Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Веб-платформа для мережі будівельних магазинів

(тема)

Виконав:

здобувач 3 року навчання,

групи КІУКІу-22-2

Денис ЦАРЕНКО
(власне ім’я, прізвище)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Тетяна ФІЛІМОНЧУК
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Царенку Денису Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи Веб-платформа для мережі будівельних магазинів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи 1) кількість програмних компонентів: три штуки; 2) головний формат передачі даних: JSON; 3) основний протокол роботи застосунку: HTTP;

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз проблеми та огляд існуючих рішень;
2) вибір технології розробки та інструментальних засобів;
3) розробка алгоритмічного забезпечення;
4) розробка програмних модулів;
5) відлагодження програмних модулів;
6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 10 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз існуючих методів вирішення задачі	10.06.2025-13.06.25	
2	Вибір програмного забезпечення та	14.06.2025-17.06.25	
	інструментів розробки		
3	Проєктування архітектури застосунку	18.06.25-21.06.25	
4	Розробка бази даних та серверної частини	22.06.25-28.06.25	
	застосунку		
5	Розробка графічного інтерфейсу користувача	29.06.25-02.07.25	
6	Тестування застосунку	03.07.25-05.07.25	
7	Подання кваліфікаційної роботи керівникам	06.07.25-09.07.25	
	для попереднього захисту		
8	Подання кваліфікаційної роботи на	10.07.25-11.07.25	
	рецензування		

Дата видачі завдання " 09 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тетяна ФІЛІМОНЧУК
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 98 с., 69 рис., 2 дод., 10 джерел.

СЕРВЕР, КЛІЄНТ, ТРИШАРОВА АРХІТЕКТУРА, БАЗА ДАНИХ, ЗАПИТ, ПРОТОКОЛ HTTP, SQL, JAVASCRIPT, PHP, HTML, CSS, REACT, ФРЕЙМВОРК, ВЕБ-РОЗРОБКА.

Метою кваліфікаційної роботи є створення веб-застосунку для мережі будівельних магазинів, що підтримує можливість адміністрування та функціонал інтернет-магазину для потенційних покупців.

У ході виконання кваліфікаційної роботи було створено веб-застосунок на основі тришарової архітектури, що складається з серверу бази даних (містить у собі адміністративні дані застосунку), серверного застосунку (виконує роль посередника при обробці запитів з клієнтської сторони) та клієнтського веб-застосунку (містить графічний інтерфейс для взаємодії користувача з сервером). Для реалізації бази даних було використано діалект MariaDB мови SQL, для написання серверного застосунку – мову PHP, для клієнтського веб-застосунку – мову розмітки сторінок HTML, мову стилів CSS із надбудовою Tailwind та фреймворк React для мови Javascript. При розробці було зроблено наголос на можливості компонентної розробки, написані відповідні універсальні функції та реалізовано відповідний робочий процес застосунку.

ABSTRACT

Bachelor’s thesis : 98 pages, 69 figures, 2 appendices, 10 sources.

SERVER, CLIENT, THREE-TIER ARCHITECTURE, DATABASE, REQUEST, HTTP PROTOCOL, SQL, JAVASCRIPT, PHP, HTML, CSS, REACT, FRAMEWORK, WEB DEVELOPMENT.

The major role of this thesis is to create a web application for a network of hardware stores that supports the administration and functionality of an online store for potential buyers.

During the thesis work process, a web application was created based on a three-tier architecture, consisting of a database server (containing the application's administrative data), a server application (acting as an intermediary in processing requests from the client side) and a client web application (containing a graphical interface for user interaction with the server). The MariaDB dialect of the SQL language was used to implement the database, the PHP language was used to write the server application, and the HTML markup language, the CSS style sheet with the Tailwind add-on, and the React framework for the Javascript language were used for the client web application. During development, emphasis was placed on the possibilities of component development, the corresponding universal functions were written, and the corresponding application workflow was implemented.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

API – інтерфейс програмування застосунків (англ., Application Programming Interface)

CLI – інтерфейс командного рядка (англ., Command Line Interface)

CMS – система керування вмістом (англ., Content Management System)

CSS – каскадні таблиці стилів (англ., Cascade Style Sheets)

DOM – об’єктна модель документа (англ., Document Object Model)

HTML – мова розмітки гіпертексту (англ., Hypertext Markup Language)

HTTP – гіпертекстовий протокол передачі (англ., HyperText Transfer Protocol)

IT – інформаційні технології (англ., Information Technology)

JS – мова веб-програмування JavaScript (англ., JavaScript)

JSON – об’єктний опис JavaScript (англ., JavaScript Object Notation)

PHP – препроцесор гіпертексту (англ., Hypertext PreProcessor)

REST – передача репрезентативного стану (англ., Representational State Transfer)

SQL – мова структурованих запитів (англ., Structured Query Language)

URL – уніфікований вказівник ресурсу (англ., Unfied Resource Locator)

WDDX – обмін розподіленими у Всесвітній мережі даними (англ., Web Distributed Data eXchange)

XML – розширювана мова розмітки (англ., eXtensible Markup Language)

ВСТУП

Торгівля – це один з найстаріших способів соціальної взаємодії між людьми. Як тільки людство дійшло до створення перших осідлих громад, бартер, наряду з війною став одним з основних способів здобуття необхідних ресурсів. Разом з соціумом еволюціонував і процес торгівлі: з'явилися протовалюти – золото та срібло, бартер перетворився на власне торгівлю – обмін за посередництвом цих самих валют, а зі збільшенням масштабу людської організації збільшувалась і площа торгівлі: обмін між племенами перетворився на торгівлю між областями, містами, феодалами та державами, а згодом і на міжнародну торгівлю.

Інформаційна епохи та стрімкий розвиток обчислювальних технологій у другій половині двадцятого сторіччя посприяло зміні у способах та підходах до торгівлі: міжнародні платіжні системи, інтернет-агрегатори товарів, збільшення популярності індивідуальних товарних доставок – усе це і є наслідки цифрової революції [1].

Робота з такою цифровізованою інфраструктурою призводить до появи нових викликів: необхідно забезпечувати інтуїтивно зрозумілу подачу інформації як для співробітників магазинів, так і для їх потенційних покупців.

Найпопулярнішим рішенням останнього часу є створення вебзастосунків, які працюють за допомогою засобів мережі Інтернет і напряму не потребують встановлення на машину.

Метою даної роботи є створення вебзастосунку, який дозволяє легко отримувати або змінювати дані про будь-який товар у мережі будівельних магазинів. Для реалізації задачі буде використано три компоненти: реляційна база даних, що зберігатиме усі необхідні дані про товари, персонал, ключі сесій тощо, вебсервер, необхідний для обробки запитів клієнта до бази даних, а також вебклієнт зі зручним та зрозумілим графічним інтерфейсом.

1 АНАЛІЗ ПРЕДМЕТОЇ ОБЛАСТІ

В ході кваліфікаційної роботи необхідно вирішити наступні задачі:

- провести аналіз існуючих рішень з метою знаходження переваг та недоліків;
- спроєктувати, розробити та створити реляційну базу даних (БД), що міститиме у собі необхідні поля інформації;
- привести БД до третьої нормальної форми: створити усі необхідні таблиці з урахуванням первинних ключів, сформувати зовнішні ключі для залежних таблиць, максимально атомізувати таблиці для забезпечення їх якнайбільшої незалежності за рахунок зв’язків ключів;
- забезпечити правильність та захищеність введених даних, зокрема перевірку некоректних значень та екранізацію (санітизацію) аргументів запитів;
- створити серверний застосунок, який матиме змогу оброблювати запити від користувача до БД: POST – для зміни та обробки даних, GET – для їхнього отримання та відображення;
- інтегрувати серверний застосунок із БД, перевірити вплив його функціональності на БД, відлагодити та виправити помилки, що проявляються;
- спроєктувати структури запитів: поля, формат відповіді, обробку недопустимих значень у запиті/відповіді тощо;
- спростити організацію роботи з БД за допомогою використання процедур користувача, функцій та макросів БД;
- спроєктувати графічний інтерфейс користувача клієнтського застосунку, що має бути інтуїтивно зрозумілим, простим у використанні;
- розробити, відлагодити та інтегрувати клієнтський вебзастосунок з серверним.

1.1 Аналіз існуючих рішень

Специфіка поставлених задач має на увазі створення застосунку, який зміг би об’єднати у собі функціональність ERP-застосунку (Enterprise Resource Planning) та інтернет-магазину. Більшість існуючих рішень у сфері ERP-додатків – це десктопні застосунки для локального використання. Для того, аби реалізувати подібний функціонал за допомогою вебінструментарію, необхідно вдатися до створення модифікованої адміністративної панелі, що матиме схожий функціонал для роботи з даними, але не потребуватиме встановлення на локальну машину користувача. У зв’язку з такою специфікою, розглянемо приклади, що містить деякі з вищеописаних аспектів майбутнього програмного продукту.

1.2 Застосунок керування бізнес-ресурсами Net Suite IRP

NetSuite (рисунок 1.1) – хмарний програмний застосунок керування бізнес-ресурсами, такими як фінанси, складські запаси, управління проєктами та взаємодією з клієнтами, eCommerce тощо. Він був створений американською компанією Oracle і розрахований на використання середнім та великим бізнесом.

Перевагами застосунку є:

- хмарна архітектура: NetSuite – це повністю хмарний ERP, який забезпечує доступ до системи з будь-якого місця з підключенням до Інтернету, що дозволяє користувачам працювати віддалено та знижує витрати на підтримку IT-інфраструктури;
- інтеграція та масштабованість: NetSuite пропонує вбудовану інтеграцію з іншими рішеннями Oracle, а також можливість масштабувати систему, що дозволяє організаціям рости, не змінюючи платформу;
- автоматизація процесів: NetSuite підтримує автоматизацію рутинних задач, що дозволяє знизити витрати часу та мінімізувати людські помилки у порівнянні з ручним виконанням таких самих задач;

- гнучкість у налаштуванні: NetSuite надає можливості для кастомізації та налаштування під специфічні потреби бізнесу, що включає налаштування бізнес-процесів, звітів, рольових прав доступу тощо;

- можливість отримувати аналітику та звітність у реальному часі: NetSuite має інструменти для отримання бізнес-аналітики та формування звітів в реальному часі, що дає можливість керівництву швидше приймати рішення на основі актуальних даних.

Рисунок 1.1 – Сторінка застосунку NetSuite ERP

Недоліками даного рішення є:

- висока вартість використання: NetSuite може бути дорогим рішенням, особливо для малого бізнесу, через вартість підписки, додаткові модулів та послуги з впровадження та підтримки;

- складність налаштування: впровадження та налаштування NetSuite

можуть вимагати значного часу та залучення фахівців, особливо для організацій зі складними бізнес-процесами;

- обмеження на налаштування для деяких галузей: хоча NetSuite пропонує широкий набір функцій, деякі спеціалізовані галузі можуть потребувати додаткових налаштувань або модулів, яких немає в базовій версії системи;

- залежність від інтернету: оскільки NetSuite є хмарною системою, продуктивність та доступність безпосередньо залежать від стабільності інтернет-з’єднання, тому у випадку перебоїв зі з’єднанням доступ до системи може бути обмеженим;

- крива навчання: для нових користувачів може знадобитися час на освоєння всіх функцій системи, оскільки NetSuite є досить потужним та комплексним інструментом.

1.3 Застосунок «1С:Підприємство»

Застосунок «1С:Підприємство» (рисунок 1.2) використовується для автоматизації бізнес-процесів компанії, таких як бухгалтерський та фінансовий облік, управління персоналом та виробництвом, моніторинг та керування виробництвом, логістикою та продажами.

Перевагами даного рішення є:

- гнучкість у налаштуванні та локалізації: «1С:Підприємство» добре налаштоване для локального ринку і може враховувати особливості законодавства, бухгалтерського обліку та податкової звітності конкретної країни;

- велика екосистема рішень: система має безліч готових конфігурацій та модулів, таких як «1С:Бухгалтерія», «1С:Управління Торгівлею», «1С:Зарплата та Управління Персоналом», що дозволяє налаштувати програму для різних галузей;

- можливість кастомізації: 1С надає засоби для розробки власних

модулів та налаштувань завдяки вбудованій мові програмування 1С, що дозволяє створювати індивідуальні рішення під конкретні вимоги бізнесу;

- локальна підтримка та навчання: популярність ПЗ сприяла появі великої кількості локальних спеціалістів з його обслуговування;

- вартість: у порівняння з більшістю програмних продуктів, розроблених західними компаніями, використання цього продукту є дешевшим, та, як результат, вигіднішим для середньо-малого бізнесу.

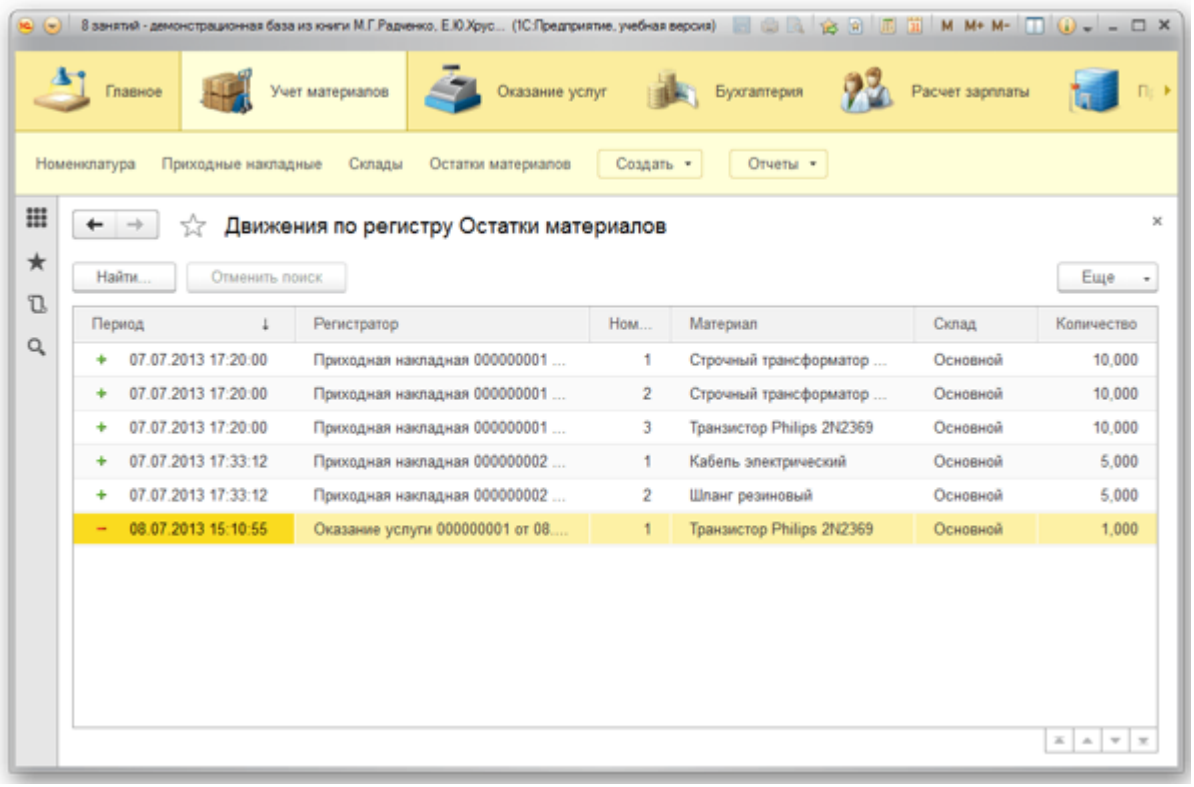


Рисунок 1.2 – Застосунок «1С:Підприємство»

Недоліками даного рішення є:

- програмний продукт країни-агресора: придбання та використання цього програмного забезпечення сприяє популяризації роботи «1С» та приносить дохід до російського бюджету;

- обмеження для міжнародних компаній: система орієнтована переважно на ринки країн СНД, тому для компаній із глобальними операціями вона може бути менш зручною через відсутність багатомовності

та підтримки міжнародних стандартів обліку;

- застаріла архітектура: деякі версії «1С:Підприємство» мають обмежену функціональність з точки зору хмарних можливостей, що ускладнює роботу з системою віддалено. Хоча є хмарні рішення 1С, вони менш інтегровані та можуть мати обмеження в порівнянні з сучасними хмарними ERP;

- крива навчання та обмеженість гнучкості в масштабуванні: незважаючи на гнучкість кастомізації, для роботи з програмою потрібні певні знання мови програмування 1С та специфіки системи. Масштабування також може бути складним для великих міжнародних компаній;

- безпека та надійність: оскільки 1С – це локально популярна система, вона часто стає ціллю для кіберзагроз та потребує ретельної внутрішньої IT-підтримки для підтримки безпеки;

- сумісність з іншими системами: інтеграція з популярними глобальними системами (наприклад, SAP, Salesforce) може бути складною та потребує додаткових модулів або рішень від партнерів.

1.4 Адміністративна панель Plesk

Plesk (рисунок 1.3) – програмне забезпечення для автоматизації адміністрування серверів та баз даних, розроблене компанією Plesk International GmbH у 2001 році. Перевагами використання даного рішення є:

- інтуїтивний інтерфейс: Plesk має зручний та зрозумілий графічний інтерфейс, що спрощує процес адміністрування навіть для користувачів з мінімальним досвідом у серверній сфері. Весь функціонал добре структурований, що дозволяє легко знайти необхідні налаштування;

- підтримка багатьох платформ та серверів: Plesk працює як на Linux, так і на Windows, що робить його універсальним вибором для різних серверних середовищ. Підтримка великих платформ і технологій (Git, Docker, Node.js) дозволяє використовувати його в різних вебпроектах;

- інтеграція з популярними інструментами: Plesk має розширення для управління сайтами на WordPress, Joomla та інших CMS, а також інструменти для розробників, такі як підтримка Docker, Git та автоматизація задач, що робить його зручним для швидкого розгортання та розробки;

- автоматизація та безпека: Plesk пропонує інструменти для автоматичного резервного копіювання, оновлення програмного забезпечення та сертифікатів SSL, що полегшує управління та забезпечує безпеку сервера. Розширення для захисту від вірусів, спаму та DDOS-атак допомагають забезпечити стабільність вебсайтів та їхню безпеку;

- можливість створення окремих акаунтів та рольових прав доступу, що особливо корисно для хостинг-провайдерів та компаній, які можуть розподіляти права доступу до різних функцій між різними користувачами та клієнтами;

- широка екосистема розширень: Plesk має маркетплейс із великим вибором додаткових розширень, що дозволяє користувачам додавати нові функції, такі як управління DNS, інструменти для SEO, аналітика та інші спеціалізовані інструменти.

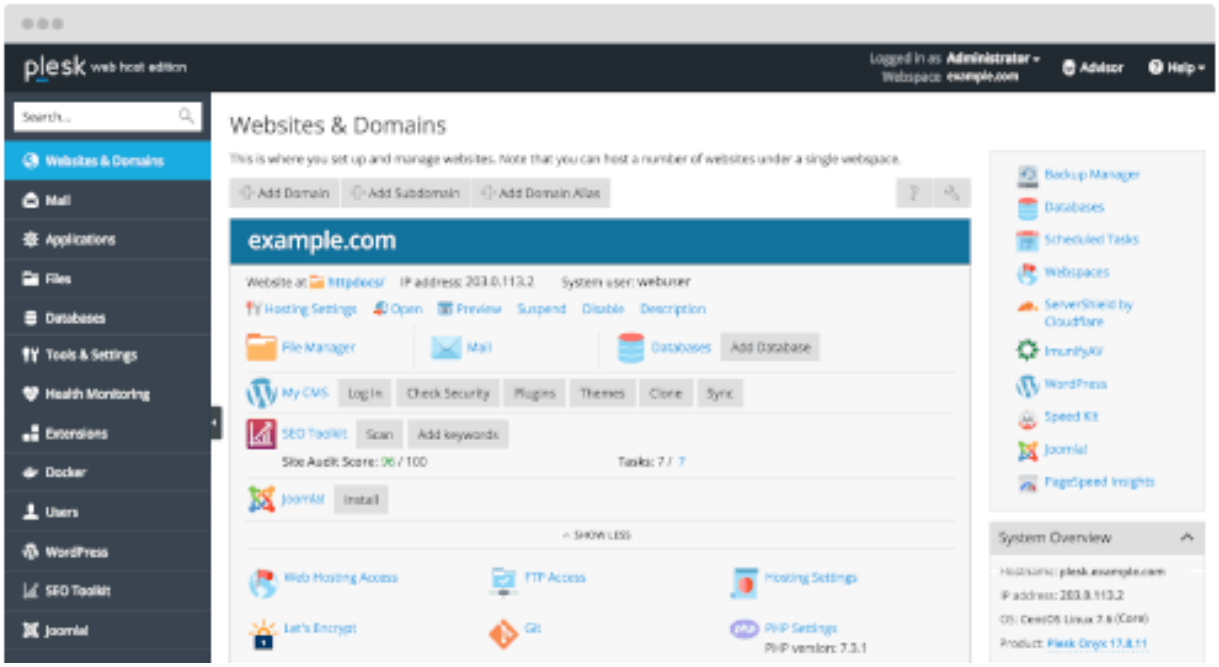


Рисунок 1.3 – Застосунок Plesk

Недоліками даного рішення є:

- вартість: Plesk – комерційний продукт, його ліцензії можуть бути дорогими для малого бізнесу чи індивідуальних користувачів, особливо якщо потрібні додаткові розширення або ліцензія на багато сайтів;

- витрати ресурсів: хоча Plesk забезпечує великий функціонал, він може бути досить вимогливим до ресурсів сервера, що може спричиняти зниження продуктивності на слабких або малопотужних серверах;

- крива навчання для новачків: незважаючи на зручний інтерфейс, Plesk має багато функцій та налаштувань, через що новачкам може знадобитися деякий час для освоєння всього потенціалу системи;

- залежність від розширень: багато додаткових функцій Plesk доступні тільки через платні розширення, що може збільшити загальну вартість системи та потребує додаткових налаштувань;

- обмеження на кастомізацію: у порівнянні з відкритими панелями управління (такими як ISPConfig або Virtualmin), Plesk має обмеження в можливостях кастомізації. Деякі користувачі можуть вважати це недоліком, особливо якщо їм потрібно змінювати або налаштовувати особливі конфігурації сервера;

- проблеми сумісності при оновленнях: у рідкісних випадках після оновлень Plesk можуть виникати проблеми з сумісністю певних розширень або додатків, особливо якщо використовуються нестандартні конфігурації сервера.

1.5 Вебзастосунок для серверного адміністрування Webmin

Webmin (рисунок 1.4) – це вебзастосунок, інструмент серверного адміністрування для Unix-подібних операційних систем. Окрім функціоналу адміністрування серверів, також підтримує функції системного адміністрування, такі як управління файлами та базами даних, брандмауером,

адміністрування користувачів та груп тощо.

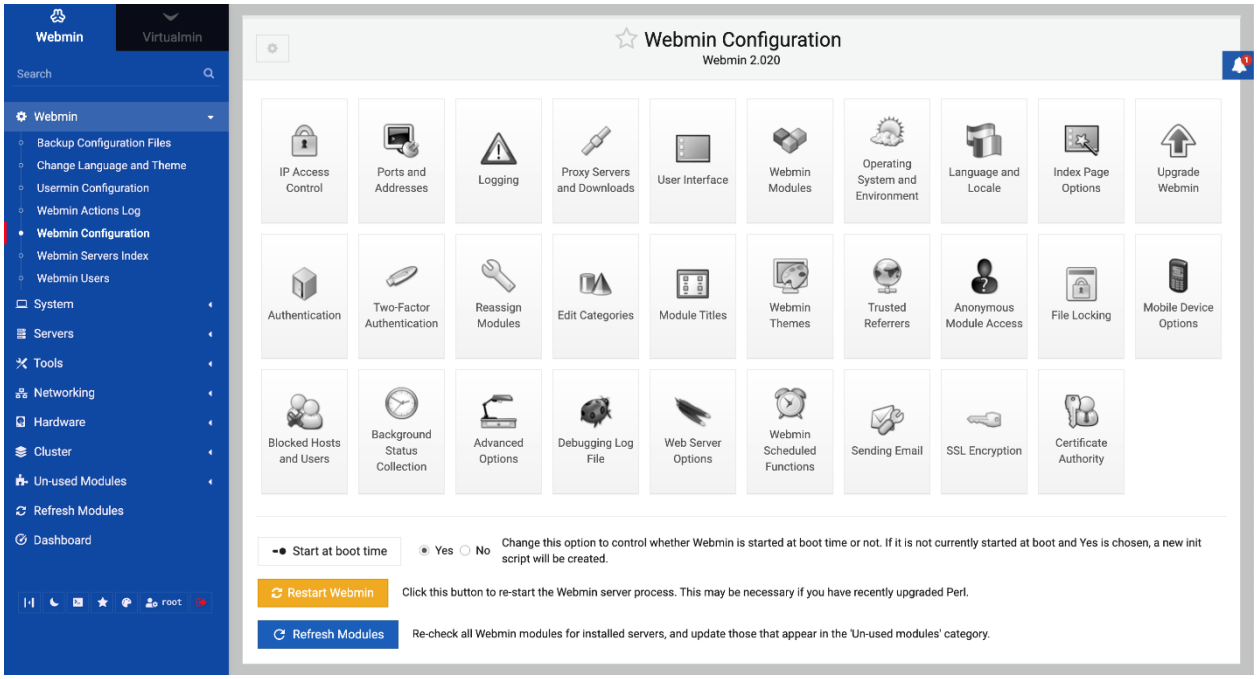


Рисунок 1.4 – Застосунок Webmin

Перевагами даного рішення є:

- безкоштовність та відкритий код: Webmin розповсюджується з відкритим вихідним кодом і є повністю безкоштовним. Це значний плюс для малого бізнесу та ентузіастів, яким потрібен доступ до базового управління сервером без додаткових витрат;

- гнучкість та кастомізація: завдяки відкритому коду Webmin можна налаштовувати відповідно до своїх потреб, він підтримує безліч модулів, і користувачі можуть змінювати існуючі модулі або створювати нові;

- можливість розширення функціоналу: Webmin має модульну структуру та підтримує різноманітні розширення для управління сервісами, такими як Apache, MySQL, Postfix, шо дозволяє налаштовувати сервер, додаючи лише потрібні функції;

- простота використання: Webmin має інтуїтивно зрозумілий вебінтерфейс, який полегшує управління сервером навіть для користувачів з обмеженими технічними знаннями, що робить його доступним для початківців;

- підтримка різних ОС: Webmin може працювати на різних Unix-подібних операційних системах, таких як Debian, Ubuntu, CentOS, та підтримує більшість основних Linux-дистрибутивів, що робить його універсальним інструментом;

- підтримка CLI та API: Webmin можна інтегрувати з командним рядком або використовувати її API для автоматизації задач, що надає додаткові можливості для досвідчених користувачів.

Недоліками даного рішення є:

- обмежений функціонал для вебхостингу: Webmin є більше системною панеллю управління, який не розрахований на вебхостинг. Для управління вебхостингом краще використовувати Virtualmin – додаток до Webmin, який додає необхідні функції. Однак навіть з Virtualmin його функціонал поступається популярним комерційним панелям, як-от cPanel або Plesk;

- порівняно базовий інтерфейс: хоча Webmin має простий інтерфейс, він може виглядати дещо застарілим та менш привабливим у порівнянні з комерційними панелями, що може бути мінусом для деяких користувачів, які надають перевагу сучаснішому інтерфейсу;

- відсутність офіційної підтримки: через те, що Webmin є безкоштовним продуктом, він не має офіційної технічної підтримки. Хоча є активна спільнота і доступна велика кількість документації, це може стати проблемою для тих, хто звик до швидкої підтримки від комерційних продуктів;

- складнощі з конфігурацією для новачків: хоча Webmin простий у використанні для виконання базових задач, його налаштування для більш просунутих конфігурацій (наприклад, налаштування брандмауера або безпеки сервера) може вимагати додаткових знань про серверну архітектуру;

- проблеми з продуктивністю на великих системах: Webmin може бути недостатньо продуктивним на серверах з великою кількістю користувачів чи вебсайтів, оскільки він не оптимізований для таких навантажень, що може

призвести до зниження продуктивності і додаткової оптимізації системи;

- обмежена підтримка Windows: хоча Webmin може працювати на системах Windows за допомогою Unix-емулятора Cygwin, це рішення не є офіційним та менш стабільне. Для користувачів ОС Windows краще розглядати інші панелі управління.

1.6 Інтернет-магазини

Наведення прикладу інтернет-магазинів не є доречним, оскільки кожен з таких застосунків реалізований приблизно однаково, а зазвичай відрізняються вони лиш стилем. Основні можливості, спільні для усіх інтернет-магазинів, такі:

- особистий кабінет клієнта;
- кошик;
- засіб фільтрування результатів пошуку;
- вікно пошуку;
- вікно сортування результатів пошуку;
- одна з множини картка товару.

Картка товару – це основний елемент інтернет-магазину, який дозволяє дізнатись основну інформацію про один зі знайдених товарів.

Обов’язкові поля інформації, які містить така картка – це назва товару, його ціна, у переважній більшості випадків його зображення, а також кнопка «Додати до кошику».

Додаткова інформація про товар може бути наступною: позначка про те, чи продається зараз товар за знижкою, чи багато людей купили його останнім часом, відгуки про товар, оцінки товару, інформація про наявність/відсутність товару у тому чи іншому магазині, код товару, можливість перегляду товару у різних кольорах тощо.

Беручи до уваги, що ПЗ розробляється для мережі будівельного магазину, окрім основної інформації, доречно буде також реалізувати показ даних про оцінки товару, відгуки на нього та наявність у різних магазинах.

2 АНАЛІЗ ТЕХНОЛОГІЙ, ЩО ВИКОРИСТОВУЮТЬСЯ ПРИ РОЗРОБЦІ ВЕБ-ЗАСТОСУНКУ

2.1 Мова розмітки HTML

Hypertext Markup Language (HTML) [2] – мова розмітки документів для їх графічної інтерпретації інтернет-браузерами. Використання HTML у сторінках є ключовим: саме ця мова слугує певним «фундаментом» сторінок: за допомогою неї описується структура сторінки та вбудовуються зображення, файли, інтерактивні форми, засоби взаємодії з користувачем тощо.

Така розмітка реалізовується за допомогою тегів (зазвичай парних блоків тексту, огорнутих кутовими дужками). Текст, обернений тегом, є окремим елементом (вузлом) моделі об’єктів документа (DOM), та в залежності від тегу може мати різні параметри за замовчуванням (рисунок 2.1).

```
<ul>
  <li>Lemon</li>
  <li>Orange</li>
  <li>Lime</li>
</ul>
```

Рисунок 2.1 – Приклад опису стилізованого списку у HTML

HTML не має верхнього рівня обмеження глибини вкладеності елементів, але на практиці рекомендується використовувати не більше 20 – це необхідно для спрощення процесу розробки, відлагодження та супроводу такої сторінки.

Мова HTML була представлена Тімом Бернерсом-Лі у 1989 році, а

перша версія випущена 3 листопада 1992 року. За більш ніж двадцять років розробки було створено п’ять номерних версій мови. Реліз останньої основної версії (HTML5) стався 22 січня 2008 року, друга ревізія п’ятої версії мови (HTML 5.2) була випущена 15 грудня 2017 року.

Для мови періодично розробляються нові стандарти та специфікації, за допомогою робочих груп вводяться нові можливості, які тестуються на працездатність та сумісність, що згодом рецензуються. Цим процесом з самого початку життєвого циклу HTML займається організація World Wide Web Consortium (W3C), створена самим Тім Бернерсом-Лі у 1994 році. Починаючи з 2004 року, супроводженням та оновленням мови W3C займається спільно з WHATWG (Web Hypertext Application Technology Working Group), що забезпечило можливість створення єдиного «живого стандарту» мови.

2.2 Мова каскадного опису стилів CSS

Cascading Style Sheets (CSS) – формальна мова опису вебсторінок, що використовується для оформлення HTML-документів та сторінок. Мова була створена Хоконом Віумом Лі (Håkon Wium Lie) у 1994 році, а її основною ідеєю було відокремлення стилю документа від його структури. Результатом роботи над CSS стали три основні рівні (аналог версій) мови [3].

Специфікація CSS1 була прийнята асоціацією W3C 17 грудня 1996 року та описувала такі можливості мови, як параметри шрифтів, кольори та атрибути тексту, вирівнювання елементів сторінки та властивості padding (внутрішній відступ) та margin (зовнішній відступ) для блокових елементів, а також додала першу ітерацію розташування елементу (display, float, clear).

Специфікація CSS2 була прийнята 12 травня 1998 року на основі CSS1 та описувала додаткову функціональність, таку як блокова верстка із застосуванням параметрів absolute, relative, sticky, fixed, block тощо,

розширений список селекторів, медіазапити (правила) для різних типів пристроїв відображення та параметрів сторінок, а також псевдоелементи ::before та ::after.

Починаючи з рівня 3, специфікація мови розбивається на модулі, основні з яких – це Selectors (селектори), Backgrounds and Borders (фони та рамки), Text (текст), Flexible Box Layout (компонування адаптивного макету), Transforms (зміщення), Transitions (переходи), Animations (анімації), Media Queries (медіазапити), Colors (кольори) та Positioned Layout (позиційне розташування).

Так само як і з HTML, ця мова розвивається згідно «живого стандарту», що використовується, доповнюється та поліпшується у ході використання CSS при розробці численних вебзастосунків.

CSS є своєрідною «надбудовою» над HTML: будь-яка сторінка може бути відображена та завантажена без використання CSS, але для звичайного користувача вона виглядатиме дещо відштовхуюче. Стиль сторінок без стилів виглядає по-старому, але у нинішніх умовах така концентрація тексту на сторінці не є зручною, і не може затримати на собі потенційного користувача (рисунок 2.2).

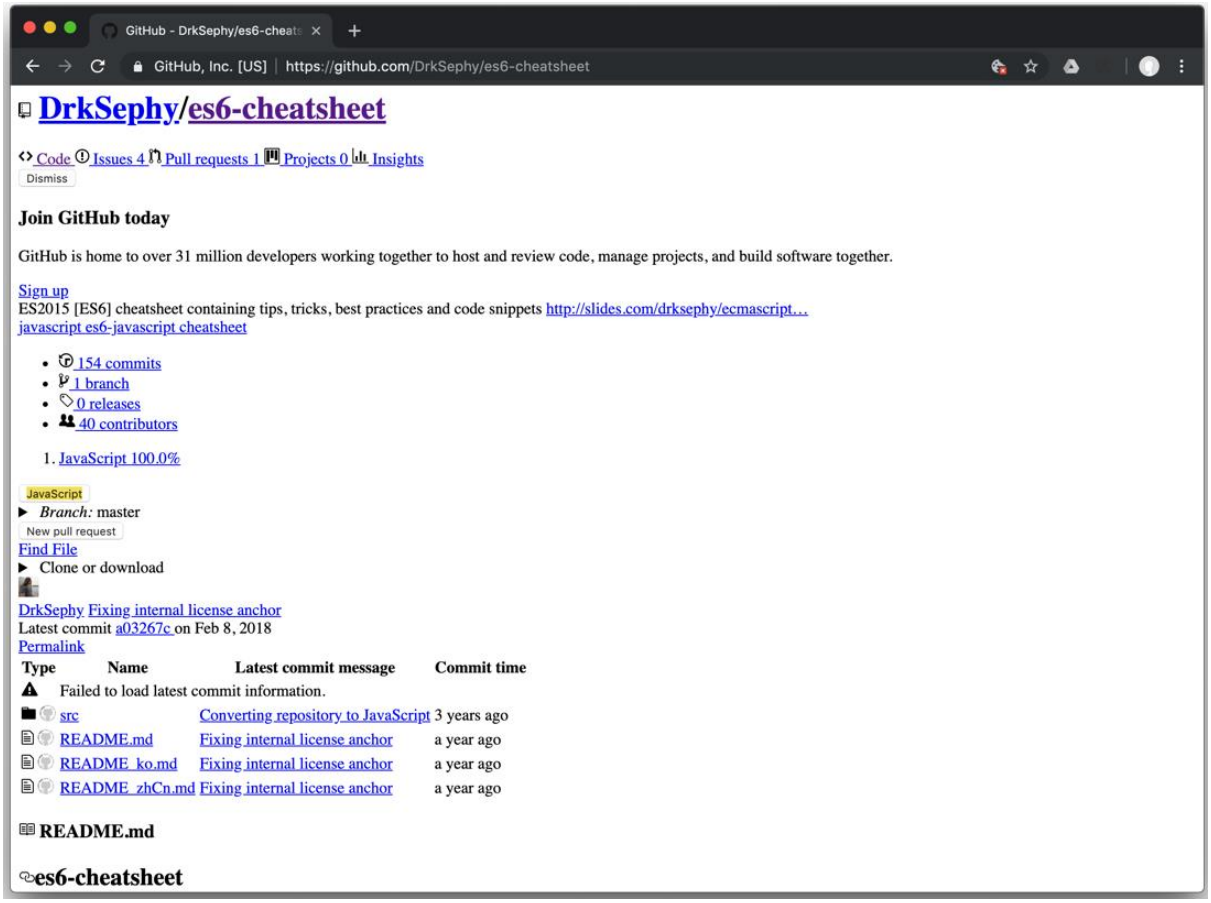
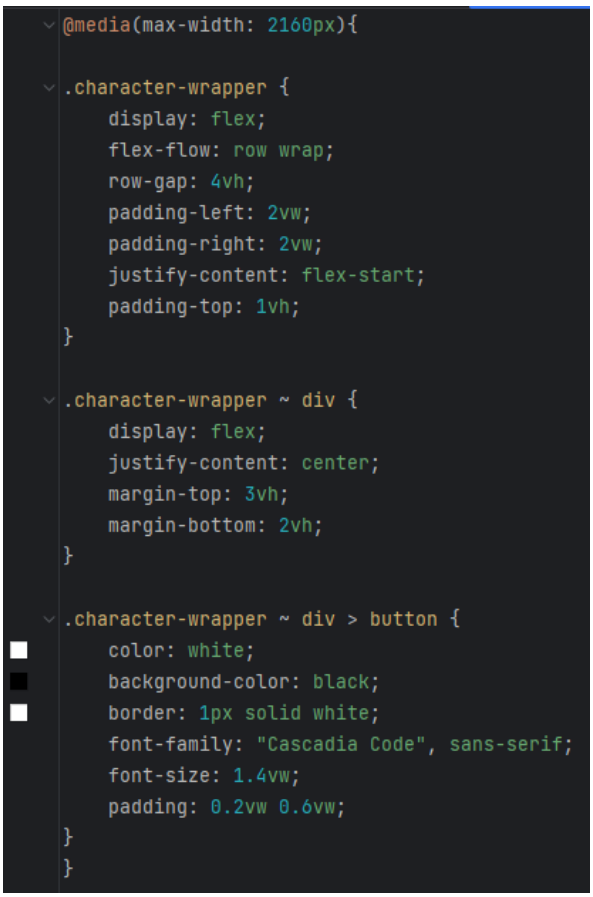


Рисунок 2.2 – Приклад HTML-сторінки без використання стилів

Робота мови CSS заснована на двох основних принципах – успадкування та каскадності. Принцип успадкування полягає у тому, що властивості CSS, які застосовуються для батьківського елементу, за замовчуванням переносяться на його дочірні елементи. Принцип каскадності є протилежністю успадкування, оскільки дозволяє описувати декілька наборів правил для одного елемента, які застосовуються згідно пріоритету. Основою механізму пріоритету є специфічність селектора, що визначає, яке правило буде застосовано до елемента. Найвищий пріоритет (ненумерований) є у властивостей, позначених ключовим словом !important, далі у вбудованих в HTML-тег стилей, згодом селекторів ідентифікаторів, потім класів і наостанок – селекторів тегів.

Загалом CSS-документ – це множина структур даних, розділених медіазапитами, що складаються з селектора, множини імен властивостей та їх значень (рисунок 2.3).



```
@media(max-width: 2160px){

.character-wrapper {
    display: flex;
    flex-flow: row wrap;
    row-gap: 4vh;
    padding-left: 2vw;
    padding-right: 2vw;
    justify-content: flex-start;
    padding-top: 1vh;
}

.character-wrapper ~ div {
    display: flex;
    justify-content: center;
    margin-top: 3vh;
    margin-bottom: 2vh;
}

.character-wrapper ~ div > button {
    color: white;
    background-color: black;
    border: 1px solid white;
    font-family: "Cascadia Code", sans-serif;
    font-size: 1.4vw;
    padding: 0.2vw 0.6vw;
}
}
```

Рисунок 2.3 – Приклад коду мовою CSS

CSS підтримує можливість масштабування коду за допомогою директиви @import. Перевагами мови CSS є:

- зменшення об’єму написаного на HTML коду;
- простота підтримки та масштабування у проєкті;
- широка підтримка усіма популярними браузерами;
- можливість створення складного та/або адаптивного дизайну сайту.

Недоліками мови CSS є:

- висока специфічність деяких селекторів ускладнює читання та розуміння створеного коду;
- часом неочевидна взаємодія елементів з певними властивостями так само може вплинути на процес відлагодження та рефакторингу коду;
- браузерні несумісності ускладнюють процес розробки необхідністю створення додаткових правил та псевдоелементів, а також відлагодженням сторінки за допомогою різних браузерів;
- відсутність логіки коду (умов та циклів).

- складність дебагінгу у комбінації з глобальністю заданих стилів.

2.3 Мова програмування JavaScript

JavaScript (JS) – мультипарадигменна мова програмування високого рівня, що (у контексті веброзробки) використовується для додавання інтерактивності, логіки та динамічних функцій до вебсторінок [4]. Перша версія мови під назвою Mocha була розроблена у 1995 році Бренданом Айхом для браузера Netscape Navigator. Стандарт мови було розроблено у 1997 році, під назвою ECMAScript.

Розробка JavaScript велась з урахуванням концепцій одразу декількох мов програмування. Так, своє ім’я та концепцію об’єктно-орієнтованого підходу до розробки JS отримав від популярної на той момент мови Java, а основний синтаксис (так само як і у випадку з мовою Java) був взятий від C. Від мови Self JavaScript взяв концепцію прототипного успадкування, при якому у кожного JavaScript-об’єкта є посилання на об’єкт-прототип, що може містити неоголошені поля та властивості. Діалект мови Lisp під назвою Scheme вплинув на функціональні аспекти JavaScript, зокрема визначення функцій як об’єктів першого класу (тобто можливість отримувати та передавати їх у функції у якості аргументів).

Мова JavaScript має наступні переваги, серед яких:

- динамічна типізація: при оголошенні змінної розробнику не потрібно вказувати її тип, а достатньо лише присвоїти їй значення;
- асинхронність: JavaScript має такі механізми як зворотні виклики (callbacks), проміси (promises) та функції async/await, що забезпечують створення асинхронних запитів та їх обробку;
- інтерпретованість: JavaScript не потребує компіляції та виконується безпосередньо в браузері, на відміну від більшості інших мов програмування.

Структуру JavaScript для веброзробки можна розбити на три складові:

- базова мова (Core Language): основні елементи мови JavaScript, без

яких неможливо писати код. Ця частина мови безпосередньо визначена специфікацією ECMAScript та включає в себе такі функції як оголошення та робота зі змінними, робота з різними типами даних (числа, рядки, бінарні значення, об’єкти та їх масиви тощо), логіка алгоритму (умови та цикли), функції та класи;

- BOM (Browser Object Model): інтерфейс взаємодії мови JavaScript з інтернет-браузером. Включає в себе функції роботи з вікнами (window.open(), window.close(), window.innerHeight, window.outerWidth тощо), отримання інформації про браузер (інтерфейси window.navigator, window.location тощо) та використання таймерів (setTimeout, setInterval);

- DOM (Document Object Model): інтерфейс, що дозволяє безпосередньо взаємодіяти з HTML-структурою сторінки за допомогою JS. Він включає в себе функції отримання вузлів дерева документа (document.querySelector(), document.querySelectorAll(), document. getElementById()), зміни вмісту елемента (element.textContent, element.innerHTML, element.outerHTML тощо), додавання вбудованих стилів елементів (element.style), обробка подій (addEventListener), зміна кількості та порядку вузлів дерева документа (element.append(), element.prepend(), element.remove() тощо) та інші.

Існує декілька способів використовувати JavaScript при веброзробці. Перший – написанная коду всередині тегу <script> (лістинг 2.1). Такий код виконуватиметься при завантаженні сторінки до ініціалізації елементів DOM та найкраще підходить для створення обробників подій або коду, що має бути запущено перед ініціалізацією елементів дерева документа.

Лістинг 2.1 – Використання JavaScript, вбудованого у сторінку

```
<script>
console.log(“Hello World!”);
</script>
```

Другий можливий спосіб використання JavaScript на вебсторінці –

безпосередній опис обробника подій всередині тегу елемента. Такий підхід найчастіше всього використовується при додаванні кнопок на сторінку для опису функції, що спрацьовує при натисканні (лістинг 2.2).

Лістинг 2.2 – Використання JavaScript всередині елемента

```
<button class=”test-button”
onclick=”() => {console.log(“button clicked!”)}”>
</button>
```

Третій і найпоширеніший спосіб використання JavaScript – винесення коду функцій у окремий файл, що згодом підключається до сторінки (лістинг 2.3). Функції, описані у цьому файлі, згодом можна використовувати аналогічно другому способу (лістинг 2.4).

Лістинг 2.3 – Підключення JS-файлу з функціями до HTML-документу

```
<head>
<script src="scripts.js"></script>
…
</head>
```

Лістинг 2.4 – Приклад використання функції з підключеного файлу

```
<button onclick="generateBlocks2()">Generate</button>
```

Мова JavaScript має такі переваги:

- підтримка сучасними браузерами: JavaScript є одним з основних інструментів веброзробки та найпопулярнішою мовою у цьому напрямку. Відкрита специфікація дозволяє розробникам браузерів імплементувати підтримку мови на програмному рівні;

- простота інтеграції з мовами HTML/CSS: разом з зазначеними мовами JS складає базовий набір веброзробника. З плином часу способи взаємодії цих мов описувались та стандартизувались сильніше та детальніше;

- широкий вибір бібліотек та фреймворків: будучи однією з найпопулярніших мов програмування на даний час, JavaScript має активну спільнотою як професійних, так і незалежних розробників, що створюють свої доповнення до мови, які облегшують вирішення тих чи інших задач.

Серед недоліків JavaScript можна виділити наступні:

- вразливість до помилок: будучи динамічно типізованою мовою, JavaScript часто може виконувати неправильні операції над змінними. Так, наприклад, порівняння двох чисел мовою JavaScript без попереднього явного приведення до чисельного типу призводить до порівняння двох строкових змінних, що призводить до некоректних результатів;

- безпека: код, що написаний мовою JavaScript та запускається у браузері, може бути достатньо легко переглянутий та проаналізований на предмет наявності у ньому слабких місць. Цю проблему частково закриває обфускація коду, але при наявності великої кількості часу та терпіння, обфускований код можна зламати та так само проаналізувати;

- несумісність різних версій стандарту: для оновлення проєктів, що використовують ECMAScript старих версій, необхідно звірятися з додатковими матеріалами, а також самостійно проводити відлагодження коду.

2.4 Фреймворк Tailwind CSS

Tailwind – це надбудова для мови CSS, що спрощує роботу без необхідності писати додатковий власний код цією мовою. На відміну від більшості інших фреймворків Tailwind представляє собою не набір попередньо визначених та стилізованих компонентів, а множину невеликих попередньо описаних утилітарних HTML-класів, більша частина з яких відповідає за одне значення тієї чи іншої властивості CSS. Так, наприклад, клас «px-8» відповідає за внутрішній відступ у 32 пікселя, а клас «h-full» визначає ширину елемента у 100% ширини батьківського елемента.

Поєднання множини таких класів у описі HTML-елемента дозволяє його стилізувати без необхідності писати додатковий код для стилізації цього ж елемента у окремому .css-файлі. Так, елемент, стилі до якого було застосовано з використанням інструментарію Tailwind (рисунок 2.4), займатиме більше місця у HTML-розмітці сторінки, проте набагато менше

місця у таблиці стилів. Враховуючи особливості роботи CSS, індексація строго заданої кількості стилів Tailwind займе набагато менше часу, аніж завантаження множини CSS-класів (рисунок 2.5).

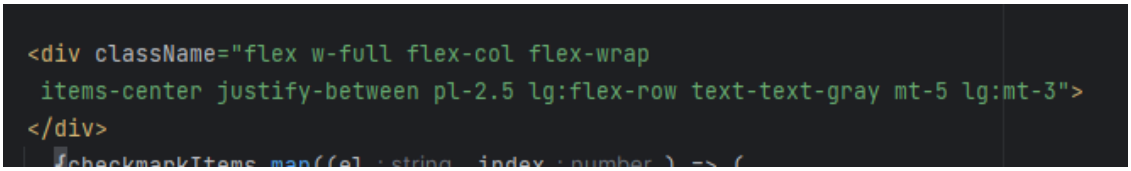


Рисунок 2.4 – Опис стилю елементу за допомогою Tailwind

Tailwind для опису стилів використовує підхід «Mobile-first». Це означає, що кожен із описаних у бібліотеці класів оптимізовано для відображення мобільними вебсторінками. Класи, що описуються без префіксів, застосовуються для усіх розмірів екранів без виключення, починаючи з найменших. Для того, аби змінити ці значення для більших екранів, необхідно поставити перед властивістю префікс брейкпоїнта відповідного розміру екрану [5]. На рисунку 2.5 можна побачити спосіб такого опису властивостей – елемент за замовчуванням описує верхній зовнішній відступ як mt-5 (тобто 1.25rem або 20 пікселів), але при ширині екрану у мінімум 1024 пікселі (префікс lg:) це значення змінюється на mt-3 (0.75rem або 12 пікселів). Фреймворк надає можливість використовувати за замовчуванням 5 брейкпоїнтів, але користувач має можливість додавати власні додаткові брейкпоїнти.

```
.test-class{
    display: flex;
    width: 100%;
    flex-direction: column;
    flex-wrap: wrap;
    align-items: center;
    justify-content: space-between;
    padding-left: 0.625rem;
    color: #616161;
    margin-top: 1.25rem;
}

@media (min-width: 1024px) {
    .test-class{
        flex-direction: row;
        margin-top: 0.75rem;
    }
}
```

Рисунок 2.5 – Опис такого ж стилю елементу за допомогою CSS-класу

Окрім власне інших точок опису екрану, розробник, що працює з бібліотекою Tailwind CSS має змогу додавати власні значення для усіх властивостей CSS, що описані у фреймворці. Це можливо завдяки коригуванню файла «tailwind.config.js», що створюється у головній директорії проекту при підключенні до нього Tailwind.

Структура файлу «tailwind.config.js» – це JSON-файл, що містить у собі масив з описом розширень файлів та директорій, які сканувatиме фреймворк, а також об’єкт «theme», що й відповідає за додавання власних значень властивостей стилю. Фрагмент значень змінної «themes» наведено на рисунку 2.6.

Використання Tailwind у проєктах докорінно змінює спосіб стилізації елементів сторінки і призводить до наявності своїх особливостей при розробці. Розглянемо плюси та мінуси Tailwind.

Отже, фреймворк має наступні переваги:

- швидкість розробки: базова комплектація фреймворка містить у собі класи, що охоплюють більшу частину вже існуючих властивостей CSS, зменшуючи час на стилізацію окремих елементів;

- адаптивність: наявність брейкпоїнтів та оптимізованість значень для мобільних пристроїв дозволяє спростити та пришвидшити розробку

адаптивних сторінок у порівнянні зі стандартним CSS;

- кастомізація: файл налаштувань «tailwind.config.js» дозволяє розробнику додавати власні значення для тих чи інших властивостей. Для значень, що використовуються у проєкті рідко, передбачена власна нотація, яка дозволяє записати значення напряму;

- мінімізація CSS-файлів: при побудові проєкту ті стилі, що не використовуються, видаляються. Це дозволяє оптимізувати використання місця та пам’яті проєктом;

- легка інтеграція: нерідко бібліотека використовується при розробці разом з іншими інструментами, такими як React.JS/Next.JS, Angular, Vue та іншими. Установку та первинне налаштування бібліотеки у проєкті можна легко провести засобами Node Packet Manager (npm).

Однак, разом із перевагами, фреймворк Tailwind CSS має також і ряд недоліків. Серед них:

- перевантаженість html-елементів: довгі списки класів безпосередньо впливають на читабельність коду, що в свою чергу може вплинути на процес відлагодження та супроводу програмного коду;

- крива навчання: бібліотека Tailwind складається з достатньо великої множини класів. Хоч шаблон класу достатньо простий та виглядає як «префікс:властивість-значення», але базова мова CSS сама по собі має велику кількість передбудованих властивостей, що призводить до того, що розробникам-новачка у Tailwind необхідно запам’ятати багато класів, аби працювати швидко та ефективно;

- інший підхід до розробки: при достатньо довгій роботі з Tailwind`ом розробник починає звикати до особливостей та переваг фреймворку, що може потім призвести до проблем при роботі на подальших проєктах без використання цієї бібліотеки.

Рисунок 2.6 – Опис власних значень шрифту, радіусу обрамлення та ширини елементу у файлі «tailwind.config.js»

2.5 Фреймворк React.js

React.js – фреймворк для створення інтерфейсів користувача, який був розроблений компанією Facebook (нині Meta), перша версія фреймоврку була випущена 29 травня 2013 року.

Основною особливістю React є використання компонентного підходу при розробці вебзастосунків [6]. Цей підхід полягає у розбитті кожної сторінки застосунку на множину частин, які згодом можна перевикористовувати з іншими значеннями – компоненти. Для реалізації такого підходу фреймворк використовує реактивне роботу коду – при кожній зміні стану відбувається ререндерінг (перемальовка) зміненого компонента. Це уможливлюється за допомогою використання віртуального дерева елементів документу (Virtual DOM), що звіряє зміни з реальним та оновлює лише ті елементи, що відрізняються. Це забезпечує високу продуктивність сторінок навіть при масштабних змінах станів.

Дані у підпорядкованих компонентах передаються лише у напрямку «батьківський компонент → дочірній компонент» за допомогою масиву

props. Основою реактивності бібліотеки є так звані хуки – спеціальні змінні, спрацювання підпорядкованих функцій яких запускає перемальовку дерева елементів (для useState) або підпорядкована функція яких запускається при зміні стану прослухованих даних (для useEffect).

Із переваг React можна виділити:

- сумісність: бібліотека легко підключається до існуючих проєктів;
- модульність: процес створення вебсторінки перетворюється на множину процесів створення менших за об’ємом автономніших компонентів, які можна використовувати у інших частинах застосунку;
- швидкість роботи: як було зазначено вище, React – оптимізований для своєї задачі фреймворк, і побудований відповідним чином.

Із недоліків же можна привести такі:

- складність маніпулювання між паралельними компонентами: для того, аби змінити стан паралельного елемента у ієрархії, необхідно створювати посилання на відповідні функції у компоненті батьківського рівня і як передавати їх у компонент-джерело змін, так і описувати логіку у компоненті-цілі. Це займає достатньо багато часу і іноді буває важко зрозумілим при відлагодженні коду;
- складність масштабування: розширення проєктів часто є навіть більшим викликом, аніж первинна розробка додатку.

2.6 Серверна мова PHP

Hypertext Preprocessor (PHP) – мова програмування вебзастосунків, націлена переважно на створення серверної їх частини. Мова була створена Расмусом Лердорфом у 1994 році у вигляді набору Perl-скриптів для обробки вебформ. Остання версія 8.3, була випущена 23 листопада 2023 року.

На сьогоднішній день PHP займає провідне місце у розробці серверних вебзастосунків, цією мовою написано приблизно сімдесят відсотків усіх таких продуктів [7]. Зокрема, на PHP написані такі сервіси, як Facebook,

Wikipedia, Wordpress та інші.

При роботі з PHP, розробник має можливість вводити код цією мовою безпосередньо у HTML-сторінку, використовуючи пару тегів <?php ?>. Таке введення дозволяє динамічно змінювати наповнення сторінки, оскільки усі зміни оброблюються на стороні сервера, а лише згодом передаються на сторону клієнта. У випадку роботи над даною кваліфікаційною роботою, така особливість PHP не є корисною, оскільки оновленням та промальовкою даних займається React.js.

Мова має достатньо простий синтаксис, і, як результат, легку криву навчання для новачків. Особливістю синтаксису PHP є використання знаку долару ($) перед назвами змінних і його відсутність у назвах флагів та функцій.

PHP – це динамічно типізована мова, а отже типи даних не присвоюються змінним на етапі компіляції, що спрощує процес розробки та додає нові можливості при створенні функцій, але взамін погіршує підтримуваність коду і може створити проблеми неправильної обробки даних.

Починаючи з версії мови 7.x, PHP додав можливість типізувати вхідні дані функцій. Оскільки ця особливість не є обов’язковою для використання, а лише додатковим інструментом керування обробкою інформації, то на типізацію мови серед інших це не вплинуло.

Будучи серверною мовою, PHP має можливість реагування на запити, наприклад, функції fetch() з JavaScript. Найбільш використовуваним типом запитів у цій роботі є Multipart Form Data – набір пар «ключ-значення», що записується у змінну $_POST мови при отриманні. Виділяючи одне з полів для опису дії, можливо реалізувати різні сценарії обробки інформації сервером.

Для уніфікації та полегшення веброзробки, мова має великий інструментарій для серіалізації та десеріалізації даних у такі формати, як JSON, XML, WDDX, YAML та інші.

Мова PHP має можливість роботи з БД, для цього використовується модуль mysqli. Для роботи з БД спочатку необхідно отримати доступ та екземпляр об’єкту бази даних, а згодом використовувати команди підготування, виставлення аргументів та перевірки виконання запиту.

Також у PHP є можливість роботи з файлами на сервері за допомогою тих же самих Multipart Form Data. Дані, передані таким чином, записуються у серверний масив $_FILES, де зберігаються дані про назву, розмір, тип, тимчасове місце розташування файлу, а також власне копія самого файлу.

Так само, як і функція fetch() у JS, у PHP є модуль cURL, що дозволяє створювати та оброблювати GET та POST-запити до зовнішніх джерел. Це буває корисно, якщо для роботи сервера необхідно використовувати віддалені дані та функції, доступ до яких напряму є ускладненим.

На сьогодні, основними сферами застосування PHP є:

- розробка CMS (систем керування змістом), таких як Wordpress, Drupal або Joomla;
- розробка інтернет-магазинів, таких як OpenCart або Magento;
- розробка CRM-систем, таких як SugarCRM;
- розробка соціальних мереж, форумів тощо;
- створення API на основі мови, які оброблюють та повертають інформацію без необхідності виведення її безпосередньо на клієнтську частину додатка;
- автоматизація обробки даних на стороні сервера, що потрібна для розвантаження фронтенду додатків.

Перевагами PHP є:

- простота у використанні: легка крива навчання для новачків дозволяє обрати цю мову для більшості серверних задач розробки вебдодатків;
- гнучкість: PHP є мовою з широким та доволі потужним інструментарієм, що дозволяє локалізувати велику частину процесів обробки даних саме у цій мові;
- швидкість: починаючи з версії 8.x мови, було введено механізм

компіляції JIT, що значно збільшило швидкість обробки інформації мовою;

- велика екосистема: будучи мовою з відкритим вихідним кодом, PHP створила навколо себе велику спільноту як користувачів, так і розробників, що вилилося у велику кількість користувацьких бібліотек та фреймворків для мови;

- доступність хостингу: мова, що займає більше половини ринку веброзробки, є першою у черзі на оптимізацію сумісності для серверів.

Однак, у PHP є і недоліки, серед яких:

- суперечливий код: динамічна типізація мови та часте дублювання функціоналу певних функцій можуть призвести до проблем з відлагодженням під час та супроводженням після процесу розробки;

- слабша продуктивність: хоч, починаючи з останніх версій, мова стала компілюватись швидше, але у порівнянні з Go або Node.js, PHP є повільнішим;

- застарілі проєкти та підходи: будучи мамонтом ринку, за всю історію веброзробки цією мовою було написано дуже велику кількість сайтів, код частини з яких не оновлювався довгий час, що може стати головним болем при супроводженні legacy-проєктів.

2.7 Мова структурованих запитів SQL

Structured Query Language (SQL) – це мова, призначена для роботи з реляційними базами даних [8]. Ядром мови є DML (Data Manipulation Language), що описує набір команд, які дозволяють отримувати (SELECT), змінювати (UPDATE), створювати (INSERT) та видаляти (DELETE) зміст реляційних таблиць даних.

Окрім DML, мова SQL також має у собі такі модулі:

- DDL (Data Definition Language), що дозволяє працювати зі структурою даних у базі, наприклад додавати (CREATE), налаштовувати (ALTER) та видаляти (DROP) таблиці або бази даних;

- DCL (Data Control Language), що дозволяє надавати (GRANT) та відкликати (REVOKE) привілеї тих чи інших користувачів;

- TCL (Transaction Control Language), що дозволяє керувати об’єднаними та цілісними послідовностями запитів до бази даних – транзакціями, зокрема підтверджувати (COMMIT), повністю (ROLLBACK) або частково (SAVEPOINT) відкочувати зміни у таких транзакціях.

Мова SQL є строго типізованою та підтримує числові (int/smallint/bigint/tinyint, decimal/numeric, float, double), строкові (char, varchar, text), датові (date, time, datetime) та інші (bit, boolean, blob) типи даних.

Мова базується на реляційній подачі даних, при якій вони організовані у таблиці з множиною стовпців (полів) та рядків (записів). Такі таблиці об’єднуються між собою за допомогою первинних та зовнішніх ключів. Множина об’єднаних таблиць записів даних і називається базою даних.

Первинним ключем називається стовпець з унікальним значенням для кожного запису. Зазвичай ним виступає цілочисленний ідентифікатор запису, що інкрементується для кожного нового запису у таблицю. У SQL існує атрибут PRIMARY KEY, що накладає саме такі обмеження: на унікальність значень стовпця, а також їх автоінкрементацію для будь-якого нового запису.

Зовнішнім ключем називається стовпець, тип даних якого повторює тип даних первинного ключа будь-якої іншої таблиці. Грубо кажучи, зовнішній ключ – це посилання на ідентифікатор запису іншої таблиці всередині заданого рядка. У SQL такі записи оголошуються у два етапи: спочатку іде опис типу даних самого стовпця (BIGINT fkey), а далі безпосереднє зв’язування його з іншою таблицею (FOREIGN KEY (`fkey`) REFERENCES sometable(`table_id`)). У пов’язаних таким чином таблицях неможливо буде видалити запис у таблиці-джерелі, якщо такий запис містить зовнішнє посилання на себе.

Окрім статичних команд роботи з базами даних, мова SQL також містить у собі збережені функції (повертають одне значення певного типу) та

процедури (повертають множину записів бази даних з налаштованою вибіркою стовпців) [10]. Їх використання корисно для випадків, коли за один запит до бази даних необхідно провести більше однієї дії різного типу або спростити виклик важких у написанні запитів, що часто використовуються під час роботи з нею.

Окрім власне даних, SQL має у собі набір інструментів для агрегації даних. Це такі функції, як COUNT() – підрахуванян кількості результатів запиту, AVG() – середнє значення результатів запиту, SUM() – сума значень результатів запиту, MAX() та MIN() – відповідно найбільший та найменший з результатів, тощо. Команда обрання даних SQL наведена у лістингу 2.5.

Лістинг 2.5 – Команда обрання даних SQL

```
SELECT поле1 [as назва1], [поле 2 [as назва2], …]
FROM назва_таблиці [join друга_таблиця]
WHERE поле_умови =/>/</>=/<=… значення_умови [AND/OR поле_умови2
… значення_умови2] GROUP BY стовпець_групування
LIMIT максимальна_кількість_записів
```

Для запитів оновлення використовується аргумент SET поле = нове_значення_поля, [поле2 = нове_значення2] у поєднанні з вибіркою WHERE. Якщо ж вибірки нема, такі значення будуть оновлені у всіх записах. Ключове слово FROM опускається, пишеться лише назва таблиці.

Для запитів додавання даних використовується аргумент VALUES, що дозволяє задавати дані нового запису у таблицю. Замість аргументу FROM використовується аргумент INTO з назвою таблиці-отримувача даних.

Вибірка використовується при оновленні та видаленні даних, при додаванні – жоден аргумент не використовується. Групування та ліміт використовуються лише у запитах вибору даних.

Загалом же, мова SQL має такі переваги:

- простота: мова задумувалася для спрощення роботи користувачів з даними, і її синтаксис зрозумілий людям, що знають базову англійську, оскільки заснований на живому описі дій мови;

- потужність: запити мови дозволяють працювати з великою кількістю записів водночас;

- гнучкість: кожен запит мови може бути ускладнено для отримання необхідних результатів, наприклад відсортовано та обмежено кількість отриманих записів;

- сумісність: мова SQL є стандартом мови при роботі з реляційними базами даних, і тому підтримується і використовується майже у всіх СУБД.

Але окрім плюсів, SQL страждає такими недоліками:

- неможливість обробляти нестандартні запити: без використання процедур, результат запиту є чітко структурованим і не допускає відхилень;

- продуктивність: без додаткової ручної оптимізації у вигляді індексування або кешування даних час обробки запитів може бути достатньо високим;

- відмінності між реалізаціями: основа мови (DML) є однаковою для усіх її діалектів та розширень, але синтаксис додаткових можливостей мови може відрізнятись від реалізації до реалізації.

3 КОДУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

Програмний застосунок створено на основі трьохшарової архітектури, сервер реалізовано як REST API, що виконує операції на локальному сервері бази даних.

3.1 Реалізація бази даних

3.1.1 Структура таблиць бази даних

База даних застосунку являє собою звичайну релятивну БД, написану діалектом MariaDB мови SQL. База даних складається з 13 таблиць. Таблиця customer_info надає дані про покупця, що зареєстрований у системі та складається з наступних полів (рисунок 3.1).

Назва	Тип даних	Обмеження	Опис
c_id	int	primary key, auto increment	Ідентифікатор кортежа даних
c_login_id	int	not null	Посилання на кортеж даних таблиці user_login_info
c_name	varchar(200)	unique	Ім’я покупця
c_phone	varchar(10)	unique, not null	Телефон покупця (10 символів, телефон без коду країни)

Рисунок 3.1 – Стовпці таблиці customer_info

Таблиця department_info надає дані про різні відділення мережі магазинів, використовується у операціях перевірки наявності відділення, створенні запитів на відправку отримання товарів між відділеннями та отримання даних про відділення. Склад таблиці наведено на рисунку 3.2.

Назва	Тип даних	Обмеження	Опис
dep_id	int	primary key, auto increment	Ідентифікатор кортежа даних
dep_address	varchar(200)	not null	Адреса відділення
dep_active	bit	–	Показує, чи активне відділення на даний момент

Рисунок 3.2 – Стовпці таблиці department_info

Таблиця employee_info (рисунок 3.3) надає загальні дані про працівника та редагується у адміністративній частині застосунку.

Назва	Тип даних	Обмеження	Опис
e_id	int	primary key, auto increment	Ідентифікатор кортежа даних
e_login_id	int	foreign key	Посилання на кортеж даних таблиці user_login_info
e_name	varchar (200)	not null	Ім’я продавця
e_dep_id	int	foreign key	Посилання на кортеж даних таблиці department_info
e_working	bit	–	Показує, чи доступний нині працівник
is_admin	bit	–	Показує, чи має працівник привілеї адміна

Рисунок 3.3 – Таблиця employee_info

Таблиця item_discounts (рисунок 3.4) описує дані про знижки на той чи інший товар, використовується працівниками зі статусом адміністратора для зміни даних про поточну ціну того чи іншого товару, а також дані з цієї таблиці відображаються для покупця.

Назва	Тип даних	Обмеження	Опис
di_id	bigint	primary key, auto increment	Ідентифікатор кортежа даних
di_item_id	bigint	foreign key, unique	Посилання на кортеж даних таблиці item_info
newprice	decimal (8, 2)	> 0.0	Ціна товару за знижкою

Рисунок 3.4 – Таблиця item_discounts

Таблиця item_images (рисунок 3.5) описує імена файлів зображень, що пов’язані з тим чи іншим товаром. Вона використовується як при роботі із адміністративною частиною застосунку (при редагуванні або додаванні зображень до товарів), так і у користувацькій частині клієнтського застосунку (при показі зображень товарів). Ця таблиця, як і попередня, містить три стовпці даних.

Назва	Тип даних	Обмеження	Опис
image_id	bigint	primary key, auto increment	Ідентифікатор кортежа даних
image_item_id	bigint	foreign key	Посилання на кортеж даних таблиці item_info
image_url	varchar (400)	not null	Назва файлу зображення, що відноситься до товару

Рисунок 3.5 – Таблиця item_images

Таблиця item_info (рисунок 3.6) містить у собі опис товарів та є опорною таблицею для багатьох інших. Ця таблиця використовується в усіх запитах клієнтського застосунку, що стосуються інформації про предмети, їх описи, ціни, знижки, запити на отримання/посилку товарів між відділеннями, формуванням відгуків тощо.

Назва	Тип даних	Обмеження	Опис
i_id	bigint	primary key, auto increment	Ідентифікатор кортежа даних
i_name	varchar (20)	not null	Назва товару
i_price	decimal (8,2)	> 0.0, not null	Ціна товару
i_unit	varchar (20)	not null	Одиниця виміру товару
i_active	bit	–	Чи активний товар

Рисунок 3.6 – Таблиця item_info

Таблиця item_requests (рисунок 3.7) містить дані про запити на перевезення тієї чи іншої кількості товару у відділення. Її дані можуть бути змінені лише за допомогою адміністративної частини клієнтського застосунку (як працівником-адміністратором, так і звичайним).

Назва	Тип даних	Обмеження	Опис
r_id	bigint	primary key, auto increment	Ідентифікатор кортежа даних
r_source_dep	int	foreign key, not null	Посилання на кортеж таблиці department_info (відправник)
r_target_dep	int	foreign key, not null	Посилання на кортеж таблиці department_info (призначення)
r_item_id	bigint	foreign key, not null	Посилання на кортеж таблиці item_info
r_amount	bit	> 0.0	Кількість замовленого товару

Рисунок 3.7 – Таблиця item_requests

Таблиця item_sort_tags (рисунок 3.8) містить дані про теґи того чи іншого товару, що використовуються для їх фільтрування та зміни у адміністративній частині клієнтського застосунку.

Назва	Тип даних	Обмеження	Опис
st_id	bigint	primary key, auto increment	Ідентифікатор кортежа даних
st_i_id	bigint	foreign key, not null	Посилання на кортеж таблиці item_info
st_tag	varchar (50)	not null	Тег товару

Рисунок 3.8 – Таблиця item_sort_tags

Таблиця orders (рисунок 3.9) містить інформацію про замовлення різноманітних товарів.

Назва	Тип даних	Обмеження	Опис
o_id	bigint	primary key, auto increment	Ідентифікатор кортежа даних
o_gr_id	bigint	not null	Ідентифікатор номеру замовлення для групування
o_it_id	bigint	foreign key, not null	Посилання на кортеж таблиці item_info
o_em_id	int	foreign key, not null	Посилання на кортеж таблиці employee_info
o_cu_id	int	foreign key, not null	Посилання на кортеж таблиці customer_info
o_dp_id	int	foreign key, not null	Посилання на кортеж таблиці department_info
o_price	decimal (8, 2)	> 0.0, not null	Ціна товару на момент придбання
o_amount	decimal (10, 2)	> 0.0, not null	Кількість придбаного товару

Рисунок 3.9 – Таблиця orders

Таблиця reviews (рисунок 3.10) містить дані про відгуки покупців на той чи інший товар, а також його оцінку.

Назва	Тип даних	Обмеження	Опис
r_id	bigint	primary key, auto increment	Ідентифікатор кортежа даних
r_item_id	bigint	foreign key, not null	Посилання на кортеж таблиці item_info
r_grade	decimal (2, 1)	not null	Оцінка товару
r_user_id	int	foreign key, not null	Посилання на кортеж таблиці customer_info
r_review_text	varchar (1000)	–	Відгук на товар

Рисунок 3.10 – Таблиця reviews

І, нарешті, таблиця user_login_info, містить дані для входу користувачів (як працівників, так і покупців). Ця таблиця одна із небагатьох, що модифікується як працівниками, так і покупцями при реєстрації у інтернет-магазині (рисунок 3.11). Повний опис структури БД наведено у додатку Б.

Назва	Тип даних	Обмеження	Опис
id	int	primary key, auto increment	Ідентифікатор кортежа даних
login	varchar (30)	not null, unique	Логін користувача
pass	varchar (100)	not null	Пароль користувача
employee	bit	–	Показує, чи є користувач покупцем, чи працівником

Рисунок 3.11 – Таблиця user_login_info

3.1.2 Процедури, що зберігаються у БД

Для одинарних взаємодій з базою даних (тобто таких, що не мають на меті повернення декількох кортежів результату водночас: прикладом таких взаємодій є перевірка наявності зареєстрованого користувача за наданими даними або відстеження транзакції реєстрації запиту певної кількості товару відділенням) використовуються збережені процедури. Результатом роботи цих процедур є кортеж, що завжди має у собі стовпець error, що позначає, чи була правильно виконана операція (або чи є співпадіння за наданими даними у БД). У залежності від значення стовпця error формуються й інші стовпці кортежу. Так, наприклад, якщо процедура завершилася з помилкою, то кортеж матиме усього два стовпці, другий з яких називатиметься errdesc та міститиме опис помилки, що виникла.

У випадку ж, якщо процедура не повертає помилку, кількість та назва стовпців результату може варіюватись у залежності від процедури, що виконувалася. Це може бути як і той самий стовпець errdesc, що повертатиме значення «Успішно!», або ж кортеж з бази даних з необхідними даними. Приклад внутрішньої процедури, що описується при створенні бази даних, наведено у лістингах 3.1-3.3 на прикладі процедури, що підтверджує отримання цільовим відділенням затребуваної кількості товару.

Лістинг 3.1 – Обробник помилок процедури confirm_incoming

```
create procedure confirm_incoming(id bigint) begin
    declare exit handler for sqlstate '45000' begin
        rollback ;
        select true as error, errdesc as errdesc; end;
```

На лістингу 3.1 описано обробник можливих помилок користувача, що виникають при роботі процедури. Звідси видно, яким чином формується її результат, і які стовпці даних будуть виведені у результаті виконання такої процедури. Важливо зазначити, що обробник будь-яких інших системних переривань БД описується по-іншому, аніж у попередньому лістингу.

Лістинг 3.2 – Опис можливих помилок процедури confirm_incoming

```
if needed > amount_t then begin set error = true; set errdesc =
'На складі немає необхідної кількості товару'; end;
     ...
        if(select row_count()) = 0 then begin set errdesc =
'Віднімання даних від отримувача провалено'; signal sqlstate
'45000'; end; end if;
...
        if(select row_count()) = 0 then begin set errdesc =
'Додавання даних до надсилача провалено'; signal sqlstate
'45000'; end; end if;
     ...
        if(select row_count()) = 0 then begin set errdesc =
'Видалення запису про запит провалено'; signal sqlstate '45000';
end; end if;
```

З лістингу 3.2 видно, що при можливій помилці значення змінної errdesc, що відповідає за опис власне помилки, змінюється відповідно. Згодом викликається стан 45000 (кастомна помилка), що був описаний у попередньому лістингу. На лістингу 3.3 показано, як відбувається завершення функції при відсутності помилок.

Лістинг 3.3 – Штатне завершення процедури confirm_incoming

```
set error = false; set errdesc = 'Успішно';
end;
end if;
select error as error, errdesc as errdesc;
end;
```

Як видно з лістингів 3.1-3.3, ця процедура слугує прикладом другого типу, оскільки і при помилці, і при успіху виконання повертає стовпець errdesc. Фрагмент коду процедури, що повертає кортеж даних у випадку успішного її закінчення, наведено у лістингу 3.4.

Лістинг 3.4 – Код результату успішного виконання функції

```
        ...
else select false as error, temp_login as login, temppass as
password;
        end if;
    end;
end if; end;
```

Більшість інших взаємодій з базою даних відбувається за допомогою серверної функції query_factory, що використовує бібліотеку mysqld для санітизації, перевірки та обробки результатів.

3.2 Серверна частина застосунку

Як було зазначено раніше, серверна частина застосунку створена на основі технології REST API, і виконує функції посередника між запитами клієнтської частини застосунку та результатами запитів серверу бази даних. Для цього сервер реалізовує два основних модулі, що зберігаються у відповідних php-файлах.Перший – це перехоплювач POST-запитів, що звіряє, яка дія має бути виконана сервером, а другий – це власне файл, що зберігає необхідні для виконання функції. Окрім цього, серверна частина застосунку також реалізовує функціонал для логування запитів користувача та відповідних функцій сервера, а також функції роботи з файлами на сервері. Усі ці модулі будуть детальніше описані нижче.

3.2.1 Перехоплювач POST-запитів

Файл requesthandler.php є фактично головним файлом для взаємодії з сервером, оскільки включає в себе файли усіх інших модулів. Структурно цей модуль являє собою великий блок switch-case, що перевіряє поле «action» вхідного POST-запиту, а в залежності від значення цього поля виконується та чи інша вкладена функція. Фрагмент функції перехоплення наведено на лістингу 3.5.

Лістинг 3.5 – Фрагмент коду функції перехоплення запитів

```
switch ($_POST['action']) {
  default:
    formlog_action_result_ua("невідома операція!",
"невизначено");
    misc_log_ua("Запит - " . $_POST['action']);
    echo 'no such action';
```

```
    break;
  case 'test-server':
    echo test_server();
    break;
  case 'test-db':
    echo external_db_check();
    break;
  case 'test-factory':
    echo factory_test();
    break;
  case 'get-items':
    echo get_items_info(); break;
```

3.2.2 Функції взаємодії серверу з базою даних

Файл mainfuncs.php містить у собі опис функцій, що викликаються модулем перехоплення запитів. Структурно складається з опису прапорців функції, опису даних для підключення до бази даних, двох функцій тестування підключення та функції отримання змінної бази даних, код якої буде наведено у лістингу 3.6.

Лістинг 3.6 – Функція get_db_handle()

```
function get_db_handle(): mysqli|false{
  global $db_name;
  global $db_pass;
  global $db_user;
  global $db_host;
  error_reporting(error_reporting() & ~E_WARNING);
  try {
    $db_handler = mysqli_connect($db_host, $db_user, $db_pass,
$db_name);
    if($db_handler === false) throw new Exception("database
connection fail!");
    else return $db_handler;
  }
  catch (Exception $e){ return false; }}
```

Функції тестування підключення використовуються для відображення статусу при завантаженні клієнтської частини застосунку та повертають текстову інформацію, а також логують цей самий результат у окремо виділений файл на сервері.

Наступним після цих двох функцій йде опис найбільшої атомарної

функції, що є основою для створення та реалізації усіх запитів до бази даних зі сторони сервера. Ця функція носить назву query_factory і буде описана по частинах. Лістинг 3.7 містить опис функції, що дозволить надалі краще зрозуміти, за що відповідає кожен з її аргументів.

Лістинг 3.7 – Опис функції query_factory()

```
function query_factory ($log_summary, $query, $inputs=[],
$sani_code_types="", $flags = 0, $error_prepend = '')
```

Аргумент log_summary використовується для логування дії та містить загальний опис запиту до бази даних. Цей аргумент є обов’язковим для виконання функції, оскільки інакше розібрати лог серверу було б майже неможливо.

Аргумент query – це власне запит до бази даних, причому у якості аргументів можуть використовуватись як знаки питання (?) при санітизованому запиті, так і самі значення. Другий підхід не реалізовано у коді, але його можливо використовувати надалі.

Аргумент inputs типу «масив» не є обов’язковим і використовується у запитах з аргументами, причому змінні у ньому мають бути описані у порядку, у якому вони зустрічаються у запиті до бази даних.

Аргумент sani_code_types типу «строка» так само не є обов’язковим і використовується у санітизованих запитах для опису типів даних, причому порядок так само має зберігатись відповідно масиву inputs. Опис санітизованих даних наступний: «s» – текстова строка, «i» – цілочисельний тип, «d» – число з плаваючою крапкою, «b» – blob, бінарні дані, що передаються у БД пакетами (зазвичай файли).

Аргумент flags – прапорці виконання функції. Можливі три прапорці: CHECK_ZERO_ROWS, CHECK_PROCEDURE_ERROR, CHECK_ROWS_ AFFECTED. Перший прапорець відповідає за перевірку нульової кількості кортежів відповіді (тобто чи існують взагалі результати вибірки), другий – перевіряє кількість змінених стовпців для запитів INSERT/DELETE до бази

даних, третій використовується у запитах виклику процедури бази даних та перевіряє, чи процедура не повернула помилку.

Аргумент error_prepend – необов'язковий, використовується для логування та відповідає за підпис перед основною інформацією щодо результату виконання логованого запиту.

У лістингу 3.8 наведено код першої частини функції, що отримує посилання на БД, а також виконує власне запит до бази даних. Перевірка даних відповідно заданих прапорців йде далі.

Лістинг 3.8 – Перша частина функції query_factory()

```
$db = get_db_handle();
  try{
    if (!$db) throw new Exception("підключення до БД відсутнє");
    $new_query = $db->prepare($query);
    if(!($sani_code_types === (null||'') || $inputs ==
(null||[])))
      $new_query->bind_param($sani_code_types, ...$inputs);
    if(!$new_query->execute()) throw new Exception('запит до БД
не було виконано');
    $data = [];
…
}
```

З коду видно, що ця частина функції відповідає за отримання екземпляру посилання на БД, аналізує його та видає текстову помилку у випадку, якщо під'єднання до бази даних не було успішним (тобто якщо функція з' єднання повернула false). Далі, функція виконує санітизований запит до БД і у залежності від переданого масиву змінних та санітарних типів, вшиває ці дані у запит. Якщо ж запит не пройшов, функція повертає помилку «Запит до БД не було виконано». У лістингу 3.9 наведено частину функції, що відповідає за перевірку значень, що були повернені при виконанні запитів вибірки та виклику процедур.

Лістинг 3.9 – Друга частина функції query_factory()

```
if(str_starts_with(strtolower($query), 'select') ||
str_starts_with(strtolower($query), 'call')) {
```

```
      $result = $new_query->get_result();
      while ($row = $result->fetch_assoc()) $data[] = $row;
      $new_query->free_result();
      $new_query->close();
      if(($flags & CHECK_ZERO_ROWS) && $data === []) throw new
Exception('запит не повернув жодного рядка');
      if($flags & CHECK_PROCEDURE_ERROR){
        if(str_starts_with(strtolower($query), 'call') &&
$data[0]['error'] == (1||true))
          throw new Exception($data[0]['errdesc']);
      }
    }
```

Цей алгоритмічний блок, як і було сказано раніше, виконується, якщо наведений запит до БД починається зі слова «select» (тобто є запитом на вибірку даних) або «call» (тобто це виклик процедури). При виконанні отримується результат запиту до БД, результат порядково записується у масив data і згодом дані цього масиву звіряються з заданими прапорцями функції. Якщо поставлено флаг CHECK_ZERO_ROWS, видається помилка при пустому масиві data, що повідомляє користувачу, що запит не повернув жодного рядка результату.

Якщо поставлено флаг CHECK_PROCEDURE_ERROR, відбувається додаткова перевірка запиту, чи починається він зі слова «call» (тобто чи є цей запит викликом процедури), і якщо перший рядок результату містить значення true у полі error, то блок повертає помилку з описом, взятим з поля errdesc результату запиту.

Однак, окрім запитів вибірки та виконання процедур, використовуються також процедури вставлення та видалення кортежів даних з БД. Алгоритмічний блок, що відповідає за перевірку отриманого результату запиту у цьому випадку, наведено у лістингу 3.10.

Лістинг 3.10 – Третя частина функції query_factory()

```
else {$data = ['affected_rows' => $new_query->affected_rows];
      if
($data['affected_rows'] === 0 &&
($flags & CHECK_ROWS_AFFECTED))
throw new Exception('insert/update: жодного рядка не змінено');
    }
```

Цей блок спрацьовує у всіх інших випадках (технічно, він спрацює і при запитах UPDATE, MODIFY, ALTER тощо, але усі вони інкапсульовані у збережених функціях БД) і замість полів результату запиту отримує службове значення «affected_rows», що відображає кількість змінених запитом INSERT або DELETE рядків. Якщо поставлено прапорець CHECK_ROWS_AFFECTED і значення кількості змінених рядків дорівнює нулю, функція повертає відповідну помилку. У лістингу 3.11 наведено алгоритм стандартного завершення роботи функції.

Лістинг 3.11 – Передостання частина функції query_factory()

```
mysqli_close($db);
    formlog_action_result_ua($log_summary,  ($error_prepend === 
'' ? '' : $error_prepend.': ').'успішний результат загальної 
функції запиту');
    return ['error' => false, 'data' => $data];  }
```

Звідси видно, що при відсутності захоплених помилок під час виконання функції, вона повертає асоціативний масив, що складається з ключа «error», що є false, та ключа «data», що власне містить дані, які повернув запит/процедура. Також результат виконання функції логується, і якщо у функцію передано префікс помилки, його також буде наведено у лозі серверу. Враховуючи, що основна частина функції обернена у блок try, вона містить блок catch, що відповідає за обробку отриманих у ході виконання функції помилок. Код блоку catch наведено у лістингу 3.12.

Лістинг 3.12 – Фінальна частина функції query_factory()

```
catch (Exception $e){  mysqli_close($db);
    formlog_action_result_ua($log_summary, str_replace("\n", '', 
($error_prepend === '' ? '' : $error_prepend.': ').$e-
>getMessage()));
    return ['error' => true, 'errdesc' => ($error_prepend === '' 
? '' : $error_prepend.': ').$e->getMessage()];  }
```

Як можна побачити з тексту блоку catch, він виконує усі ті ж самі дії, що й звичайний кінець функції, але у масиві повертає значення true поля

«error», а замість поля «data» з результатами запиту до БД, однак, містить поле «errdesc», що є описом помилки, що виникла під час роботи функції (або кастомної помилки, викликаної внутрішніми перевірками).

Окрім функції query_factory(), файл також містить функцію result_check(), що використовується для додаткових перевірок масиву даних результату запиту, сформованого функцією query_factory(). Код вторинної функції наведено у лістингу 3.13.

Лістинг 3.13 – Функція result_check()

```
function result_check(&$data, $bool_converts=[]){
  formlog_action_result_ua("Перевірка результатів запиту: ",
($data['error'] === false ? 'успіх' : 'помилка –
'.$data['errdesc']));
  if($data['error'] === false && $bool_converts !== [])
    foreach ($data['data'] as &$el)
      foreach($bool_converts as $f)
        if(isset($el[$f])) $el[$f] = $el[$f] === 1;
  return !$data['error'];}
```

З лістингу 3.13 видно, що ця функція спрощує перевірку значення поля «error» результату, а в якості додаткового функціоналу містить можливість перетворення обраних користувачем полів у булевий (двійковий) тип даних. Для цього використовується аргумент bool_converts, за замовчуванням є пустим масивом. Функція повертає true, якщо запит «пройшов». Пара функцій query_factory() та check_result() є основою при виконанні запитів до бази даних зі сторони сервера та використовується у більшості з них.

3.2.3 Модуль роботи з файлами на сервері

Модуль роботи з файлами серверу використовується при роботі з зображеннями у адміністраторському застосунку, містить опис функцій додавання, зміни та видалення зображень із директорії сервера, а також функцію генерації випадкового рядку даних заданої довжини. Файлові функції використовують інструментарій PHP, а саме функції unlink(),

move_uploaded_file(), array_key_exists для перевірки, чи FORM-запит містить файл та функцію file_exists() для перевірки наявності файлу вже на сервері. Код функції remove_image наведено у лістингу 3.14.

Лістинг 3.14 – Функція remove_image()

```
function remove_image($filename){
  $log = "Видалення зображення товару";
  global $imagedir;
  try {    unlink($imagedir . $filename);
    if(file_exists($imagedir.$filename)) throw new
Exception('Файл не було видалено!');
    $res = remove_image_entry($filename);
    if($res['error']) throw new Exception($res['errdesc']);
    formlog_action_result_ua($log, "Успішно");
    return ['error'=>false, 'res' => 'успіх'];} catch (Exception
$e){
    formlog_action_result_ua($log, $e->getMessage());
    return ['error' => true, 'errdesc' => $e->getMessage()];  }}
```

Як видно з коду 3.14, половину функції займає перевірка, чи існує файл та чи було його видалено. Аналогічні перевірки описані і у двох інших функціях. Як і у випадку з функцією створення запиту, ця повертає асоціативний масив з обов’язковим полем «error».

Функція генерації випадкового текстового рядка не потребує пояснення, її код наведено у лістингу 3.15.

Лістинг 3.15 – Функція generate_random_strings_nums()

```
function generate_random_strings_nums($len){
  $alphabet = '0123456789abcdefghijklmnopq
rstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
  $r = '';
  for($i = 0; $i < $len; $i++)
    $r .= $alphabet[mt_rand(0, strlen($alphabet)-1)];
  return $r; }
```

3.2.4 Модуль логування

Модуль логування даних представлений файлом logging.php та складається з шести функцій, три з яких стосуються безпосередньо

зберігання даних на сервері, а три інших – відповідають за роботу з історією запитів між відділеннями. Найбільш використовуваною функцією є formlog_action_result_ua(), код якої наведено у лістингу 3.16.

Лістинг 3.16 – Функція formlog_action_result_ua

```
function formlog_action_result_ua($action, $result){
  $log = "\n[". $_SERVER['HTTP_ORIGIN'] . "], FormData: " .
str_replace("     ", " ", str_replace("\n", ' ',
print_r($_POST, true, ))) . "\n" .
    date('Y-m-d H:i:s') . ': виконано дію – ' . $action . ",
результат – " . $result;
  file_put_contents(__DIR__."/log.txt",    $log.PHP_EOL,
FILE_APPEND);
}
```

Як видно з лістингу 3.16, ця функція, окрім власне дії і її результату, записує час запиту до сервера та власне поля запиту. Усе це потрібно для відлагодження та можливого спостереження за поведінкою сервера при різних заданих даних POST-запиту.

3.3 Клієнтська частина застосунку

Завдяки використанню фреймворку React, клієнтська частина застосунку реалізована як множина компонентів, що взаємодіють між собою за допомогою переданих змінних та функцій. Кожен окремий компонент має велику кількість коду. На лістингу 3.17 наведено фрагмент файлу App.js, що є кореневим файлом усього застосунку, а саме його return-частину, що показує, як оголошуються використані компоненти у коді. Хоч передані дані і не наведені у лістингу, цей лістинг слугує для наочного показу загального уявлення про код, написаний на React.

Лістинг 3.17 – Фрагмент файлу App.js

```
return (
      <>
        <PopupWindow message={popupInfo[0]}
useTimer={popupInfo[1]} timeout={popupInfo[2]}
```

```
modal={popupInfo[3]}/>
       <LoginForm apiUrl={server_URL}
popupHandler={setPopupInfo} setAdminVisibility={(visible) =>
{setAdminVisibility(visible);}} workerRef={workerInfo}
            outerVisibility={loginVisibility}
setThisVisible={(value) => setLoginVisibility(value)}
setStoreVisibility={(value) => setStoreVisibility(value)}
        customerInfo={customerInfo}/>
        <AdminPage outerVisibility={adminVisibility}
apiURL={server_URL} popupHandler={setPopupInfo}
workerInfo={workerInfo.current}
            thisVisibilityHandler={(val) =>
{setAdminVisibility(val)}} loginVisibilityHandler={(val) =>
setLoginVisibility(val)}
                    clearLoginInfo={() => {workerInfo.current =
[]; console.log(workerInfo.current)}}/>
        <Webstore visibility={storeVisibility}
apiURL={server_URL} popupHandler={setPopupInfo}
customerInfo={customerInfo.current}/>
      </>
  );
```

Як було зазначено раніше, для роботи з сервером клієнтський застосунок використовує POST-запити з множиною полів, обов’язковим з яких є поле «action». Для створення таких запитів і їх відправки використовуються чотири функції, що будуть розглянуті надалі.

Перша з них – це функція arrayToFormData(), що, як виходить з назви, перетворює заданий масив строк у JS-об’єкт FormData, тіло якого і посилається у POST-запиті. Код цієї функції наведено у лістингу 3.18.

Лістинг 3.18 – Функція arrayToFormData()

```
const arrayToFormData = (data) => {

    let formData = new FormData();
    if(!Array.isArray(data) || data.length % 2 === 1) return
null;
    for(let i = 0; i < data.length / 2; i++)
        formData.append(data[i*2], data[i*2 + 1]);
    return formData;
}
```

Друга функція – це fetchPostRequest(), що власне й посилає одинарний POST-запит на сервер. Її код наведено у лістингу 3.19.

Лістинг 3.19 – Функція fetchPostRequest()

```
const fetchPostRequest = async (formData, callbackFunction =
undefined, reduceToSingle = false) => {
    const apiAddress = 'http://192.168.0.108/kurs-project-
server/requesthandler.php';
    try{ const rq = await fetch(apiAddress, {method: "POST",
body: formData});
        if(!rq.ok) throw new Error('Запит не повернув статус
200!');
        let rp = await rq.json();
        if(rp.error)
          throw new Error(rp.errdesc);
        if(callbackFunction !== undefined &&
Array.isArray(rp.data))
            rp.data = rp.data.map(callbackFunction);
        return {
error: false, data: (rp.data.length === 1 && reduceToSingle) ?
rp.data[0]:rp.data}
    } catch (e){ return {error: true, errdesc: e.toString()}  }}
```

Функція приймає три аргументи, два з яких не є обов’язковими. Параметр formData формується попередньо описаною функцією, callbackFunction – це функція, що застосовується для усіх елементів масиву даних, значення за замовчуванням – функція, яка жодним чином не змінює елемент та повертає його ж. Параметр reduceToSingle дозволяє обрати, чи буде повернено масив з даними результату запиту, чи лише його перший елемент. Це корисно для запитів, що повертають виключно одне поле, а також допомагає у адаптації результату до раніше написаного коду.

Третя функція – makeMassRequestObject(), що використовується для створення об’єкту массового запиту на сервер (лістинг 3.20).

Лістинг 3.20 – Функція makeMassRequestObject()

```
const makeMassRequestObject = (formdata, callback = ((el) =>
el), makeSingular = false) => {
    return {formdata: formdata, callback: callback, makeSingle:
makeSingular}}
```

Об’єкт, створений цією функцією, надалі використовується у функції massPostRequestFetch(), що необхідна для массової відправки запитів на сервер та обробки отриманих результатів. Код цієї функції наведено у

лістингу 3.21.

Лістинг 3.21 – Функція massPostRequestFetch()

```
const massPostRequestFetch = async(fetchData) =>
{    if(!Array.isArray(fetchData))
       return null;
    const responses = await Promise.all(
        fetchData.map(el => fetchPostRequest(el.formdata,
el.callback, el.makeSingle)));
    if(responses.find(el => el.error) === undefined)
        return {error: false, data: responses.map(el => el.data)
}
    else
{        let errorMessage = '';
        responses.map((el, index) => {
            if(el.error) errorMessage += `Запит ${index} -
${el.errdesc}!\n`;
        }
)
        return {error: true, errdesc: errorMessage}
    }
}
```

Параметр fetchData – це і є об’єкт, створений попередньо описаною функцією, makeMassRequestObject(). Ця ж функція запускає декілька паралельних запитів до сервера та очікує на закінчення їх усіх (тобто час виконання цієї функції є часом найдовшої з відповідей сервера) незалежно від того, чи був запит успішним, чи ні. Згодом, для кожного запиту, що повернув помилку, функція повертає об’єкт з полями error:true та errdesc, що власне описує усі помилки. Якщо ж жодної помилки не виникло, функція повертає об’єкт з полем error = false та полем data, що є масивом результатів запитів до сервера.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Адміністративна частина застосунку

При переході за адресою застосунку, користувача зустрічає сторінка з формою входу, яка наведена на рисунку 4.1.

Рисунок 4.1 – Форма входу користувача

При введенні неправильних або неактуальних даних користувача буде повідомлено про помилку за допомогою статичного компоненту PopupWindow (рисунок 4.2). У ході роботи програми, цей компонент в основному використовується для обробки помилок.

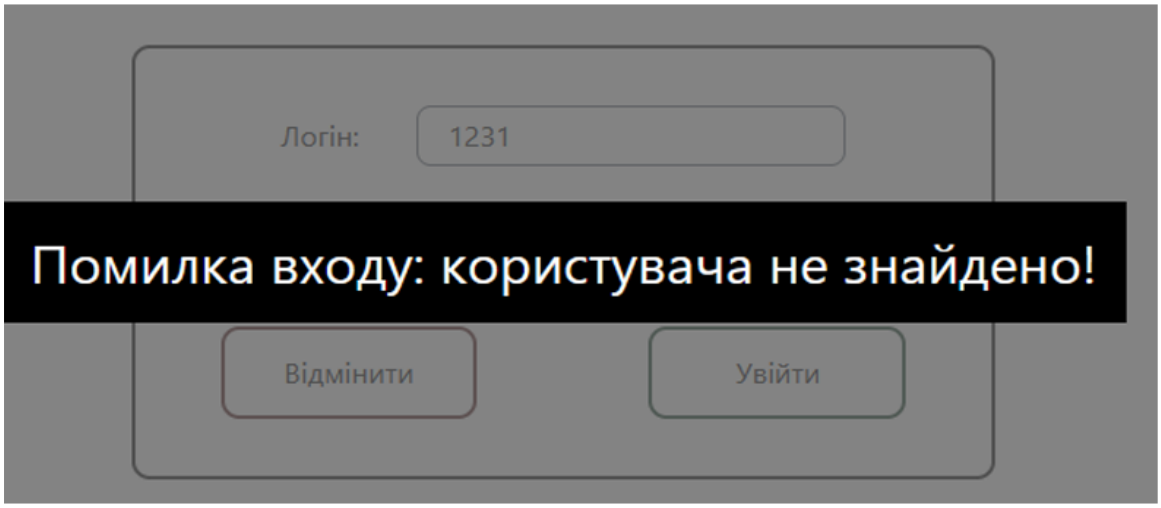


Рисунок 4.2 – Приклад помилки, яку може отримати користувач

Після валідації даних користувача зустріне головне меню застосунку, що містить кнопки переходу до вкладок та кнопку виходу (рисунок 4.3).

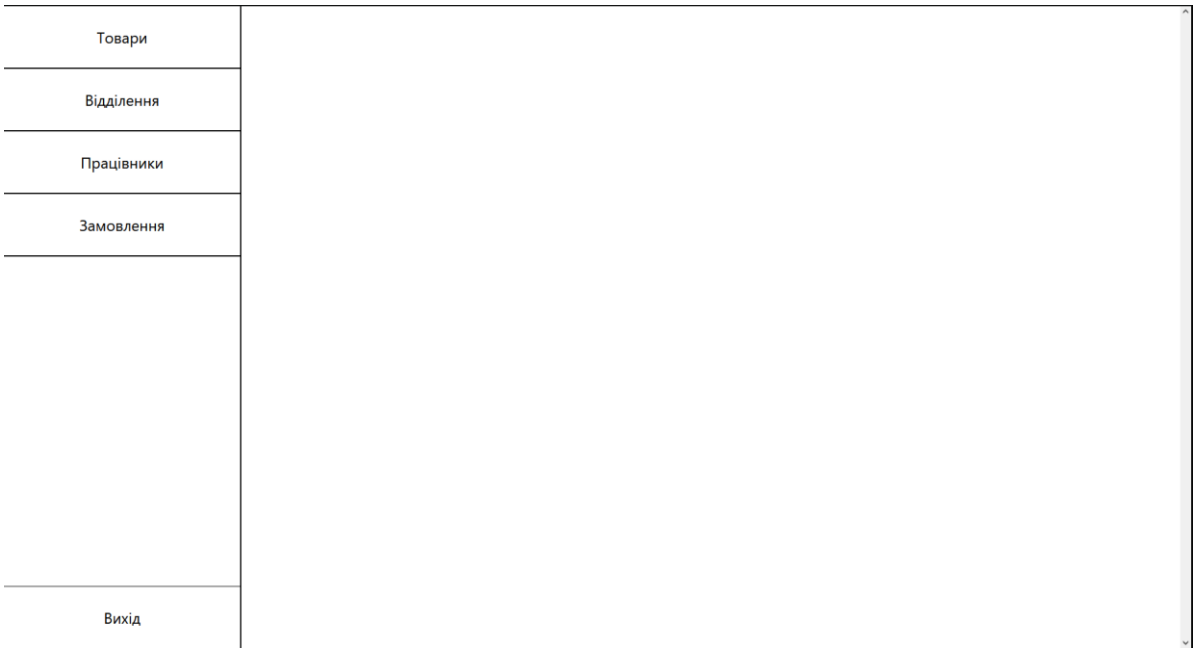


Рисунок 4.3 – Головне меню додатка

При виборі вкладки «Товари» користувача зустріне меню з кнопками переходу до підменю «Загальна інформація», «Зображення», «Описи» та «Теги» (рисунок 4.4).

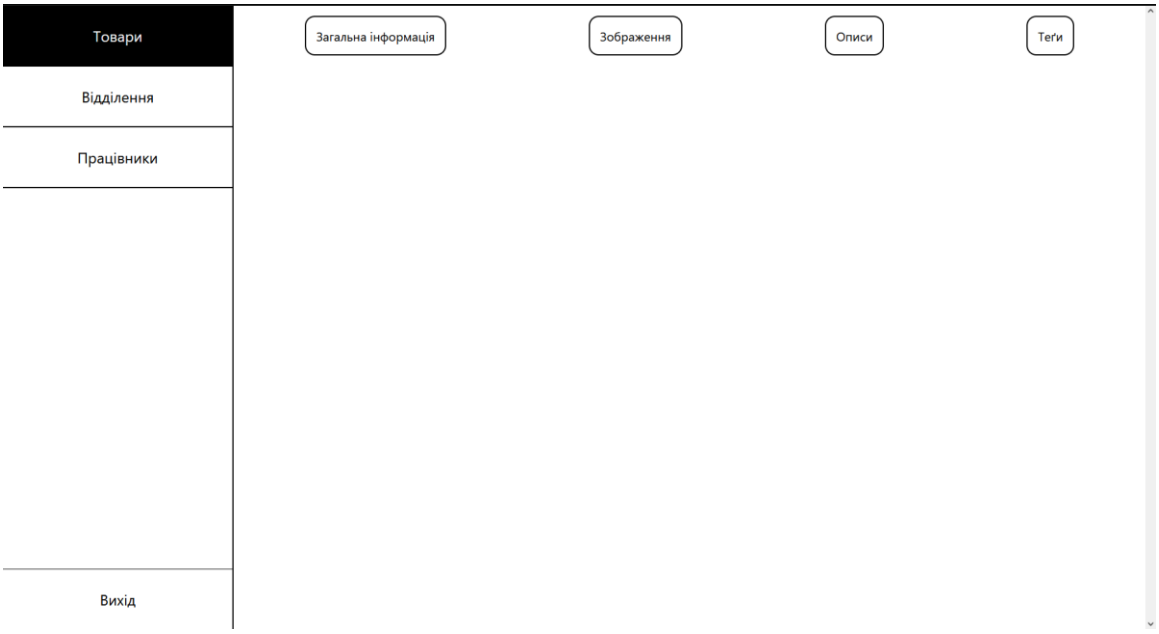


Рисунок 4.4 – Вкладка «Товари» додатка

Підменю «Загальна інформація» дозволяє користувачу редагувати або додавати дані про товари (рисунок 4.5), «Зображення» – додавати, змінювати або видаляти зображення для товарів (рисунок 4.6), «Описи» – змінювати текст опису того чи іншого товару (рисунок 4.7), а «Теги» – змінювати або додавати теги до товарів (рисунок 4.8).

Рисунок 4.5 – Підменю «Загальна інформація»

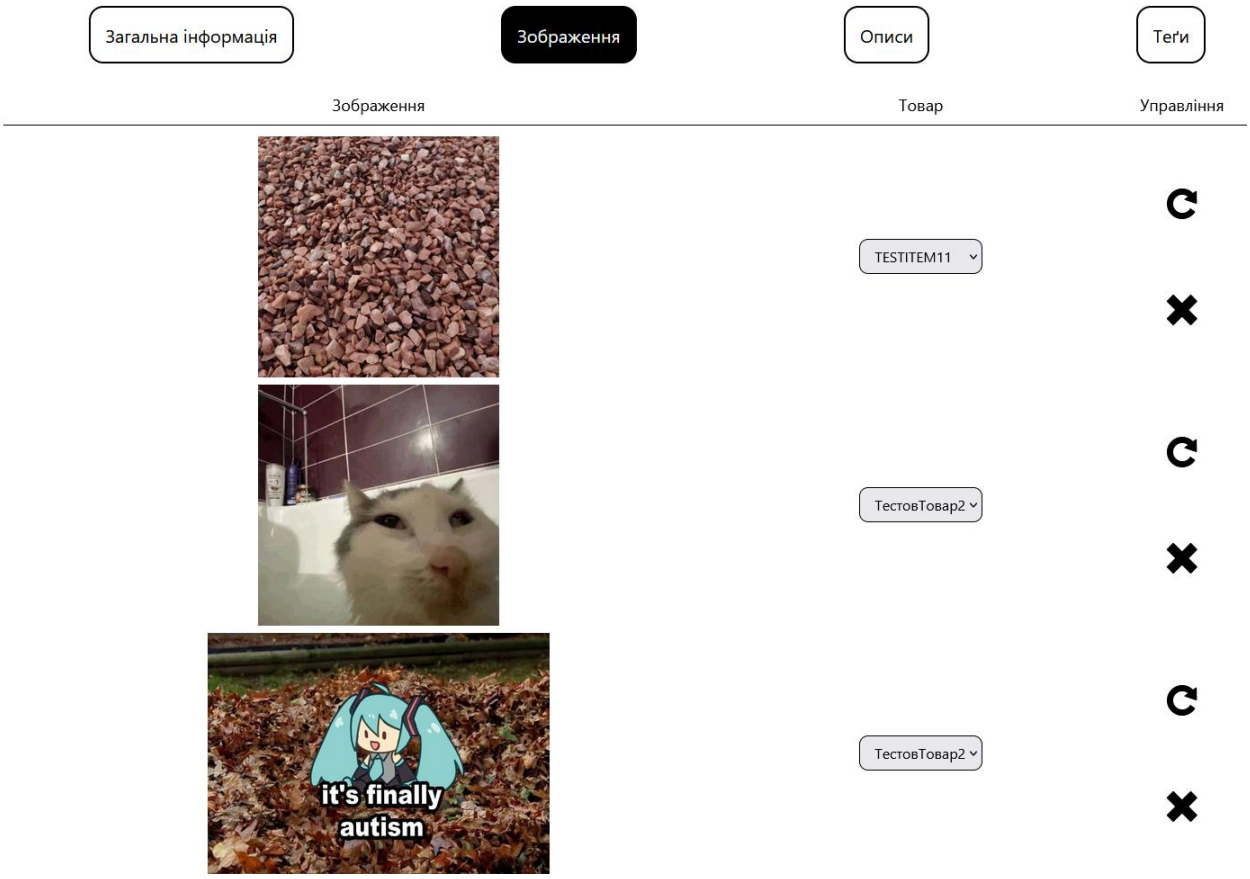


Рисунок 4.6 – Підменю «Зображення»

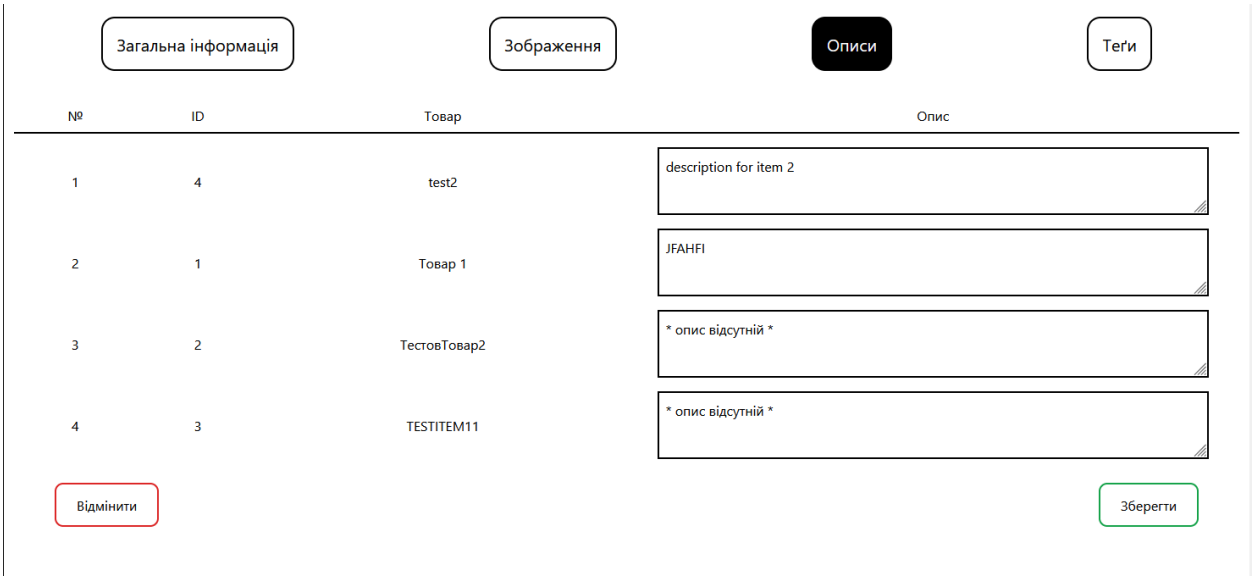


Рисунок 4.7 – Підменю «Описи»

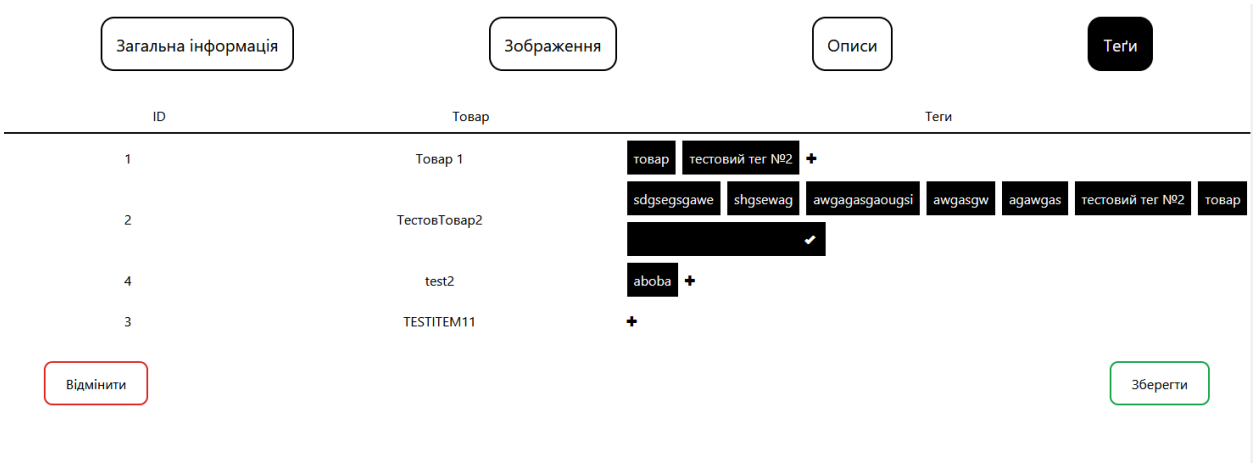


Рисунок 4.8 – Підменю «Теги»

Елементи керування на сторінках та меню виконані у вигляді піктограм, тож не потребують додаткових пояснень. Текстові кнопки («Відмінити» та «Зберегти») відповідно відміняють внесені зміни та посилають запит на збереження даних на сервер. Вкладка «Товари» є однаковою як для звичайного робітника, так і для адміністратора. Ці вкладки відрізняються між цими двома ролями, і вони будуть наведені у двох екземплярах. Вкладка «Відділення» надає дані про відділення, історію запитів та дані про наявність тих чи інших товарів у відділенні. Адміністраторам ця вкладка також дозволяє створювати та керувати

запитами на отримання товарів між відділеннями.

На рисунку 4.9 наведено загальний вигляд вкладки «Товари» для адміністратора. При вході у систему працівника, що не має адміністративних привілеїв, ця вкладка не матиме підменю «Вхідні запити» та «Вихідні запити».

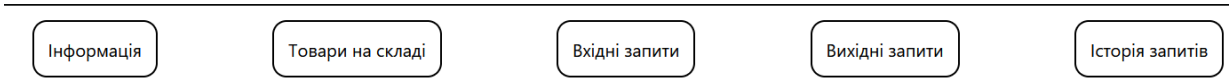


Рисунок 4.9 – Підменю вкладки «Відділення»

Підменю «Інформація» використовується для отримання даних про наявні відділення, а у випадку адміністративного доступу до системи дозволяє змінити статус активності відповідного адміністратору відділення. При цьому, галочки активності інших відділень, хоч і є, але вони неактивні – користувач не може з ними взаємодіяти, а отже і не може змінити статус активності іншого, аніж своє, відділення. На рисунках 4.10, 4.11 наведено зовнішній вигляд підменю «Інформація» відповідно для адміністратора та працівника, що не має адміністративного доступу.

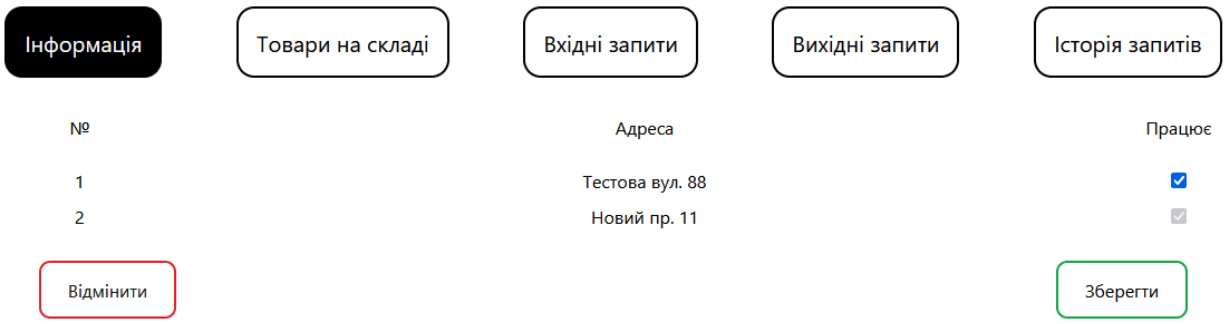


Рисунок 4.10 – Підменю «Інформація», адміністративна версія

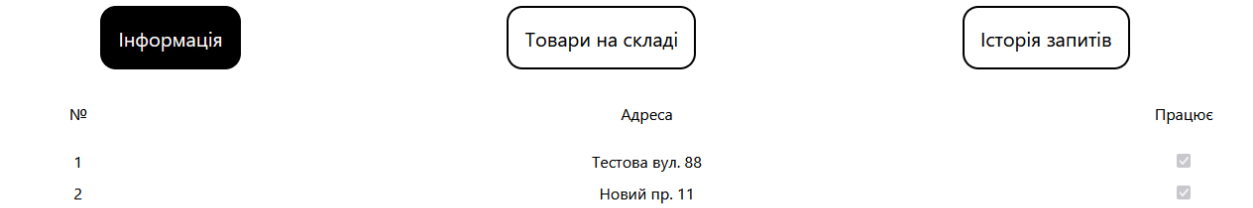


Рисунок 4.11 – Підменю «Інформація», версія без адміністративного доступу

Підменю «Товари на складі» не може бути змінена жодною роллю, відповідно є однаковою як для працівників-адміністраторів, так і для інших працівників. Це підменю надає інформацію про наявні на складі відповідного працівнику відділення товари та їх кількість. Зовнішній вигляд підменю наведено на рисунку 4.12. Натискання на стовпець «ID», «Назва» чи «Кількість» дозволяє відсортувати елементи списку за відповідним полем.

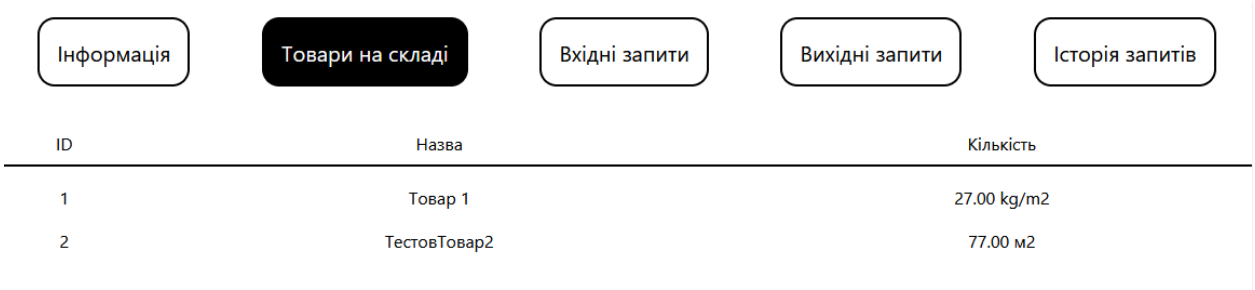


Рисунок 4.12 – Підменю «Товари на складі»

Аналогічно попередньому, підменю «Історія запитів», що доступне усім користувачам, виглядає однаково як для адміністраторів, так і для не-адміністраторів. Підменю так само підтримує функцію сортування та зображено на рисунку 4.13.

Рисунок 4.13 – Підменю «Історія запитів»

Наступні дві вкладки, «Вхідні запити» та «Вихідні запити», як і було зазначено раніше, доступні лише адміністраторам. Вони використовуються для створення та керування запитами на отримання товарів між відділеннями. За допомогою цих же вкладок можна побачити поставки зі складу до різних відділень. Склад є відділенням 0 у базі даних, є за замовчуванням неактивним і не показується у відповідній вкладці адміністративної частини застосунку. Зовнішній вигляд підменю «Вхідні запити» показано на рисунку 4.14. Так само, як і попередні, список у цій вкладці підтримує сортування.

Рисунок 4.14 – Підменю «Вхідні запити»

Як видно з рисунку, кожен запит має два елементи керування ним: галочку та хрестик, що відповідно підтверджують або спростовують факт отримання товару за цим запитом. На рисунку 4.15 зображено зовнішній вигляд підменю «Вихідні запити», що використовується для перевірки,

створення та відміни запитів з відділення користувача до інших відділень. Аналогічно, є можливість сортування даних за стовпцями. Варто зазначити, що вихідні запити мають лише одну кнопку керування – це «Відмінити» (мінус у крузі), що дозволяє відмінити створений запит.

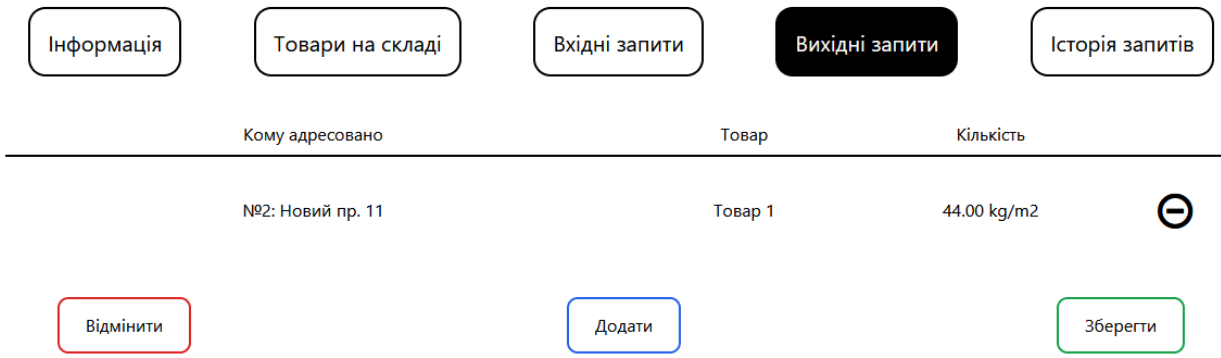


Рисунок 4.15 – Підменю «Вихідні запити»

Зазначте, що прийнятий або спростований іншим відділенням запит не з’явиться тут, а одночасна спроба виконання двох дій призведе до помилки. При натисканні кнопки «Додати» з’явиться нове поле запиту (рисунок 4.16), що дозволяє власне створити новий запит до іншого відділення (в тому числі і до відділення №0, складу).

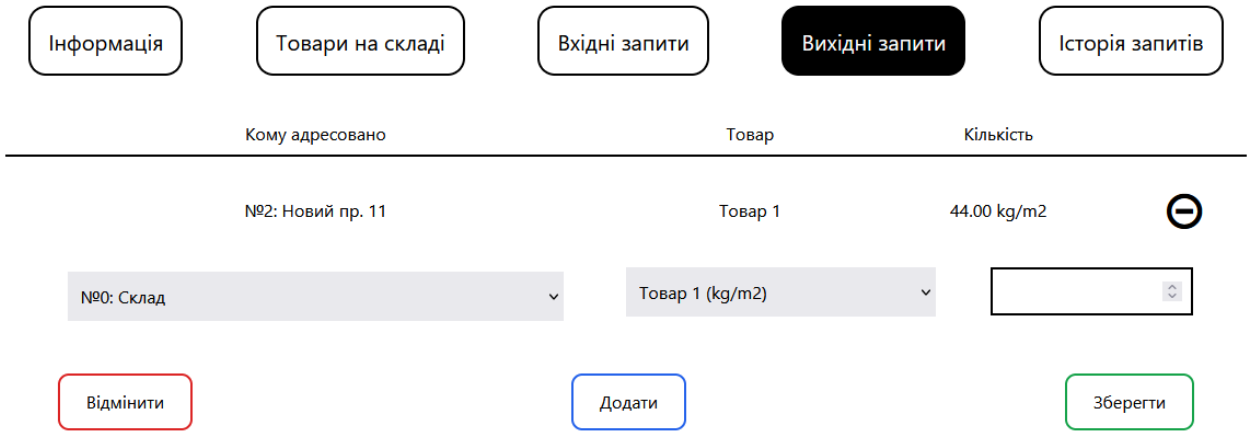


Рисунок 4.16 – Зовнішній вигляд підменю при додаванні нового запиту

Перший випадаючий список відповідає за відділення-ціль запиту, другий – за товар, що потребує переміщення, а текстове поле – за кількість

товару, яку необхідно перемістити. При неправильному введенні даних у текстове поле буде створено запит на 0 одиниць товару.

Вкладка «Працівники» відповідає за отримання та зміну даних про працівників. Вона містить єдине для усіх ролей підменю «Інформація», а друге підменю відрізняється у залежності від ролі працівника, що користується системою. Для ролі «Адміністратор» це «Логін/пароль працівників», для робітника – «Мій логін/пароль». Зовнішній вигляд вкладки наведено на рисунку 4.17.

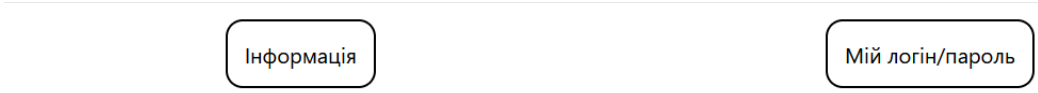


Рисунок 4.17 – Вкладка «Працівники», версія не-адміністратора

Для ролі адміністратора, вкладка «Інформація» використовується для отримання даних про усіх працівників або зміни активності та адміністраторських привілеїв для працівників того ж відділення. Зовнішній вигляд адміністраторської версії підменю «Інформація» наведено на рисунку 4.18.

Рисунок 4.18 – Підменю «Інформація» вкладки «Працівники», адміністраторська версія

Зауважу, що при додаванні нового працівника адміністратор може обрати будь-яке відділення для нового працівника, але при виборі відмінного

відділення, не зможе редагувати дані та дозволи цього працівника. Система автоматично згенерує для нього тимчасовий логін та пароль, що можна буде побачити на вкладці «Логін/пароль працівників». Зовнішній вигляд підменю при додаванні нового працівника наведено на рисунку 4.19.

Рисунок 4.19 – Додавання нового працівника

Вкладка «Інформація» для працівника не-адміністратора використовується виключно для ознайомлення та отримання загальної інформації про всіх працівників, зареєстрованих у системі, разом із можливістю подивитись, чи активний той чи інший працівник, і чи має він адміністративні привілеї. Зовнішній вигляд цієї вкладки наведено на рисунку 4.20.

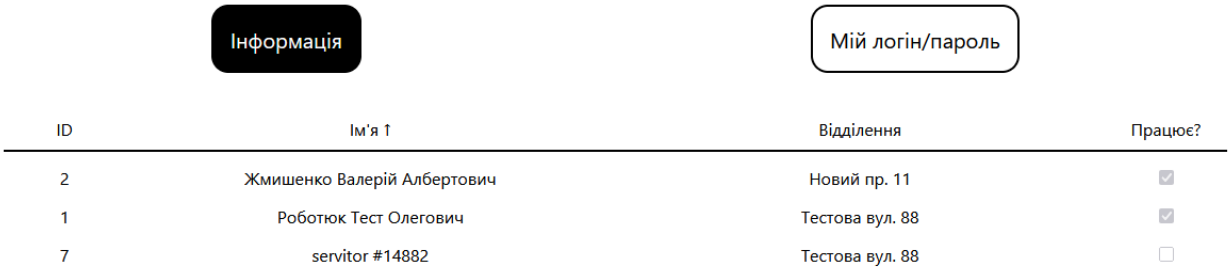


Рисунок 4.20 – Підменю «Інформація» для не-адміністратора

Адміністративне підменю «Логін/пароль працівників» використовується для ознайомлення та отримання облікових даних усіх працівників. Зовнішній вигляд цього підменю наведено на рисунку 4.21.

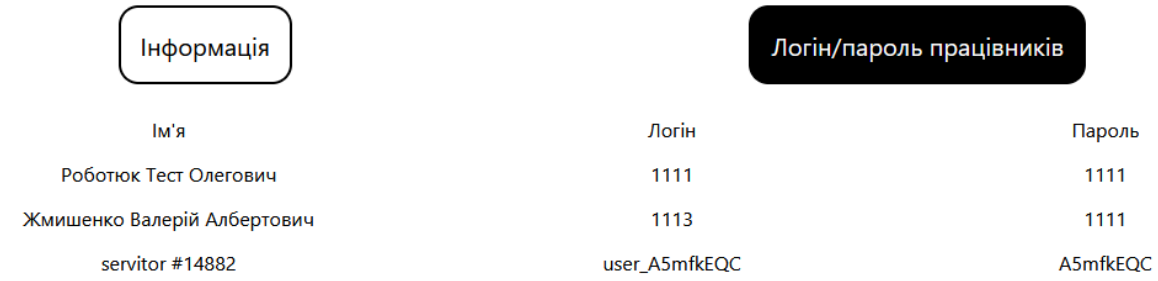


Рисунок 4.21 – Підменю «Логін/пароль працівників»

Для працівника без адміністративних привілеїв це підменю має назву «Мій логін\пароль», і може використовуватися для перегляду або зміни своїх облікових даних у системі. Зовнішній вигляд цього підменю для працівника без адміністративних привілеїв наведено на рисунку 4.22.

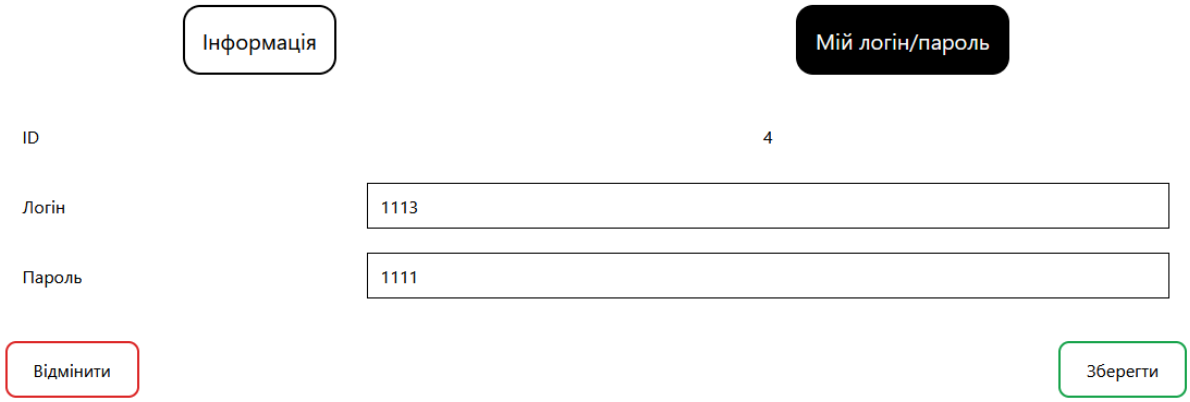


Рисунок 4.22 – Підменю «Мій логін/пароль»

Вкладка «Замовлення» складається з двох підменю, що виглядають однаково для обох ролей. Ці підменю: «Замовлення у процесі» та «Історія замовлень». Перше підменю використовується для затвердження або відхилення з будь-яких причин того чи іншого замовлення клієнта та містить дані про власне замовлення (рисунок 4.23).

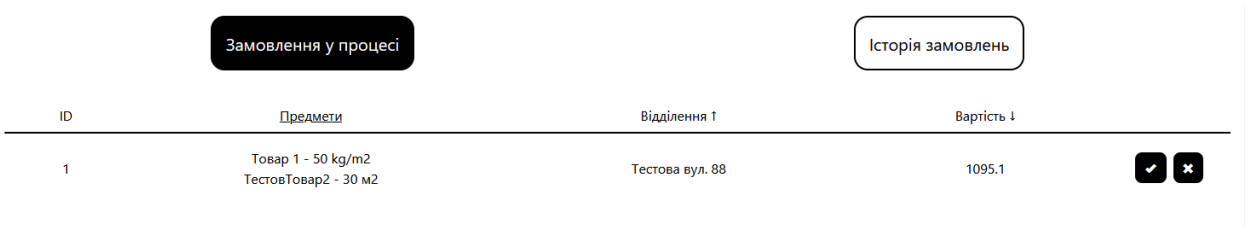


Рисунок 4.23 – Підменю «Замовлення у процесі»

При підтвердженні замовлення буде виконано перевірку на наявність товарів спочатку у відділенні, а потім на складі. Якщо товари є на складі, замовлення буде створено з відповідним повідомленням (рисунок 4.24).

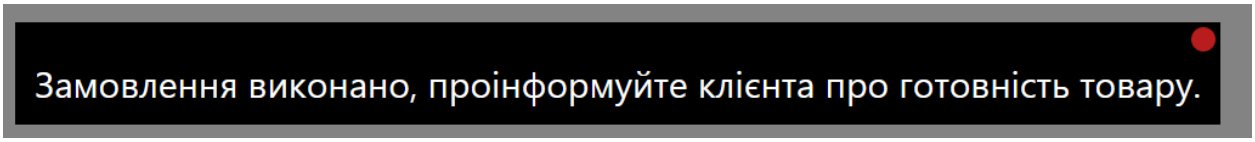


Рисунок 4.24 – Повідомлення про успішне створення замовлення

Якщо ж товарів немає у відділенні, але необхідна їх кількість є на складі (відділення 0), то користувач отримає повідомлення про створення нової заявки на товар (рисунок 4.25).

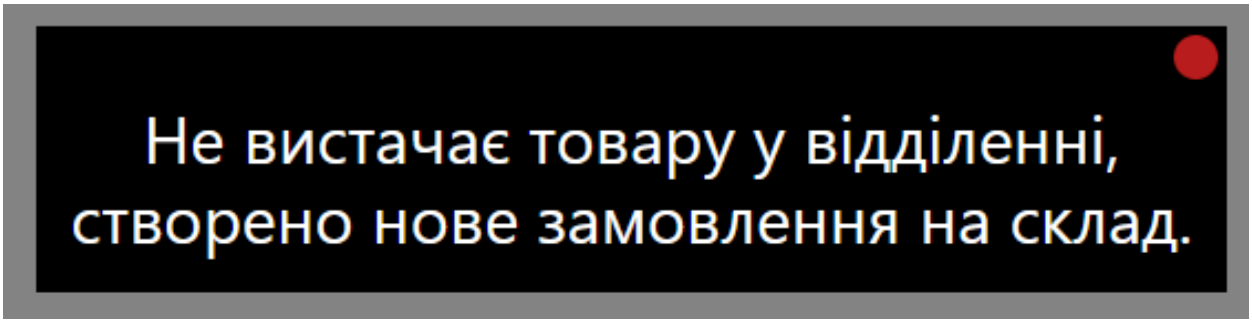


Рисунок 4.25 – Повідомлення про створення замовлення на склад

Якщо ж ані у відділенні, ані на складі немає потрібної кількості, то користувач отримає повідомлення, що пропонує вручну створити замовлення на той чи інший товар у інші відділення (рисунок 4.26).

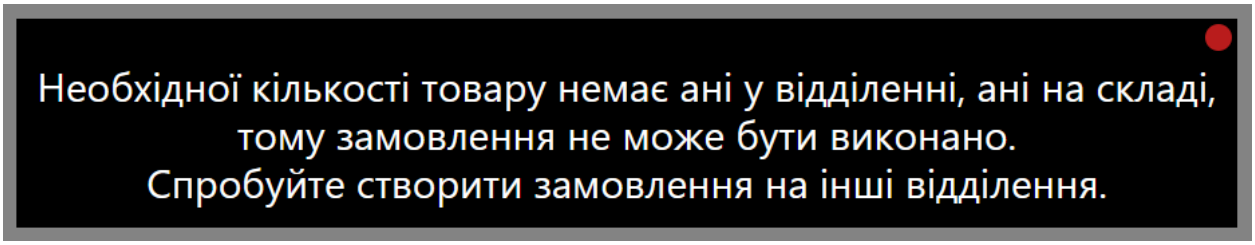


Рисунок 4.26 – Повідомлення про відсутність товару на складі

При натисненні кнопки скасування замовлення буде скасовано, а користувачу показано відповідне повідомлення (рисунок 4.27).

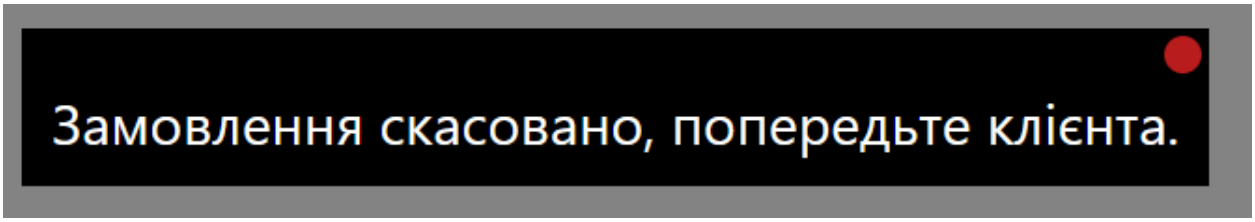


Рисунок 4.27 – Повідомлення про скасування замовлення

Підменю «Історія замовлень» є ознайомчим та відображає історію створених раніше замовлень, що були затверджені (рисунок 4.28).

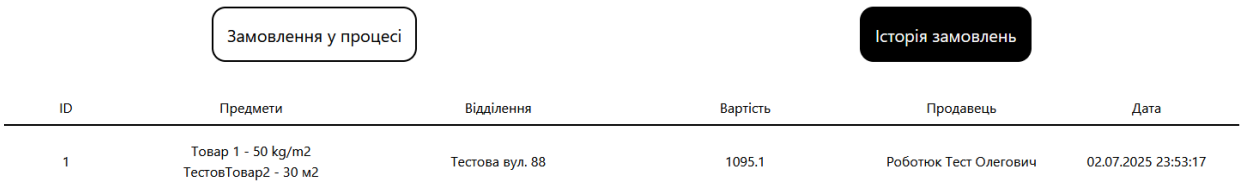


Рисунок 4.28 – Підменю «Історія замовлень»

Останньою кнопкою у адміністративній частині додатку є «Вихід», що повертає користувача до форми входу (рисунок 4.1).

4.2 Веб-магазин

Для переходу у веб-магазин користувачу у формі входу необхідно ввести дані користувача-покупця або обрати опцію «Увійти як гість».

Остання дозволить переглянути загальний асортимент мережі магазинів без можливості формування кошика. При вході у секцію веб-магазину, покупця зустріне вікно, наведене на рисунку 4.29. Головні елементи керування у ньому знаходяться у верхній частині сторінки, а піктограми є інтуїтивно зрозумілими. Розглянемо сторінку детальніше.

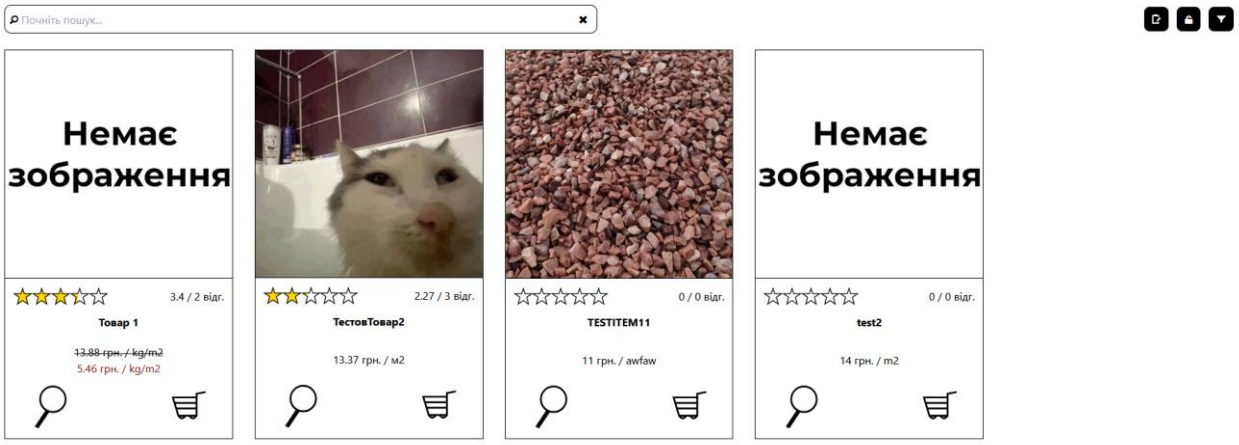


Рисунок 4.29 – Головне вікно веб-магазину

При введенні тексту в поле «Почніть пошук», наявні елементи списку будуть відфільтровані за назвою, і буде видно лише товари, що містять введений у пошуку текст. Приклад пошуку наведено на рисунку 4.30.

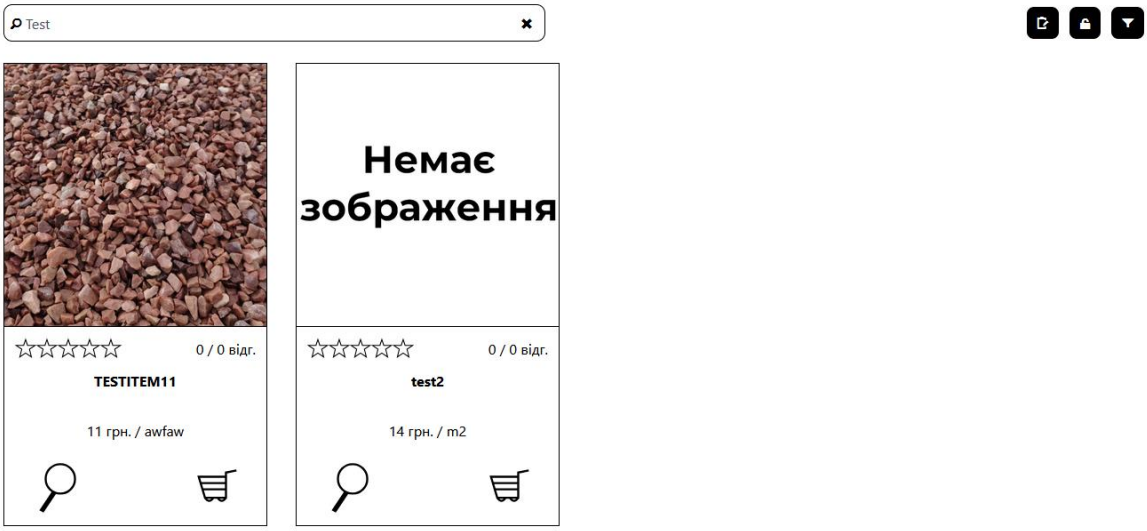


Рисунок 4.30 – Пошук товарів

При введенні тексту у поле пошуку з’явиться кнопка-хрестик, що дозволяє очистити поле та оновити список товарів. Кнопка зі знаком фільтру відповідає за фільтрування елементів (товарів) за їх тегами (теги обираються у відповідній вкладці адміністративного додатку). На рисунку 4.31 наведено процес фільтрування товарів за тегами.

Рисунок 4.31 – Фільтрація товарів за тегами

При натисненні на кнопку збільшувального скла на картці товару відкриється повнорозмірне відображення товару, що містить його зображення, опис, теги та огляди на товар. Зовнішній вигляд такого відображення наведено на рисунку 4.32.

Рисунок 4.32 – Повне відображення товару

При натисненні на кнопку напроти тексту «Відгуки» відкриється форма створення нового відгуку на товар, що наведена на рисунку 4.33.

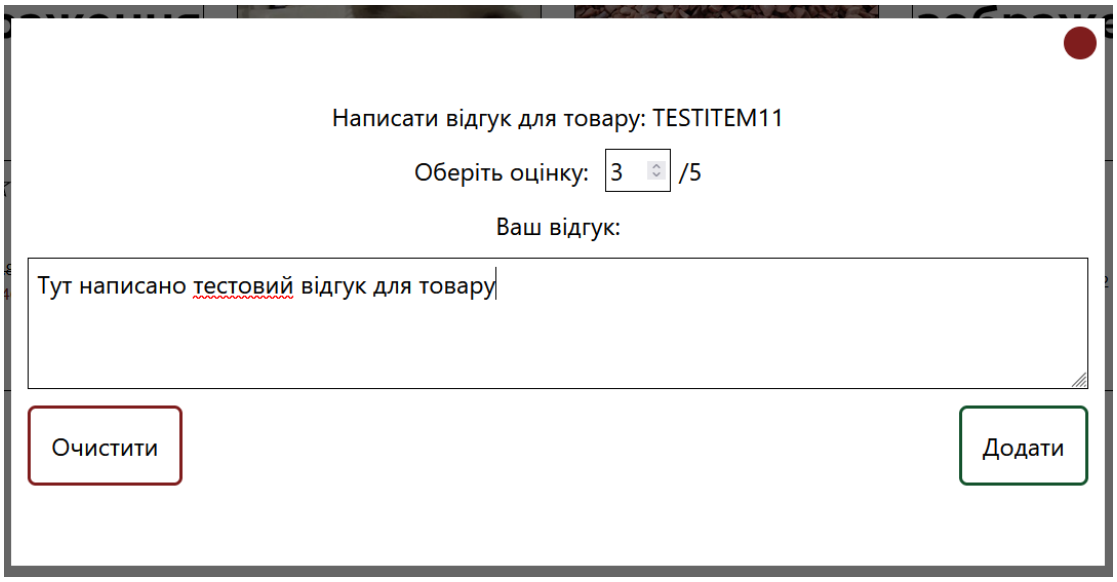


Рисунок 4.33 – Форма створення відгуку на товар

Кнопка «Очистити» очистить текстове поле та виставить значення оцінки у значення за замовчуванням – 1, кнопка «Додати» – додасть відгук у базу даних. Варто зауважити, що відгуки можуть писати лише зареєстровані користувачі, інакше при натисненні на кнопку створення відгуку користувачу буде показана відповідна помилка (рисунок 4.34).

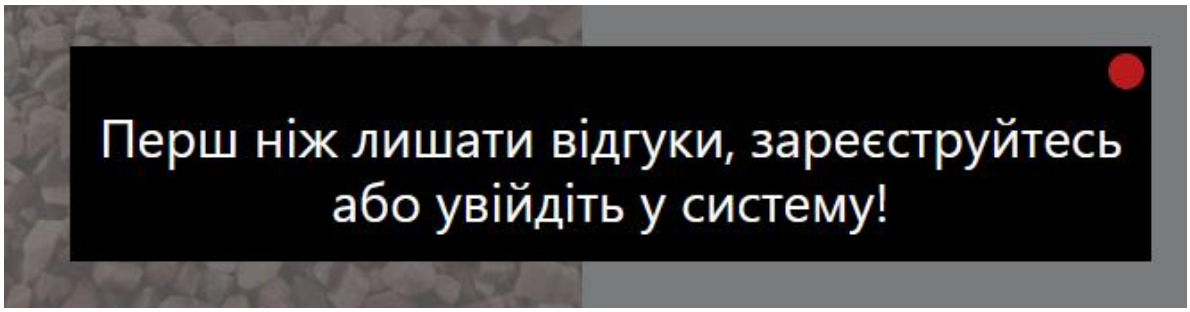


Рисунок 4.34 – Сповіщення про незареєстрованого користувача

Аналогічне повідомлення зустріне незареєстрованого користувача, що спробує додати товар до кошика (рисунок 4.35).

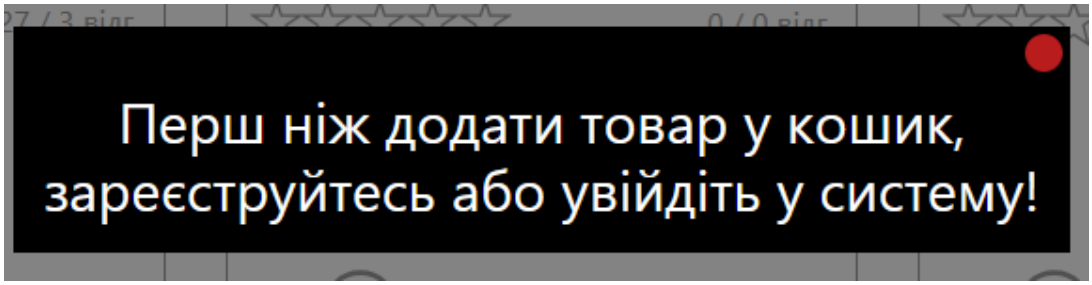


Рисунок 4.35 – Сповіщення про незареєстрованого користувача при додаванні товару

Якщо ж товар до кошика додає зареєстрований користувач, його так само зустрічає відповідне повідомлення (рисунок 4.36).

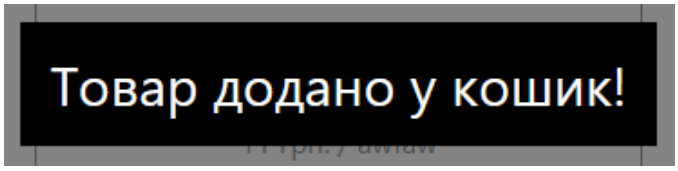


Рисунок 4.36 – Повідомлення про успішне додання товару

Для реєстрації та входу у верхній частині сторінки веб-магазину передбачено дві відповідні кнопки. Кнопка з зображенням планшету відповідає за відкриття вікна реєстрації, що показано на рисунку 4.37.

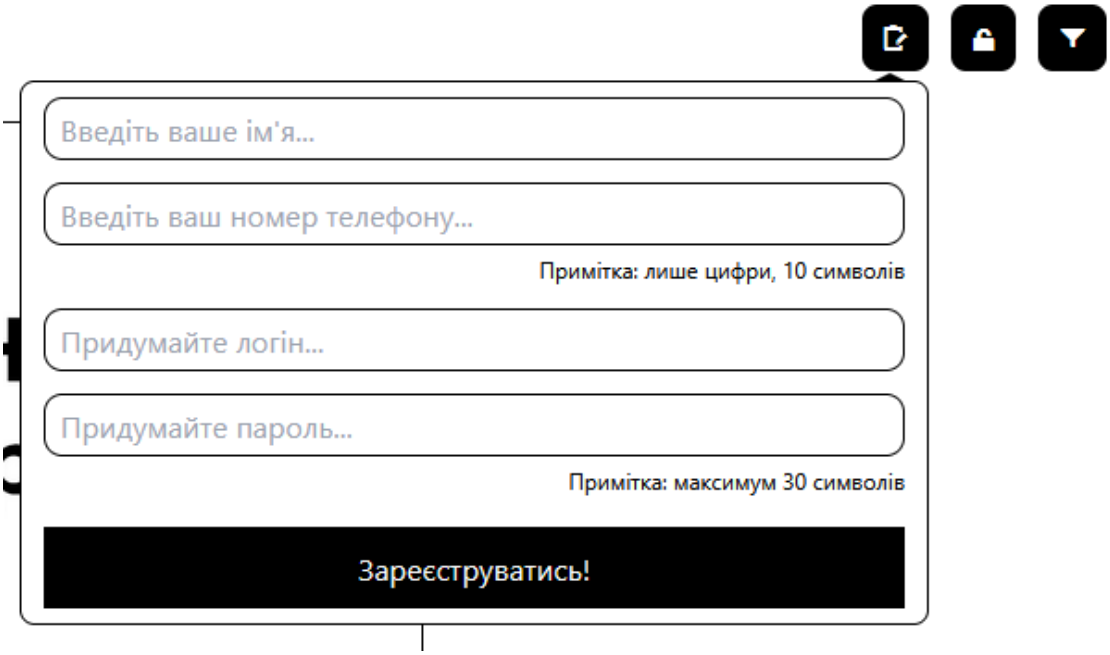


Рисунок 4.37 – Розгорнуте вікно реєстрації користувача

В залежності від результату реєстрації користувач отримає або повідомлення про успішну реєстрацію (рисунок 4.38), або повідомлення про необхідність додаткової перевірки полів інформації (рисунок 4.39).

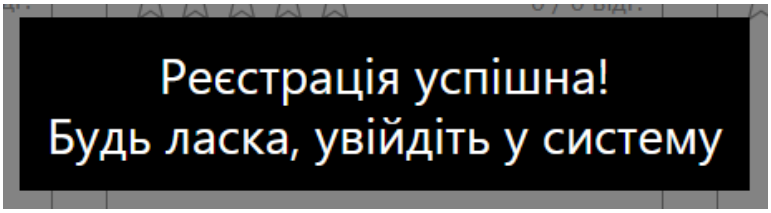


Рисунок 4.38 – Повідомлення про успішну реєстрацію користувача

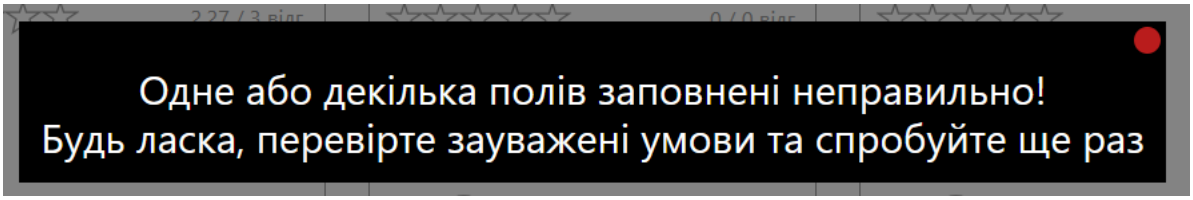


Рисунок 4.39 – Повідомлення про додаткову перевірку полів реєстрації

Кнопка із зображенням відчиненого амбарного замку у верхній частині сторінки магазину відповідає за відкриття форми входу користувача (рисунок 4.40).

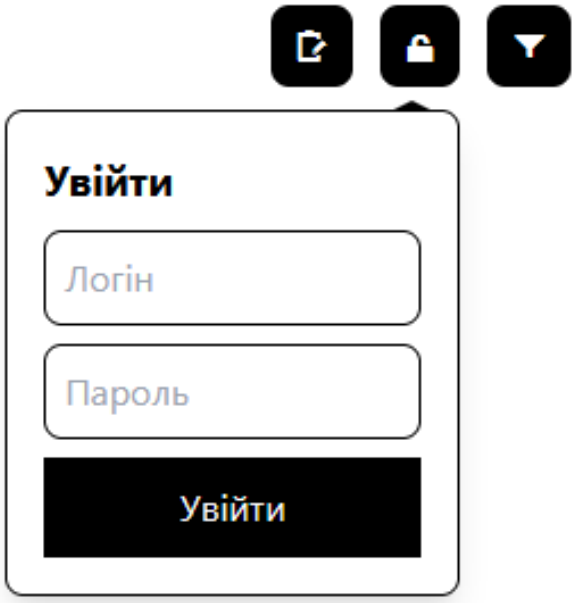


Рисунок 4.40 – Розгорнуте вікно входу користувача

В залежності від результату, користувач отримає або вітальне сповіщення (рисунок 4.41), що є показником успішного входу, або повідомлення про помилку (рисунок 4.42).

Рисунок 4.41 – Вітальне сповіщення у випадку успішного входу користувача

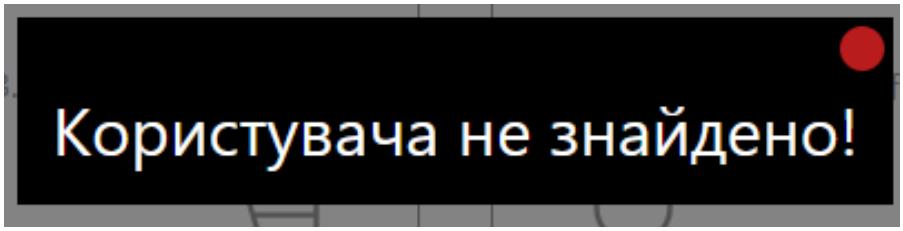


Рисунок 4.42 – Сповіщення про помилку входу користувача

У разі успішного входу користувача, дві кнопки у верхній частині сторінки магазину зникнуть, а замість них з’явиться кнопка з зображенням торса, що є аналогом особистого кабінету користувача (рисунок 4.43).

Рисунок 4.43 – Розгорнуте вікно особистих даних користувача

При натисненні на кнопку з зображенням олівця покупець може змінити свої облікові дані. В такому випадку текстове поле буде замінено на елемент введення даних, а кнопка зміни буде прихована. Замість неї з’явиться дві кнопки: підтвердження та скасування змін (рисунок 4.44).

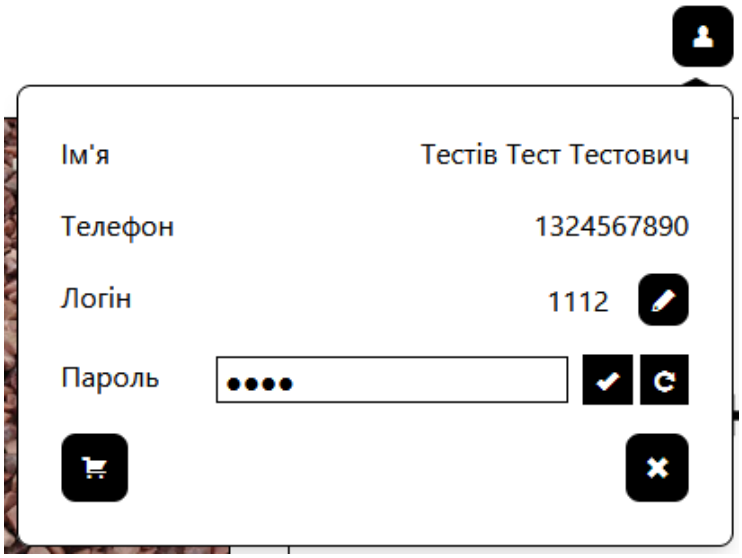


Рисунок 4.44 – Вікно даних користувача при зміні поля паролю

При натисненні на кнопку з зображенням хреста буде виконано вихід користувача з системи, а зовнішній вигляд верхньої частини сторінки буде скинуто до першпочаткового вигляду. Натискання ж на кнопку кошика перенесе користувача до окремого меню, що дозволить обрати необхідну кількість товару (рисунок 4.45).

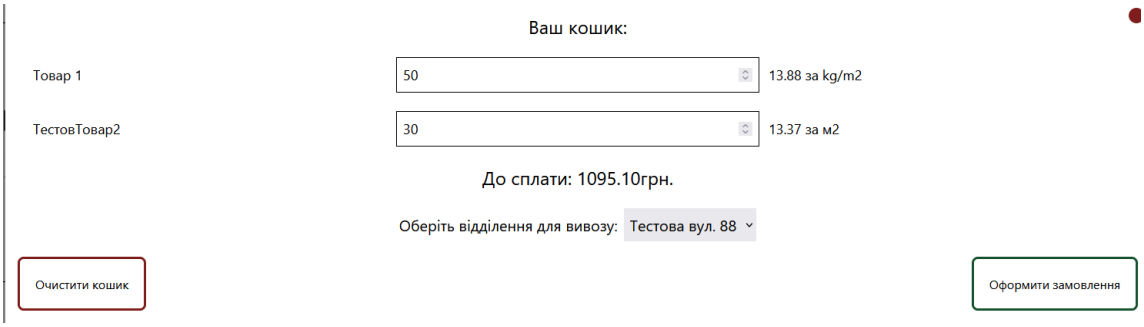


Рисунок 4.45 – Кошик користувача

При зміні кількості товару ціна буде перерахована автоматично.

Натискання на кнопку «Очистити кошик», власне, очистить кошик та перезавантажить зовнішній вигляд кошика (рисунок 4.46).

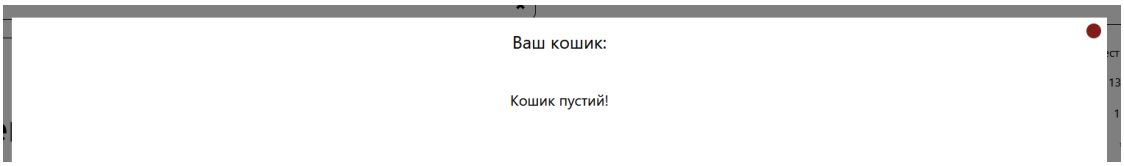


Рисунок 4.47 – Пустий кошик користувача

При натисненні кнопки «Оформити замовлення» кошик так само буде очищений, а користувач отримає повідомлення про успішне створення замовлення (рисунок 4.48).

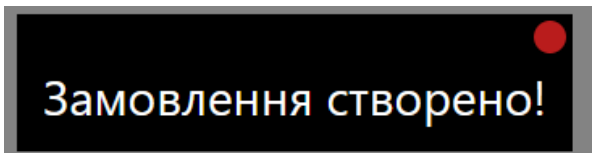


Рисунок 4.48 – Повідомлення про створення нового замовлення

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було створено веб-застосунок на основі тришарової архітектури з чітким розподіленням функціоналу між шарами. Для роботи з базою даних було зроблено акцент на формуванні правильних та безпечних запитів до неї з використанням техніки санітизації, а також за допомогою використання збережених функцій та процедур.

Для серверної частини застосунку було створено універсальні методи формування та перевірки результатів запитів до БД, що дозволяє у майбутньому розширювати або модифікувати функціонал застосунку відповідно потреб. Для сервера також було реалізовано логування запитів користувачів, що дозволяє легше відлагоджувати можливі проблеми та помилки як клієнтської, так і саме серверної частини застосунку.

Клієнтська частина застосунку була реалізована з урахуванням можливості розділення ролей потенційних користувачів: працівник, адміністратор, покупець. Для перших двох доступна адміністративна панель з урахуванням ролі, для останньої ролі (так само як і для незареєстрованого гостя) – веб-магазин.

Клієнтську частину розроблено за допомогою компонентного підходу до веб-розробки, написано відповідні функції для спрощення рутинних процесів, а також створено компоненти для спрощення відображення як адміністративної, так і користувацької інформації.

Застосунок є легко масштабованим, за необхідності розширення його функціоналу реалізовано та створено усі необхідні інструменти для цього як у серверній, так і клієнтській частині. Для того, аби не відходити від основної теми роботи, зміна бази даних є важчою, але процедурний підхід так само може слугувати інструментом масштабування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шлапак А., Яценко О., Іващенко О., Зарицька Н., Осадчук В. Цифрова трансформація міжнародної торгівлі в контексті глобальної конкуренції: технологічні інновації та інвестиційні пріоритети. Фінансово-кредитна діяльність: проблеми теорії та практики. Київ: Фінтехальянс. Том 6(53). 2023. PP. 334-347. https://doi.org/10.55643/fcaptp.6.53.2023.4241

2. Thomas A. Powell. HTML & CSS: The Complete Reference, Fifth Edition. Publisher: McGraw-Hill Companies. New York, 2010. 857 p.

3. Sufyan bin Uzayr. Mastering CSS: A Beginner’s Guide. Publisher: Boca Raton. 2023. 446 p.

4. Sukyong Ryu, Jihyeok Park. JavaScript Language Design and Implementation in Tandem. Communications of the ACM, Volume 67, Issue 5. New York: Association for Computing Machinery, 2024. PP. 86-95.

5. Kartik Bhat. Ultimate Tailwind CSS Handbook: Build sleek and modern websites with immersive UIs using Tailwind CSS. Publisher: Orange Education Pvy Ltd. Delhi, 2023. 293 p.

6. Keshari P., Maurya P., Kumar P., Katiyar A. Web Development Using ReactJS. 2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N). Greater Noida: IEEE, 2023. PP. 1571-1575.

7. Коцюба А.Г., Єрохін А.Л. Дослідження технологій PHP і MySQL як інструментів розробки сучасних web-додатків. Радіоелектроніка та молодь у XXI столітті: матеріали 27-го Міжнародного молодіжного форуму: збірник матеріалів форуму. Т. 6. Конференція "Інформаційні інтелектуальні системи". Ч. 1. Харків: ХНУРЕ, 2023. С. 311-312.

8. Булатецька Л.В., Булатецький В.В. Мова запитів SQL: текст лекцій нормативної навчальної дисципліни “Бази даних та розподілені інформаційно-аналітичні системи”. Луцьк: СНУ ім. Л. Українки, 2018. 92 с.

9. Царенко Д.О., Філімончук Т.В., Волощук О.Б. Вебплатформа для

мережі будівельних магазинів. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: тези доповідей п’ятнадцятої міжнародної науково-технічної конференції. Т.2: секція 2. Баку: ІСУ АР; Харків: НТУ «ХПІ»; Харків: ХНУРЕ; Харків: НАУ «ХАІ»; Жиліна: УмЖ. 2025. с. 37.

10. Larry Rockoff. The Language of SQL. Publisher: Addison-Wesley Professional. Boston, 2021. 272 p.