

# Sprawozdanie z projektu z przedmiotu:

## Programowanie Defensywne

### Temat projektu:

System zarządzania sklepem z elektroniką wraz z obsługą magazynu

---

Uczelnia:	Politechnika Świętokrzyska
-----------	----------------------------

---

Wydział:	Elektrotechniki, Automatyki i Informatyki
Rok akademicki:	2024/2025
Kierunek studiów:	Informatyka
Specjalizacja:	Cyberbezpieczeństwo
Semestr:	I letni
Grupa:	1ID25A
Student	Mateusz Szczerek
Student	Wiktor Chabera
Student	Mateusz Sidło

---

## Cel i zakres zadania

Celem projektu jest zaprojektowanie i implementacja systemu zarządzania sklepem z elektroniką wraz z obsługą magazynu. Do budowy strony serwera wykorzystano Spring oraz bazę danych PostgreSQL, natomiast interfejs użytkownika powstał przy pomocy React oraz Bootstrap.

System powinien obsługiwać między innymi:

- tworzenie nowych użytkowników
  - obsługę spisu posiadanych produktów
  - obsługę spisu zamówień
  - bezpieczny sposób logowania do systemu
- 

## Implementacja

### Model Użytkownika – Katalog `auth`

Katalog `auth` reprezentuje pracownika sklepu będącego użytkownikiem systemu oraz bezpieczny sposób na przechowywanie haseł dostępu.

### Model Produktu – Plik `Product.java`

Klasa `Product` reprezentuje produkt posiadany przez sklep

**Atrybuty:**

- `productName` – nazwa produktu,
- `price` – cena,
- `description` – opis produktu,
- `producer` – nazwa producenta.

## Model Zamówienia - Pliki `Purchase.java` oraz `PurchaseDetails.java`

Klasy `Purchase` oraz `PurchaseDetails` reprezentują dokonane przez użytkownika zamówienie.

### Atrybuty:

- `clientId` - identyfikator nabywcy
- `date` - data złożenia zamówienia
- `price` - kwota
- `status` - status zamówienia
- `orderId` - numer zamówienia
- `productId` - numer produktu
- `quantity` - ilość produktu w zamówieniu

## Model Magazynu - Plik `Warehouse.java`

Klasa `Warehouse` reprezentuje magazyn z towarami.

### Atrybuty:

- `name` - nazwa magazynu
- `productsList` - lista towarów znajdujących się w magazynie

## Model Klienta - Plik `Customer.java`

Klasa `Customer` reprezentuje klienta sklepu.

### Atrybuty:

- `customerName` - nazwisko klienta
- `customerAddress` - adres domowy klienta
- `customerEmail` - email klienta
- `customerPhone` - numer telefonu klienta

## Repozytoria - Katalog repository

Katalog `repository` zawiera interfejsy potrzebne do przechowywania danych w aplikacji na których opierają się kontrolery.

---

## Bezpieczeństwo - Katalog security

Katalog `security` zawiera klasy odpowiedzialne za bezpieczne działanie aplikacji, takie jak weryfikacja dwuetapowa wykorzystana podczas logowania czy koder i dekodek Base32 zgodny ze standardem RFC 4648.

---

## Logika Biznesowa - Katalog services

Katalog `services` zawiera interfejsy oraz klasy odpowiedzialne za logikę działania poszczególnych funkcjonalności do których odwołują się kontrolery.

---

## Kontroler REST dla zamówień – `OrderController.java`

Kontroler `OrderController` zapewnia REST API do zarządzania zamówieniami.

### Endpointy (`/api/orders`):

- GET / – pobranie wszystkich zamówień,
- GET /{id} – pobranie danych konkretnego zamówienia,
- POST / - dodanie nowego zamówienia,
- PUT /{id} – aktualizacja danych,
- DELETE /{id} – usunięcie zamówienia.

## Kontroler REST dla produktów – `ProductController.java`

Kontroler `ProductController` zapewnia REST API do zarządzania listą produktów.

### Endpointy (`/api/products`):

- GET / – pobranie wszystkich produktów,
- GET /{id} – pobranie danych konkretnego produktu,
- POST / - dodanie nowego produktu,
- PUT /{id} – aktualizacja danych,
- DELETE /{id} – usunięcie produktu.

## Kontroler REST dla użytkowników – `UserController.java`

Kontroler `UserController` zapewnia REST API do zarządzania kontami użytkowników.

### Endpointy (`/api/users`):

- GET / – pobranie wszystkich użytkowników,
- GET /{id} – pobranie danych konkretnego użytkownika,
- POST / - dodanie nowego użytkownika,
- PUT /{id} – aktualizacja danych,
- DELETE /{id} – usunięcie użytkownika,
- POST /login - zalogowanie użytkownika,
- POST /register - rejestracja nowego użytkownika.

## Kontroler REST dla magazynów – `WarehouseController.java`

Kontroler `WarehouseController` zapewnia REST API do zarządzania zasobami magazynów.

### Endpointy (`/api/warehouses`):

- GET / – pobranie zawartości wszystkich magazynów,
- GET /{id} – pobranie danych konkretnego magazynu,
- POST / - dodanie nowego magazynu,

- PUT `/id` – aktualizacja danych,
- DELETE `/id` – usunięcie magazynu.

## Układ strony - `layout.tsx`

Plik zawierający układ stron.

## Panel Logowania - `authentication/login`

Strona zawierająca panel logowania użytkownika.

## Panel Rejestracji - `authentication/register`

Strona zawierająca panel rejestracji nowego użytkownika.

## Spis produktów - `products`

Strona wyświetlająca spis posiadanych produktów.

---

## Podsumowanie

Na obecnym etapie realizacji projektu, zgodnie z harmonogramem obejmującym trzeci kamień milowy, udało się rozbudować kluczowe podstawowe funkcjonalności systemu zarządzania sklepem z elektroniką oraz obsługi magazynu. Zrealizowano modele danych odpowiadające strukturze bazy danych, przygotowano repozytoria umożliwiające komunikację z bazą, a także zaimplementowano kontrolery REST API dla wszystkich głównych encji: produktów, zamówień, użytkowników oraz magazynów. Po stronie klienta stworzono podstawowy interfejs użytkownika obejmujący panele logowania, rejestracji oraz prezentacji spisu produktów.

System umożliwia obecnie:

- rejestrację i logowanie użytkowników,

- komunikację z bazą danych PostgreSQL poprzez odpowiednio przygotowane kontrolery REST i repozytoria.

W ostatnim etapie planowana jest dalsza rozbudowa funkcjonalności, poprawa ergonomii interfejsu użytkownika oraz dokładne testowanie aplikacji w celu zapewnienia jej wysokiej jakości i stabilności działania.