

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій

(повне найменування факультету)

Кафедра комп'ютерних систем та мереж

(повне найменування кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ СЕРВІСНОГО
ЦЕНТРУ З АВТОМАТИЗОВАНИМ УПРАВЛІННЯМ ТОВАРАМИ ТА
ЗАМОВЛЕННЯМИ**

Виконав: студент 4 курсу, групи КНТ-522
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерна інженерія»

ЛЬВОВ Р.Є.

(ПРІЗВИЩЕ та ініціали)

Керівник ЗЕЛЕНЬОВА І.Я.

(ПРІЗВИЩЕ та ініціали)

Рецензент ТЯГУНОВ Д.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерні системи та мережі
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.

“14” квітня 2026 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ЛЬВОВА Руслана Євгеновича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка інформаційної системи сервісного центру з автоматизованим управлінням товарами та замовленнями
керівник проєкту (роботи) к.т.н., доцент, ЗЕЛЕНЬОВА І.Я.

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом навчального закладу від «14» квітня 2026 року №160.

Строк подання студентом проєкту (роботи) 01.06.2026

3. Вихідні дані до проєкту (роботи) опис предметної області, технології розробки клієнтської та серверної частини

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз предметної області

2) Порівняння існуючих інформаційних систем

3) Розробка інформаційної системи вебзастосунку інтернет-магазину та сервісного центру

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	Зеленьова І. Я., доцент		
Нормоконтроль	Голуб Т.В., доцент		

7. Дата видачі завдання 01.04.2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області та теоритичні відомості, що знадобляться під час роботи	15.04.2026 р.	
2	Огляд та порівння аналогів	19.04.2026 р.	
3	Постановка завдань на проєктування	21.04.2026 р.	
3	Розробка інформаційної системи	05.05.2026 р.	
4	Оформлення отриманих результатів у ПЗ	20.05.2026 р	
5	Оформлення презентації	01.06.2026 р	

Студент (ка)

(підпис)

Руслан ЛЬВОВ

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Ірина ЗЕЛЕНЬОВА

(Ім'я ПРИЗ

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
58с., 1 табл., 25 рис., 19 джерел, 3 додатки.

ВЕБЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, СЕРВІСНИЙ ЦЕНТР,
NODE.JS, TYPESCRIPT, MYSQL, EXPRESS.JS, SEQUELIZE,
АВТОМАТИЗАЦІЯ ЗАМОВЛЕНЬ, ОБРОБКА ДАНИХ

Об'єкт дослідження – процеси продажу електронної техніки, обробки замовлень, ведення обліку товарів та сервісного обслуговування мобільних пристроїв.

Предмет дослідження – система автоматизації продажу електронної техніки та сервісного обслуговування клієнтів магазину і сервісного центру.

Мета роботи – розробка інформаційної системи магазину електронної техніки та сервісного центру з обробкою товарів, замовлень та сервісних заявок.

У роботі проаналізовано сучасні інтернет-магазини електронної техніки, досліджено технології створення вебзастосунків, серверної частини та баз даних. Розроблено структуру вебсистеми, API для роботи з товарами та замовленнями, а також реалізовано взаємодію між клієнтською та серверною частинами застосунку.

Практична цінність – реалізовано вебсайт з модальним відкриттям товарів без перезавантаження сторінки та системою перевірки HTTP-запитів для підвищення безпечності роботи вебресурсу.

ЗМІСТ

Скорочення та умовні позначки	6
Вступ	7
1 Аналітичний огляд.....	8
1.1 Опис предметної області.....	8
1.2 Постановка задачі розробки.....	10
1.3 Огляд аналогів	13
2. Вибір та реалізація структури системи, яка проектується.....	17
2.1 Вибір мови програмування та технологій розробки.....	17
2.2 Архітектура та алгоритм роботи системи.....	18
2.3 Реалізація системи пошуку товарів.....	23
2.4 JWT - авторизація	25
2.5 Хешування та система захисту	26
3. Тестування та запуск інформаційної системи	17
3.1 Вибір мови програмування та технологій розробки.....	17
3.2 Архітектура та алгоритм роботи системи.....	18
3.3 Реалізація системи пошуку товарів.....	23
3.4 JWT - авторизація	25
3.5 Вхід та реєстрація користувача.....	37
3.6 Панель адміністратора.....	39
3.7 Блок додавання товарів.....	41
Висновки.....	43
Перелік джерел посилання	45
Додаток А Схеми обробки заявок	46
Додаток Б Лістинги основних модулів програми	49
Додаток В Слайди презентації.....	56

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	– База даних
TSS	– Trigram Similarity Search, Пошук за триграмною схожістю
JWT	– Json Web Token, Токен автентифікації користувача
API	– Application Programming Interface, Інтерфейс програмування застосунків
ПК	– Персональний комп'ютер
CSS	– Cascading Style Sheets, Каскадні таблиці стилів
HTML	– HyperText Markup Language, Мова гіпертекстової розмітки
JS	– JavaScript, Мова програмування для створення інтерактивних вебзастосунків
TS	– TypeScript, Типізоване розширення мови JavaScript
React	– JavaScript-бібліотека для створення користувацьких інтерфейсів
Node.js	– Серверне середовище виконання JavaScript
HTTP	– Graphical User Interface, Графічний інтерфейс користувача
IDE	– Integrated Development Environment, Інтегроване середовище розробки

ВСТУП

На сьогоднішній день інформаційні технології займають невід’ємне місце у повсякденному житті більшості людей. Завдяки мережі Інтернет людина має можливість дізнатися багато інформації за лічені секунди, здійснити онлайн-покупки в будь-якому магазині світу. Саме тому інтернет-технології стали необхідною складовою роботи сучасних підприємств і магазинів.

Сфера інтернет-магазинів активно розширюється та розвивається з кожним роком, стаючи зручнішою й популярнішою серед користувачів. Онлайн-платформи дозволяють доволі швидко побачити весь асортимент товарів, порівняти характеристики та ціну й оформити замовлення, не виходячи з дому.

На відміну від звичайних магазинів, веб-сайт може працювати цілодобово, що дає тисячам клієнтів змогу одночасно переглядати вигідні пропозиції та робити покупки в будь-який зручний час. Також ще однією перевагою є те, що можна одночасно обрати кілька товарів і порівняти їхні характеристики, а також подивитися відгуки інших користувачів про кожен із них.

Для власників бізнесу з продажу це також є ефективним рішенням, адже створення такого середовища автоматизує багато процесів. За допомогою сучасних систем можна легко вести облік продажів, контролювати наявність і кількість товарів на складі, швидко обробляти замовлення та аналізувати статистику відвідувань. Ці фактори дозволяють зменшити витрачений час на оформлення, запобігти помилкам і підвищити продуктивність компанії.

Метою даної дипломної роботи є розробка вебсайту магазину техніки, яка є актуальною в сучасних умовах розвитку електронної комерції, оскільки значна кількість користувачів здійснює пошук та придбання техніки саме через Інтернет. Такий вебресурс дає змогу швидко отримувати інформацію про товари, їхні характеристики, вартість і наявність у магазині, а також оформлювати замовлення без потреби відвідувати фізичний магазин.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Опис предметної області

Одним із основних інструментів для представлення інформації в мережі Інтернет є вебсайт.

Вебсайт — це програмно-інформаційний ресурс, який складається зі сукупності пов'язаних між собою сторінок, що розміщуються на сервері та мають спільне доменне ім'я. Вебсайт дає можливість користувачу отримати доступ до різних видів інформації, таких як текстова, графічна та мультимедійна.

Один вебсайт може складатися з головної сторінки та багатьох додаткових сторінок, пов'язаних між собою системою навігації. Усі сторінки вебсайту мають однакову тематику, дизайн і структуру.

Основна мета створення вебсайту — забезпечити легкий і зручний доступ користувачів до потрібної інформації, сервісів, товарів або цифрових послуг. Такі ресурси часто створюють і використовують для того, щоб автоматизувати бізнес-процеси, обробляти інформацію, спілкуватися з клієнтами або рекламувати товари компанії. У сучасному світі майже кожна компанія вже має свій вебресурс, адже це дає змогу ефективно вести діяльність підприємства та автоматизувати завдання.

Кожен сучасний вебсайт має мати сучасний, зручний і зрозумілий дизайн та інтерфейс користувача. Найважливішими частинами інтерфейсу вебсайту є: меню навігації, чітке інформаційне текстове наповнення та функціональні модулі.

Існує велика кількість різновидів вебсайтів. Такі, як інформаційні сайти, блоги, корпоративні ресурси, форуми та інтернет-магазини. Кожен із цих сайтів має своє призначення та певні особливості функціоналу й наповнення. Розглянемо детальніше деякі з них.

Інформаційні сайти — це вебресурси, головне завдання яких — надання користувачу інформації за запитом. Це можуть бути новини, довідкові матеріали, статті або історична інформація.

Блоги — це вебсайти, які постійно наповнюються інформацією у вигляді текстових повідомлень, фотографій або відеоматеріалів. Блог зазвичай веде одна людина або група авторів.

Корпоративні сайти — це вебресурси, головна задача яких: представляти компанію або підприємство в мережі Інтернет. Зазвичай на такому сайті міститься інформація про саму компанію, її діяльність, послуги, контакти та новини компанії.

Інтернет-магазин — це веб-сайт, який спеціалізується на продажу різних категорій товарів або послуг через мережу Інтернет. Такі сайти дають можливість користувачам переглядати каталог продукції, порівнювати різні товари за певною характеристикою, оформлювати замовлення та здійснювати оплату. Вони можуть містити широкий асортимент продукції — від електронної техніки до друкованих видань. Асортимент продукції інтернет-магазинів може включати в себе різноманітні категорії: електронну техніку, аксесуари, побутові товари, одяг, книги та багато іншого.

Вебсайт, розроблений у межах дипломної роботи, призначений для реалізації електронної техніки та аксесуарів і надання послуг з ремонту мобільних телефонів. Ресурс забезпечує користувачам можливість перегляду каталогу товарів, ознайомлення з технічними характеристиками продукції, отримання інформації щодо ремонтних послуг, а також оформлення замовлень через мережу Інтернет.

Також користувачі можуть швидко дізнаватися інформацію про вартість товарів, наявність продукції та умови ремонту без необхідності особисто відвідувати магазин. Саме тому вебсайти магазинів електроніки та сервісних центрів є актуальними і допомагають забезпечити швидку взаємодію між клієнтом і продавцем або сервісом.

1.2 Постановка задачі розробки

Метою даного дипломного проекту є розробка інформаційної системи сервісного центру з автоматизованим управлінням товарами та замовленнями, яка забезпечить зручну взаємодію користувачів із вебресурсом, а також автоматизацію процесів продажу техніки та обробки заявок на ремонт мобільних пристроїв. Даний вебресурс повинен спростити процес вибору товарів, отримання консультацій та подання заявки на ремонт техніки без необхідності особистого відвідування магазину клієнтом, а також забезпечувати працівнику зручне керування товарами, контроль замовлень, ведення обліку товарів на складі, а також спрощення адміністрування ресурсу.

Функціональні вимоги:

Пошук і сортування товарів: користувач повинен мати можливість здійснювати пошук і фільтрацію продукції за назвою, категорією та брендом. Система пошуку повинна підтримувати обробку неточного введення та орфографічних помилок під час введення запиту користувачем, відображаючи результат, найбільш відповідний введеному запиту.

Інформація про товари на вебсайті: система повинна підтримувати відкриття інформації про товар у модальному вікні без повного перезавантаження сторінки, що забезпечує швидку навігацію та зручнішу взаємодію покупця з вебсайтом. Кожен товар повинен мати опис, характеристики, відгуки, фотографії, вартість товару та доставки, а також інформацію про наявність на складі.

Оформлення покупки товару: користувачі повинні мати можливість переглядати каталог товарів, додавати продукцію до кошика, змінювати кількість обраних позицій, видаляти непотрібні товари з кошика, оформлювати замовлення та залишати контактні дані для подальшої комунікації з продавцем.

Особистий кабінет: сайт має надавати змогу користувачу авторизуватися та створити власний профіль.

Розділ ремонту техніки: вебсайт повинен містити сторінку з інформацією про ремонт мобільних телефонів, описом доступних послуг, орієнтовними цінами та можливістю залишити заявку на ремонт.

Вебсайт має складатися з двох частин: інтерфейсу користувача та панелі адміністратора.

Панель адміністратора: система повинна надавати адміністратору доступ до редагування каталогу товарів, оновлення інформації щодо кількості та ціни, контролю замовлень та керування роботою вебресурсу.

Нефункціональні вимоги:

Зручність та зрозумілість інтерфейсу: навігація сайту має бути простою й зрозумілою для користувачів з різним рівнем цифрових навичок.

Такий сайт повинен забезпечувати:

- перегляд і пошук товарів за обраними категоріями товарів або фільтрів;
- зручну навігацію та зрозумілий інтерфейс;
- швидке відображення потрібної інформації про товар без зайвого перезавантаження сторінки;
- можливість оформлення онлайн-замовлень з вибором варіантів отримання;
- додатковий розділ для замовлення послуг ремонту техніки;
- систему керування товарами та заявками на ремонт;
- стабільну та швидку взаємодію між клієнтською та серверною частинами сайту.

Продуктивність: вебсайт має забезпечувати швидке завантаження інформації на сторінці та зменшувати повторні переходи між сторінками для оновлення завдяки використанню динамічного оновлення контенту.

Безпека: особисті дані покупців та інформація про їхні замовлення повинні бути захищені від несанкціонованого доступу. Система має передбачати систему автентифікації користувачів, різні права доступу на сайті та шифрування даних.

Підтримка проєкту: архітектура системи повинна забезпечувати можливість оновлень і розширення функцій у подальшому, а також додавання нових позицій товарів без суттєвих змін у структурі програмного коду.

Локалізація: система повинна підтримувати інтерфейс українською мовою для всіх основних компонентів.

1.3 Огляд аналогів

Одним із аналогів інформаційної системи магазину електронної техніки є Rozetka. Цей вебресурс займає одне з перших місць за обсягом продажів товарів різних категорій, зокрема електронної техніки та аксесуарів. Rozetka надає змогу покупцю обрати товар, переглянувши каталог, здійснити пошук за певними фільтрами та оформити онлайн-покупку товару, який сподобався.

До переваг можна віднести те, що цей магазин має гарну репутацію серед покупців і користується довірою клієнтів під час вибору платформи для замовлень. На сайті доступна велика кількість товарних пропозицій, фільтр пошуку працює доволі непогано, а також є кабінет покупця з історією замовлень.

Однак цей вебсайт має певні недоліки. Через те, що пропозицій товарів дуже багато і вебсайт доволі популярний, деякі сторінки перевантажені великою кількістю рекламних та інформаційних банерів, що часом заважає зосередитися саме на цьому товарі та його характеристиках. Коли користувач відкриває обраний товар у каталозі, система завантажує нову сторінку, через що під час навігації каталогом товарів збільшується кількість переходів між сторінками. Окрім цього, немає прайс-листа сервісного обслуговування техніки; є лише функція гарантійного ремонту у постачальника або повернення товару. Поза межами гарантійного терміну сервісне обслуговування не надається. Головну сторінку сайту «Rozetka» наведено на рисунку 1.1.

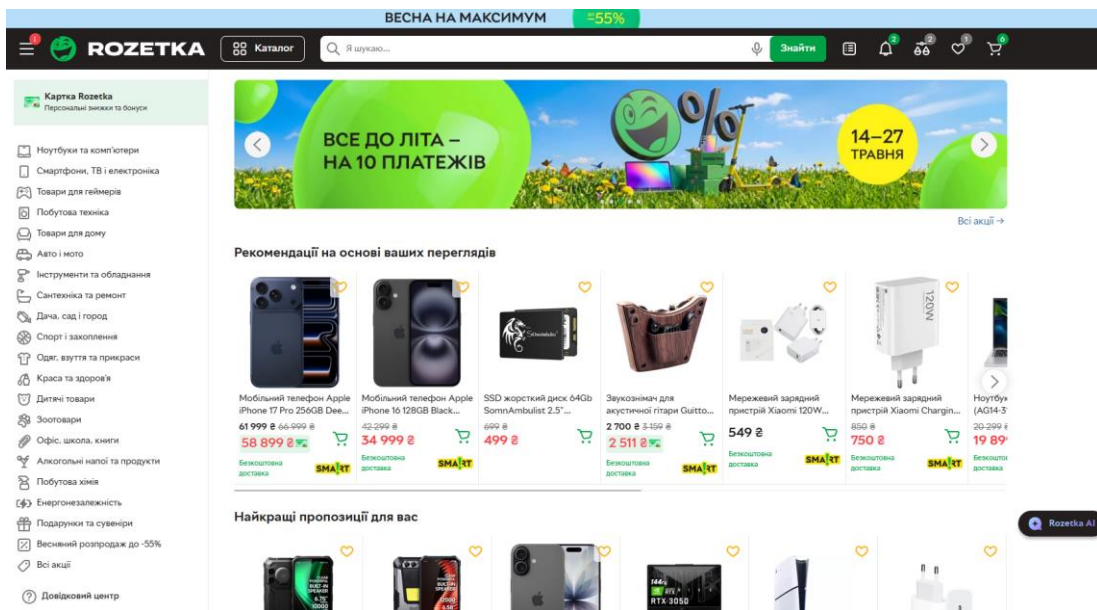


Рисунок 1.1 – Головна сторінка сайту «Rozetka»

Ще одним прикладом дуже добре відомого мені вебсайту в подібній сфері продажу є магазин «Aspor». Цей магазин спеціалізується саме на аксесуарах та запчастинах до техніки і має великий асортимент товарів. До переваг можна віднести зрозумілий інтерфейс, доволі просту та зрозумілу структуру сайту, показ кількості залишків саме цього товару на складі в цифрах та те, що майже весь асортимент є в наявності прямо зараз і немає потреби робити передзамовлення товару.

Серед недоліків можна відзначити недостатньо оптимізовану систему пошуку потрібного товару. Зокрема, пошук товарів здійснюється лише за повним збігом введеного запиту з назвою товару. У разі допущення користувачем навіть незначної орфографічної помилки система не відображатиме результати, що ускладнює пошук необхідної продукції та негативно впливає на зручність використання вебресурсу.

Також цей вебресурс має нижчу пропускну здатність роботи серверної частини сайту порівняно з іншими подібними інтернет-магазинами, і через це деякі сторінки завантажуються доволі довго. Також під час повернення товару в магазин гроші повертаються на бонусний рахунок, але в особистому кабінеті це не відображається, що часом викликає незручності для клієнта, адже він не пам'ятає свій баланс.

Головна сторінка сайту «Aspor» наведена на рисунку 1.2.

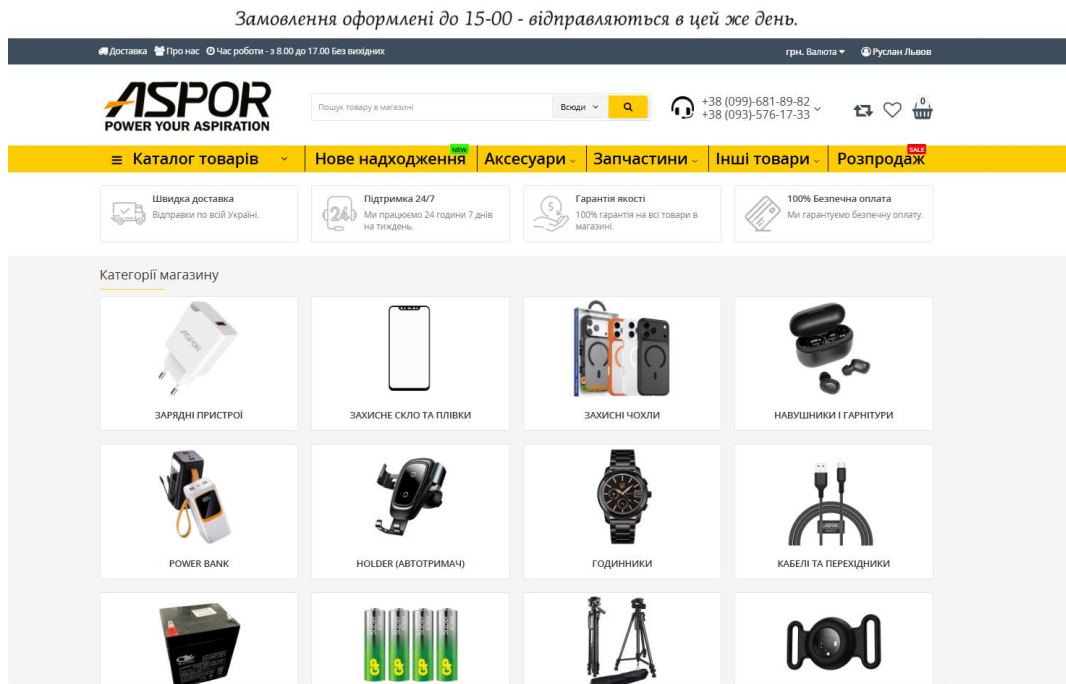


Рисунок 1.2 – Головна сторінка сайту «Aspor»

Також подібним аналогом є магазин «Stylus», який також спеціалізується на продажі комп'ютерної техніки, смартфонів та аксесуарів до них. Даний вебсайт має переваги, такі як: широкий асортимент, доволі непоганий опис товарів та їх характеристик, зручне оформлення каталогу товарів за категоріями. Непогано зроблено слайдер з промоакціями компанії. Також цей магазин надає послуги з ремонту та сервісного обслуговування техніки, зроблено прайс-лист послуг. Додатковою перевагою є те, що у продажу є вживана техніка, , адже це означає можливість купити за вигідною ціною телефон у непоганому стані.

З недоліків, можна відзначити — те, що на головній сторінці надто багато інформації. Якщо прокрутити сторінку донизу, то вона доволі довга, що впливає на час завантаження головної сторінки та на швидку орієнтацію користувача на ній. Крім того, велика кількість інформації на головній сторінці може відволікати покупця від потрібного пошуку товару. Головну сторінку сайту «Stylus» наведено на рисунку 1.3.

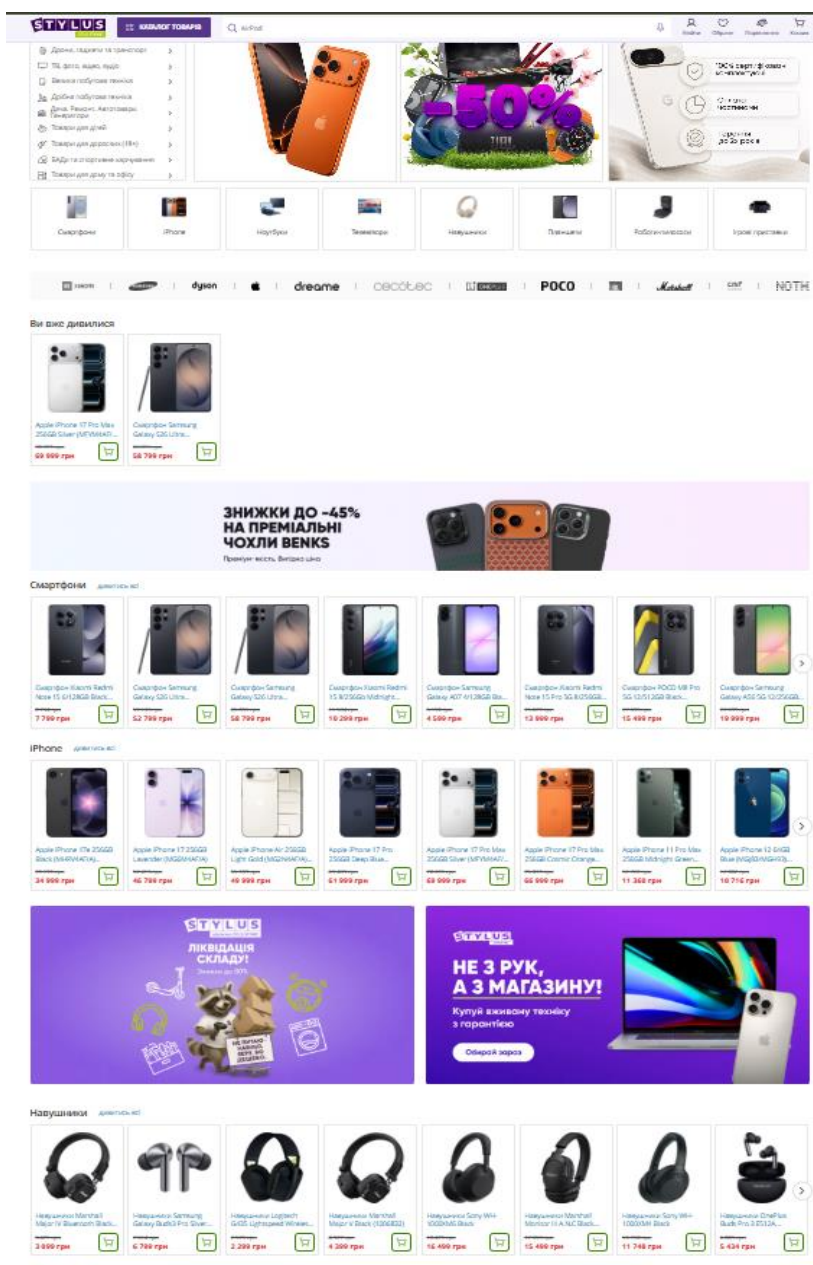


Рисунок 1.3 – Головна сторінка сайту «Stylus»

Наведемо таблицю (табл 1.1.) для порівняння вищенаведених сайтів, критерії порівняння оберемо такі:

- швидкість завантаження сторінок;
- модальне відкриття товарів;
- адаптація та якість пошукової системи;
- наявність сервісного ремонту;
- продаж вживаної техніки.

Таблиця 1.1 – Порівняння аналогів вебсайтів

Назва вебсайту	Швидкість завантаження сторінок	Модальне відкриття товарів	Якість системи пошуку	Наявність сервісного ремонту	Продаж вживаної техніки	Особливості вебресурсу
1	2	3	4	5	6	7
Rozetka	+	-	+	-	±	Великий асортимент товарів та довіра клієнтів
Stylus	±	-	+	+	+	Сервісне обслуговування
Aspor	-	-	-	-	-	Спеціалізація на мобільних аксесуарах та запчастинах
Розроблювана система	+	+	+	+	+	Стильний дизайн

Після проведення аналізу та порівняння цих вебсайтів можемо зробити висновок, що кожен з розглянутих вебресурсів мають свій достатньо широкий функціонал та надають можливість користувачам переглядати товари, шукати саме те, що їм цікаво завдяки пошуку та фільтрам та оформляти інтернет-замовлення не виходячі з будинку. Кожен із наведених магазинів має певні особливості, що відрізняють його від інших, а також свої переваги та недоліки, які впливають на комфорт під час пошуку та замовлення, а також на швидкість взаємодії покупця та сервера вебсайту.

Побачивши, що більш великі інтернет-магазини мають значно ширший асортимент товарів і функціонал, проте певні вебресурси також можуть бути перенавантажені інформацією та рекламними банерами й мати велику кількість переходів користувача між сторінками. І загалом усі ці фактори можуть впливати на комфорт користувача та збільшувати час очікування завантаження сторінок.

Саме тому під час розробки вебсайту дипломної роботи основна увага буде звернута на створення оптимізацій інтерфейсу, швидкості та зручності пошукової системи, мінімізації переходів між сторінками та додавання розділу сервісного

центру в одну єдину систему вебресурсу, де буде можливість керувати товарами компанії та замовленнями клієнтів.

2 ВИБІР ТА РЕАЛІЗАЦІЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір мови програмування та технологій розробки

Для створення вебсайту магазину техніки та сервісного центру з автоматизованим управлінням товарами, замовленнями та заявками на ремонт необхідно правильно підібрати мову написання сайту, щоб усі частини проєкту працювали стабільно, швидко та забезпечували надійну взаємодію між серверною частиною та клієнтською.

Сучасні розробники вебзастосунків використовують різні мови програмування. Серед найпопулярніших — Python, JavaScript і TypeScript. Розглянемо кожен з цих мов.

Python — це універсальна, багатоцільова мова програмування, яка зазвичай використовується у веб-розробці, машинному навчанні та аналізі даних. Програми, написані на цій мові, починаються з невеличких скриптів і завершуються великими складними системами. Одним з невеличких мінусів цієї мови є те, що Python — не є повністю компільованою мовою, а належить до інтерпретованих мов. У більшості випадків під час створення веб-додатків на цій мові порівняно з іншими мовами вона є трохи повільнішою. Ця мова має доволі простий синтаксис та велику кількість бібліотек, які підійдуть для різних сфер розробки.

JavaScript — одна з найпоширеніших мов програмування. Ця мова не надає низькорівневого доступу до процесора або до пам'яті, тому що була створена саме для використання у браузері, тобто для вебдодатків і вебсайтів. JavaScript використовується для написання інтерактивних елементів сайтів, аналізу та

обробки дій користувача, а також для втілення динамічного оновлення сторінки без потреби повторно перезавантажувати сайт. Широко використовується для написання як клієнтської, так і серверної частини вебсайту.

TypeScript — це також мова програмування, яка є розширенням JavaScript і підтримує типізацію даних. Однією з найбільших переваг цієї мови є можливість створення більш структурованого та безпечного коду, що дозволяє покращити й спростити обслуговування вебзастосунку згодом. Також ще однією перевагою є підтримка інтеграції з фрейворками та бібліотеками. Серед таких: Angular, React і Vue.

Для реалізації дипломного проєкту було обрано JavaScript та TypeScript, адже ці мови добре підходять для написання сучасних вебзастосунків із використанням клієнт-серверної архітектури. Для написання клієнтської частини буде частково задіяно CSS для побудови структури сторінок і стилізації інтерфейсу користувача.

У процесі розробки інформаційної системи як основне середовище програмування було обрано редактор коду WebStorm. Вибір даного редактора обумовлений тим, що він має доволі великий функціонал інструментів для розробки. Крім того, всі важливі інструменти під час встановлення середовища вже завантажені, що дозволяє почати роботу над проєктом одразу.

2.2 Архітектура та алгоритм роботи системи

Важливу роль під час створення вебсайту відіграє правильно підібрана система зберігання, обробки та передачі даних між сервером і користувачем. Майже кожен сучасний сайт потребує використання бази даних і сервера. Особливу роль це відіграє саме під час створення інтернет-магазину, адже

потрібно забезпечити швидке завантаження інформації про товари, клієнтів і замовлення та обробити всі ці дані.

Для коректної та швидкої роботи вебсайту інтернет-магазину техніки й сервісного центру буде використовуватися база даних. База даних — це сховище взаємопов'язаної інформації, яка розподілена та організована певним чином для пошуку, обробки та аналізу цих даних. Існують різні системи керування базами даних, такі як PostgreSQL, SQLite, MySQL та інші. Кожна з цих систем має свої певні відмінності та особливості й використовується залежно від потреб проєкту.

Наприклад, PostgreSQL зазвичай використовують у достатньо великих проєктах, де буде велика кількість операцій із даними та значне навантаження на систему. SQLite більше підходить для невеличких проєктів, які зберігаються локально.

MySQL — це реляційна система керування базами даних, яка допомагає здійснювати збереження, обробку та пошук структурованої інформації. Ця система є однією з найпоширеніших серед веброзробників завдяки тому, що вона стабільна, забезпечує високу швидкість передавання даних і може одночасно обробляти велику кількість запитів.

У проєкті з розробки системи магазину електронної техніки та сервісного центру буде використано базу даних MySQL та частково PostgreSQL. MySQL буде використовуватися для зберігання інформації про товари, категорії цих товарів, дані про користувачів, інформацію про замовлення та заявки на ремонт смартфонів. База даних PostgreSQL з додатковим розширенням для організації пошуку товарів.

Загальна структура роботи інформаційної системи вебсайту магазину електроніки та сервісного центру наведена на рисунку 2.1. Ця схема відображає основні процеси взаємодії користувачів із вебзастосунком, обробку запитів сервером та обмін інформацією з базою даних.

Користувач взаємодіє із клієнтською частиною вебсайту: переглядає товари на сторінці, користується фільтрами та пошуком для більшої зручності, реєструє свій обліковий запис покупця, оформлює замовлення/заявку в сервісний

центр на ремонт техніки. Адміністратор, водночас, має можливість додавати товари, обробляти замовлення та бачити контактні дані покупця. Серверна частина забезпечує обробку запитів, перевірку даних та допомагає взаємодії між клієнтом та БД

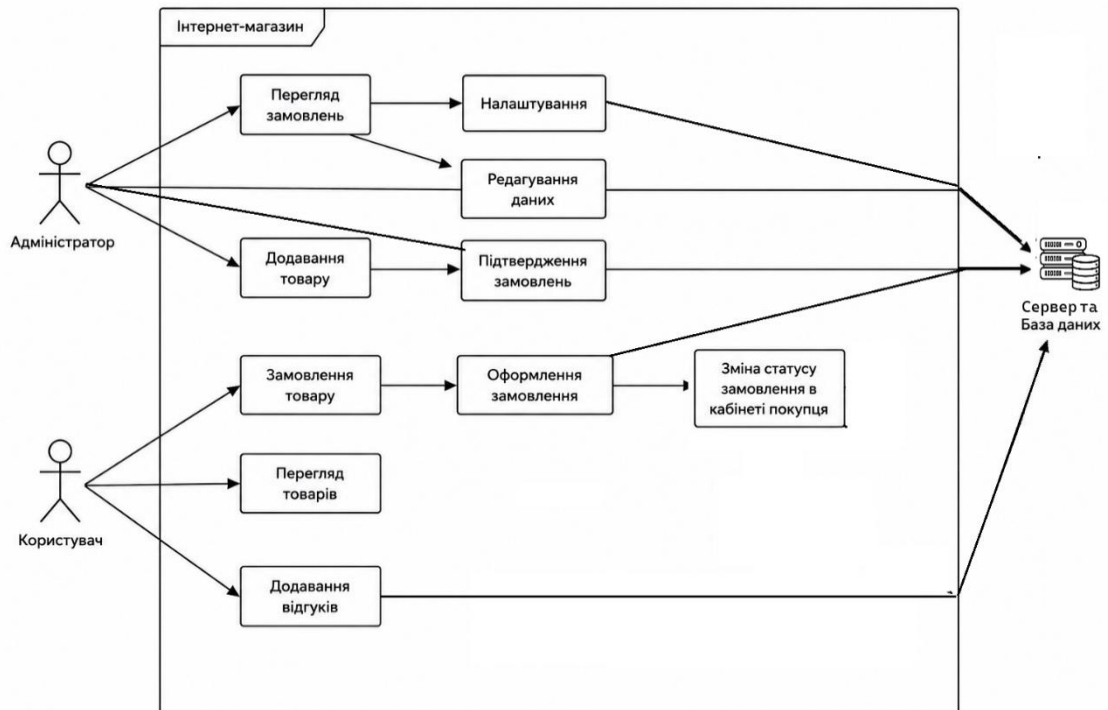


Рисунок 2.1 – Взаємодія клієнта, сервера та бази даних

Серверна частина проєкту буде реалізована за допомогою Node.js та Express.js. Node.js — це платформа, яка дозволяє створювати серверні вебзастосунки та обробляти безліч запитів користувачів безпосередньо на сервері. Express.js — це фреймворк для Node.js, який спрощує організацію серверної логіки та API. Серверна частина виступає посередником між користувачем і базою даних, забезпечуючи контрольований доступ до потрібної інформації та захист від несанкціонованого доступу до даних.

Додатково у проєкті буде використовуватися власна npm-бібліотека *inferpress*, яку було розроблено та яка вже активно використовується у робочих проєктах для спрощення процесу валідації та типізації HTTP-запитів у середовищі Express. *Inferpress* — бібліотека, яка дозволяє автоматично

перевіряти вхідні дані, виконувати перетворення типів і формувати TypeScript-типи на основі створених схем валідації. Вона дозволяє значно скоротити повтори коду та підвищити безпеку серверної частини вебсайту. Завдяки використанню бібліотеки Inferpress забезпечується автоматична перевірка параметрів body, query та params. Ці дії дозволяють уникати більшості помилок під час роботи з даними користувачів, а також спрощують подальшу підтримку серверного коду проєкту.

Нижче на рисунку 2.2 наведено приклад використання бібліотеки inferpress, а в Додатку В додано її лістинг.

На цьому прикладі показано, як цю бібліотеку було використано для автоматичної валідації параметрів типізації даних і HTTP-запиту. Перед початком виконання серверної логіки система перевіряє правильність отриманих даних відповідно до визначеної схеми. Параметр page має містити числове значення, параметр search приймає або текстовий рядок, або може мати значення null. Якщо виявляється невідповідність типів, бібліотека автоматично видасть помилку та припинить обробку цього запиту.

```

1  import { Router } from "express";
2  import { publicRoute, v } from "inferpress";
3
4  const router = Router();
5  // Маршрут отримання списку товарів
6  router.get("/",
7    // Схема валідації параметрів запиту
8    publicRoute({
9      query: v.object({
10        page: v.number(),
11        search: v.nullable(v.string())
12      }) },
13
14    // Обробка HTTP-запиту
15
16    async ({ query }) => {
17      // Повернення відповіді клієнту
18      return {
19        success: true,
20        data: query
21      }; }) );
22
23  export default router;

```

Рисунок 2.2 – Приклад використання бібліотеки inferpress

У межах даної дипломної роботи зазначена бібліотека використовується в серверній частині під час створення API-маршрутів, перевірки й обробки запитів користувачів, а також перевірки вхідних даних перед тим, як розпочнеться подальша їх обробка.

Підсумовуючи, можемо помітити, що бібліотека `inferpress` скорочує кількість допоміжного коду та допомагає спростити створення API. Це допомагає у розробці великих проєктів, де є потреба у великій кількості HTTP-маршрутів та у забезпеченні обробки даних користувачів вебсайту.

2.3 Реалізація системи пошуку товарів

Одна з найважливіших функцій, без якої зазвичай не розробляють жодного магазину, — це пошукова система. Від якості розробки цієї системи залежатиме комфорт під час пошуку потрібного товару та можливої покупки клієнтом, а також його загальне враження про магазин. Для покращення взаємодії потенційного покупця з системою інтернет-магазину та сервісного центру було реалізовано не тільки звичайний пошук за назвою товару, а й додано можливість нечіткого пошуку. Ця технологія має назву `Trigram Similarity Search` і дозволяє покупцю знаходити потрібний товар, навіть якщо він допустить орфографічну помилку під час введення назви.

Ця функція реалізована на серверній частині проєкту завдяки базі даних PostgreSQL і розширенню `pg_trgm`. Це розширення дає змогу порівнювати подібність текстових рядків.

Перед виконанням операції пошуку введений користувачем текст проходить обробку, під час якої прибираються зайві пробіли, а текст переводиться на нижній регістр (маленькі літери). Завдяки цим діям забезпечується точніше порівняння тексту, адже запобігається написанню запиту різними символами.

Коли введений текст оброблено, сервер формує SQL-запит через Sequelize із використанням функції `similarity()`. Ця функція порівнює подібність між текстом, який ввів користувач, і назвами товарів, що містяться в БД. Для реалізації такого пошуку використовувався код, наведений на рисунку 2.3.

```
const discountedPrice : Literal = Sequelize.literal(`val: `
  ROUND("products"."price" * (1 - COALESCE("products"."discount", 0) / 100.0))
`);

if (search && search.trim()) {
  const normalizedSearch : string = search.trim().toLowerCase();
  where[Op.and] = [
    ...(where[Op.and] || []),
    {[Op.or]: [
      Sequelize.where(Sequelize.fn( fn: "similarity",
        Sequelize.fn( fn: "LOWER", Sequelize.col( col: "products.name")), normalizedSearch
      ), Op.gte, rightOperand: 0.1
    )},
    {name: {[Op.iLike]: `${normalizedSearch}%`}}
  ]
}
];
}
```

Рисунок 2.3- Лістинг коду для реалізації функції Trigram Similarity Search

Коротко розглянемо реалізацію цього лістингу. Змінна `normalizedSearch` містить введений пошуковий запит користувача вже після попередньої обробки тесту. Така обробка може включати в себе приведення символів до єдиного регістру та видалення зайвих пробілів. Це дозволяє забезпечити коректніший пошук користувачу, незалежно від способу введення даних.

Функція `similarity` виконує пошук подібності з тими значеннями назв товарів, що є в БД. Далі отримується значення подібності 0.1 , і такий товар додається до результату пошуку. Також є оператор `iLike`, який відповідає за перевірку за частковим співпадінням символів, тобто, коли назву введено неповністю.

Після виконання цього процесу користувач отримає запит із результатами пошуку, навіть коли введено лише частину слова або з помилкою, адже система буде шукати найбільш схожі результати.

2.4 JWT-авторизація

Також ще одна з найважливіших технологій, яку обов'язково потрібно розробити, — це система захищеної авторизації. Через те, що на вебсайті є можливість реєструватися та робити замовлення, вказуючи персональні дані для зв'язку з клієнтом, в інформаційній системі під час розробки були використано JWT-авторизацію за допомогою бібліотеки `jsonwebtoken`, яка дає змогу ідентифікувати користувачів під час їхньої взаємодії з серверною частиною вебзастосунку. У моїй системі використовуються два токени : `access-токен` та `refresh-токен`.

Коли клієнт успішно входить у систему, створюється спеціальний `access-токен` доступу, у якому міститься додаткова інформація про цього користувача. Це допомагає потім ідентифікувати та підтвердити авторизацію під час подальших запитів до системи. Термін дії цього токена встановлено на 10 хвилин. Після завершення цього періоду користувач не зможе використовувати старий токен для виконання захищених запитів. Реалізацію даного токена в проєкті показано на рисунку 2.4 .

```
16
17   export function signAccess(payload: { userId: number; deviceId: number; }) {
18       return jwt.sign(payload, access_token, {expiresIn: "10m"});
19   }
20
```

Рисунок 2.4 - Лістинг коду для реалізації `access-токена`

Для подальшої роботи в системі використовується `refresh-токен`. Його завдання — автоматично оновлювати авторизацію в системі. На відміну від `access-токену`, термін його дії встановлено довше, адже `refresh-токен` має створювати новий `access-токен` без потреби повторного введення логіну та пароля. У цьому випадку `refresh token` буде активним протягом 30 днів.

Реалізацію цього токена в проєкті показано на рисунках 2.5 і 2.6.

```

9      const {access_token, refresh_token} = Env.jwt;
10     const REFRESH_TTL = "30d";
11     const tokens_db = db.users.users_tokens;

```

Рисунок 2.5 - Лістинг коду для реалізації часу дії refresh токена

```

20
21     export function signRefresh(payload: { userId: number; deviceId: number; jti: string; }) {
22         return jwt.sign(payload, refresh_token, {expiresIn: REFRESH_TTL});
23     }

```

Рисунок 2.6 - Лістинг коду для реалізації refresh токена

Також завдяки використанню deviceId реалізовано підтримку входу в особистий кабінет одночасно з декількох пристроїв, адже для кожного пристрою створюється окрема сесія та окремий refresh-токен.

Збереження активних refresh-сесій використовується у БД. Коли створюється нова сесія, інформацію про неї вносять до спеціальної таблиці. У таблиці зберігаються такі дані, як дані про користувача, id пристрою, ідентифікатор токена та дата завершення сесії.

Тому використання JWT-авторизації та токенів дозволило реалізувати додатковий захист серверної частини та безпечну систему авторизації користувачів на вебсайті, а також можливість мультипристроїв, де не потрібно постійно перезалогинюватися в акаунт, адже токени оновлюються автоматично

2.5 Хешування та система захисту

У розроблюваній інформаційній системі є потреба у введенні особистих даних клієнта, тому ці дані потрібно обов'язково захистити. Для цього була використана бібліотека bcrypt. Ця бібліотека використовується для створення захищеного хешу даних для входу.

Хешування — це метод захисту даних, за якого певна інформація перетворюється на спеціальний набір символів фіксованої довжини за допомогою математичного алгоритму. Особливістю цього процесу є те, що отриманий результат неможливо повернути до початкового вигляду. Саме тому хешування широко використовується для безпечного зберігання паролів та іншої конфіденційної інформації у вебзастосунках і базах даних.

Коли користувач зареєструвався, перед збереженням його пароля виконується хешування даних, зображене на рисунку 2.7. На відміну від звичайного шифрування, хешування є одностороннім процесом, і вже отриманий хеш неможливо повернути в пароль у звичайному вигляді стандартними методами, тому навіть адміністратор системи не має доступу до паролів клієнтів.

```
27      const hash = await bcrypt.hash(password, 12);  
28      const user = await db.users.users.create({  
29          ...user_data, password: hash,  
30      });
```

Рисунок 2.7 - Лістинг коду для реалізації хешування паролю у проєкті

Параметр з номером 12 визначає складність хешування. Чим більше це значення, тим захищенішим буде пароль і складніше його буде підібрати. Водночас, збільшення складності призводить до зростання навантаження на сервер та подовження часу обробки під час реєстрації або авторизації користувача на вебресурсі. Тому, значення цього параметру підбирається таким чином, щоб забезпечити баланс між безпекою та продуктивністю системи.

Розглянемо приклад того, як проходить хешування, взявши за основу пароль Qwerty123. Після застосування алгоритмів хешування його вигляд буде приблизно таким: \$2b\$12\$X7F5mQ9LrT2vY8nKcP1uOeJ4hD6sW3aBzN0xCqR7. У такому вигляді кеш-пароль записується в БД, і навіть якщо два користувачі мають однаковий пароль, завдяки хешуванню ці записи будуть виглядати по-різному.

2.6 Стилiзацiя та iнтерфейс користувача

Ще однією важливою складовою розробки вебсайту є його візуальна частина — веб-дизайн та сучасне оформлення сторінки.

Веб-дизайн — це частина розробки вебсайту, до завдань якої належать проектування та оформлення інтерфейсу користувача. Це важлива складова — те, що людина бачить першим, коли відкриває цей вебсайт.

Інтерфейс — сукупність різних графічних елементів, за допомогою яких у користувача є можливість взаємодіяти зі сторінкою сайту. Макети інтерфейсу користувача було розроблено та додано у Додаток В.

Для оформлення інтерфейсу вебсайту буде використано бібліотеку Styled Components. Вона дозволяє створювати CSS-стилі безпосередньо всередині компонентів React. Завдяки цьому кожен елемент інтерфейсу отримує власний набір стилів, який не впливає на інші частини та компоненти сайту. Такий підхід забезпечує локалізацію стилів і підвищує зручність підтримки коду під час його подальшого розвитку. Щоб підтримувати єдиний візуальний стиль сайту, будуть використані CSS-змінні, які застосовуються в різних компонентах інтерфейсу. Приклад їх використання наведено на рисунку 2.8.

```
1  @import url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap');
2  @import url('https://fonts.googleapis.com/css2?family=Manrope:wght@400;500;600;700;800&display=swap');
3
4  :root, html, body, #root {
5    /*font-family: "SF UI Display", sans-serif;*/
6
7    --font-ui: 'Inter', -apple-system, BlinkMacSystemFont,
8    'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell,
9    'Helvetica Neue', Arial, sans-serif;
10   --font-logo: 'Manrope', sans-serif;
11
12   font-family: var(--font-ui);
13
14   margin: 0;
15   padding: 0;
16   width: 100%;
17   height: 100%;
18   overflow: hidden;
19 }
```

Рисунок 2.8 - Лістинг коду для створення CSS-змінних

У разі необхідності змінити шрифт або якихось інших загальних параметрів, достатньо лише відредагувати один файл, і зміни застосуються автоматично до всіх компонентів системи.

Сайт складається з функціональних блоків, і кожна частина сайту є окремим React-компонентом. Зі сторони користувачів це можна побачити ніби як один єдиний вебресурс, однак насправді інтерфейс складається з окремих і незалежних блоків. Ці блоки можуть повторно використовуватися на різних сторінках сайту. Для зручності роботи з сайтом верхня панель закріплена, тому навіть під час прокручування сторінки вона завжди буде доступною зверху.

```
109
110   const Container = styled.header`
111     width: 100%;
112     height: 72px;
113
114     position: sticky;
115     top: 0;
116
117     z-index: 100;
118
119     display: flex;
120     justify-content: center;
121
122     backdrop-filter: blur(16px);
123
```

Рисунок 2.9 - Лістинг коду для створення верхньої панелі

Наведена частина коду на рисунку 2.9 реалізує позиціонування верхньої панелі. Завдяки властивості `position: sticky` вдалося забезпечити постійну видимість елементів навігації сторінки. Параметр `z-index` зі значенням 100 надає панелі пріоритет відображення, через що панель навігації розташовується поверх інших компонентів інтерфейсу.

Головна сторінка є центральною частиною сайту та має містити каталог товарів. Каталог буде побудований за принципом карткового відображення, де кожна картка товару при відкритті міститиме основну інформацію. Щоб правильно реалізувати каталог і розставити рядки товарів, буде використано технологію CSS Grid. CSS Grid — це технологія вистроювання двовимірних макетів сторінки, яка дозволяє краще розташовувати елементи у вигляді рядків і

стовпців. Лістинг коду для оформлення каталогу наведено на рисунку 2.10. Завдяки використанню функції `auto-fill` система автоматично визначатиме оптимальну кількість карток товарів у рядку, завдяки чому не треба робити горизонтальне прокручування.

Окрім каталогу, на головній сторінці розміщено елементи фільтрації товарів і рядок пошуку. Усі ці елементи було розроблено в єдиному стилі та інтегровано в єдину загальну концепцію дизайну вебсайту.

```

111
112     const ProductsGrid = styled.div`
113         display: grid;
114
115         grid-template-columns: repeat(auto-fill, minmax(220px, 1fr));
116
117         gap: 8px;
118     `;

```

Рисунок 2.10 - Лістинг коду для оформлення каталогу

Крім основних модулів, також зроблено кошик, список бажань, адміністративну панель і модуль авторизації користувача. Всі ці елементи стилізовані на основі `Styled Components`, що дозволило дотримуватися єдиного стилю в усіх частинах вебсайту.

3 ТЕСТУВАННЯ ТА ЗАПУСК ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Запуск серверу з сайтом

Для коректної роботи розроблюваної інформаційної системи магазину електронної техніки та сервісного центру її потрібно запустити в середовищі розробки. Щоб здійснити запуск системи необхідно відкрити проєкт та в терміналі середовища використовується команда `"npm run dev"`, після чого запуститься локальний сервер і сайт буде доступний за посиланням. Це зручно і

цілком достатньо для тестування коректності роботи різних функцій у проєкті, оскільки існує можливість швидко перевіряти та тестувати щойно додані функції та працювати над помилками. Доки сервер на ПК увімкнений, сайт буде працювати в браузері на тому ж комп'ютері, адже якщо закрити WebStorm, вебсайт перестане працювати.

Для постійної роботи інформаційної системи магазину електронної техніки та сервісного центру її потрібно розмістити на сервері або на хостингу. У межах тестування саме цього вебсайту його було розміщено на сервері однієї з компаній із відповідним дозволом, тому сайт може працювати без постійно ввімкненого ПК, на будь-якому пристрої в будь-який час. Для того щоб запустити систему і розмістити на сервері, потрібно було створити оптимізовану версію вебсайту. Це робиться після повного завершення роботи з проєктом командою в терміналі `"npm run build"`. Після чого вже готові файли проєкту мають бути завантажені на сервер і проєкт можна запускати.

3.2 Огляд головної сторінки вебсайту

Після відкриття вебсайту розроблюваної інформаційної системи магазину електронної техніки та сервісного центру в браузері з'явиться головна сторінка, представлена на рисунку 3.1.

У верхній частині сторінки ліворуч розміщено логотип магазину, його назва, кошик, вподобані товари та меню для переходу між магазином і сервісним центром. Інтерфейс сторінки було розроблено таким чином, щоб користувач міг легко та швидко знаходити необхідну інформацію та виконувати потрібні дії без зайвих переходів між сторінками.

Нижче представлений асортимент товарів та строка пошуку, а ліворуч — фільтри для зручнішого вибору потрібної категорії товарів. Доступні фільтри пошуку : за ціною та категорією товарів.

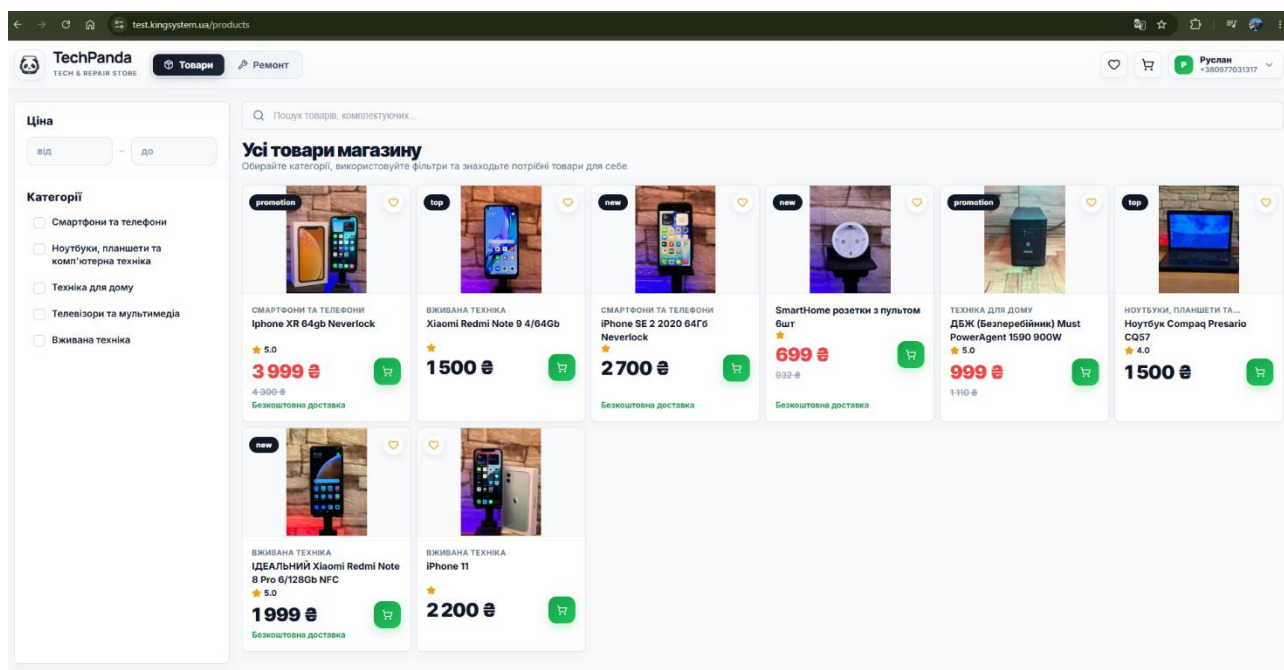


Рисунок 3.1 – Головна сторінка інтернет магазину «TechPanda»

При натисканні на певну категорію товарів відкриється список товарів, що належать до обраної категорії. Пошукова строка має функцію нечіткого пошуку, що дозволяє знайти потрібний товар, навіть якщо під час введення допущено помилку. Реалізація даної функції показано на рисунку 3.2.

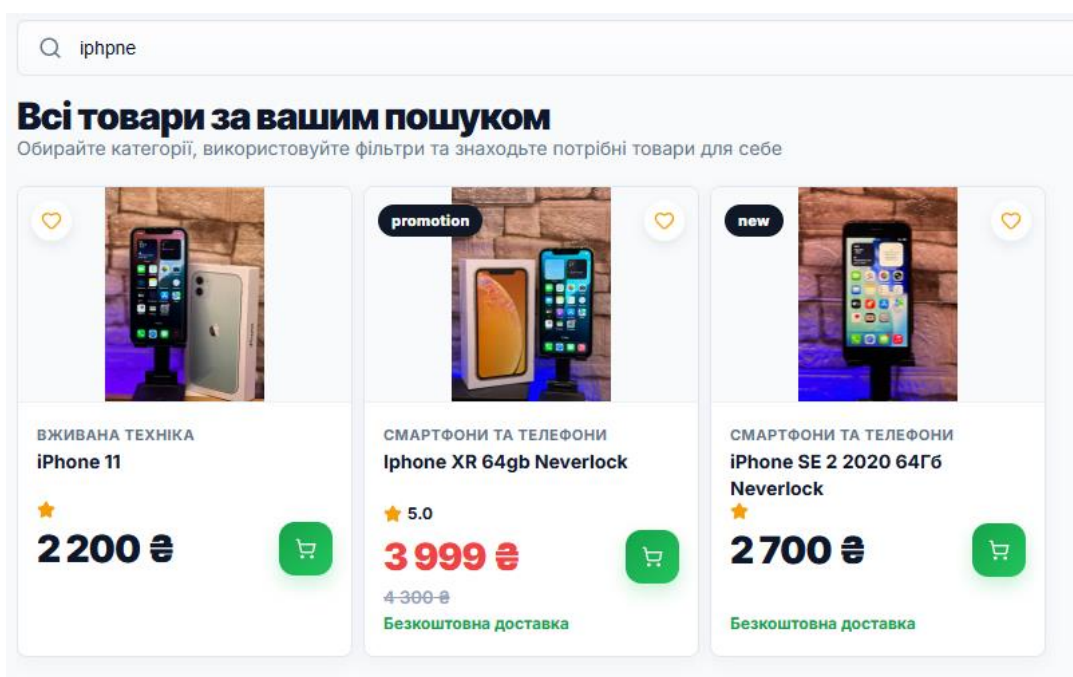


Рисунок 3.2 – Реалізація нечіткого пошуку

3.3 Створення замовлення товарів

Для того, щоб зробити замовлення, потрібно обрати товар і додати його в кошик. Після натискання на обраний товар відкриється модальне вікно без переходу на іншу сторінку з фотографіями, описом та відгуками про товар. Також доступні кнопки Додати до кошика та Додати у вподобані товари. На рисунку 3.3 показано модальне вікно з інформацією про товар.

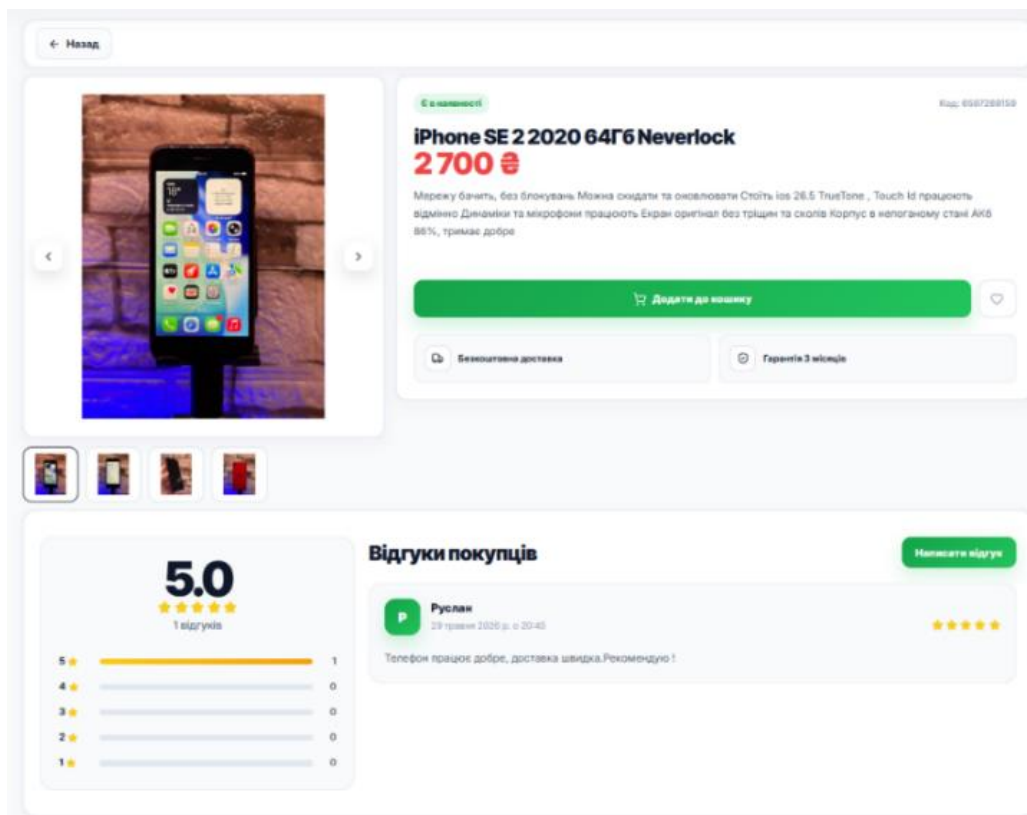


Рисунок 3.3 – Модальне вікно з інформацією про товар

У розробленому вебзастосунку реалізовано функцію кошика, яка дозволяє користувачам зручно формувати список товарів для подальшого оформлення замовлення. Коли натиснути на кнопку «Додавання в кошик», товар автоматично переміщується в кошик, де зберігається разом з іншими товарами. Є можливість змінювати кількість товару, видаляти непотрібні товари з кошика або оформлювати замовлення. При цьому, загальна вартість замовлення буде

автоматично перераховуватись, відповідно до внесених змін. У нижній частині сторінки знаходиться блок , де вказана загальна ціна товарів у кошику, а також кнопки «Очистити кошик» та «Зробити замовлення». Блок кошику представлено на рисунку 3.4.

Після натискання кнопки «Зробити замовлення» без перезавантаження вебсайту відкривається модуль із формою для заповнення даних про користувача та номер телефону, а також обирається зручний спосіб отримання замовлення й оплати.

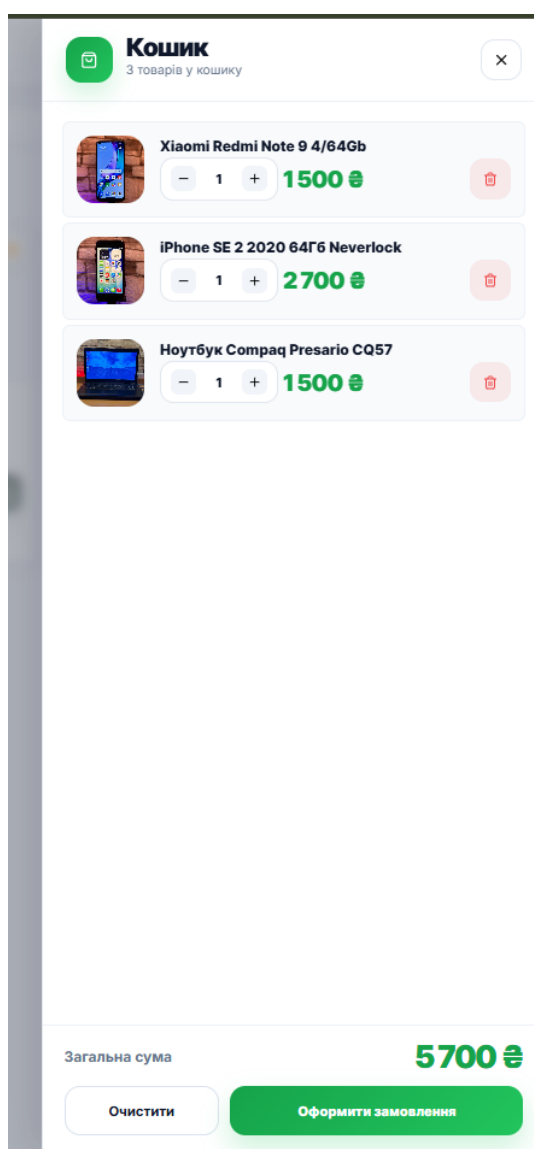


Рисунок 3.4 – Блок кошику

Реалізовано автоматичне заповнення даних із минулих замовлень, якщо покупець увійшов до початку оформлення замовлення у свій особистий кабінет.

Після того, як покупець перевірів введену інформацію, кількість вибраних товарів і обрав спосіб оплати, він натискає кнопку праворуч «Підтвердити замовлення». На цьому етапі оформлення замовлення завершено, і покупець очікує на зв'язок із менеджером. Весь процес оформлення замовлення показано на рисунку 3.5.

The screenshot displays the checkout interface of the TechPanda website, organized into two main columns.

Left Column (Form Fields):

- Контактні дані (Contact Details):** Includes fields for 'Ім'я та прізвище' (Name and Surname) with the value 'Руслан', 'Номер телефону' (Phone Number) with '+380 97 703 1317', and 'Email' with 'lvovruslan21@gmail.com'.
- Доставка (Delivery):** Offers two methods: 'Нова Пошта' (Nova Poshta) for delivery to a branch or mailbox, and 'Самовивіз' (Self-pickup) for free pickup from the store. Below these are fields for 'Місто' (City) set to 'Запоріжжя' and 'Відділення' (Branch) set to '14'.
- Оплата (Payment):** Offers 'Онлайн оплата' (Online payment) where the merchant provides payment data, and 'Оплата при отриманні' (Payment upon receipt) where the user pays after receiving the goods.

Right Column (Order Summary):

- Ваше замовлення (Your Order):** Lists 4 items:
 - iPhone XR 64gb Neverlock (1 x 3 999 ₴)
 - iPhone SE 2 2020 64Гб Neverlock (1 x 2 700 ₴)
 - Ноутбук Compaq Presario CQ57 (1 x 1 500 ₴)
 - ІДЕАЛЬНИЙ Xiaomi Redmi Note 8 Pro 6/128Gb NFC (1 x 1 999 ₴)
- Summary:**
 - Товари (Goods): 10 198 ₴
 - Сума знижки (Discount sum): 301 ₴
 - Доставка (Delivery): Тариф перевізника (Carrier rate)
 - До сплати (Total to pay): 10 198 ₴**
- Підтвердити замовлення (Confirm order):** A large green button with a right arrow.
- Натискаючи кнопку, ви підтверджуєте оформлення замовлення

Рисунок 3.5 – Блок замовлення

3.4 Створення заявки на сервісне обслуговування

TechPanda — це не тільки інтернет-магазин, а й додатково має можливість обслуговувати техніку та володіє власним сервісним центром. Завдяки цьому користувач може не лише придбати необхідний товар, але й звернутися до компанії у разі виникнення несправностей або необхідності проведення сервісного обслуговування.

Щоб користувач міг подати заявку на сервісне обслуговування техніки, потрібно на головній сторінці обрати розділ «Ремонт», після чого він перейде до модуля сервісного центру. Загальний вигляд цього модуля наведено на рисунку 3.6.

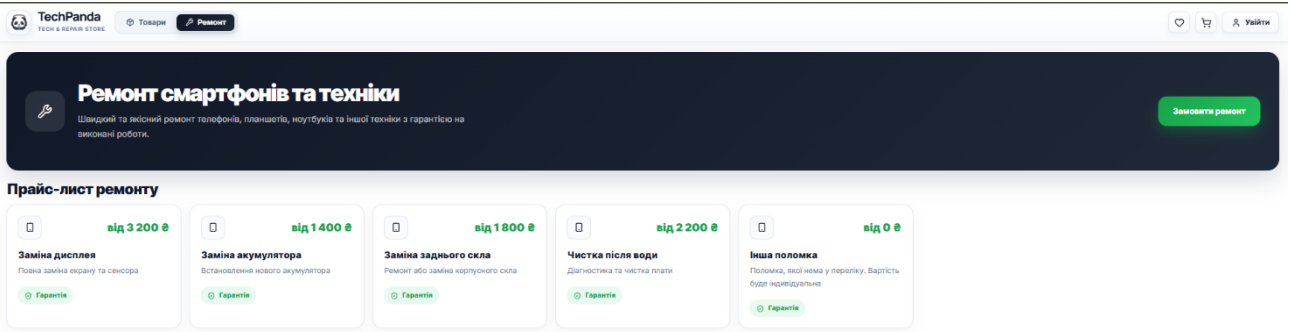


Рисунок 3.6 – Блок сервісного центру

Щоб подати заявку на ремонт, користувачу потрібно натиснути кнопку «Замовити ремонт». Автоматично відкриється модуль заповнення заявки, зображений на рисунку 3.7.

The image shows a mobile app interface for 'Замовлення ремонту' (Repair order). At the top, there's a title bar with a close button. Below it, the section 'Оберіть послуги ремонту' (Select repair services) contains five checkboxes with service names and prices: 'Заміна дисплея 3 200 ₴' (checked), 'Заміна акумулятора 1 400 ₴', 'Заміна заднього скла 1 800 ₴', 'Чистка після води 2 200 ₴' (checked), and 'Інша поломка 0 ₴'. Below this are input fields for 'ПІБ' (Full name) and 'Номер телефону' (Phone number). Then, a field for 'Назва/Модель пристрою' (Device name/model) with the example 'Наприклад iPhone 15 Pro Max'. A large text area for 'Опис поломки' (Describe the problem or symptoms...). At the bottom, a summary box shows 'Загальна вартість ремонту' (Total repair cost) as '5 400 ₴' (without delivery and additional diagnostics). A large green button 'Підтвердити замовлення ремонту' (Confirm repair order) is at the very bottom, with a small note below it: 'Після відправки заявки менеджер зв'яжеться з вами для уточнення деталей ремонту.' (After sending the request, the manager will contact you to clarify the details of the repair.)

Рисунок 3.7 – Форма для заповнення заявки сервісного обслуговування

У даній формі користувач має обрати необхідний вид обслуговування або ремонту, вказати інформацію про пристрій, описати несправність та заповнити контактні дані для подальшого зв'язку. Після успішного надсилання заявки інформація автоматично зберігається в базі даних системи і стає доступною для перегляду адміністраторами. Далі з клієнтом зв'яжеться менеджер, який спеціалізується на обслуговуванні техніки, і уточнить важливі моменти, а також зорієнтує щодо часу роботи.

Завдяки перевірці полів реалізовано механізм обов'язкового заповнення даних. Якщо не будуть заповнені обов'язкові поля, замовник не зможе надіслати заявку на обслуговування. Коли всі дані заповнені, кнопка автоматично стане доступною для натискання.

3.5 Вхід та реєстрація користувача

Реалізовано функціонал реєстрації та авторизації користувачів на вебсайті. Під час першого входу на сайт користувач реєструється. Для цього він заповнює форму, де вказує свої персональні дані. Туди входять ім'я, прізвище (за бажанням), номер телефону, електронна пошта, а також придумує пароль. Після заповнення рядки автоматично перевіряються на коректність введення.

Коли реєстрація пройшла успішно, дані про користувача зберігаються в БД, щоб він мав можливість знову заходити у свій профіль. Особлива увага була приділена безпеці даних користувача, тому такі дані зберігаються в хешованому вигляді, що унеможливорює їх крадіжку з БД та відновлення звичайними методами.

Під час наступного входу користувача введені дані також хешуються та порівнюються з хешем, збереженим у таблиці БД, що дозволяє безпечно перевірити коректність введеного пароля без його розшифровки у відкритому звичайному тексті.

Якщо дані були введені некоректно, система покаже повідомлення про помилку авторизації, наведене на рисунку 3.8. В іншому випадку, коли все було правильно введено, користувачу генерується та надається унікальний токен доступу. Він надалі зберігатиметься на стороні клієнта в браузері та передаватиметься разом із запитом на сервер. Такий підхід дозволяє спростити клієнту користування вебресурсом та прибирає потребу в постійному вході в систему під час дії токена, а також забезпечує більш надійний рівень безпеки ресурсу.

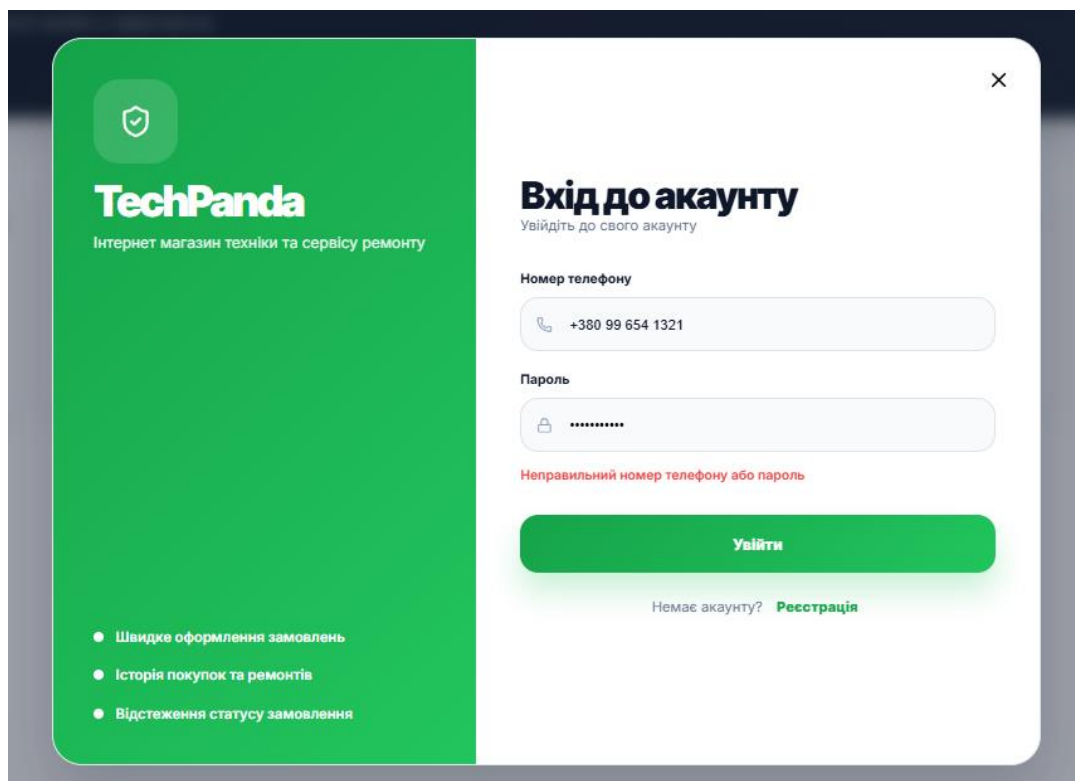


Рисунок 3.8 – Загальний вид модуля входу в систему та помилка авторизації

3.6 Панель адміністратора

Адміністративна панель інформаційної системи магазину техніки та сервісного центру наразі забезпечує контроль над замовленнями та зміну їх

статусу, додавання товарів, перегляд статистики замовлень і ремонтів кожного користувача, а також дає можливість призначати нових адміністраторів у системі.

Доступ до адміністративної панелі мають лише користувачі з відповідними правами. Надання прав адміністратора здійснюється адміністратором системи через спеціальний інтерфейс керування користувачами. Для надання доступу адміністратора потрібно лише обрати потрібного користувача та натиснути відповідну кнопку. Після цього обраний користувач отримує доступ до адміністративної панелі та всіх передбачених системою інструментів керування.

Процес призначення нового адміністратора продемонстровано на рисунку 3.9.

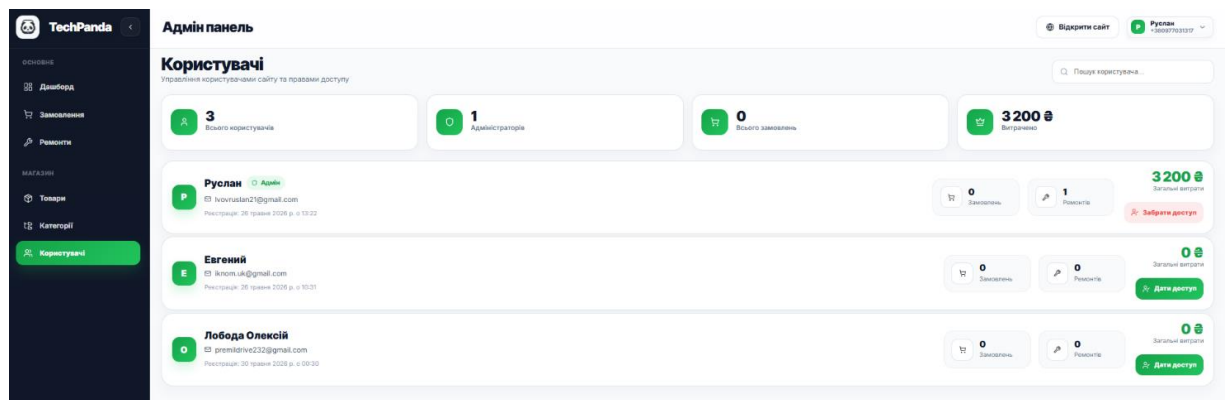


Рисунок 3.9 – Блок сайту з інформацією про користувачів та їх доступ

Дана панель також надає можливість переглядати детальну інформацію про кожного зареєстрованого користувача. На цій вкладці адміністратор може ознайомитися з кількістю оформлених замовлень, зверненнями до сервісного центру та загальною сумою коштів, витрачених користувачем у системі. Завдяки цій інформації є можливість аналізувати активність клієнтів, визначати найбільш активних покупців та оцінювати ефективність роботи магазину й сервісного центру.

Загальний вигляд панелі адміністратора з відкритою вкладкою «Замовлення» наведено на рисунку 3.10.

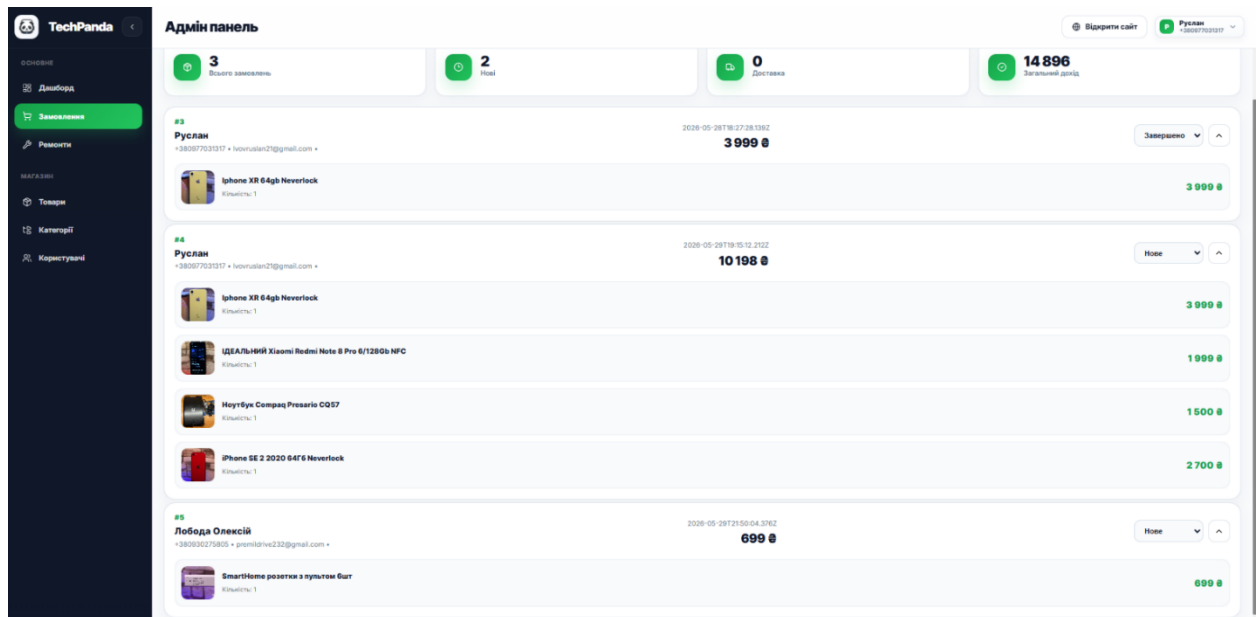


Рисунок 3.10 - Загальний вигляд панелі адміністратора

Адміністратор має можливість переглядати контактну інформацію про покупця та змінювати статус кожного замовлення залежно від етапу, на якому воно перебуває. Це дозволяє контролювати процес обробки замовлень та швидко взаємодіяти з клієнтами у разі необхідності уточнення деталей щодо замовлення.

Він може обрати один з варіантів статусу:

- нове замовлення;
- в обробці;
- доставка;
- завершено.

Так само інший менеджер має можливість змінювати статус сервісного обслуговування, адже замовлення на ремонт надходять автоматично в іншу вкладку адміністративної панелі. Це дозволило розділити процеси продажу товарів та сервісного обслуговування, забезпечуючи тим більш ефективну та зручну організацію роботи працівників. Вигляд модуля сервісного ремонту та варіанти статусу виконання замовлення наведено на рисунку 3.11

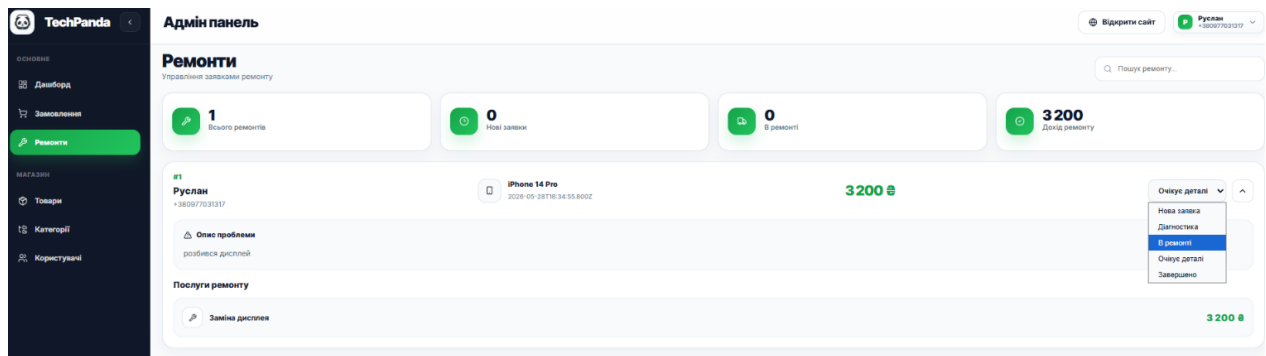


Рисунок 3.11 – Керування сервісним обслуговуванням

3.7 Блок додавання товарів

Щоб на сайті завжди були актуальні пропозиції товарів, їх потрібно регулярно додавати. Це можна зробити декількома варіантами. Або напямуч в базі даних — що довше й не дуже зручно. До того ж, такий підхід потребує спеціальних технічних знань та є менш зручним для щоденного використання. А також, пряме редагування даних у базі може підвищувати ризик виникнення помилок, які здатні вплинути на коректну роботу системи. Саме тому в межах розробленої інформаційної системи було реалізовано спеціальний модуль додавання товарів, який додано до адміністративної панелі вебзастосунку.

Доступ до функцій створення та редагування товарів мають виключно користувачі з правами адміністратора. Після авторизації в такому акаунті, він має перейти в розділ «Товари» та відкрити форму для створення нової позиції, показану на рисунку 3.12.

Реалізація такого механізму додавання дозволяє значно спростити процес адміністрування інтернет-магазину, надає можливість швидко оновлювати асортимент продукції і підтримувати актуальність інформації на вебресурсі, без необхідності прямого втручання в структуру БД або програмний код системи.

Створення товару
Новий товар магазину

Фотографії

Основна інформація

Назва товару
iPhone 11 64Gb

Ціна
5800

Кількість
0

Знижка
10

Гарантія
12

Категорія
Смартфони та телефони

Позначка
NEW

Доставка
Безкоштовна

Код товару
8978546

Опис
Корпус в дуже доброму стані
Неруху бачить, дзвонить
Wi-Fi працює
Батарея тримає заряд нормально, телефон сам по собі не вимикається
Face ID, TrueTone, NFC - працюють

Характеристики

Бренд
Apple

Пам'ять
64Gb

Колір
Білий

NFC
Є

Скасувати Створити товар

Рисунок 3.12 – Зразок заповненого блоку додавання товарів

Під час додавання є можливість вказати:

- назву товару;
- опис;
- характеристики товару (додавання характеристики та її значення);
- ціну;
- обрати категорію товару;
- вказати відсоток знижки;
- спосіб доставки;
- додати код товару;
- позначка товару (NEW, ТОП, АКЦІЯ);
- фотографії товару.

Коли під час додавання товару заповнено всі потрібні значення, сервер обробляє введені дані та створює новий запис у БД.

У розроблюваному вебзастосунку для зберігання фотографій товарів реалізовано окрему таблицю. Тобто самі фотографії товару не зберігаються

безпосередньо в базі даних; для цього використовується окрема таблиця `product_images`. Кожній фотографії призначається запис в таблиці та пов'язується з певним товаром. Це було зроблено за допомогою ідентифікатора `productId`.

Такий підхід корисний тим, що фотографії не містяться безпосередньо в БД, а мають там лише власне текстове поле з назвою файлу та його шляхом. Це є більш ефективним рішенням порівняно зі зберіганням зображень прямо в БД, тому що фотографії можуть займати багато місця й приторможувати роботу системи.

Коли відкривається блок з фотографією, система швидко отримує дані з бази, створює посилання на потрібний файл і передає його браузеру для відображення користувачу.

ВИСНОВКИ

У даній дипломній роботі на тему «Розробка інформаційної системи магазину електронної техніки та сервісного центру з автоматизованим управлінням товарами та замовленнями» було реалізовано вебзастосунок інтернет-магазину та сервісного центру для обслуговування смартфонів, що дозволяє користувачу даного вебсайту отримати необхідні послуги в межах однієї компанії. Ця інформаційна система поєднує в собі можливість продажу товарів, керування замовленнями, додавання нових товарних позицій, приймання заявок на сервісне обслуговування та автоматизацію звичайних бізнес-процесів компанії.

Перед початком розробки системи було проаналізовано аналоги з подібною сферою діяльності, досліджено особливості кожної з систем, зроблено висновки та сформовано вимоги щодо можливих покращень під час проєктування власного вебсайту.

Під час розробки системи було реалізовано клієнтські та серверну частини вебсайту, які поєднані між собою та забезпечують взаємодію між користувачем, сервером і базою даних.

Для потенційного покупця було розроблено сучасний і зручний каталог продукції, який має фільтри для вибору категорій і фільтрацію за ціновим діапазоном, систему пошуку з неточним пошуком, що дозволяє знаходити необхідні позиції навіть у разі неточного введення назви, а також безліч інших корисних функцій. Додано розділ сервісного обслуговування, що дозволяє переглянути ціни та варіанти ремонтних робіт, а за потреби залишити заявку на потрібне обслуговування.

У процесі розробки системи також було додано можливість створення особистого кабінету користувача, де зберігається контактна інформація клієнта. Для забезпечення безпеки зберігання персональних даних клієнтів було впроваджено автентифікацію за допомогою JWT-токенів, а також автоматичне подовження сесії за допомогою refresh-токенів. Паролі надійно зберігаються в

хешованому вигляді, тому навіть адміністратор системи не зможе отримати доступ до них.

Після завершення процесу розробки було перевірено працездатність функцій і модулів системи. Було перевірено працездатність функцій, таких як авторизація користувачів, модальне відкриття блоків вебсайту, можливість замовити товар або послугу сервісного обслуговування, а також функцій адміністративного керування інформаційною системою та обробкою замовлень.

Окрему увагу було приділено перевірці коректності роботи механізму модального відкриття блоків сайту, таких як кошик, оформлення замовлень, детальна інформація про товар та інші. У результаті було визначено, що поставлені завдання перед дипломною роботою виконані.

Створена інформаційна система інтернет-магазину та сервісного центру має практичну цінність. Адже в подальшому планується до використання в комерційній сфері після певних доопрацювань та інтеграцій, що допоможе перейти від сторонніх платформ до власної.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Sigma Software University. Стаття про БД URL: <https://university.sigma.software> (дата звернення 05.05.2026).

2 Wezom. Стаття про детальний огляд і розбір Node.js URL: <https://wezom.com.ua/ua/blog/vse-hto-nuzhno-znat-o-nodejs> (дата звернення 05.05.2026).

3 Guido van Rossum. Python Tutorial. Розділ 1 URL: <https://docs.python.org/uk/3/tutorial/appetite.html> (дата звернення 07.05.2026).

4 Стаття про визначення та опис мови Python URL: <https://pythonguide.rozh2sch.org.ua/> (дата звернення 07.05.2026).

5 Стаття з інформацією про хешування даних URL: <https://cyberset.com.ua/beginners/cryptography/data-hashing-methods> (дата звернення 07.05.2026).

David Flanagan. JavaScript Guide URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide%20> (дата звернення 08.05.2026).

6 Офіційний ресурс Bcrypt Node.js бібліотеки URL: <https://www.npmjs.com/package/bcrypt> (дата звернення 09.05.2026).

7 Офіційний ресурс PostgreSQL Documentation URL: <https://www.postgresql.org/docs/current/intro-what-is.html> (дата звернення 11.05.2026).

8 Niels Provos, David Mazières. A Future-Adaptable Password Scheme URL: <https://www.usenix.org/publications/library/provos.html> (дата звернення 15.05.2026).

9 Mozilla Developer Network. Document Object Model (DOM) URL: <https://developer.mozilla.org/en> (дата звернення 15.05.2026).

10 W3C. CSS Grid Layout Module Level 1 URL: <https://www.w3.org/TR/css-grid-1/> (дата звернення 16.05.2026).

11 Rachel Andrew. CSS Grid Layout URL: https://developer.mozilla.org/en-US/docs/Web/CSS/Guides/Grid_layout

12 Офіційний ресурс Introduction to JSON Web Tokens URL: <https://www.jwt.io/introduction#what-is-json-web-token> (дата звернення 1.05.2026).

13 OpenID Foundation. JSON Web Token (JWT) URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення 18.05.2026).

14 Access Tokens and Refresh Tokens URL: <https://auth0.com/docs/secure/tokens/access-tokens> (дата звернення 19.05.2026).

15 GitHub з кодом розроблюваної системи URL: <https://github.com/Ruslannn211/Site> (дата звернення 24.05.2026).

16 GitHub з кодом серверної частини URL: <https://github.com/Ruslannn211/Server> (дата звернення 24.05.2026).

17 GitHub з кодом бібліотеки inferpress URL: <https://github.com/Ruslannn211/inferpress2026> (дата звернення 01.06.2026).

18 Веб сайт розроблюваної інформаційної системи завантажений на сервер URL: <https://test.kingsystem.ua/products> (дата звернення 01.06.2026).

ДОДАТОК А

СХЕМИ ОБРОБКИ ЗАЯВОК

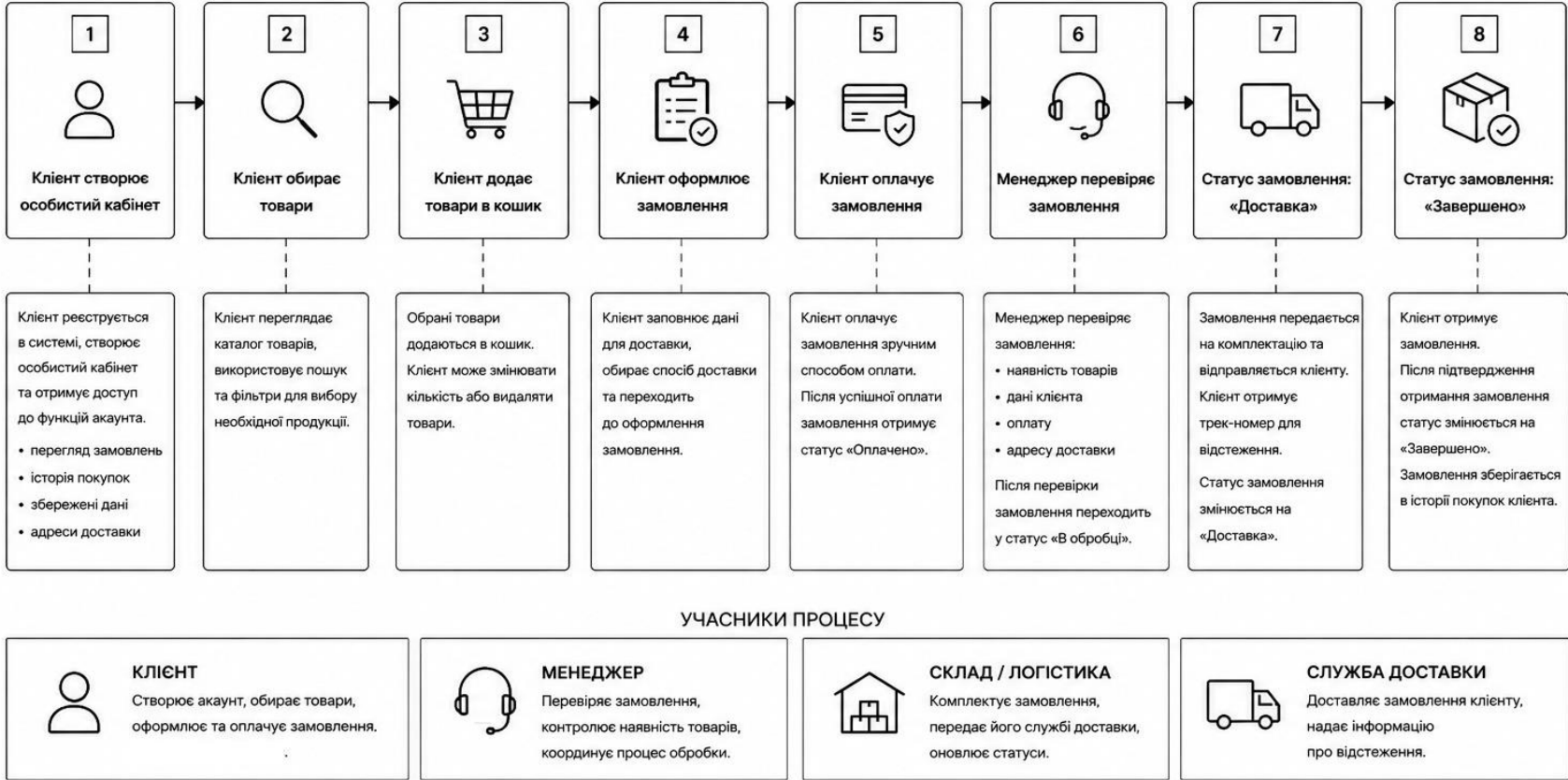


Рисунок А1– Схема обробки замовлення товару

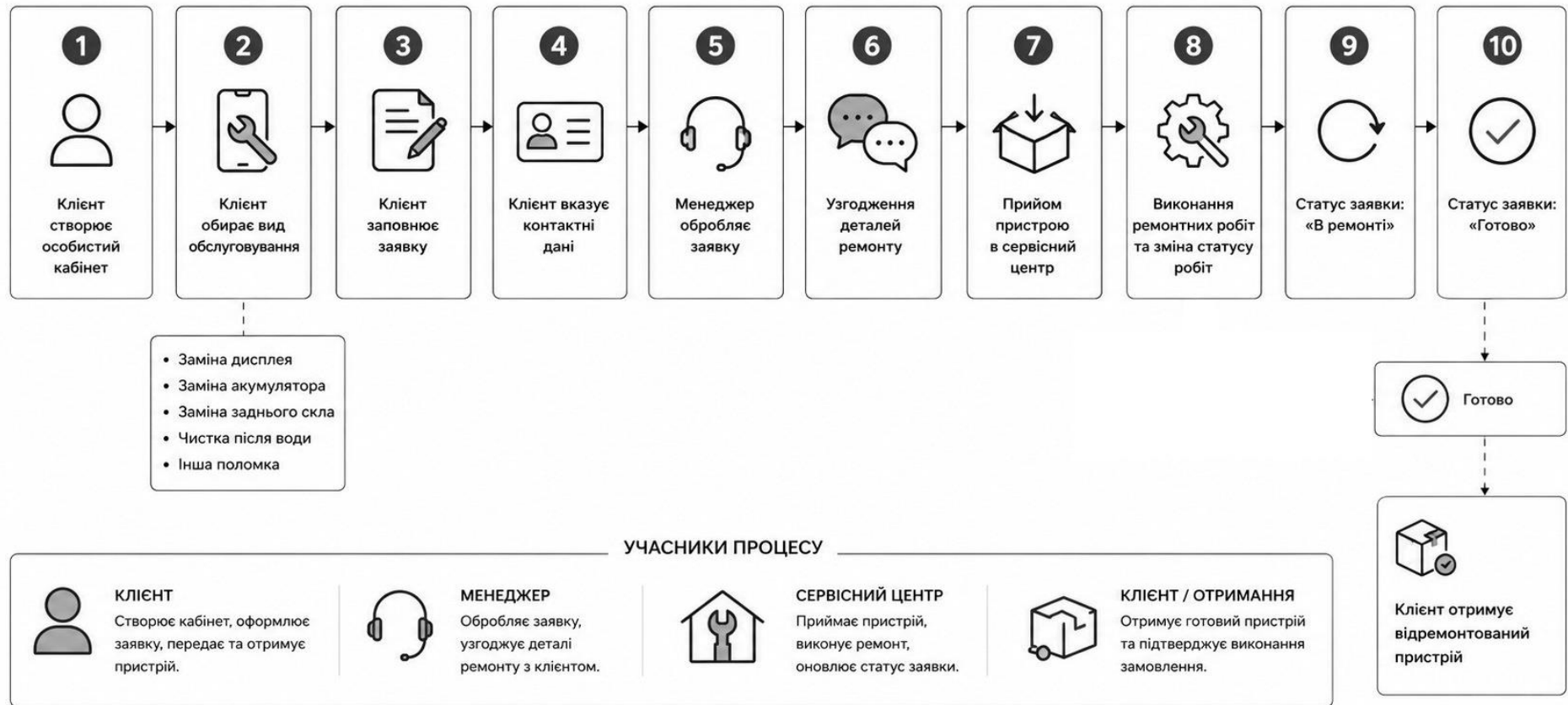


Рисунок А2 – Схема обробки заявки на ремонт

ДОДАТОК Б

ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ ПРОГРАМИ

Лістинг Б.1 - Реалізація механізму створення схем валідації

```

export enum DataTypes {
  STRING = "string",
  NUMBER = "number",
  BOOLEAN = "boolean",
  OBJECT = "object",
  NULLABLE = "nullable",
  ARRAY = "array",
  BLOB = "blob",
  ANY = "any",
}
export const v = {
  string: () => ({
    type: DataTypes.STRING,
  }) as const,
  number: () => ({
    type: DataTypes.NUMBER,
  }) as const,
  boolean: () => ({
    type: DataTypes.BOOLEAN,
  }) as const,
  any: (): SchemaType => ({
    type: DataTypes.ANY,
  }),
  nullable: <T extends SchemaType> (
    value: T
  ) => ({
    type: DataTypes.NULLABLE,
    value,
  }),
  array: <T extends SchemaType> (
    items: T
  ) => ({
    type: DataTypes.ARRAY,
    items,
  }),
  object: <T extends Record<string, SchemaType>> (
    schema: T
  ) => ({
    type: DataTypes.OBJECT,
    schema,
  }),
};

```

Лістинг Б.2 - Автоматичне формування типів TypeScript на основі описаної схеми валідації без дублювання коду

```
import {DataTypes, SchemaType} from "../schema";
export type InferType<S> =
  S extends { type: DataTypes.STRING }
    ? string
    : S extends { type: DataTypes.NUMBER }
      ? number
      : S extends { type: DataTypes.BOOLEAN }
        ? boolean
        : S extends {
            type: DataTypes.NULLABLE;
            value: infer V;
          }
          ? InferType<V> | null
          : S extends {
              type: DataTypes.ARRAY;
              items: infer I;
            }
            ? InferType<I>[]
            : S extends {
                type: DataTypes.OBJECT;
                schema: infer O;
              }
              ? {
                  [K in keyof O]:
                    InferType<O[K]>
                }
              : never;
```

Лістинг Б.3 - Перевірка отриманих даних відповідно до описаної схеми

```
export default function validator(
  value: any,
  schema: SchemaType,
): any {
  switch (schema.type) {
    case DataTypes.ANY:
      return value;
    case DataTypes.STRING:
      return String(value);
    case DataTypes.NUMBER: {
      const num = Number(value);
      if (Number.isNaN(num)) {
        throw new Error("Expected number");
      }
      return num;
    }
    case DataTypes.BOOLEAN: {
      if (typeof value === "boolean") {
```

```

        return value;
    }
    if (typeof value === "string") {
        const v = value.toLowerCase();
        if (v === "true" || v === "1") {
            return true;
        }
        if (v === "false" || v === "0") {
            return false;
        }
    }
    throw new Error("Expected boolean");
}

```

Лістинг Б.4 - створення публічного маршруту без необхідності авторизації користувача

```

function publicRoute<T extends RouteSchema>(
  schema: T,
  endpoint: (
    data: InferRequest<T> & {
      req: Request;
      res: Response;
    }
  ) => Promise<any>
) {
  return async (req: Request, res: Response,) => {
    try {
      const data = {
        body: schema.body
          ? validator(
              req.body,
              schema.body
            )
          : undefined,
        query: schema.query
          ? validator(
              req.query,
              schema.query
            )
          : undefined,
        params: schema.params
          ? validator(
              req.params,
              schema.params
            )
          : undefined,
      };
      const result = await endpoint({
        ...data,
        req,

```

```

        res,
    ));
    if (res.headersSent) {
        return;
    }
    if (result === undefined) {
        return res.status(200).json({
            success: true,
            data: null,
        });
    }
    return res.json(
        toPlain(result)
    );
} catch (e: any) {
    return res.status(400).json({
        success: false,
        error: e.message,
    });
}
};
}

```

Лістинг Б.5 – Механізм роботи приватних маршрутів, доступ до яких можливий лише після успішної авторизації користувача

```

function privateRoute<T extends RouteSchema>(
    schema: T,
    endpoint: (
        data: InferRequest<T> & {
            req: Request;
            res: Response;
            user: UserType;
        }
    ) => Promise<any>
) {
    return async (req: Request, res: Response,) => {
        if (!config.authenticate) {
            throw new Error(
                "authenticate function is required for
privateRoute"
            );
        }
        try {
            const user = await config.authenticate(req, res);
            if (!user) return;
            const data = {
                body: schema.body
                ? validator(
                    req.body,
                    schema.body

```

```

        )
        : undefined,
    query: schema.query
      ? validator(
          req.query,
          schema.query
        )
        : undefined,
    params: schema.params
      ? validator(
          req.params,
          schema.params
        )
        : undefined,
  };
  const result = await endpoint({
    ...data,
    req,
    res,
    user,
  });

```

Лістинг Б.6 - Для підключення бібліотеки у проєкт використовується цей файл експорту

```

export * from "./schema";
export * from "./types";
export * from "./routes";

```

ДОДАТОК В

СЛАЙДИ ПРЕЗЕНТАЦІЇ

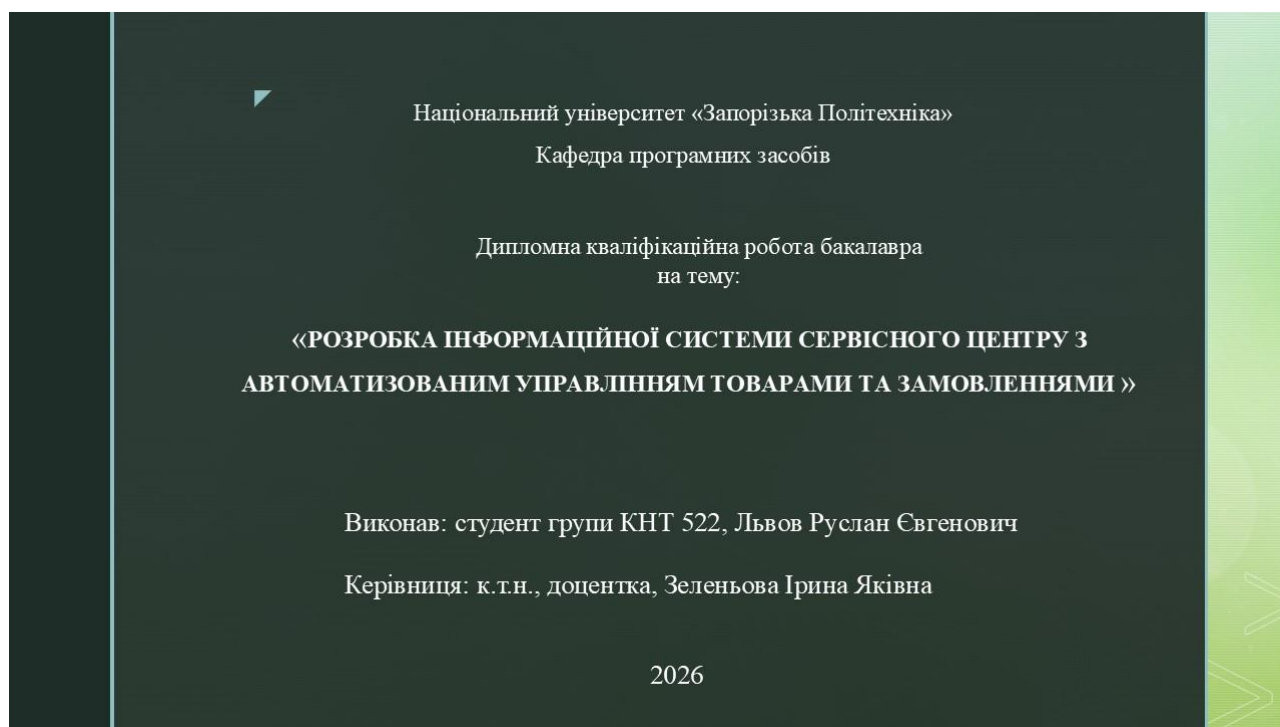


Рисунок В.1 – Слайд №1

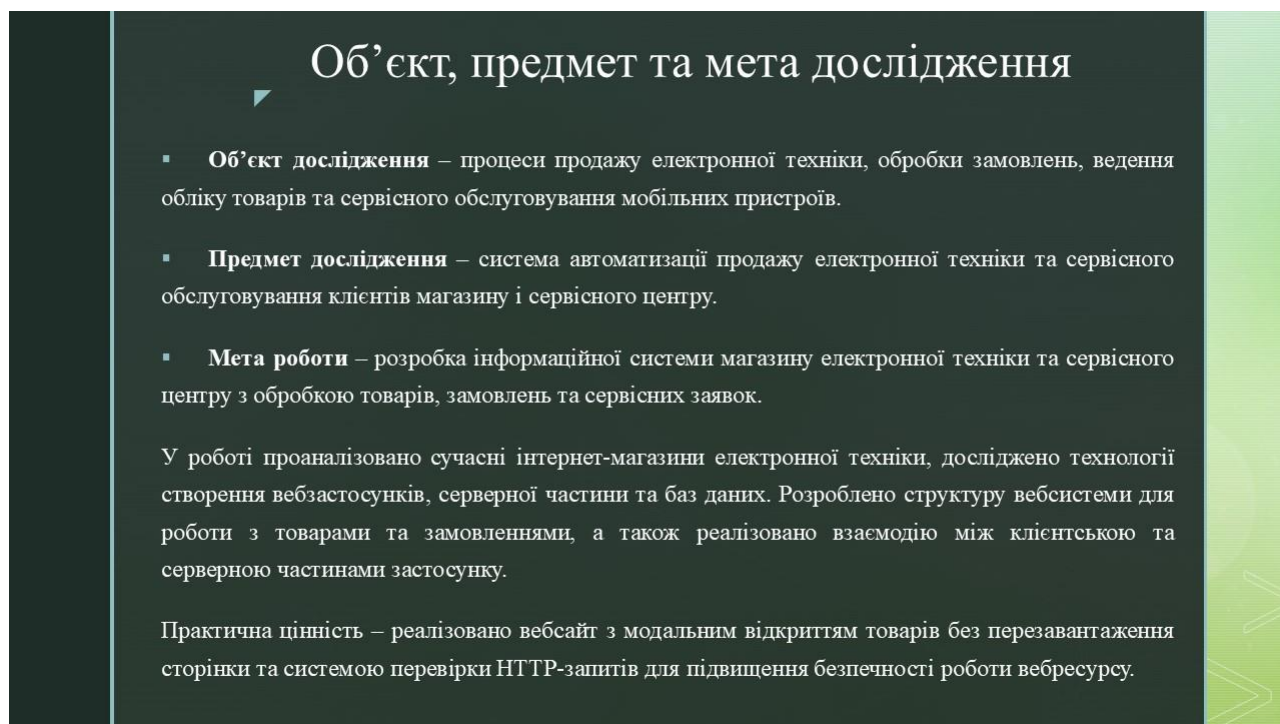


Рисунок В.2 – Слайд №2

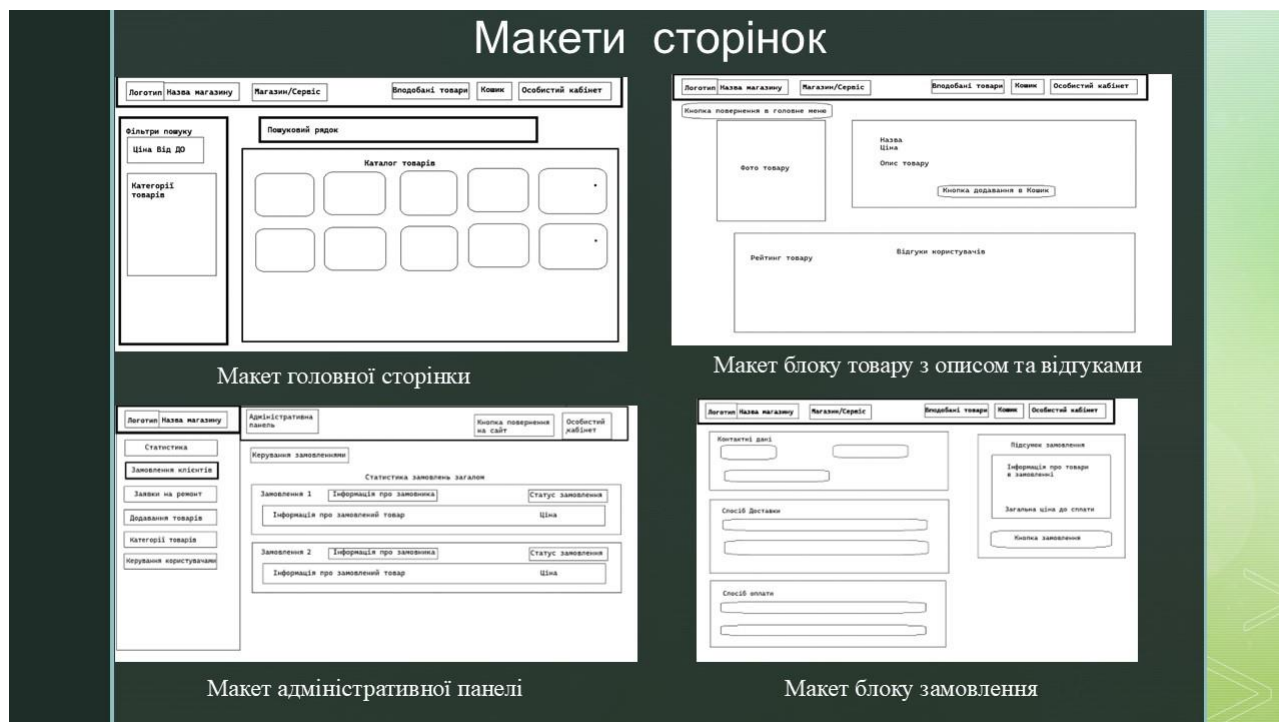


Рисунок В.3 – Слайд №3

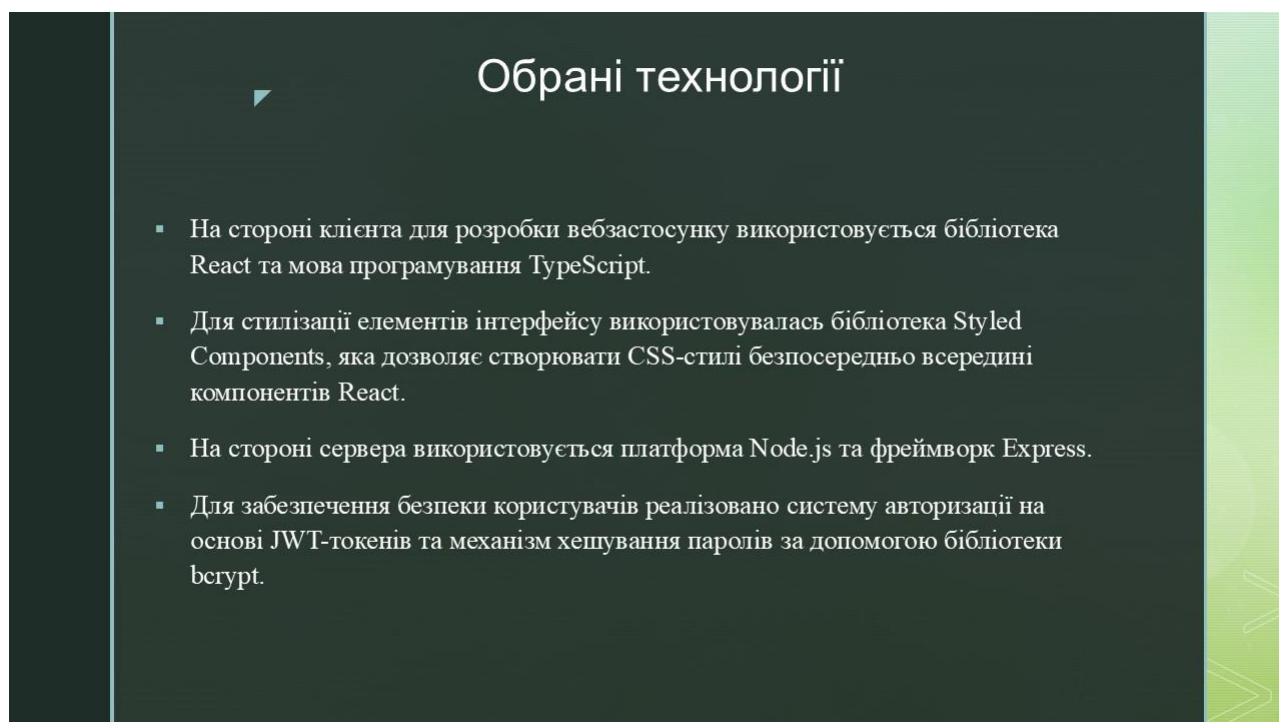


Рисунок В.4 – Слайд №4

Взаємодія клієнта, сервера та бази даних

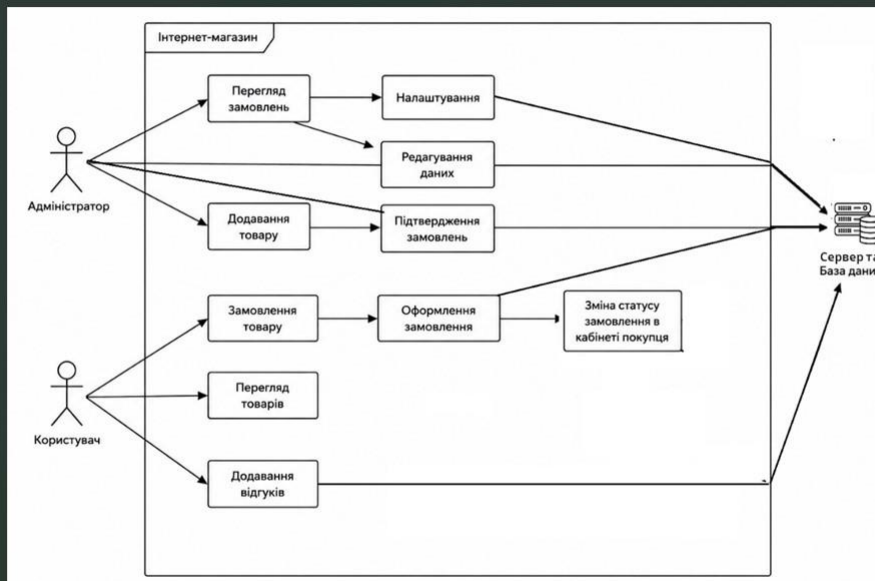


Рисунок В.5 – Слайд №5

Приклади розробленої сторінки та модулів

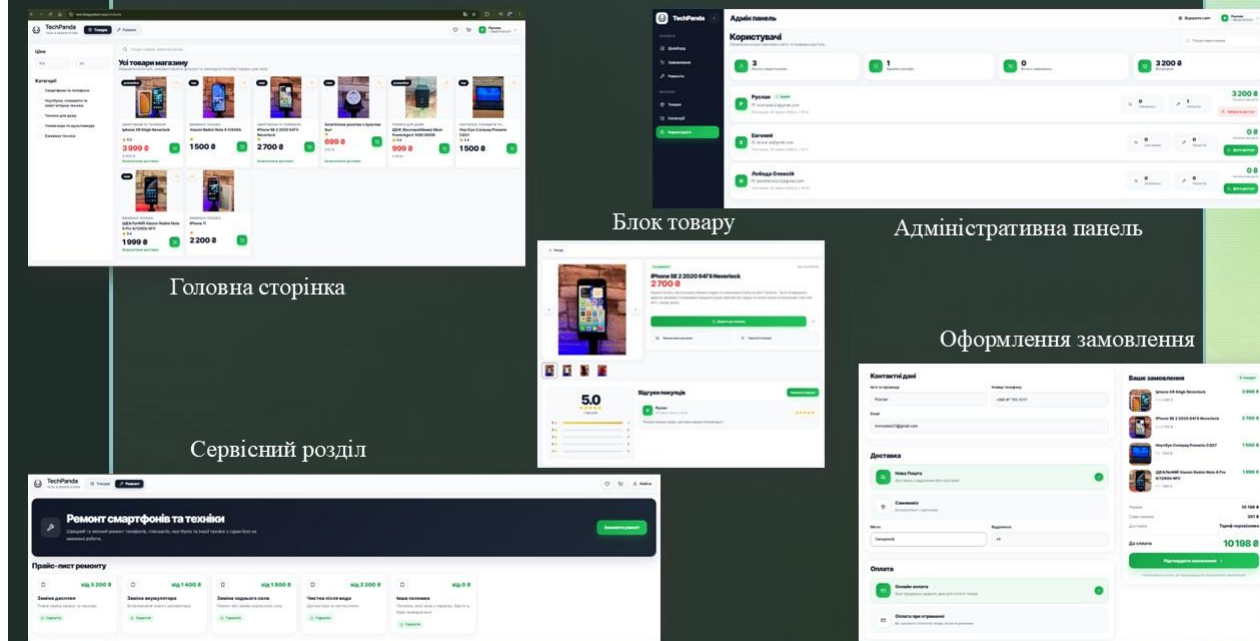
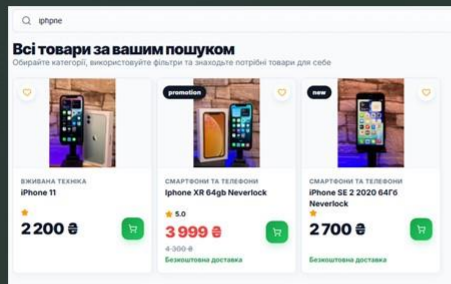


Рисунок В.6 – Слайд №6

Реалізація нечіткого пошуку



Запит написаний з помилкою : `iphrpe`

Ця функція реалізована на серверній частині проєкту завдяки базі даних PostgreSQL і розширенню «`pg_trgm`». Це розширення дає змогу порівнювати подібність текстових рядків.

Trigram Similarity Search

Запит з частковою фразою : `хочу розетку`

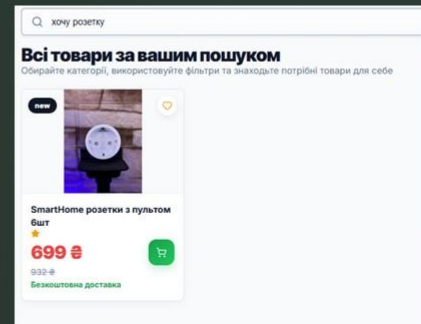


Рисунок В.7 – Слайд №7

Npm-бібліотека InferPress

- InferPress — бібліотека, яка дозволяє автоматично перевіряти вхідні дані, виконувати перетворення типів і формувати TypeScript-типи на основі створених схем валідації.
- Дозволяє значно скоротити повтори коду та підвищити безпеку серверної частини вебсайту.
- Завдяки використанню бібліотеки Inferpress забезпечується автоматична перевірка параметрів `body`, `query` та `params`. Ці дії дозволяють уникати більшості помилок під час роботи з даними користувачів, а також спрощують подальшу підтримку серверного коду проєкту.
- Наприклад: параметр `page` має містити числове значення, параметр `search` приймає або текстовий рядок. Якщо виявляється невідповідність типів, бібліотека автоматично видасть помилку та припинить обробку цього запиту.
- У межах даної дипломної роботи зазначена бібліотека використовується в серверній частині під час створення API-маршрутів, перевірки й обробки запитів користувачів, а також перевірки вхідних даних перед тим, як розпочнеться подальша їх обробка.
- Це допомагає у розробці великих проєктів, де є потреба у великій кількості HTTP-маршрутів та у забезпеченні обробки даних користувачів вебсайту.

Рисунок В.8 – Слайд №8

Хешування та система захисту

- Під час розробки були використано JWT-авторизацію за допомогою бібліотеки jsonwebtoken, яка дає змогу ідентифікувати користувачів під час їхньої взаємодії з серверною частиною вебзастосунку. У моїй системі використовуються два токени : access-токен та refresh-токен.
- Хешування. Після хешування пароль Qwerty123 буде приблизно таким: \$2b\$12\$X7F5mQ9LrT2vY8nKcP1uOeJ4hD6sW3aBzN0xCqR7. У такому вигляді кеш-пароль записується в БД.

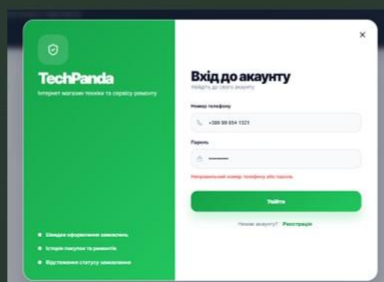


Рисунок В.9 – Слайд №9

ВИСНОВКИ

- Реалізовано вебзастосунок інтернет-магазину та сервісного центру з можливістю продажу товарів, керування замовленнями, додавання нових товарних позицій, приймання заявок на сервісне обслуговування та автоматизацію звичайних бізнес-процесів компанії.
- Для автентифікації користувачів впроваджено механізм JWT-токенів із підтримкою автоматичного оновлення сесії за допомогою refresh-токенів.
- Захист персональних даних забезпечується шляхом зберігання паролів у хешованому вигляді, що унеможливорює крадіжку паролів, навіть адміністратором системи.
- Розроблено адміністративну панель для керування товарами, замовленнями та сервісними заявками.
- Для серверної частини використано власну бібліотеку InferPress, що забезпечує безпечну валідацію HTTP-запитів та спрощує написання коду.
- Створена інформаційна система має практичну цінність і буде використана як основа для подальшого впровадження та розвитку в комерційній діяльності.

Рисунок В.10 – Слайд №10