



POLITECHNIKA RZESZOWSKA  
im. Ignacego Łukasiewicza  
WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

Michał Koryl, Filip Kosiorowski, Katarzyna Kośniowska  
173160, 173161, 173162

Aplikacje internetowe  
Projekt zaliczeniowy

## Spis treści

<b>1. WSTĘP .....</b>	<b>3</b>
<b>2. UŻYTE STRUKTURY .....</b>	<b>4</b>
MAVEN .....	4
SPRING .....	6
VAADIN .....	8
BAZA DANYCH H2.....	11
<b>3. KOMPONENTY .....</b>	<b>12</b>
WIDOKI VAADIN.....	12
INTERFEJS STRONY.....	14
INTERFEJS UŻYTKOWNIKA (UI) .....	18
<b>4. BAZA DANYCH .....</b>	<b>22</b>
STRUKTURA BAZY DANYCH .....	22
KORZYSTANIE ZE SPRING DATA JPA .....	26
<b>5. ORGANIZACJA KODU APLIKACJI.....</b>	<b>28</b>
<b>6. PODSUMOWANIE.....</b>	<b>30</b>

# 1. WSTĘP

Celem projektu było stworzenie aplikacji sklepu internetowego, która spełnia współczesne wymagania użytkowników oraz zapewnia łatwość obsługi zarówno dla klientów, jak i administratorów systemu. W trakcie realizacji projektu skupiliśmy się na stworzeniu intuicyjnego interfejsu, płynnego działania oraz możliwości rozszerzania funkcjonalności w przyszłości.

Jako kluczowe cechy serwisu ustaliliśmy:

- 1) **Przejrzystość strony:** Aplikacja została zaprojektowana w taki sposób, aby użytkownicy mogli w łatwy sposób znaleźć interesujące ich produkty oraz poruszać się po serwisie.
- 2) **Przeglądanie przedmiotów i kategorii, wyszukiwarka:** Użytkownik ma możliwość przeglądania dostępnych produktów oraz korzystania z wyszukiwarki, która pozwala szybko odnaleźć interesujące przedmioty, uwzględniając ich kategorie, nazwy lub inne szczegóły.
- 3) **System dodawania, usuwania i edytowania koszyka:** Użytkownicy mogą zarządzać swoimi zakupami za pomocą intuicyjnego systemu koszyka, który umożliwia dodawanie, usuwanie i edytowanie produktów.
- 4) **Możliwość złożenia zamówienia:** Proces składania zamówienia został zaprojektowany w sposób prosty i intuicyjny, aby użytkownik mógł sfinalizować swoje zakupy szybko i bez zbędnych komplikacji.

Dodatkowo, w ramach projektu wdrożyliśmy:

- 1) **Responsywny design:** Strona jest w pełni dostosowana do różnych rozdzielczości ekranów, dzięki czemu użytkownicy mogą korzystać z aplikacji zarówno na komputerach, jak i na urządzeniach mobilnych.
- 2) **Podstawowy system zarządzania użytkownikami:** Aplikacja umożliwia rejestrację i logowanie użytkowników, co pozwala na personalizację doświadczeń oraz umożliwia dostęp do historii zamówień.
- 3) **Modularność i możliwość rozbudowy:** Aplikacja została zaprojektowana w sposób umożliwiający łatwe dodawanie nowych funkcji, takich jak integracja z systemami płatności, zaawansowany system rekomendacji czy dynamiczne promocje.

## 2. UŻYTE STRUKTURY

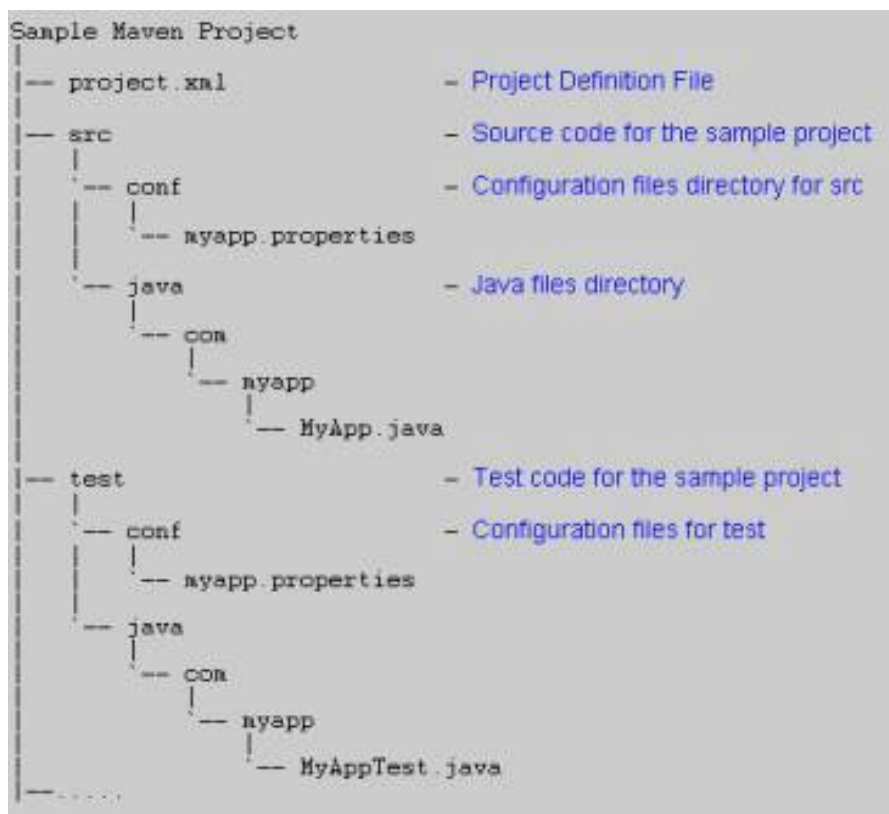
### MAVEN

Maven to narzędzie do zarządzania budowaniem, zależnościami i cyklem życia projektu w Javie. Używając go w swoim projekcie, zyskaliśmy korzyści:

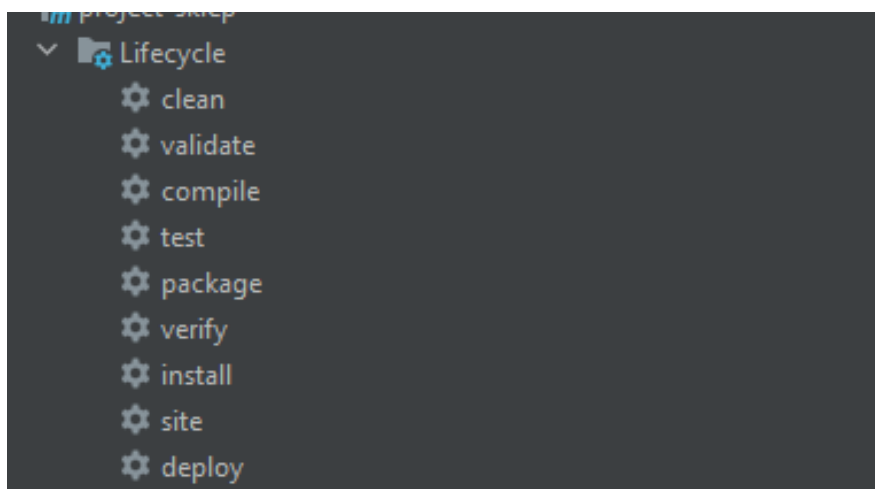
- 1) Automatyczne pobieranie bibliotek i innych zależności z repozytoriów centralnych, takich jak Maven Central. Dzięki temu nie trzeba ręcznie dodawać plików .jar do projektu. Zależności są definiowane w pliku pom.xml i mogą być łatwo aktualizowane.

```
<dependency>
  <groupId>com.vaadin</groupId>
  <artifactId>vaadin-spring-boot-starter</artifactId>
</dependency>
```

- 2) wymuszenie standardowej struktury projektu, co ułatwia pracę w zespole i utrzymanie projektu. Struktura ta wygląda tak:



- 3) automatyzacja cyklu życia projektu - Maven definiuje tzw. lifecycle, który pozwala na automatyczne wykonywanie zadań, takich jak: kompilacja, testowanie, budowanie aplikacji czy wdrożenie na produkcję



- 4) repozytoria lokalne oraz zdalne
- 5) Narzędzia takie jak IntelliJ posiadają pełne wsparcie dla Mavena. Plik pom.xml jest automatycznie analizowany, a zależności i wtyczki są konfigurowane bez konieczności ręcznego ustawiania.

# SPRING

Znacząco ułatwia tworzenie nowoczesnych, skalowalnych i elastycznych aplikacji. W swoim projekcie wykorzystaliśmy go, aby zbudować aplikację szybciej i bardziej efektywnie. Wiele ustawień jest prekonfigurowanych, co pozwala uniknąć rozbudowanych plików XML.

Wszystkie ustawienia, tłumaczenia zawierają się w pliku application.properties:

```
application.properties
1  server.port=${PORT:8080}
2  logging.level.org.atmosphere = warn
3  spring.mustache.check-template-location = false
4
5  # Launch the default browser when starting the application in development mode
6  vaadin.launch-browser=false
7  # To improve the performance during development.
8  # For more information https://vaadin.com/docs/latest/integrations/spring/configuration#sp
9  vaadin.allowed-packages = com.vaadin,org.vaadin,com.example.application,com.floodingcode
10 spring.jpa.defer-datasource-initialization = true
11 spring.datasource.url=jdbc:h2:file:./data/sklepdb;AUTO_SERVER=TRUE
12 spring.datasource.driverClassName=org.h2.Driver
13 spring.datasource.username=admin
14 spring.datasource.password=admin
15 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
16 spring.h2.console.enabled=true
17 spring.jpa.hibernate.ddl-auto=update
18 spring.jpa.show-sql=true
19 spring.main.allow-circular-references=true
20 spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
```

Oraz pliku startowym Application.java

```
@SpringBootApplication(scanBasePackages = "com.example.application")
@Theme(value = "project-sklep")
public class Application implements AppShellConfigurator {

    //private static final Logger log = LoggerFactory.getLogger(Applica

    public static void main(String[] args) {
        System.setProperty("spring.devtools.restart.enabled", "true");
        SpringApplication.run(Application.class, args);
    }
}
```

Użyte przez nas moduły Spring:

- 1) **Spring Core:** Podstawa frameworka, oferująca IoC i DI (Dependency Injection).
- 2) **Spring MVC:** Umożliwia tworzenie aplikacji webowych w architekturze Model-View-Controller.
- 3) **Spring Data:** Upraszcza interakcje z bazami danych, nasza aplikacja korzysta z JPA

```
public interface UserRepository extends JpaRepository<User, Integer> {  
    public User findByUsername(String username); 2 usages mfrontiera  
    boolean existsByUsername(String username); 1 usage mfrontiera  
    User findById(Long Id); 2 usages mfrontiera  
}
```

- 4) **Spring Security:** Dodaje mechanizmy uwierzytelniania i autoryzacji do aplikacji.
- 5) **Spring Boot:** Ułatwia konfigurację i uruchamianie aplikacji dzięki gotowym ustawieniom.

Posiada on również wbudowany serwer Tomcat, który pozwala uruchomić aplikację bez zewnętrznego serwera.

# VAADIN

To framework do budowy nowoczesnych, responsywnych aplikacji webowych w Javie, który wyróżnia się możliwością tworzenia pełnego interfejsu użytkownika (UI) bez konieczności pisania kodu w JavaScript, HTML czy CSS co upraszcza proces tworzenia i utrzymania aplikacji.

Vaadin składa się z kilku komponentów, które pomagają w tworzeniu aplikacji:

- 1) **Vaadin Flow:** Narzędzie do budowy aplikacji webowych opartych na architekturze serwerowej, wszystko działa po stronie serwera, a UI jest automatycznie renderowane w przeglądarce użytkownika.
- 2) **Vaadin Components:** Kolekcja gotowych komponentów, takich jak formularze, przyciski, siatki danych, a także bardziej zaawansowane elementy, jak mapy czy wykresy.
- 3) **Vaadin PWA:** Umożliwia tworzenie aplikacji typu Progressive Web Apps.

Vaadin to idealne rozwiązanie dla programistów Javy, którzy chcą tworzyć nowoczesne interfejsy użytkownika w jednym środowisku programistycznym, bez potrzeby nauki technologii frontendowych. Dzięki gotowym komponentom i wsparciu dla Spring, Vaadin ułatwia tworzenie aplikacji biznesowych o wysokiej jakości.



## Przykład widoku aplikacji (strona główna):

```
@Route(value = "", layout = StoreLayout.class) @mfrontiera *
@PageTitle("STRONA GŁÓWNA")
@AnonymousAllowed
@ParentLayout(MainLayout.class)

public class MainViewView extends VerticalLayout implements RouterLayout {

    private OrderedList itemsContainer; 5 usages
    private List<Item> items; 2 usages
    private final ItemRepository itemRepository; 2 usages

    public MainViewView(ItemRepository itemRepository) { @mfrontiera *
        this.itemRepository = itemRepository;

        itemsContainer = new OrderedList();
        itemsContainer.addClassNames(LumoUtility.Gap.MEDIUM, LumoUtility.Display.FLEX, LumoUtility.FlexDirection.ROW, LumoUtility.FlexWrap.WRAP);

        setJustifyContentMode(JustifyContentMode.START);
        setAlignItems(Alignment.CENTER);
        Slide s1 = new Slide(createSlideContent( url: "arduino-learning-kit", image: "https://u.cubeupload.com/korylek/arduinokit.jpg"));
        Slide s2 = new Slide(createSlideContent( url: "ender3-v3", image: "https://u.cubeupload.com/korylek/ender3v3.jpg"));
        Slide s3 = new Slide(createSlideContent( url: "flipper-zero", image: "https://u.cubeupload.com/korylek/flipperzero30.jpg"));

        Carousel c = new Carousel(s1,s2,s3).withAutoProgress()
            .withSlideDuration(4)
            .withStartPosition(1)
            .withoutSwipe();
        c.setWidth("550px");
        c.setHeight("300px");


        H3 akapit = new H3( text: "CIEKAWY PRZEDMIOTY:");

        updateItems();

        add(c, akapit, itemsContainer);
    }

    public static Component createSlideContent(String url, String image) { 3 usages @mfrontiera
        Div result = new Div();
        result.addClassName("carousel-item");
        result.getStyle().set("background-image", "url('" + image + "')");
        result.addClickListener(event -> {
            UI.getCurrent().navigate(url);
        });
        return result;
    }
}
```

Oraz jej odwziewiedlenie w przeglądarce:



[KONTO](#)  
[KOSZYK \(0\)](#)

[Druk 3D](#)  
[Filamenty](#)  
[Drukarki 3D](#)  
[Żywice](#)

[Smart Home](#)  
[GŁÓWNA](#)


DRUK 3D

Filamenty

Drukarki 3D


Żywice

SMART HOME



Filament PLA 1.75mm  
czarny 1kg  
Indeks: 51 Ilość: 12


**79.99 zł**



Drukarka 3D Ender 3 Pro

Indeks: 6 Ilość: 3


**899.99 zł**



Zestaw Arduino Uno R3

Indeks: 7 Ilość: 25


**99.99 zł**



Podgrzewany stół do drukarki 3D

Indeks: 9 Ilość: 7


**199.99 zł**



Czujnik temperatury DS18B20

Indeks: 10 Ilość: 30

**19.99 zł**



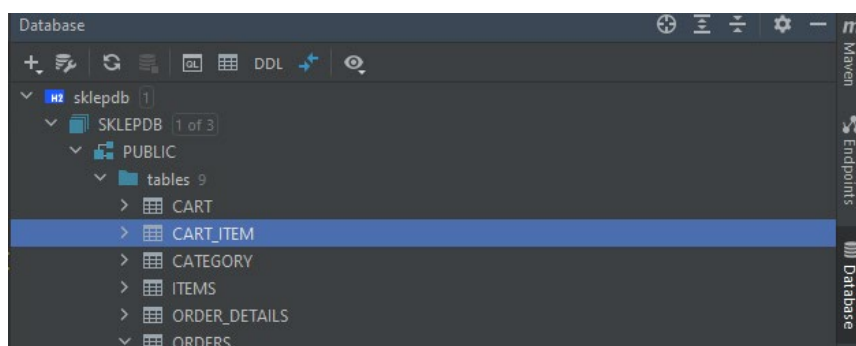
**CIEKAWY PRZEDMIOTY:**

# BAZA DANYCH H2

Jako bazę danych wybraliśmy H2, która jest lekka, szybka i łatwa w obsłudze. Zdecydowaliśmy się na tryb pracy w pliku, co pozwala na trwałe przechowywanie danych między kolejnymi uruchomieniami aplikacji.

Jako kluczowe cechy implementacji bazy danych ustaliliśmy:

- 1) **Trwałość danych:** Wszystkie dane aplikacji (np. produkty, użytkownicy, zamówienia) są zapisywane do pliku, co pozwala na ich odtworzenie po ponownym uruchomieniu aplikacji.
- 2) **Łatwość konfiguracji:** H2 wymaga minimalnego nakładu pracy podczas konfiguracji i integracji z aplikacją dzięki prostym ustawieniom w pliku konfiguracyjnym.
- 3) **Wsparcie dla SQL:** H2 obsługuje pełny zestaw poleceń SQL, co ułatwia zarządzanie danymi oraz pisanie zapytań do bazy.
- 4) **Integracja z Javą:** Baza danych H2 została zintegrowana z aplikacją przy użyciu **Spring Data JPA**, co pozwala na łatwe operacje CRUD (tworzenie, odczyt, aktualizacja, usuwanie) przy użyciu obiektów Java.
- 5) **Tryb pliku:** Dane są przechowywane w lokalnym pliku `.mv.db`, co sprawia, że aplikacja nie wymaga instalacji zewnętrznego serwera bazy danych.
- 6) **Szybkość działania:** H2 zapewnia wysoką wydajność, co sprawia, że aplikacja działa płynnie nawet przy większej liczbie operacji na danych.
- 7) **Dostęp do konsoli H2:** W trakcie rozwoju aplikacji możliwy jest podgląd danych bezpośrednio w narzędziu IntelliJ, co ułatwia debugowanie i testowanie.



## 3.KOMPONENTY

### WIDOKI VAADIN

W aplikacji wykorzystano framework **Vaadin**, który pozwala na tworzenie nowoczesnych, responsywnych i interaktywnych interfejsów użytkownika w przeglądarce, wykorzystując wyłącznie język Java. Vaadin umożliwia tworzenie widoków za pomocą komponentów UI, co pozwoliło na zaprojektowanie spójnego i przyjaznego interfejsu użytkownika sklepu internetowego.

W aplikacji użyto szerokiej gamy komponentów Vaadin, takich jak:

- 1) **Grid**: Wyświetlanie tabeli z produktami lub zamówieniami.
- 2) **FormLayout**: Tworzenie formularzy w widokach rejestracji, logowania i realizacji zamówienia.
- 3) **VerticalLayout i HorizontalLayout**: Organizacja elementów na stronie w sposób responsywny.
- 4) **ComboBox**: Wybór opcji dostawy w widoku realizacji zamówienia.
- 5) **Button**: Obsługa akcji użytkownika, takich jak dodawanie produktów do koszyka czy finalizacja zamówienia.
- 6) **Notification**: Wyświetlanie powiadomień o działaniach użytkownika, np. „Produkt dodany do koszyka”.

Zalety zastosowania Vaadin

- 1) **Responsywność**: Aplikacja działa poprawnie zarówno na komputerach, jak i urządzeniach mobilnych.
- 2) **Prostota tworzenia widoków**: Dzięki Vaadin całość interfejsu można tworzyć w Javie, bez potrzeby korzystania z osobnych technologii frontendowych (HTML, CSS, JavaScript).
- 3) **Spójny wygląd**: Gotowe komponenty Vaadin zapewniają profesjonalny i jednolity design interfejsu użytkownika.
- 4) **Integracja z backendem**: Vaadin pozwala na płynną komunikację z warstwą backendową, co umożliwia np. łatwe wyświetlanie danych z bazy H2.

Przykład widoku vaadin dla strony głównej:

```
public class MainViewView extends VerticalLayout implements RouterLayout {

    private OrderedList<Item> itemsContainer; // 5 usages
    private List<Item> items; // 2 usages
    private final ItemRepository itemRepository; // 2 usages

    public MainViewView(ItemRepository itemRepository) { // 1 mfrontiera
        this.itemRepository = itemRepository;

        itemsContainer = new OrderedList();
        itemsContainer.addClassNames(LumoUtility.Gap.MEDIUM, LumoUtility.Display.FLEX, LumoUtility.FlexDirection.ROW, LumoUtility.FlexWrap.WRAP, LumoUtility.Margin);

        setJustifyContentMode(JustifyContentMode.START);
        setAlignItems(Alignment.CENTER);
        Slide s1 = new Slide(createSlideContent( url: "arduino-learning-kit", image: "https://u.cubeupload.com/korylek/arduinoakit.jpg"));
        Slide s2 = new Slide(createSlideContent( url: "ender3-v3", image: "https://u.cubeupload.com/korylek/ender3v3.jpg"));
        Slide s3 = new Slide(createSlideContent( url: "flipper-zero", image: "https://u.cubeupload.com/korylek/flipperzero30.jpg"));

        Carousel c = new Carousel(s1,s2,s3).withAutoProgress()
            .withSlideDuration(4)
            .withStartPosition(1)
            .withoutSwipe();
        c.setWidth("550px");
        c.setHeight("300px");

        H3 akapit = new H3( text: "CIEKAWY PRZEDMIOTY:");

        updateItems();

        add(c, akapit, itemsContainer);
    }

    public static Component createSlideContent(String url, String image) { // 3 usages, 1 mfrontiera
```

Kafelek przedmiotu:

```
public ItemViewCard(Item item) { // 3 usages, 1 mfrontiera
    this.securityService = SpringContext.getBean(SecurityService.class);
    this.cartService = SpringContext.getBean(CartService.class);
    this.sessionCartService = SpringContext.getBean(SessionCartService.class);

    Dialog cartDialog = new Dialog();

    IntegerField selectCart = new IntegerField();
    selectCart.setMin(0);
    selectCart.setValue(0);
    selectCart.setStepButtonsVisible(true);
    selectCart.setLabel("Koszyk");
    cartDialog.add(selectCart);

    //setWidth("20%");
    addClassNames(LumoUtility.Background.CONTRAST_S, LumoUtility.Display.FLEX, LumoUtility.FlexDirection.COLUMN, LumoUtility.AlignItems.START, LumoUtility.Padding, LumoUtility.BorderRadius.LARGE);

    Div div = new Div();
    div.addClassNames(LumoUtility.Background.CONTRAST, LumoUtility.Display.FLEX, LumoUtility.AlignItems.CENTER, LumoUtility.JustifyContent.CENTER, LumoUtility.Margin.Bottom.MEDIUM, LumoUtility.Overflow.HIDDEN, LumoUtility.BorderRadius.MEDIUM, LumoUtility.Width.FULL);
    div.setHeight("160px");

    Image image = new Image(item.getImageUrl(), item.getShortDescription());
    image.setWidth("100%");
    div.add(image);
    result = slug.slugify(item.getName());
    Anchor nameLink = new Anchor(item.getFullPath(), item.getName());

    Span header = new Span(nameLink);
    header.addClassNames(LumoUtility.FontSize.MEDIUM, LumoUtility.FontWeight.LIGHT);
    header.getStyle().set("word-wrap", "break-word");
    header.getStyle().set("white-space", "normal");

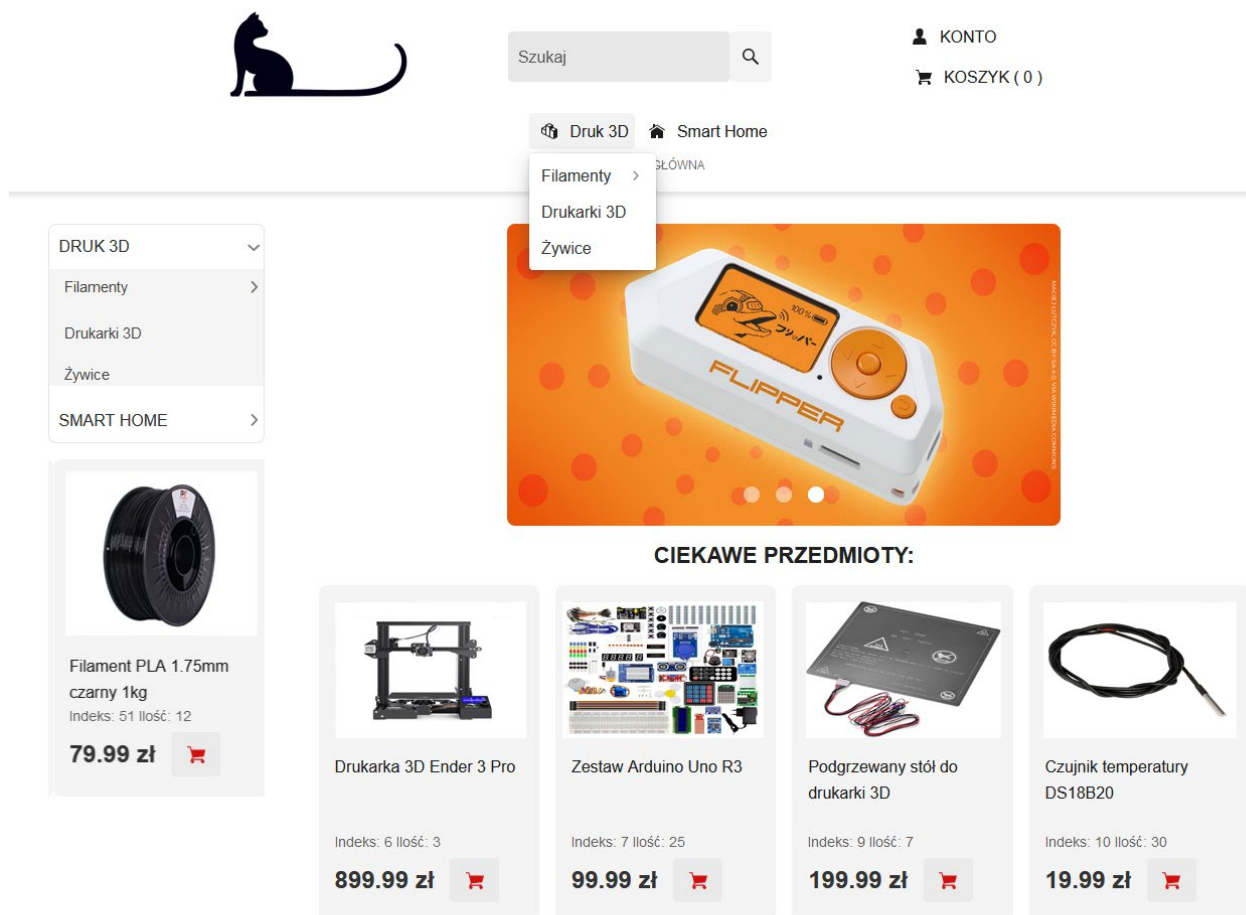
    Div headerContainer = new Div(header);
    headerContainer.setHeight("30%");
    //setWidth("100%");
```

# INTERFEJS STRONY

## Strona główna

Na stronie głównej mamy takie komponenty jak:

- 1) Karuzele zdjęć która przedstawia wyróżnione produkty, informacje o wydarzeniach lub zapowiedzi
- 2) Karuzele najnowszych produktów
- 3) Kilka produktów które mogą zaciekać użytkownika



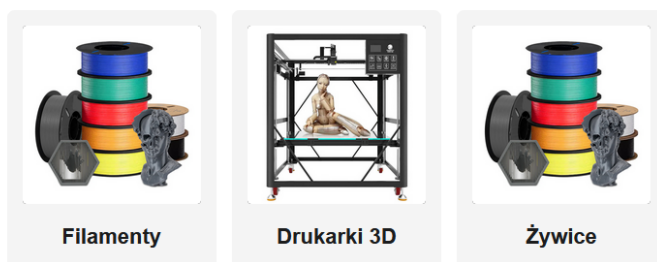
## Kategorie

Strona każdej kategorii jest tworzona dynamicznie, zawiera przejścia do podkategorii oraz swoje przedmioty

STRONA GŁÓWNA > **DRUK3D**

## Druk 3D

Druk 3D to technologia wytwarzania obiektów poprzez nakładanie kolejnych warstw materiału zgodnie z cyfrowym modelem. Proces rozpoczyna się od stworzenia projektu w programie CAD, który następnie jest przesyłany do drukarki 3D. Urządzenie, wykorzystując materiały takie jak plastik, metal, żywice czy ceramika, buduje przedmiot warstwa po warstwie.



Sortuj 56

Nazwa A-Z

← Poprzednia

Strona 1 z 3

→ Następna

## Filamenty

Filamenty to materiały wykorzystywane w drukarkach 3D działających w technologii FDM (Fused Deposition Modeling). Mają formę cienkiego drutu nawiniętego na szpulę, który podczas druku jest podgrzewany i nakładany warstwami w celu stworzenia obiektu.



Sortuj 41

Nazwa A-Z

← Poprzednia

Strona 1 z 2

→ Następna

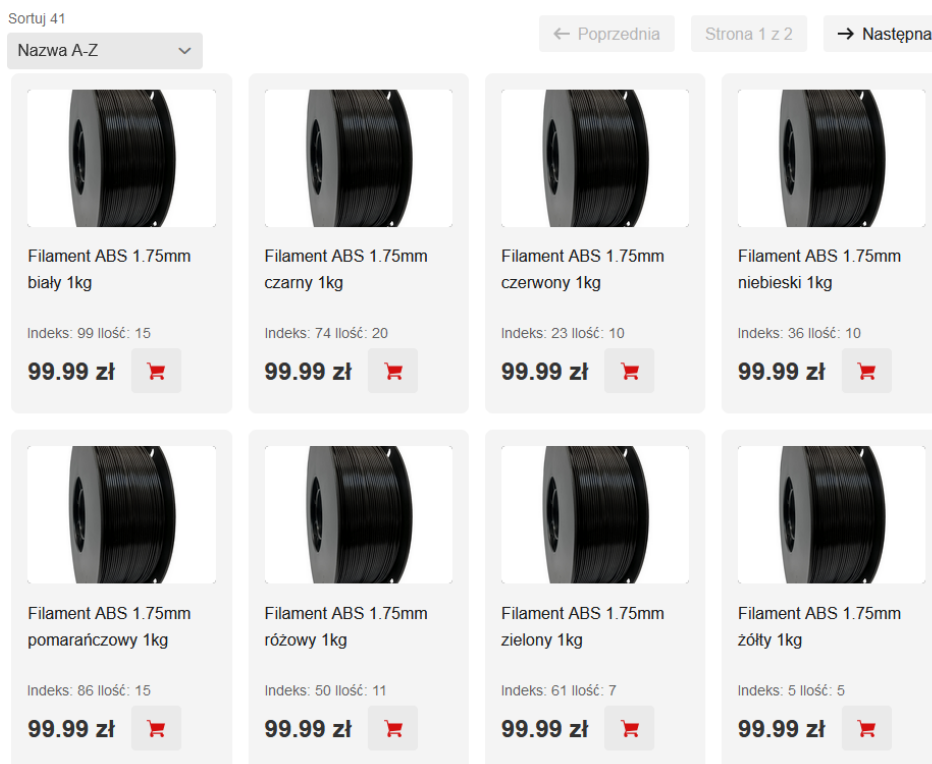


## Karty przedmiotów

Zawierają zdjęcie przedmiotu, nazwę, indeks, cenę oraz przycisk dodania do koszyka oraz ilość na stanie



Aby zachować przejrzystość, jeśli przedmiotów jest więcej niż 20 są one dzielone na strony. Podgląd posiada również możliwość sortowania ceną lub nazwą.





Sortuj 41

Cena rosnąco

← Poprzednia

Strona 1 z 2

→ Następna



Szpulka filamentu PLA  
1.75mm zielony 0.5kg

Indeks: 8 Ilość: 12

**45.99 zł**



Filament PLA 1.75mm  
biały 1kg

Indeks: 1 Ilość: 10

**79.99 zł**



Filament PLA 1.75mm  
czarny 1kg

Indeks: 51 Ilość: 12

**79.99 zł**



Filament PLA 1.75mm  
fioletowy 1kg

Indeks: 89 Ilość: 6

**79.99 zł**



Widok kategorii oraz przedmiotów dodaje się automatycznie przy zmianie ilości / nazw w bazie danych.

# INTERFEJS UŻYTKOWNIKA (UI)

Interfejs użytkownika na stronie sklepu internetowego został zaprojektowany z myślą o intuicyjności i wygodzie użytkownika. Kluczowym celem było stworzenie przejrzystego układu, który umożliwi użytkownikom szybkie poruszanie się po stronie, przeglądanie produktów oraz realizację zamówień.

Najważniejsze elementy UI:

## Breadcrumbs:

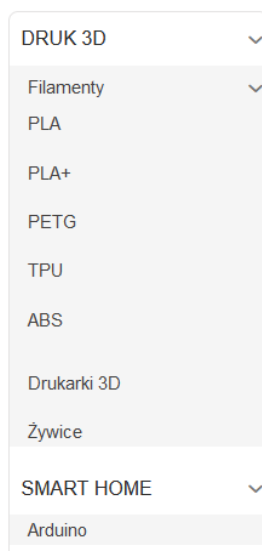
- 1) Pozwalają użytkownikowi śledzić swoją aktualną lokalizację w strukturze strony.
- 2) Umożliwiają szybki powrót do poprzednich kategorii lub głównej strony.
- 3) Dzięki breadcrumbs użytkownicy mogą poruszać się w głąb kategorii produktów, zachowując pełną orientację w strukturze sklepu.

STRONA GŁÓWNA > DRUK3D > DRUKARKI3D > DRUKARKA-3D-ENDER-3-PRO

## Produkt: Drukarka 3D Ender 3 Pro

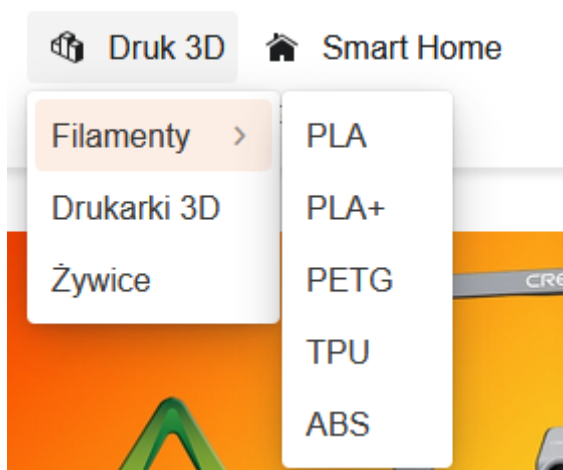
## Rozwijane menu kategorii po lewej stronie:

- 1) Umieszczone w widocznym miejscu na stronie, menu umożliwia użytkownikom łatwy dostęp do głównych kategorii produktów.
- 2) Po rozwinięciu pokazuje podkategorie, co pozwala na szybkie dotarcie do konkretnych grup produktów.
- 3) Rozwijane menu jest idealnym rozwiązaniem dla sklepów z dużą ilością produktów, ponieważ eliminuje konieczność przeszukiwania całej oferty.



### Menu górne z kategoriami i podkategoriami:

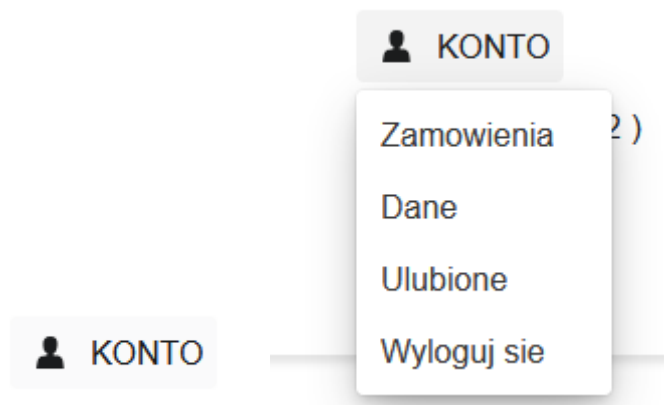
- 1) Menu górne zawiera główne kategorie produktów, które są łatwo dostępne na każdej podstronie.
- 2) Podkategorie rozwijają się w przejrzysty sposób po najechaniu kursorem, co pozwala na szybki wybór interesującej sekcji.
- 3) Menu zaprojektowano tak, aby było responsywne i działało poprawnie na urządzeniach mobilnych.



Zarówno jak menu lewe i górne aktualizuje się automatycznie po dodaniu nowych kategorii do bazy danych.


### Przycisk przejścia do konta/logowania:

- 1) Widoczny w prawym górnym rogu strony, umożliwia użytkownikom szybki dostęp do panelu logowania lub zakładki konta.
- 2) Po zalogowaniu przycisk zmienia swoją funkcję, oferując dostęp do opcji zarządzania kontem, takich jak przeglądanie historii zamówień czy edycja danych.
- 3) Dzięki temu elementowi proces rejestracji i logowania jest szybki i intuicyjny.




## Koszyk:

- 1) Ikona koszyka znajduje się w prawym górnym rogu strony i jest widoczna na każdej podstronie.
- 2) Zawiera licznik produktów, który informuje użytkownika o liczbie dodanych przedmiotów bez potrzeby otwierania koszyka.
- 3) Kliknięcie w koszyk prowadzi użytkownika do strony z podsumowaniem zamówienia, gdzie może edytować swój wybór, usunąć produkty lub przejść do finalizacji zamówienia.
- 4) Jeśli kupujący jest zalogowany, zawartość koszyka zapisuje się na jego koncie
- 5) Dodanie więcej niż jednego takiego samego produktu powiększa jego ilość w koszyku


 **KOSZYK ( 2 )**

**Produkty**







**Moduł sterownika LED WS2812**  
Indeks: 97

79.98



**Filament ABS 1.75mm czarny 1kg**  
Indeks: 74

99.99

	Produkt	Ilość	Razem	
	<b>Moduł sterownika LED WS2812</b> Indeks: 97	2	79.98 zł	
	<b>Filament ABS 1.75mm czarny 1kg</b> Indeks: 74	1	99.99 zł	

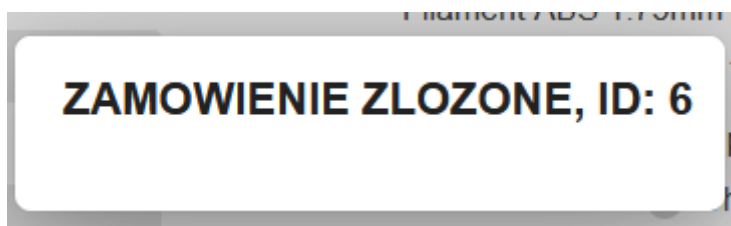
**REALIZUJ ZAMOWIENIE**

## Strona realizacji zamówienia:

Po przejściu do finalizacji zamówienia użytkownik jest przekierowany na stronę, gdzie może wprowadzić szczegóły potrzebne do wysyłki:

- 1) **Adres dostawy:** Użytkownik wprowadza dane, takie jak imię, nazwisko, ulica, numer domu/mieszkania, miasto, kod pocztowy oraz kraj.
- 2) **Wybór kuriera:** Na stronie dostępne są opcje wyboru dostawcy (np. DHL, InPost, DPD) wraz z kosztami przesyłki oraz szacowanym czasem dostawy.
- 3) **Podgląd zamówienia:** Użytkownik może sprawdzić listę produktów w koszyku, ich ilości oraz łączny koszt zamówienia.
- 4) **Przycisk potwierdzenia zamówienia:** Po uzupełnieniu danych użytkownik klika przycisk „Złóż zamówienie”, który przesyła informacje do systemu, generuje podsumowanie zamówienia oraz potwierdzenie.

Imię •	Jan	<b>Produkty</b>  Moduł sterownika LED WS2812 79,98 zł Filament ABS 1.75mm czarny 1kg 99,99 zł  Całość: 179,97 zł  <input checked="" type="radio"/> Inpost 10 zł <input type="radio"/> Dhl 20 zł <input type="radio"/> DPD 15 zł  <b>Złóż zamówienie</b>
Nazwisko •	Kowalski	
Adres •	Gdansk	
Kod pocztowy •	01-420	
Miasto •	Gdansk	
Numer telefonu •	666333666	



Po pomyślnym przetworzeniu zamówienia ilość przedmiotów „na stanie” zmniejsza się.

## 4. BAZA DANYCH

W projekcie wykorzystano bazę danych H2 w trybie zapisu do pliku, co zapewnia trwałość danych między kolejnymi uruchomieniami aplikacji. Baza została zintegrowana z aplikacją za pomocą Spring Data JPA, co umożliwia wygodne zarządzanie danymi w formie obiektów Java. Dzięki użyciu JPA proces tworzenia, odczytu, aktualizacji i usuwania danych został uproszczony i zautomatyzowany.

### STRUKTURA BAZY DANYCH

Baza danych zawiera zestaw tabel, które odzwierciedlają kluczowe funkcjonalności aplikacji sklepu internetowego.

Opis najważniejszych tabel:

#### **CART:**

- 1) Przechowuje dane dotyczące koszyków użytkowników.
- 2) Kluczowe kolumny:
  - a) id – identyfikator koszyka.
  - b) user\_id – powiązanie z użytkownikiem, który posiada koszyk.
  - c) total\_price – całkowity koszt produktów w koszyku.

#### **CART\_ITEM:**

- 1) Zawiera informacje o produktach dodanych do koszyka.
- 2) Kluczowe kolumny:
  - a) id – identyfikator pozycji w koszyku.
  - b) cart\_id – powiązanie z tabelą cart.
  - c) item\_id – identyfikator produktu.
  - d) quantity – liczba sztuk danego produktu.

#### **CATEGORY:**

- 1) Reprezentuje kategorie produktów w sklepie.
- 2) Kluczowe kolumny:
  - a) id – identyfikator kategorii.
  - b) name – nazwa
  - c) description – opis
  - d) image\_url – ścieżka do zdjęcia
  - e) parent\_id – id nadkategorii, null jeśli to korzeń

## **ITEMS:**

- 1) Przechowuje dane o produktach dostępnych w sklepie.
- 2) Kluczowe kolumny:
  - a) id – identyfikator produktu.
  - b) quantity – ilość produktu w magazynie
  - c) name – nazwa
  - d) price – cena
  - e) category\_id – powiązanie z tabelą category.
  - f) image\_url – ścieżka do zdjęcia
  - g) long\_description – długi opis produktu
  - h) short\_description – skrócony opis
  - i) created\_at – data dodania produktu (używane do wyświetlania najnowszych produktów)

## **ORDER:**

- 1) Zawiera dane o zamówieniach złożonych przez użytkowników.
- 2) Kluczowe kolumny:
  - a) id – identyfikator zamówienia.
  - b) user\_id – identyfikator użytkownika, który złożył zamówienie.
  - c) total\_price – całkowity koszt zamówienia.
  - d) status – status zamówienia (np. „złożone”, „w realizacji”, „zakończone”).

## **ORDER\_DETAILS:**

- 1) Szczegóły dotyczące zamówienia, takie jak produkty i ich ilość.
- 2) Kluczowe kolumny:
  - a) id – identyfikator szczegółów zamówienia.
  - b) order\_id – powiązanie z tabelą order.
  - c) item\_id – identyfikator produktu.
  - d) quantity – liczba sztuk zamówionego produktu.
  - e) price – cena za przedmiot

**ROLES:**

- 1) Definiuje role użytkowników (np. „ADMIN”, „USER”).
- 2) Kluczowe kolumny:
  - a) id – identyfikator roli.
  - b) name – nazwa roli.

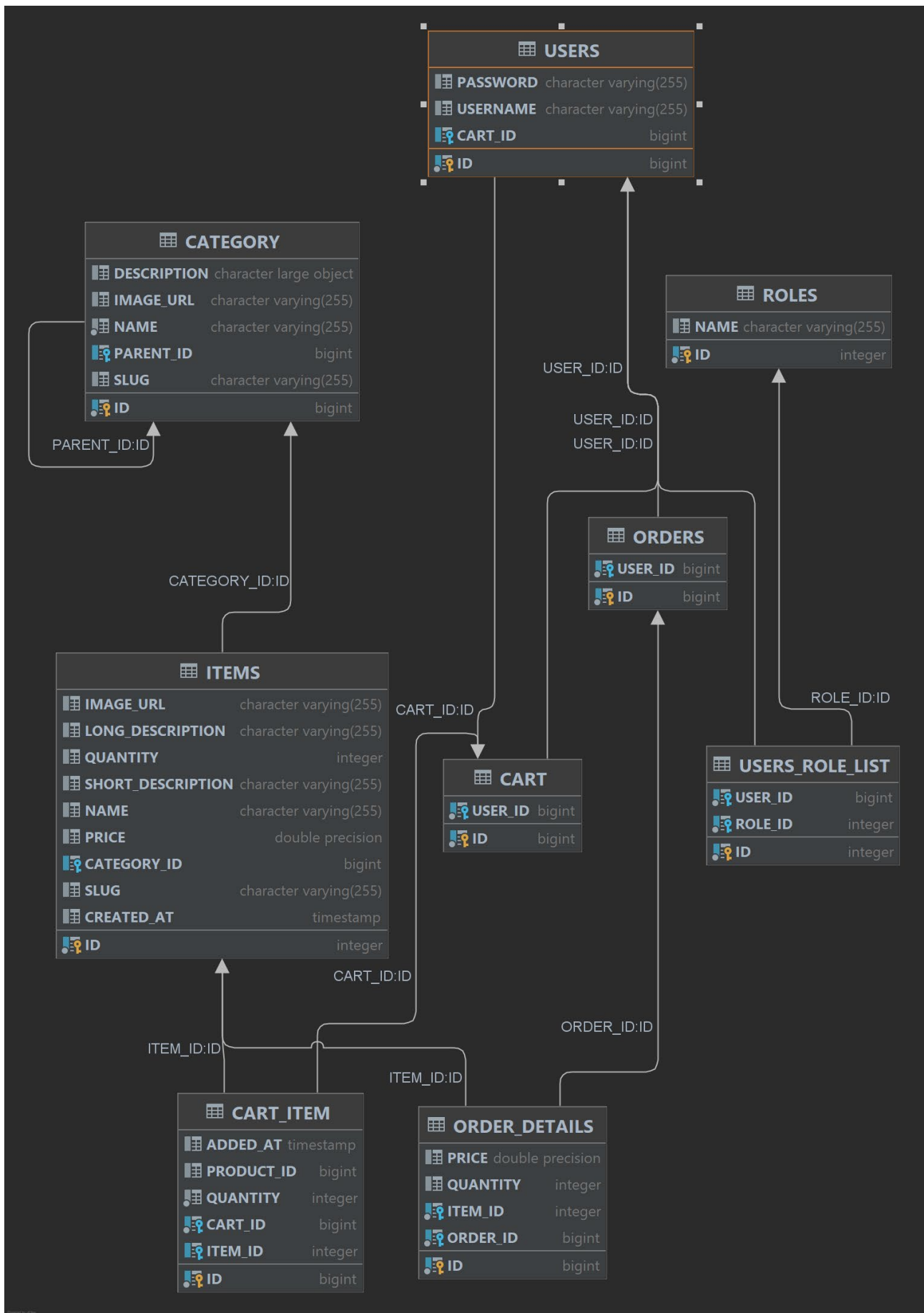
**USERS\_ROLE\_LIST:**

- 1) Łączy użytkowników z ich rolami w systemie.
- 2) Kluczowe kolumny:
  - a) user\_id – identyfikator użytkownika.
  - b) role\_id – identyfikator roli.

**USERS:**

- 1) Zawiera dane użytkowników sklepu.
- 2) Kluczowe kolumny:
  - a) id – identyfikator użytkownika.
  - b) username – nazwa użytkownika.
  - c) password – hasło użytkownika (zaszyfrowane).
  - d) email – adres e-mail użytkownika.





# KORZYSTANIE ZE SPRING DATA JPA

Dzięki użyciu **Spring Data JPA** dane w bazie są zarządzane za pomocą encji, co ułatwia ich obsługę w aplikacji.

Przykład konfiguracji encji oraz repozytorium dla items:

Encja:

```
1 package com.example.application.model;
2
3 import ...
4
5 @Data
6 @Entity
7 @EqualsAndHashCode(of = "id")
8 @Table(name = "items") // Określenie nazwy tabeli w bazie danych
9 public class Item {
10
11     @Getter
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY) // Auto-generowanie ID
14     private int id;
15
16     @Getter
17     @Column(name = "name", nullable = false)
18     private String name;
19
20     @Getter
21     @Column(name = "quantity", nullable = false) // Określenie nazwy kolumny w tabeli
22     private int quantity;
23
24     @Getter
25     @Column(name = "image_url")
26     private String imageUrl;
27
28     @Getter
29     @Column(name = "short_description")
30     private String shortDescription;
31
32     @Getter
33     @Column(name = "slug")
34     private String slug;
35
36     @Getter
37     @Column(name = "long_description")
38     private String longDescription;
39
40     @Getter
41     @Column(name = "price", nullable = false)
42     private double price;
43
44     @ManyToOne(fetch = FetchType.LAZY)
45     @JoinColumn(name = "category_id")
46     private Category category;
47
48     @Column(name = "created_at", updatable = false)
49     private LocalDateTime createdAt;
50
51     @PrePersist
52     public void prePersist() {
53         this.createdAt = LocalDateTime.now();
54     }
55 }
```

## Repozytorium:

```
@Repository  mfrontiera
public interface ItemRepository extends JpaRepository<Item, Integer> {
    List<Item> findByNameContainingIgnoreCase(String name, PageRequest pageRequest); 1 usage  mfrontiera
    List<Item> findByNameContainingIgnoreCase(String name); 2 usages  mfrontiera
    Item findById(int id); 5 usages  mfrontiera
    long countByNameContainingIgnoreCase(String name); no usages  mfrontiera
    List<Item> findByIdByCategoryId(Long id); 1 usage  mfrontiera
    List<Item> findTop5ByOrderByCreatedAtDesc(); 1 usage  mfrontiera
}
```

### Zalety zastosowania bazy H2 i Spring Data JPA

- 1) **Łatwość obsługi:** Spring Data JPA upraszcza zarządzanie danymi, eliminując konieczność pisania ręcznych zapytań SQL.
- 2) **Automatyczne mapowanie danych:** Dane w tabelach są automatycznie mapowane na obiekty Java, co ułatwia manipulację danymi w kodzie.
- 3) **Trwałość danych:** Dzięki trybowi zapisu do pliku dane w bazie są zachowywane między sesjami aplikacji.
- 4) **Prototypowanie i rozwój:** H2 idealnie nadaje się do szybkiego prototypowania i testowania aplikacji, dzięki czemu czas wdrożenia aplikacji jest krótszy.

## 5. ORGANIZACJA KODU APLIKACJI

Aby utrzymać porządek w projekcie i ułatwić jego rozwój, kod aplikacji został odpowiednio podzielony na foldery, z których każdy odpowiada za inne aspekty funkcjonalności. Taka struktura pozwala na lepszą czytelność kodu, ułatwia jego utrzymanie oraz rozwój w przyszłości.

Podział na foldery:

### 1. Widoki (Views)

- 1) Wszystkie klasy odpowiedzialne za tworzenie interfejsu użytkownika (UI) zostały umieszczone w dedykowanym folderze.
- 2) Każdy widok, np. CartView, OrderView, czy CategoryView, odpowiada za konkretną część aplikacji.
- 3) Dzięki temu UI jest odseparowane od logiki biznesowej i danych.

### 2. Encje (Entities)

- 1) Encje reprezentujące tabele w bazie danych znajdują się w folderze entities.
- 2) Każda encja, np. Cart, Item, czy User, odpowiada bezpośrednio strukturze tabeli w bazie H2.
- 3) Dzięki zastosowaniu **Spring JPA** encje automatycznie mapują dane z bazy na obiekty w Javie, co pozwala na łatwą manipulację nimi w kodzie.

### 3. Serwisy (Services)

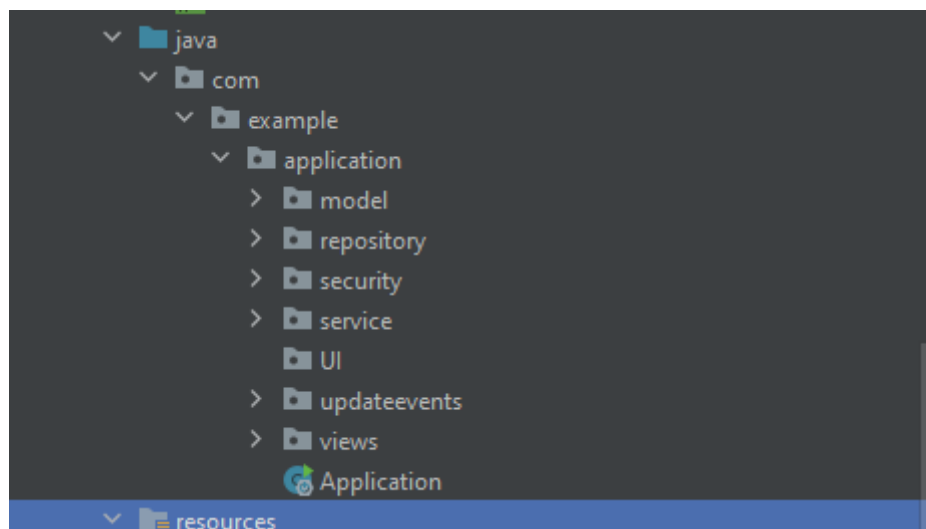
- 1) Logika biznesowa została przeniesiona do warstwy serwisowej, która znajduje się w folderze services.
- 2) Przykładowe klasy serwisowe:
  - a) CartService: Obsługuje operacje związane z koszykiem (dodawanie, usuwanie produktów, obliczanie ceny całkowitej).
  - b) OrderService: Zarządza składaniem zamówień i powiązanymi szczegółami.
  - c) UserService: Odpowiada za zarządzanie użytkownikami, np. rejestrację, logowanie, czy nadawanie ról.
- 3) Serwisy oddzielają logikę aplikacji od jej widoków i zapewniają jej ponowne wykorzystanie w różnych częściach aplikacji.

#### 4. Repozytoria (Repositories)

- 1) Obsługa bazy danych została umieszczona w folderze repositories.
- 2) Każda encja ma dedykowane repozytorium (np. CartRepository, ItemRepository), które korzysta z mechanizmów **Spring Data JPA**.
- 3) Repozytoria zawierają podstawowe operacje CRUD oraz niestandardowe zapytania (np. wyszukiwanie produktów po nazwie).

#### 5. Konfiguracja (Configuration)

- 1) Konfiguracje aplikacji, np. konfiguracja bazy danych H2 czy zarządzanie bezpieczeństwem (Spring Security), zostały wydzielone do folderu config.
- 2) Przykłady:
  - a) SecurityConfig: Definiuje zasady autoryzacji i dostęp do poszczególnych zasobów aplikacji.
  - b) VaadinConfig: Dostosowuje ustawienia Vaadin do specyfiki projektu.



## 6. PODSUMOWANIE

Celem projektu było stworzenie kompleksowej aplikacji internetowej sklepu online, która łączy w sobie nowoczesne technologie i dobre praktyki programistyczne. W trakcie realizacji projektu skupiono się na funkcjonalności, przejrzystości interfejsu użytkownika oraz modularnej architekturze kodu. Dzięki zastosowanym technologiom i dobrej organizacji projektu, powstała aplikacja, która spełnia założone cele. Jest to nowoczesny, funkcjonalny sklep internetowy, który umożliwia użytkownikom intuicyjne przeglądanie produktów, zarządzanie koszykiem i składanie zamówień. Struktura kodu pozwala na łatwe wdrożenie nowych funkcji i utrzymanie aplikacji w przyszłości, co czyni projekt zarówno użytecznym, jak i elastycznym w kontekście dalszego rozwoju.