УДК 004.738.5:339.1

DOI 10.31651/2076-5886-2023-1-69-78

PACS 89.20.Hh

БРАГІНЕЦЬ Ростислав Леонідович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького
e-mail: rostyslav.brahinets@gmail.com

ДІДКОВСЬКИЙ Руслан Михайлович
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

РОЗРОБКА ВЕБ-СЕРВІСУ ДЛЯ ОРГАНІЗАЦІЇ ОНЛАЙН ПОКУПОК ТА ПРОДАЖУ ТОВАРІВ

Розглядається процес розробки веб-сервісу для організації онлайн-покупок та продажу товарів. Проведено аналіз існуючих аналогічних платформ — «Rozetka» та «Prom» — і визначено функціональні вимоги до сервісу, що розробляється. Обґрунтовано вибір стеку технологій: Java та Spring Boot для серверної частини, TypeScript і Angular — для клієнтської, PostgreSQL — як система управління базами даних, Flyway — для міграцій схеми БД, Stripe — для обробки платіжних транзакцій. Описано архітектуру RESTful API, що забезпечує взаємодію між клієнтом і сервером, наведено ключові фрагменти коду серверної та клієнтської частин. Реалізовано чотири режими роботи сервісу: гість, авторизований покупець, адміністратор та супер-адміністратор. Функціонал включає реєстрацію й авторизацію користувачів, перегляд і сортування товарів за категоріями, кошик покупок, оплату банківською карткою з автоматичним генеруванням звіту про покупку, а також адміністративний інтерфейс для управління товарами, категоріями та обліковими записами. Наведено демонстрацію роботи сервісу з відповідними ілюстраціями.

Ключові слова: *онлайн-покупки, веб-сервіс, інтернет-магазин, PostgreSQL, Spring Boot, Angular, TypeScript, REST API, Stripe, електронна комерція.*

Вступ

В сучасному світі люди постійно щось купують. Це може бути щось життєво необхідним, а може бути просто заради розваги. Розвиток сучасних технологій постійно спрощує життя людей. Навіть така річ, як покупка чогось, сьогодні може відбуватися дуже швидко та зручно. Для цього не потрібно виходити з дому. Достатньо лише мати доступ до Інтернету. Онлайн-покупки є дуже популярними в сучасному світі, а ще більшої актуальності, навіть необхідності, вони набули через пандемію COVID-19 та збройний конфлікт в Україні.

Мета статті полягає у тому, щоб показати процес розробки веб-сервісу для організації онлайн покупок та продажу товарів. Для досягнення поставленої мети потрібно виконати наступні завдання:

1) розглянути існуючі сервіси для продажу та покупок товарів;
2) дослідити технології для роботи з базою даних;
3) визначити технологічний стек для серверної частини сервісу;
4) розглянути технології для клієнтської частини сервісу;

5) описати та продемонструвати процес розробки й функціонал веб-сервісу.

Виклад основного матеріалу

1. Використані технології та мови програмування

Розглянемо існуючі сервіси для продажу та покупок у мережі Інтернет. Зокрема візьмемо такі інтернет-магазини, як «Rozetka» та «Prom».

«Rozetka» та «Prom» є двома популярними онлайн-магазинами в Україні, які надають платформу для покупки та продажу товарів через Інтернет. Обидва магазини мають великий вибір товарів і надійну репутацію серед споживачів.

«Rozetka» є одним з найбільших онлайн-магазинів в Україні. Він пропонує широкий асортимент товарів, включаючи електроніку, побутову техніку, одяг, косметику та інше. Клієнти можуть знайти продукти різних брендів і виробників на платформі «Rozetka». Магазин також відомий своєю швидкою доставкою та якісним обслуговуванням клієнтів.

«Prom» є іншим популярним онлайн-магазином в Україні. Ця платформа також пропонує широкий асортимент товарів від різних продавців. «Prom» орієнтований на бізнес-клієнтів та підприємства, а також на приватних споживачів. Він надає можливості для оптової торгівлі та бізнесу, включаючи спеціальні умови для великих замовлень.

Як і багато інших онлайн-магазинів, обидва цих магазини забезпечують зручність покупок через Інтернет, широкий вибір товарів, можливість порівняння цін та відгуків клієнтів, а також швидку та надійну доставку. Крім того, вони мають розвинені системи оплати і забезпечують безпеку та конфіденційність покупців.

Онлайн-магазини, які мають широкий асортимент товарів, такі як «Rozetka» та «Prom», зазвичай мають структуровану інформаційну архітектуру. Вони організовують свої товари в категорії та підкатегорії для полегшення навігації користувачів. На головній сторінці зазвичай розміщено акційні пропозиції, найбільш популярні товари або нові надходження. Крім того, обидва магазини мають систему пошуку, яка дозволяє користувачам швидко знаходити потрібні товари.

«Rozetka» та «Prom» надають широкий спектр функціональних можливостей для зручності покупця. Це можуть бути фільтри для пошуку, порівняння товарів, рейтинги та відгуки клієнтів, які допомагають зробити обґрунтований вибір. Також, можуть бути надані інструменти для створення облікових записів, збереження списків бажань, отримання сповіщень про акції та знижки.

Обидва магазини звичайно мають привабливий та інтуїтивно зрозумілий інтерфейс користувача. Це означає, що вони намагаються забезпечити легку навігацію, зрозумілі піктограми та кнопки, чітке розміщення товарів та інші деталі, щоб забезпечити зручне та ефективне користування сайтом.

Онлайн-магазини використовують різні технології для своєї розробки та функціонування.

Для розробки клієнтської частини сайту (фронтенду) можуть використовуватись такі технології, як HTML, CSS та JavaScript. Фреймворки та бібліотеки, такі як Angular, React або Vue.js, можуть бути використані для побудови більш складних інтерфейсів користувача.

Для обробки бізнес-логіки та управління даними використовуються різні технології. Наприклад, популярні мови програмування, такі як Java, C#, Python або PHP, можуть бути використані для написання серверної частини. Використання фреймворків, таких як Spring (для Java) або Django (для Python), дозволяє прискорити розробку та забезпечити кращу структуру проекту.

Для зберігання та керування даними магазинів можуть використовуватись реляційні бази даних, наприклад, MySQL або PostgreSQL. Також можуть використовуватись нереляційні (NoSQL) бази даних, такі як MongoDB або Redis, для певних аспектів, наприклад, кешування або швидкодії.

Деякі магазини можуть використовувати спеціалізовані CMS для управління контентом на сайті. Популярні CMS, такі як WordPress або Drupal, можуть бути налаштовані для реалізації функціональності магазину.

Для прийому платежів та інтеграції з платіжними системами магазини можуть використовувати API платіжних провайдерів, таких як PayPal, Stripe або LiqPay.

Під час розробки власного веб-сервісу було поставлено наступні завдання:

1. Розробити меню для реєстрації.
2. Розробити меню для авторизації.
3. Розробити головні меню для неавторизованого гостя, авторизованого покупця, адміністратора та супер-адміністратора.
4. Розробити меню з категоріями товарів для всіх користувачів.
5. Розробити меню з товарами для всіх користувачів.
6. Розробити меню профілю та його редагування для авторизованих користувачів.
7. Додати кошик для авторизованого покупця.
8. Додати оплату товарів для авторизованого покупця із генеруванням звіту про покупку.
9. Розробити меню для додавання та видалення унікальних номерів адміністратора.
10. Розробити меню для перегляду та видалення зареєстрованих користувачів.
11. Розробити меню для додавання, редагування та видалення категорій товарів.
12. Розробити меню для додавання, редагування та видалення товарів.

Для розробки власного сервісу було використано Java та Spring для серверної частини, TypeScript та Angular для розробки клієнтської частини і базу даних PostgreSQL. Для обробки платежів обрано Stripe.

Stripe – це платіжна платформа, яка надає розробникам та бізнесам інструменти для прийому онлайн-платежів. Вона дозволяє підприємствам приймати платежі через Інтернет і забезпечує безпечну та зручну інтеграцію платіжних функцій у веб-сайти та додатки.

Stripe дозволяє приймати різні види платежів, включаючи кредитні та дебетові картки (Visa, Mastercard, American Express), цифрові гаманці (Apple Pay, Google Pay), банківські перекази та інші способи платежів.

Розробники можуть використовувати Stripe для інтеграції платіжних функцій безпосередньо у свої веб-сайти або мобільні додатки. Stripe надає зручне API та набір готових бібліотек для різних мов програмування, що спрощує процес інтеграції.

TypeScript є мовою програмування, розробленою компанією Microsoft в 2012 році [11]. Вона позиціонується як розширення мови JavaScript і призначена для розробки веб-застосунків.

Gradle є системою автоматичного збирання, яка поєднує принципи Apache Ant та Apache Maven. Вона використовує предметно-орієнтовану мову (DSL) на основі мови Groovy для налаштування та опису залежностей проекту. Головна мета Gradle – забезпечити зручне підключення необхідних бібліотек та залежностей.

У випадку з серверною стороною веб-сервісу, Gradle використовується для підключення необхідних бібліотек. Код, який визначає залежності та налаштування проекту, міститься у файлі build.gradle.

Для запуску серверної сторони веб-сервісу використовується фреймворк Spring Boot. У класі ApplicationStarter, в методі main(), ініціалізується серверна частина, а саме

локальний сервер Tomcat, який дозволяє працювати з веб-сервісом без підключення до Інтернету.

REST (Representational State Transfer) – це підхід до архітектури мережевих протоколів, який надає доступ до інформаційних ресурсів. Він був описаний та популяризований Роєм Філдінгом у 2000 році, одним з творців протоколу HTTP. REST базується на принципах функціонування Всесвітньої павутини та можливостях протоколу HTTP. Філдінг розробив REST одночасно з HTTP 1.1, ґрунтуючись на попередньому протоколі HTTP 1.0.

Комунікація між клієнтом та сервером у RESTful API є безстановною, що означає, що кожен запит клієнта до сервера містить усю необхідну інформацію для обробки цього запиту. Сервер не зберігає жодної інформації про стан клієнта між запитами.

2. Реалізація сервісу

Для створення сервісу необхідно створити сервер, який буде працювати з нашою базою даних. Додавати до неї інформацію, видаляти та редагувати її, а також просто переглядати. І потрібно створити інтерфейс клієнтської сторони. Він має відображати інформацію про товари, давати можливість взаємодіяти з цими товарами та мати всі необхідні функції.

Серверна частина побудована за трирівневою REST-архітектурою: контролери обробляють HTTP-запити, сервіси реалізують бізнес-логіку, репозиторії відповідають за доступ до бази даних PostgreSQL. Нижче наведено ключові фрагменти реалізації.

Розглянемо кілька частин всього функціоналу сервісу. Зокрема для серверної сторони існує метод update(), у класі ProductRepository, який оновлює дані певного продукту. Він приймає ідентифікаційний номер товару, який потрібно оновити, і об’єкт з оновленими даними. Далі за допомогою Spring Data JDBC дані про товар оновлюються в базі даних. Для цього використовується SQL-скрипт. Код цього методу наведено нижче.

```
@Override
public void update(long id, Product product) {
    jdbcTemplate.update(
        "UPDATE product "
            + "SET name=:name, describe=:describe, price=:price, "
            + "barcode=:barcode, in_stock=:in_stock, image=:image "
            + "WHERE id=:id",
        Map.ofEntries(
            Map.entry("name", product.getName()),
            Map.entry("describe", product.getDescribe()),
            Map.entry("price", product.getPrice()),
            Map.entry("barcode", product.getBarcode()),
            Map.entry("in_stock", product.isInStock()),
            Map.entry("image", product.getImage()),
            Map.entry("id", id)
        )
    );
}
```

Метод saveProduct(), у класі ProductController, додає новий товар до бази даних. Він приймає об’єкт нового товару. В методі перевіряється чи отриманий об’єкт містить зображення товару. Якщо його немає, то встановлюється стандартне зображення. І вже потім новий товар додається до бази даних.

Для категорій існує клас-валідатор CategoryValidator. Він має кілька методів validate() для перевірки категорії. Перший приймає в якості параметрів

ідентифікаційний номер категорії та список всіх категорій, які існують. Потім створюється порожній список для ідентифікаційний номерів. Які за допомогою циклу отримуються від кожної категорії і записуються у цей список. Далі перевіряється чи місить цей список ідентифікаційний номер, отриманий методом у параметрах. Якщо він існує, то нічого не відбувається. Але якщо такого номера немає, то виникає помилка.

Інший метод validate() приймає назву категорії та список всіх існуючих категорій. Метод перевіряє чи назва не порожня. І викидає помилку, якщо це не так. Далі перевіряється чи існує вже така назва категорії, так само, як і з ідентифікаційним номером. Якщо її не існує, то виникає помилка.

Кожен користувач має якусь роль: клієнт або адміністратор. Є сервіс RoleService в якому є метод findByName(), який повертає об'єкт ролі за назвою. Він приймає назву ролі, які треба знайти і звертається до репозиторія RoleRepository, щоб отримати роль. Код класу наведено нижче.

```
package com.shop.role;
import org.springframework.stereotype.Service;
import java.util.Optional;
@Service
public class RoleService {
    private final RoleRepository roleRepository;
    public RoleService(RoleRepository roleRepository) {
        this.roleRepository = roleRepository;
    }
    public Role findByName(String name) {
        Optional<Role> role = roleRepository.findByName(name);
        return role.orElseGet(Role::new);
    }
}
```

Розглянемо тепер функціонал клієнтської частини. Для сторінки авторизації є розмітка на HTML. Вона формує на веб-сторінці потрібні елементи. А за допомогою стилів CSS відображає їх у красивому для користувача вигляді. На сторінці авторизації є назва сторінки, форма з полями для введення електронної адреси або номера телефону і пароля. Також є кнопки для підтвердження авторизації та переходу на сторінку реєстрації.

На всіх веб-сторінках є кнопки. Для них існує компонент «shop-button». Його можна помістити на сторінку за допомогою тега в HTML <shop-button>. Цей тег заміняється розміткою, яка наведена нижче.

```
<button
  class="btn"
  type="button"
  [ngStyle]="{'background-color': color}"
  (click)="onClick()">
  {{text}}
</button>
```

Для такої заміни існує клас «ButtonComponent». Він вказує, яку розмітку та стилі використовувати для компонента «shop-button». Компонент має кілька властивостей, які потрібно задавати під час його використання. Потрібно вказувати текст, який має відображатися на кнопці і колір самої кнопки. Також кнопка має обробник подій, який

спрацьовує під час натискання на неї. А що саме має зробити кнопка потрібно теж вказати під час використання компонента. Код цього класу наведено нижче.

```
import {Component, EventEmitter, Input, OnInit, Output} from '@angular/core';
@Component({
  selector: 'shop-button',
  templateUrl: './button.component.html',
  styleUrls: ['./button.component.css']
})
export class ButtonComponent implements OnInit {
  @Input() text: string;
  @Input() color: string;
  @Output() btnClick;
  constructor() {
    this.btnClick = new EventEmitter();
  }
  ngOnInit(): void {
  }
  onClick(): void {
    this.btnClick.emit();
  }
}
```

Для того щоб звертатися до сервера та працювати з користувачами існує сервіс UserService. Він використовує HttpClient щоб посилали запити на сервер. Цей сервіс дозволяє отримувати всіх користувачів, певного користувача по його ідентифікаційному номеру, оновлювати дані користувача та видаляти його.

3. Використання сервісу

Ознайомимося з розробленим сервісом та розглянемо його можливості. Сервіс має чотири режими роботи: гість, користувач, адміністратор та супер-адміністратор. Для кожного відображаються різні головні сторінки. На них відображаються випадкові товари та меню, відповідне для кожного режиму (рис. 1).

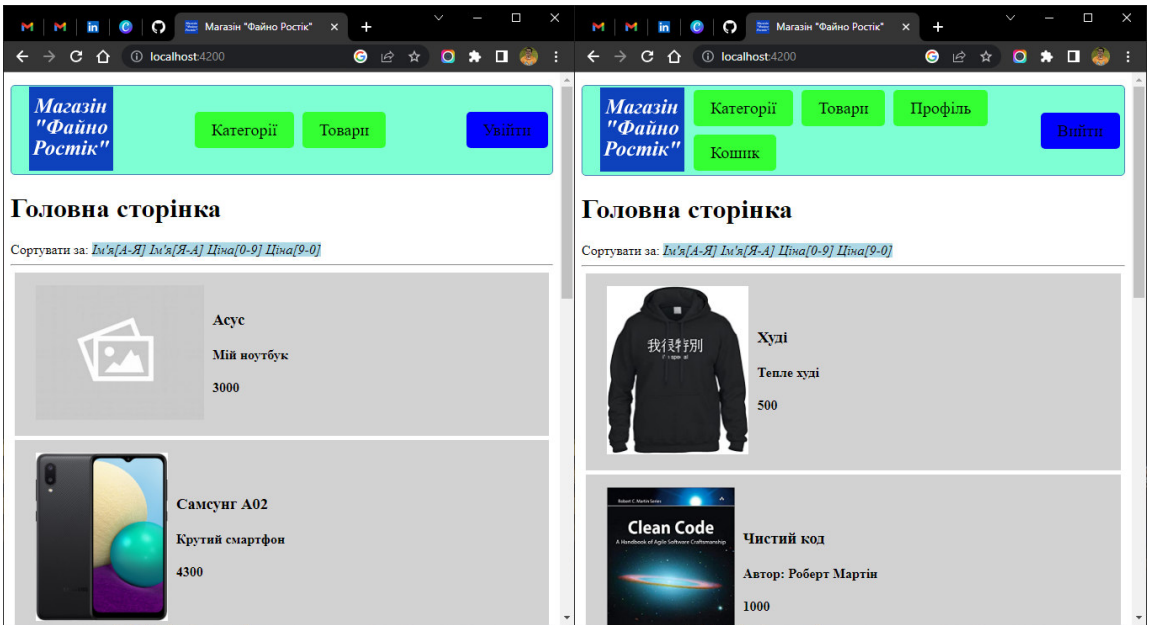

Рис. 1. Головні сторінки для гостя та авторизованого покупця

На веб-сторінці авторизації є поля для введення електронної адреси або номера телефону та пароля і кнопка підтвердження (рис. 2). На веб-сторінці реєстрації є поля для введення прізвища та ім’я, електронної адреси, номера телефону, пароля та підтвердження пароля, номеру адміністратора, кнопка підтвердження і кнопка переходу на веб-сторінку авторизації.

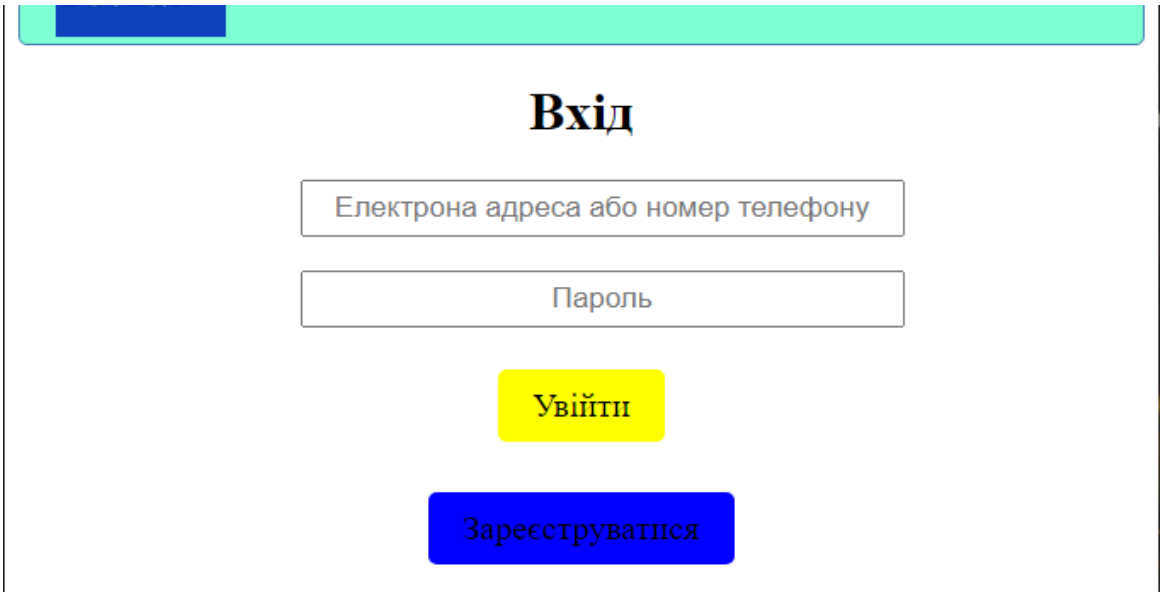

Рис. 2. Сторінка авторизації

На веб-сторінці «Товари» є кнопки переходу на інші веб-сторінки та кілька випадкових товарів. Також товари можна сортувати за назвою та ціною у порядку зростання або спадання (рис. 3). Якщо користувач не авторизований, то на кожній сторінці є кнопка входу. А якщо авторизований, то кнопка виходу. Товари можуть бути присутні та відсутні на складі. Про це вказується на веб-сторінці з товаром. Якщо товар відсутній на складі, то купити його неможливо. Кожен товар належить до якоїсь категорії. На веб-сторінці «Категорії» є кнопки переходу на інші веб-сторінки та список усіх категорій товарів.

Рис. 3. Сторінка «Товари»

Адміністратор може додавати нові товари та категорії. Також адміністратор може видаляти існуючі товари та категорії. Ще адміністратор може редагувати існуючі товари та категорії. Додати товар можна із зображенням або без. Якщо його немає, то встановлюється стандартне зображення автоматично.

На веб-сторінці «Профіль» є посилання для переходу на інші веб-сторінки та інформація про користувача, яку було введено під час реєстрації. Користувач може змінювати інформацію про себе.

Кожен покупець має кошик, куди додаються товари, які він хоче купити (рис. 4). Після додавання товарів до кошика їх можна купити. Після натискання на кнопку «Оплатити» з'являється форма для введення реквізитів банківської карти й кнопка «Оплатити». Після оплати пропонується завантажити звіт про покупку (рис. 5).

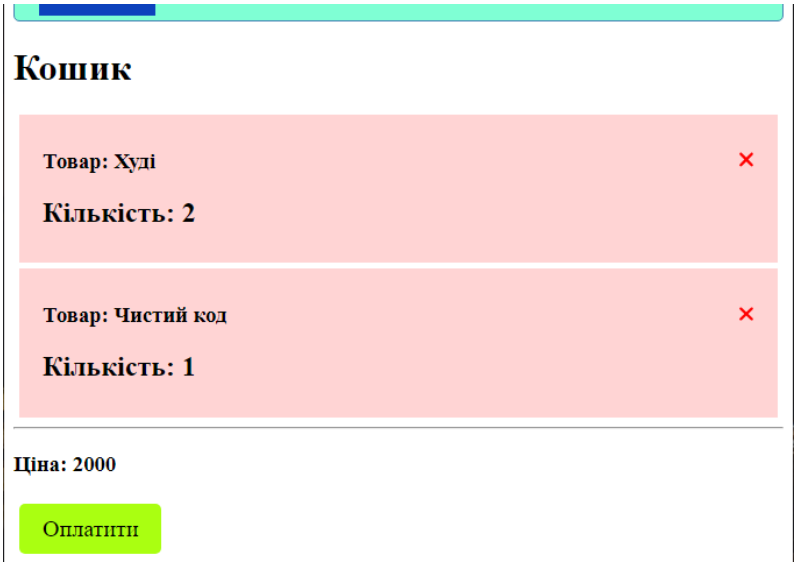

Рис. 4. Сторінка кошика з товарами

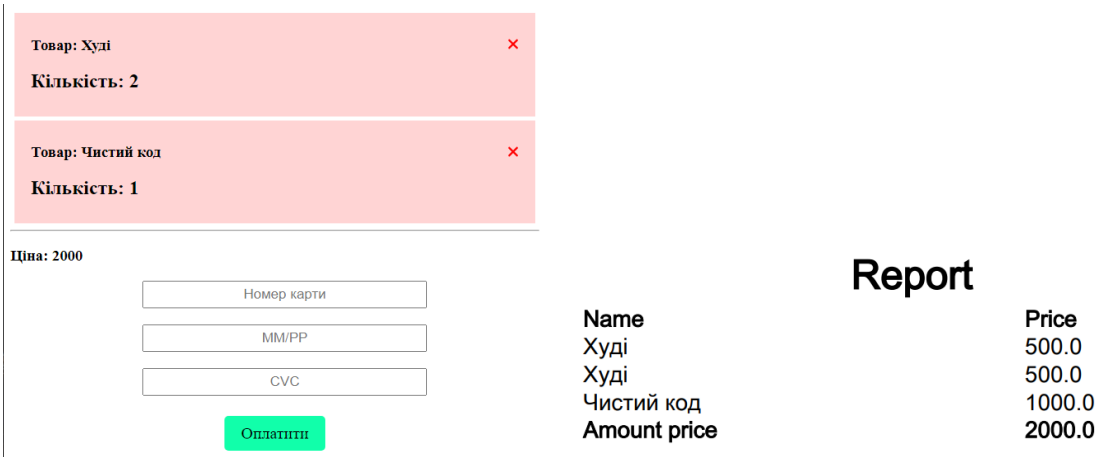

Рис. 5. Форма оплати та згенерований звіт про покупку

Супер-адміністратор може переглядати всіх зареєстрованих користувачів. На веб-сторінці конкретного користувача є кнопка переходу на інші веб-сторінки та інформація про користувача, яку було введено під час реєстрації. А також відображається роль даного користувача. Він може бути адміністратором або клієнтом. Також супер-адміністратор має «Адмін-панель». На ній є кнопки переходу на веб-сторінки для видалення користувачів і додавання та видалення унікального номера адміністратора.

Весь код можна знайти за посиланнями на репозиторії на GitHub: https://github.com/rbrahinets/shop-backend/ та https://github.com/rbrahinets/shop-frontend/.

Висновки.

У статті розглянуто процес проектування та реалізації веб-сервісу для організації онлайн-покупок і продажу товарів. Проведено порівняльний аналіз існуючих аналогічних платформ – «Rozetka» та «Prom» – і сформовано функціональні вимоги до розробленого сервісу. Обґрунтовано вибір технологічного стеку: Java і Spring Boot для серверної частини, TypeScript та Angular – для клієнтської, PostgreSQL – як система управління базами даних, Flyway – для керування міграціями схеми, Stripe – для обробки платіжних транзакцій. Реалізовано RESTful API та чотири рольові режими роботи: гість, авторизований покупець, адміністратор і супер-адміністратор. Наведено ключові фрагменти коду серверної та клієнтської частин і продемонстровано повний функціонал сервісу. Результати підтверджують, що розроблений сервіс є повнофункціональним аналогом сучасних торговельних платформ і може слугувати основою для комерційних рішень у сфері електронної торгівлі.

Список використаної літератури:

1. Schildt H. Java: The Complete Reference. 9th ed. – New York : McGraw-Hill Education, 2014. – 1312 p.
2. Spring Framework. Core Technologies. Official Reference Documentation. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html (дата звернення: 01.06.2023).
3. Spring Security. Official Reference Documentation. URL: https://docs.spring.io/spring-security/reference/ (дата звернення: 01.06.2023).
4. Spring Data. Official Reference Documentation. URL: https://docs.spring.io/spring-data/commons/docs/current/reference/html/ (дата звернення: 01.06.2023).
5. Spring Data JDBC. Official Reference Documentation. URL: https://docs.spring.io/spring-data/jdbc/docs/current/reference/html/ (дата звернення: 01.06.2023).
6. Obe R., Hsu L. PostgreSQL: Up and Running. 3rd ed. – Sebastopol : O'Reilly Media, 2017. – 336 p.
7. Flyway. Official Documentation. URL: https://flywaydb.org/documentation/ (дата звернення: 01.06.2023).
8. Stripe. Developer Documentation. URL: https://stripe.com/docs (дата звернення: 01.06.2023).
9. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 01.06.2023).
10. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 01.06.2023).
11. TypeScript. Official Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 01.06.2023).
12. Angular. Official Documentation. URL: https://angular.io/docs (дата звернення: 01.06.2023).
13. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation. University of California, Irvine, 2000. URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата звернення: 01.06.2023).

References:

1. Schildt, H. (2014). Java: The Complete Reference (9th ed.). New York: McGraw-Hill Education. 1312 p.
2. Spring Framework Core Technologies. Official Reference Documentation. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html
3. Spring Security Official Reference Documentation. URL: https://docs.spring.io/spring-security/reference/
4. Spring Data Official Reference Documentation. URL: https://docs.spring.io/spring-data/commons/docs/current/reference/html/
5. Spring Data JDBC Official Reference Documentation. URL: https://docs.spring.io/spring-data/jdbc/docs/current/reference/html/
6. Obe, R., Hsu, L. (2017). PostgreSQL: Up and Running (3rd ed.). Sebastopol: O'Reilly Media. 336 p.
7. Flyway Official Documentation. URL: https://flywaydb.org/documentation/
8. Stripe Developer Documentation. URL: https://stripe.com/docs
9. MDN Web Docs. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML
10. MDN Web Docs. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS

11. TypeScript Official Documentation. URL: https://www.typescriptlang.org/docs/
12. Angular Official Documentation. URL: https://angular.io/docs
13. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Ph.D. dissertation, University of California, Irvine. URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm

BRAHINETS Rostyslav,
Student, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DEVELOPMENT OF A WEB SERVICE FOR ONLINE SHOPPING AND PRODUCT SALES

Summary. Introduction. *In today's digital world, online commerce has become an essential component of everyday life, further accelerated by the COVID-19 pandemic and ongoing armed conflict in Ukraine. The growing demand for convenient, secure, and scalable online trading platforms motivates the development of purpose-built web services. The primary goal of this work is to design and implement a full-stack web service supporting both the purchase and sale of goods through the Internet.*

Purpose. *The purpose of this article is to present the process of developing a web service for online shopping and product sales: from analysis of existing analogues and technology selection to implementation and functional demonstration of the finished service.*

Results. *The article examines existing Ukrainian e-commerce platforms – "Rozetka" and "Prom" – and identifies functional requirements for the service under development. The technology stack is justified: Java and Spring Boot for the server side, TypeScript and Angular for the client side, PostgreSQL as the relational database, Flyway for database migration management, and Stripe for secure online payment processing. The RESTful API architecture ensuring stateless client-server communication is described, and key code fragments from both server-side and client-side components are provided. The implemented web service supports four operation modes – guest, authorized buyer, administrator, and super-administrator – each with a dedicated interface. Core functionality includes user registration and authentication, product browsing with category-based navigation and sorting, a shopping cart, bank card payment with automatic purchase report generation, and a full administrative interface for managing products, categories, and user accounts. The payment module was tested using Stripe's sandbox environment, confirming correct transaction processing.*

Conclusion. *The article presents the design and implementation of a web service for online shopping and product sales. The results of the study can be applied in the development of commercial trading platforms and the improvement of existing e-commerce solutions. The main outcomes include a comparative analysis of analogous services, justification of the selected technology stack, full implementation of the service using Java/Spring Boot and TypeScript/Angular, integration of the Stripe payment platform, and demonstration of all major functional modules.*

Keywords: online shopping, web service, internet store, PostgreSQL, Spring Boot, Angular, TypeScript, REST API, Stripe, e-commerce.

Одержано редакцією 20.10.2023 р.
Прийнято до публікації 06.12.2023 р.