConfirmationService, MessageService } from 'primeng/api';

@Component({
    selector: 'admin-products-list',
    templateUrl: './products-list.component.html',
    styles: []
})
export class ProductsListComponent implements OnInit {
    products: any = [];

    constructor(
        private productsService: ProductsService,
        private router: Router,
        private messageService: MessageService,
        private confirmationService: ConfirmationService
    ) {}

    ngOnInit(): void {
        this._getProducts();
    }

    private _getProducts() {
        this.productsService.getProducts().subscribe((products) => {
            this.products = products;
        });
    }

    updateProduct(productid: string) {
        this.router.navigateByUrl(`products/form/${productid}`);
    }

    deleteProduct(productId: string) {
        this.confirmationService.confirm({
```

```
            message: 'Do you want to delete this Product?',
            header: 'Delete Product',
            icon: 'pi pi-exclamation-triangle',
            accept: () => {
                this.productsService.deleteProduct(productId).subscribe(
                    () => {
                        this._getProducts();
                        this.messageService.add({
                            severity: 'success',
                            summary: 'Success',
                            detail: 'Product is deleted!'
                        });
                    },
                    () => {
                        this.messageService.add({
                            severity: 'error',
                            summary: 'Error',
                            detail: 'Product is not deleted!'
                        });
                    }
                );
            }
        });
    }
}
```

Apps/admin/src/app/pages/products/products-form/products-form.component.html

```
<p-toast></p-toast>
<div class="admin-page">
    <p-card [header]="editmode ? 'Edit Product' : 'Add Product'" subheader="You can add or edit products here">
        <div class="grid my-5">
            <div class="col-12">
                <p-toolbar>
                    <div class="p-toolbar-group-left"></div>
                    <div class="p-toolbar-group-right">
                        <p-button
                            styleClass="p-button-success"
                            [label]="editmode ? 'Update' : 'Create'"
                            [icon]="editmode ? 'pi pi-pencil' : 'pi pi-plus'"
                            (click)="onSubmit()"
                        >
                        </p-button>
                        <p-button styleClass="p-button-danger mx-2" label="Cancel" icon="pi pi-arrow-circle-left" (click)="onGoBack()"> </p-button>
                    </div>
                </p-toolbar>
```

```
            </div>
        </div>
        <div class="grid">
            <div class="col-12">
                <form [formGroup]="form">
                    <div class="fluid formgrid grid">
                        <div class="col-4">
                            <label for="name">Name</label><br />
                            <input
                                id="name"
                                type="text"
                                formControlName="name"
                                class="text-base   text-color   surface-
overlay p-2 border-1 border-solid surface-border border-round appearance-
none outline-none focus:border-primary w-full"
                                pInputText
                            />
                            <small   *ngIf="productForm.name.invalid   &&
isSubmitted" class="p-error">Name is required!</small>
                        </div>
                        <div class="col-4">
                            <label for="brand">Brand</label><br />
                            <input
                                id="brand"
                                type="text"
                                formControlName="brand"
                                class="text-base   text-color   surface-
overlay p-2 border-1 border-solid surface-border border-round appearance-
none outline-none focus:border-primary w-full"
                                pInputText
                            />
                            <small   *ngIf="productForm.brand.invalid   &&
isSubmitted" class="p-error">Brand is required!</small>
                        </div>
                        <div class="col-4">
                            <label for="price">Price</label><br />
                            <p-inputNumber        formControlName="price"
mode="decimal" inputId="price" [useGrouping]="false"></p-inputNumber>
                            <small   *ngIf="productForm.price.invalid   &&
isSubmitted" class="p-error">Price is required!</small>
                        </div>
                    </div>
                    <br />
                    <div class="fluid formgrid grid">
                        <div class="col-4">
                            <label      for="countInStock">Count      in
Stock</label><br />
                            <p-inputNumber formControlName="countInStock"
mode="decimal"     inputId="countInStock"     [useGrouping]="false">></p-
inputNumber>
```

```
                            <small
*ngIf="productForm.countInStock.invalid && isSubmitted" class="p-error">A
number is required!</small>
                        </div>
                        <div class="col-4">
                            <label for="category">Category</label><br />
                            <p-dropdown
                                [options]="categories"
                                [filter]="true"
                                filterBy="name"
                                [showClear]="true"
                                placeholder="Select a Category"
                                formControlName="category"
                                optionLabel="name"
                                optionValue="id"
                                [style]="{ width: '100%' }"
                            ></p-dropdown>
                            <small *ngIf="productForm.category.invalid &&
isSubmitted" class="p-error">Category is required!</small>
                        </div>
                        <div class="col-4">
                            <label   for="isFeatured">Featured</label><br
/>
                            <p-inputSwitch
formControlName="isFeatured"></p-inputSwitch>
                        </div>
                        <div class="col-12 my-3">
                            <label
for="description">Description</label><br />
                            <textarea      pInputTextarea      [rows]="7"
style="width: 100%" formControlName="description"></textarea>
                            <small *ngIf="productForm.description.invalid
&& isSubmitted" class="p-error">Description is required!</small>
                        </div>
                        <div class="col-12">
                            <label          for="richDescription">Product
Details</label><br />
                            <p-editor   formControlName="richDescription"
[style]="{ height: '320px' }"></p-editor>
                        </div>
                        <div class="col-12 my-3">
                            <label for="name">Main Image</label><br />
                            <input    type="file"    class="p-inputtext"
accept="image/*" (change)="onImageUpload($event)" />
                            <div class="col-2 my-3 pl-0">
                                <img  [src]="imageDisplay"  style="width:
100%" alt="" />
                                <small   *ngIf="productForm.image.invalid
&& isSubmitted" class="p-error">Image is required!</small>
                            </div>
                        </div>
```

```
                    </div>
                </form>
            </div>
        </div>
    </p-card>
</div>
```

Apps/admin/src/app/pages/products/products-form/products-form.component.ts

```
import { Location } from '@angular/common';
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute } from '@angular/router';
import { CategoriesService, Product, ProductsService } from '@my-team/products';
import { MessageService } from 'primeng/api';
import { timer } from 'rxjs';

@Component({
    selector: 'admin-products-form',
    templateUrl: './products-form.component.html',
    styles: []
})
export class ProductsFormComponent implements OnInit {
    editmode = false;
    form: FormGroup;
    isSubmitted = false;
    categories: any = [];
    imageDisplay: string | ArrayBuffer;
    currentProductId: string;

    constructor(
        private formBuilder: FormBuilder,
        private location: Location,
        private categoriesService: CategoriesService,
        private productsService: ProductsService,
        private messageService: MessageService,
        private route: ActivatedRoute
    ) {}

    ngOnInit(): void {
        this._initForm();
        this._getCategories();
        this._checkEditMode();
    }

    onSubmit() {
        this.isSubmitted = true;
        if (this.form.invalid) return;
```

```
        const productFormData = new FormData();

        Object.keys(this.productForm).map((key) => {
            productFormData.append(key, this.productForm[key].value);
        });
        if (this.editmode) {
            this._updateProduct(productFormData);
        } else {
            this._addProduct(productFormData);
        }
    }

    private _initForm() {
        this.form = this.formBuilder.group({
            name: ['', Validators.required],
            brand: ['', Validators.required],
            price: ['', Validators.required],
            category: ['', Validators.required],
            countInStock: ['', Validators.required],
            description: ['', Validators.required],
            richDescription: [''],
            image: ['', Validators.required],
            isFeatured: [false]
        });
    }

    private _getCategories() {
        this.categoriesService.getCategories().subscribe((categories)  =>
{
            this.categories = categories;
        });
    }

    private _addProduct(productData: FormData) {
        this.productsService.createProduct(productData).subscribe(
            (product: Product) => {
                this.messageService.add({
                    severity: 'success',
                    summary: 'Success',
                    detail: `Product ${product.name} is created!`
                });
                timer(2000)
                    .toPromise()
                    .then(() => {
                        this.location.back();
                    });
            },
            () => {
                this.messageService.add({
                    severity: 'error',
```

```
                    summary: 'Error',
                    detail: 'Product is not created!'
                });
            }
        );
    }

    private _updateProduct(productFormData: FormData) {
        this.productsService.updateProduct(productFormData,
this.currentProductId).subscribe(
            () => {
                this.messageService.add({
                    severity: 'success',
                    summary: 'Success',
                    detail: 'Product is updated!'
                });
                timer(2000)
                    .toPromise()
                    .then(() => {
                        this.location.back();
                    });
            },
            () => {
                this.messageService.add({
                    severity: 'error',
                    summary: 'Error',
                    detail: 'Product is not updated!'
                });
            }
        );
    }

    private _checkEditMode() {
        this.route.params.subscribe((params) => {
            if (params.id) {
                this.editmode = true;
                this.currentProductId = params.id;

this.productsService.getProduct(params.id).subscribe((product) => {
                    this.productForm.name.setValue(product.name);

this.productForm.category.setValue(product.category?.id);
                    this.productForm.brand.setValue(product.brand);
                    this.productForm.price.setValue(product.price);

this.productForm.countInStock.setValue(product.countInStock);

this.productForm.isFeatured.setValue(product.isFeatured);

this.productForm.description.setValue(product.description);
```

```
this.productForm.richDescription.setValue(product.richDescription);
                    this.imageDisplay = product.image as string;
                    this.productForm.image.setValidators([]);
                    this.productForm.image.updateValueAndValidity();
                });
            }
        });
    }

    onGoBack() {
        this.location.back();
    }

    onImageUpload(event: any) {
        const file = event.target.files[0];
        if (file) {
            this.form.patchValue({ image: file });
            this.form.get('image')?.updateValueAndValidity();
            const fileReader = new FileReader();
            fileReader.onload = () => {
                this.imageDisplay = fileReader.result as string;
            };
            fileReader.readAsDataURL(file);
        }
    }

    get productForm() {
        return this.form.controls;
    }
}
```

Додаток Б

ВІДГУК

Керівника економічного розділу

Додаток В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім'я файлу	**Опис**
Пояснювальні документи	
Диплом_Левченко.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_Левченко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Левченко.ppt	Презентація кваліфікаційної роботи