**Dokumentacja aplikacji sklepu wędkarskiego.**

**Zespół:**

Dariusz Dąbkowski

Kinga Prochowicz

Kacper Fiediukiewicz

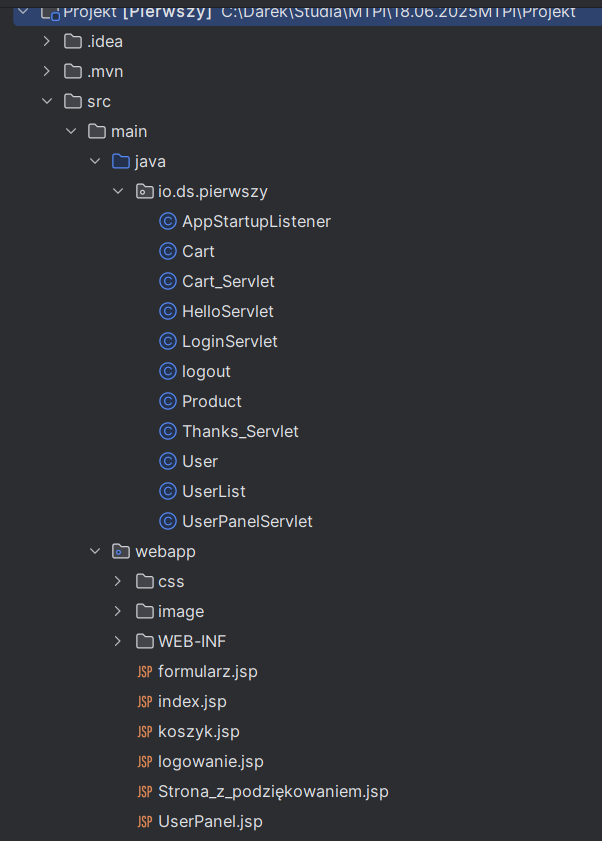
**Wstęp**

Projekt sklepu wędkarskiego to aplikacja webowa napisana w języku Java, umożliwiająca rejestrację oraz logowanie użytkownika, przeglądanie produktów, dodawanie ich do koszyka i składanie zamówień. Kluczowym elementem projektu jest wykorzystanie struktury danych HashMap do zarządzania zawartością koszyka zakupów. Mapowanie odbywa się na zasadzie klucz-wartość, gdzie kluczem jest identyfikator produktu, a wartością obiekt przechowujący informacje o ilości i cenie. Dzięki HashMap możliwe jest szybkie dodawanie, aktualizowanie i usuwanie pozycji z koszyka. Takie rozwiązanie zapewnia wydajność i przejrzystość kodu w obsłudze operacji koszykowych.

**Link:**

<https://github.com/Niewyczes/Projekt.git>

**Struktura kodu**



**1.) AppStartupListener -** jest komponentem odpowiedzialnym za inicjalizację danych aplikacji w momencie jej uruchomienia. Implementuje interfejs ServletContextListener, dzięki czemu wykonuje swoją logikę w metodzie contextInitialized(), która zostaje wywołana automatycznie podczas startu kontekstu serwletowego. Została oznaczona adnotacją @WebListener, co pozwala serwerowi aplikacji (np. Tomcatowi) automatycznie wykryć i uruchomić ją bez konieczności ręcznej konfiguracji w pliku web.xml. Główne zadanie tej klasy polega na przygotowaniu listy produktów, które następnie zostają zapisane w kontekście aplikacji jako atrybut globalny pod nazwą "globalProducts". Lista ta zawiera obiekty klasy Product, które reprezentują różnorodne produkty dostępne w sklepie wędkarskim. Produkty zostały pogrupowane w kategorie takie jak: Wędziska (np. spinningowe, gruntowe), Akcesoria (m.in. spławiki, żyłki, krzesła), Przynęty (woblery, pellet, kulki proteinowe) oraz Podbieraki (siatki, chwytaki, magnety). Każdy produkt zawiera unikalny identyfikator, nazwę kategorii, nazwę własną, cenę, opis oraz nazwę pliku graficznego powiązanego z produktem. Zapisanie listy produktów w kontekście (ServletContext) umożliwia ich późniejsze wykorzystanie w różnych częściach aplikacji, np. w widokach JSP lub w serwletach, bez potrzeby każdorazowego pobierania tych danych z bazy czy ponownego tworzenia ich w pamięci. Metoda contextDestroyed(), będąca częścią kontraktu interfejsu ServletContextListener, nie zawiera żadnej logiki – jest pusta, ponieważ w tym projekcie nie przewidziano żadnych operacji czyszczących przy zamknięciu aplikacji. Rozwiązanie to jest przykładem prostego mechanizmu inicjalizacji danych na czas działania aplikacji. Choć w środowiskach produkcyjnych dane zwykle ładowane są z bazy danych lub plików konfiguracyjnych, to tutaj ich umieszczenie w kodzie pozwala na szybki start aplikacji i testowanie jej funkcjonalności bez dodatkowego zaplecza.

**2.) Cart -** pełni funkcję pomocniczą w aplikacji sklepu wędkarskiego, reprezentując pojedynczą pozycję w koszyku użytkownika. Zawiera informacje o konkretnym produkcie oraz o jego ilości. Klasa ta działa jako kontener danych, umożliwiając przechowywanie i przetwarzanie pozycji koszykowych w sposób obiektowy i przejrzysty. Obiekt klasy Cart tworzony jest na podstawie przekazanego do konstruktora produktu (instancji klasy Product) oraz liczby sztuk tego produktu. Produkt jest przechowywany jako pole finalne, co oznacza, że po utworzeniu obiektu Cart nie można już go zmienić – można jedynie modyfikować liczbę sztuk poprzez setter setQuantity(int quantity). Do odczytu pól udostępnione są metody getProduct() oraz getQuantity(). Dodatkowo, klasa posiada metodę getTotalPrice(), która oblicza całkowitą cenę pozycji w koszyku, mnożąc cenę jednostkową produktu przez jego ilość. To rozwiązanie pozwala szybko i efektywnie uzyskać wartość danej pozycji, co jest przydatne np. podczas wyświetlania podsumowania zamówienia lub obliczania sumy końcowej. Klasa ta, choć niewielka, stanowi istotny element logiki koszyka zakupowego i współpracuje bezpośrednio z kolekcją typu HashMap, gdzie kluczem może być identyfikator produktu, a wartością obiekt klasy Cart.

**3.)Cart\_Servlet -** jest serwletem odpowiedzialnym za obsługę żądań związanych z dodawaniem produktów do koszyka zakupowego w aplikacji sklepu wędkarskiego. Została zarejestrowana pod adresem URL /cart-servlet dzięki adnotacji @WebServlet, co pozwala użytkownikowi dodawać produkty do koszyka poprzez formularz lub przycisk w interfejsie użytkownika. W metodzie doPost() serwlet przetwarza dane przesłane w żądaniu HTTP POST, pobierając identyfikator produktu (productId) oraz jego ilość (quantity). Następnie, na podstawie listy produktów zapisanej w kontekście aplikacji pod nazwą globalProducts, odnajduje obiekt Product o wskazanym identyfikatorze. Ta lista została wcześniej utworzona w klasie AppStartupListener podczas startu aplikacji. Koszyk użytkownika jest przechowywany w sesji jako mapa (Map<Integer, Integer>), w której kluczem jest identyfikator produktu, a wartością – ilość sztuk danego produktu. Jeśli koszyk nie istnieje jeszcze w sesji, tworzony jest nowy obiekt HashMap. Następnie metoda aktualizuje zawartość koszyka, dodając lub zwiększając ilość danego produktu. Użycie metody getOrDefault() pozwala bezpiecznie pobrać aktualną liczbę sztuk danego produktu, nawet jeśli nie został wcześniej dodany do koszyka. Dodatkowo w sesji zapisywane są dwie informacje: nazwa ostatnio dodanego produktu (LastAddedProductName) oraz liczba dodanych sztuk (LastAddedQuantity). Umożliwia to np. wyświetlenie komunikatu potwierdzającego dodanie produktu na stronie głównej. Po wykonaniu operacji serwlet przekierowuje użytkownika z powrotem na stronę index.jsp z parametrem ?success=1, co może być używane do wyświetlenia odpowiedniego komunikatu informacyjnego w interfejsie użytkownika.Serwlet ten stanowi istotny element logiki obsługi koszyka i ściśle współpracuje z sesją użytkownika oraz wcześniej zainicjalizowanymi danymi produktów.

**4.) HelloServlet -** pełni funkcję serwletu obsługującego rejestrację nowych użytkowników w aplikacji sklepu wędkarskiego. Zarejestrowana została pod adresem URL /hello-servlet za pomocą adnotacji @WebServlet, co umożliwia jej wywołanie np. z formularza rejestracyjnego znajdującego się w pliku formularz.jsp. Metoda doGet() przekierowuje użytkownika do strony formularza rejestracyjnego – formularz.jsp. Dzięki temu możliwe jest wyświetlenie formularza po wejściu na odpowiedni adres URL. Główna logika serwletu znajduje się jednak w metodzie doPost(), która obsługuje przesyłanie danych z formularza rejestracji. W metodzie doPost() serwlet odczytuje z żądania dane przesłane przez użytkownika: imię, nazwisko, login i hasło. Następnie przeprowadza szczegółową walidację wszystkich pól. Sprawdzane są m.in. długość pól imienia i nazwiska (od 4 do 20 znaków), zawartość tylko liter (dla imienia i nazwiska), długość loginu (od 5 do 20 znaków), unikalność loginu (czy nie jest już zajęty), a także złożoność hasła (wymagana długość, przynajmniej jeden znak specjalny oraz jedna cyfra). Jeśli którykolwiek z warunków walidacyjnych nie zostanie spełniony, użytkownik jest odsyłany z powrotem na formularz rejestracyjny, a do obiektu request dodawany jest komunikat o błędzie pod atrybutem error, który następnie może być wyświetlony w interfejsie. Po przejściu walidacji tworzony jest nowy obiekt klasy User, który zostaje dodany do listy użytkowników za pomocą UserList.addUser(newUser). Następnie serwlet uzyskuje dostęp do sesji użytkownika (HttpSession) i zapisuje w niej dane nowo zarejestrowanego użytkownika. Na koniec przeglądarka zostaje przekierowana na stronę index.jsp, co oznacza pomyślną rejestrację i automatyczne zalogowanie nowego użytkownika. Metody init() oraz destroy() są puste, ale ich obecność pozwala na ewentualną przyszłą rozbudowę, np. inicjalizację zasobów lub ich zwalnianie przy zamykaniu serwletu. HelloServlet jest więc kluczowym komponentem rejestracji użytkowników, zawierającym nie tylko mechanizm tworzenia kont, ale również zabezpieczenia w postaci walidacji danych i mechanizmu sesji, umożliwiającego zalogowanie użytkownika zaraz po rejestracji.

**5.)LoginServlet -** odpowiada za przetwarzanie żądań logowania użytkownika w aplikacji sklepu wędkarskiego. Został on zarejestrowany pod adresem /login-servlet dzięki adnotacji @WebServlet, co umożliwia wysyłanie formularza logowania (z poziomu widoku logowanie.jsp) bez konieczności konfiguracji w pliku web.xml. W metodzie doPost() serwlet pobiera z obiektu HttpServletRequest wartości parametrów login oraz haslo. W przypadku, gdy któryś z tych parametrów jest pusty lub nie został dostarczony, ustawia atrybut error z komunikatem „Wprowadź login i hasło.” i przekazuje żądanie z powrotem do widoku logowanie.jsp, aby użytkownik miał możliwość poprawienia danych. Jeśli oba pola wypełniono, serwlet odwołuje się do klasy UserList, aby pobrać obiekt User o podanym loginie. Gdy użytkownik o takim loginie istnieje i hasło się zgadza (user.getPassword().equals(password)), tworzona jest (lub pobierana istniejąca) sesja HTTP, do której trafia atrybut user zawierający dane zalogowanego użytkownika. Następnie serwlet przekierowuje przeglądarkę na stronę główną index.jsp. W przypadku nieistniejącego loginu lub niezgodnego hasła serwlet ustawia atrybut error z komunikatem „Nieprawidłowy login lub haslo” i z powrotem wywołuje widok logowanie.jsp, co pozwala na wyświetlenie komunikatu użytkownikowi. Metoda destroy() jest zaimplementowana, ale w tej wersji nie zawiera specjalnych operacji czyszczenia zasobów. Dzięki tej klasie aplikacja realizuje podstawową logikę uwierzytelniania użytkowników oraz zarządzanie stanem sesji.

**6.)logout -** jest serwletem odpowiedzialnym za wylogowanie użytkownika z aplikacji internetowej. Serwlet jest zarejestrowany pod adresem URL /logout dzięki adnotacji @WebServlet, co pozwala na obsługę żądań wylogowania przesyłanych metodą POST. W metodzie doPost() serwlet pobiera bieżącą sesję HTTP użytkownika i unieważnia ją, co skutkuje usunięciem wszystkich danych sesji oraz zakończeniem sesji użytkownika. Dzięki temu użytkownik jest skutecznie wylogowany z aplikacji. Następnie serwlet przekierowuje użytkownika na stronę główną index.jsp, co zwykle oznacza powrót do ekranu startowego aplikacji lub strony logowania. Serwlet ten pełni ważną rolę w zarządzaniu sesją użytkownika i bezpieczeństwie aplikacji, zapewniając prawidłowy mechanizm wylogowania.

**7.)Product -** reprezentuje pojedynczy produkt dostępny w aplikacji sklepu internetowego.

Klasa przechowuje podstawowe dane o produkcie, takie jak unikalny identyfikator (id), kategoria (category), nazwa (name), cena (price), opis (description) oraz ścieżka do obrazu (image). Podczas tworzenia obiektu produktu, wszystkie wymienione pola są inicjalizowane poprzez konstruktor. Dzięki temu każdy produkt ma przypisane kompletne informacje niezbędne do wyświetlenia i obsługi w aplikacji. Dostęp do danych produktu realizowany jest poprzez metody dostępowe (gettery), które zwracają wartości poszczególnych atrybutów. Pola klasy są w większości niemodyfikowalne po utworzeniu obiektu (finalne), co zapewnia bezpieczeństwo danych i zapobiega ich przypadkowej modyfikacji. Klasa Product jest podstawowym elementem modelu danych aplikacji, który służy do zarządzania i prezentacji oferty sklepu.

**8.)User -** reprezentuje użytkownika aplikacji sklepu internetowego. Zawiera podstawowe dane osobowe i informacje niezbędne do logowania oraz kontaktu, takie jak imię, nazwisko, login, hasło, numer telefonu oraz adres. Konstruktor klasy umożliwia utworzenie obiektu użytkownika z kompletem danych wymaganych do rejestracji i dalszej obsługi konta. Klasa udostępnia metody do pobierania (gettery) oraz modyfikowania (settery) tych danych, co pozwala na zarządzanie profilem użytkownika w trakcie działania aplikacji. Obiekt klasy User jest kluczowy dla funkcji takich jak logowanie, zarządzanie kontem, składanie zamówień oraz komunikacja z użytkownikiem.

**9.)UserList -** zarządza kolekcją użytkowników aplikacji. Jest to klasa pomocnicza przechowująca listę obiektów typu User w pamięci aplikacji, umożliwiająca dodawanie nowych użytkowników oraz wyszukiwanie ich na podstawie loginu. Lista użytkowników jest przechowywana jako statyczna kolekcja typu List<User>, co pozwala na dostęp do niej bez tworzenia instancji klasy. Metoda addUser(User user) dodaje nowego użytkownika do listy, pod warunkiem że w kolekcji nie istnieje już użytkownik o takim samym loginie. Jeśli użytkownik z podanym loginem już istnieje, metoda zwraca false, w przeciwnym razie dodaje użytkownika i zwraca true. Metoda getUser(String login) przeszukuje listę użytkowników i zwraca obiekt User o dopasowanym loginie lub null, jeśli użytkownik o takim loginie nie został znaleziony. Metoda exists(String login) sprawdza, czy w liście znajduje się użytkownik o podanym loginie, zwracając wartość logiczną true lub false. Klasa UserList stanowi prosty, ale istotny komponent do zarządzania danymi użytkowników w pamięci aplikacji.

**10.) UserPanelServlet -** jest serwletem odpowiedzialnym za obsługę panelu użytkownika w aplikacji internetowej.

Serwlet jest dostępny pod adresem URL /user-panel dzięki adnotacji @WebServlet i obsługuje zarówno żądania GET, jak i POST. W metodzie doGet() serwlet pobiera z sesji obiekt aktualnie zalogowanego użytkownika. Jeśli użytkownik nie jest zalogowany (obiekt user jest null), następuje przekierowanie na stronę logowania (logowanie.jsp). W przeciwnym przypadku obiekt użytkownika jest dodawany do atrybutów żądania i następuje przekazanie sterowania do widoku UserPanel.jsp, który wyświetla dane użytkownika. W metodzie doPost() serwlet również sprawdza, czy użytkownik jest zalogowany, a jeśli nie, przekierowuje na stronę logowania. Gdy użytkownik jest dostępny, serwlet pobiera dane przesłane w formularzu (imię, nazwisko, hasło, numer telefonu, adres) i weryfikuje, czy pola nie są puste. Jeśli tak, aktualizuje odpowiednie dane w obiekcie użytkownika. Po aktualizacji danych serwlet przekierowuje użytkownika z powrotem na adres /user-panel, co powoduje odświeżenie panelu i wyświetlenie zaktualizowanych informacji. Serwlet ten jest kluczowym elementem zarządzania profilem użytkownika, umożliwiającym przeglądanie i modyfikację danych osobowych w bezpieczny sposób, wykorzystując sesję do identyfikacji zalogowanego użytkownika.

**11) Thanks\_Servlet** - jest prostym serwletem, którego głównym zadaniem jest przekierowanie użytkownika na stronę główną po zakończeniu procesu zakupowego. Serwlet jest zarejestrowany pod adresem URL /thanks-servlet za pomocą adnotacji @WebServlet, co umożliwia jego automatyczną obsługę bez potrzeby dodatkowej konfiguracji w pliku web.xml. W metodzie doPost() serwlet uzyskuje dostęp do bieżącej sesji użytkownika przy użyciu request.getSession(), jednak w tej wersji kodu nie korzysta z niej do dalszego przetwarzania danych. Następnie natychmiast wykonuje przekierowanie na stronę index.jsp, co ma na celu powrót użytkownika do strony głównej sklepu – najprawdopodobniej po zakończeniu zamówienia lub potwierdzeniu transakcji. Choć serwlet nie zawiera obecnie logiki czyszczącej lub przetwarzającej dane (np. opróżniania koszyka), może on stanowić punkt końcowy procesu zakupowego i być łatwo rozbudowany o dodatkowe operacje, takie jak rejestrowanie zamówień, wysyłanie potwierdzeń czy usuwanie atrybutów sesyjnych.Dzięki swojej prostocie Thanks\_Servlet pełni rolę przekierowującą i może być wykorzystywany w sytuacjach wymagających zakończenia interakcji użytkownika w sposób uporządkowany i intuicyjny.

**12)** **formularz.jsp** - strona JSP służącą do rejestracji użytkownika. Zawiera formularz HTML wysyłający dane metodą POST do servletu hello-servlet. Formularz umożliwia wpisanie loginu, hasła, imienia i nazwiska, przy czym pola mają zdefiniowane ograniczenia długości (min i max). W przypadku wystąpienia błędu (np. nieprawidłowych danych), komunikat o błędzie jest wyświetlany na czerwono – pobierany jest z atrybutu request o nazwie "error". Strona wykorzystuje CSS do stylizacji, m.in. ustawienia tła, przycisków oraz rozmieszczenia elementów. Formularz ma estetyczny wygląd dzięki użyciu jasnych kolorów i zaokrąglonych krawędzi. Strona jest responsywna i dostosowana do różnych szerokości ekranów przez zastosowanie max-width i flexbox  
  
  
  
**13.)index.jsp -** stanowi stronę główną aplikacji sklepu wędkarskiego i pełni funkcję dynamicznego katalogu produktów. Jest odpowiedzialny zarówno za prezentację oferty sklepu, jak i za podstawową obsługę użytkownika, w tym logowanie, rejestrację oraz dodawanie produktów do koszyka. Na początku strony zaimportowane zostały klasy User i Product z pakietu aplikacyjnego oraz klasa List z biblioteki Javy. Strona jest kodowana w UTF-8 i korzysta z frameworka Bootstrap do podstawowej stylizacji oraz z własnych reguł CSS zdefiniowanych w nagłówku. W głównej części logiki serwera (<% %>) wczytywana jest aktualna sesja użytkownika (User user = session.getAttribute("user")) oraz globalna lista produktów zapisanych w kontekście aplikacji (application.getAttribute("globalProducts")). Wczytywany jest także parametr kategoria, który pozwala filtrować widoczne produkty według kategorii (np. "Wędziska", "Akcesoria"). Jeśli produkt został właśnie dodany do koszyka, a parametr success w URL wynosi 1, to użytkownikowi wyświetlany jest komunikat potwierdzający operację, z nazwą i ilością dodanego produktu. Informacje te są pobierane z sesji (LastAddedProductName, LastAddedQuantity), co umożliwia spersonalizowany komunikat. Dalsza część strony zależy od tego, czy użytkownik jest zalogowany. Jeśli nie, wyświetlane są przyciski „ZALOGUJ SIĘ” oraz „REJESTRACJA”, kierujące odpowiednio do plików logowanie.jsp i formularz.jsp. Jeśli użytkownik jest zalogowany, witany jest imiennie, i ma możliwość przejścia do panelu użytkownika (user-panel) lub wylogowania się (logout). Na stronie znajduje się również link do koszyka (koszyk.jsp), a także menu z kategoriami produktów, pozwalające przefiltrować widoczne pozycje. Kategorie są przekazywane jako parametr kategoria w adresie URL. W sekcji prezentacji produktów tworzona jest lista produktów (<ul>), w której każdy produkt wyświetlany jest w estetycznie stylizowanym bloku (<li>). Dla każdego produktu prezentowana jest jego kategoria, nazwa, cena, opis oraz zdjęcie. Użytkownik ma również możliwość wyboru ilości i dodania produktu do koszyka za pomocą formularza, który przesyła dane do serwletu cart-servlet. Jeśli produkty nie zostały załadowane poprawnie, wyświetlany jest komunikat błędu. W przeciwnym razie użytkownik otrzymuje informację o liczbie załadowanych produktów (uwzględniając ewentualne filtrowanie po kategorii). Dzięki zastosowaniu technologii JSP oraz sesji HTTP strona index.jsp łączy funkcje katalogu produktów, centrum nawigacyjnego oraz punktu dostępu do najważniejszych funkcji sklepu, takich jak logowanie, koszyk czy profil użytkownika. Strona została również przygotowana pod kątem responsywności i prostego stylu graficznego przy użyciu kolorystyki kojarzącej się z naturą (jasna zieleń, żółć).

**14.)koszyk.jsp -** pełni funkcję dynamicznej strony wyświetlającej zawartość koszyka zakupowego użytkownika w aplikacji sklepu wędkarskiego. Został zaprojektowany tak, aby umożliwiać zarówno przegląd dodanych produktów, jak i interakcję z koszykiem – m.in. usuwanie pozycji lub zmniejszanie ich ilości. Strona korzysta z danych zapisanych w sesji użytkownika, w szczególności z obiektu Map<Integer, Cart> przechowywanego pod kluczem "cart". Każdy wpis w tej mapie reprezentuje produkt (zawinięty w obiekt Cart) wraz z informacją o jego ilości i cenie. W górnej części pliku znajduje się logika przetwarzająca żądanie usunięcia określonej liczby sztuk danego produktu. Jeśli w parametrze action przesłanym metodą POST znajdzie się wartość "remove", a dołączone zostaną też identyfikator produktu (productId) i liczba sztuk do usunięcia (removeQuantity), to ilość tego produktu w koszyku zostaje zredukowana lub – jeśli usuwana liczba przekracza bieżącą – całkowicie usunięta z koszyka. Następnie następuje renderowanie zawartości koszyka. Dla każdego produktu wyświetlane są jego nazwa, ilość, cena jednostkowa pomnożona przez ilość (czyli wartość pozycji), a także formularz umożliwiający usunięcie wybranej liczby sztuk. Łączna wartość całego koszyka (suma częściowa wszystkich pozycji) jest obliczana dynamicznie i prezentowana pod listą produktów. Jeśli koszyk jest pusty lub nie istnieje, użytkownik otrzymuje komunikat informujący, że koszyk jest pusty. W przeciwnym razie ma on również możliwość złożenia zamówienia przyciskiem „Zamawiam”, który wysyła formularz POST do strony Strona\_z\_podziękowaniem.jsp, co oznacza zakończenie zakupów. Cała strona jest wizualnie dostosowana do estetyki pozostałych elementów sklepu – zastosowano jasne kolory (tło lightyellow) oraz zieleń jako kolor przewodni przycisków i nagłówków. Interfejs jest prosty, czytelny i dostosowany do obsługi koszyka nawet przez początkujących użytkowników. Strona koszyk.jsp jest centralnym punktem w procesie zakupowym aplikacji – łączy funkcję podsumowania zamówienia z możliwością jego modyfikacji i przejścia do finalizacji.

**15.)logowanie.jsp -** odpowiada za prezentację formularza logowania w aplikacji sklepu wędkarskiego. Strona została zaprojektowana z myślą o prostocie obsługi oraz estetyce dopasowanej do reszty aplikacji. Umożliwia użytkownikowi wprowadzenie loginu oraz hasła, które następnie są przesyłane metodą POST do serwletu login-servlet, odpowiedzialnego za weryfikację danych uwierzytelniających. Formularz składa się z dwóch pól: tekstowego (dla loginu) oraz hasła. Oba pola mają estetyczną, nowoczesną formę – zaokrąglone rogi, jasne kolory oraz spójne rozmieszczenie. Stylizacja strony oparta jest na kolorystyce jasnożółtej i jasnozielonej, z ciemnozielonymi akcentami przycisków, co nadaje stronie naturalny i przyjazny charakter, zgodny z tematyką sklepu. Strona wykorzystuje komponenty flexbox do centralnego wypozycjonowania formularza w pionie i poziomie, dzięki czemu prezentuje się czytelnie zarówno na komputerach, jak i urządzeniach mobilnych. Cały formularz umieszczony jest w kontenerze o maksymalnej szerokości 400px, co zapewnia jego responsywność. W przypadku nieprawidłowego logowania (np. brak danych, zły login lub hasło), serwlet LoginServlet ustawia atrybut error w żądaniu. Jeśli taki komunikat istnieje, zostaje on dynamicznie wyświetlony użytkownikowi na czerwono na górze strony, informując o problemie. Na dole formularza umieszczony został link pozwalający użytkownikowi wrócić do strony głównej (index.jsp). Całość zapewnia przejrzysty i funkcjonalny interfejs logowania, który może być łatwo rozbudowany o dodatkowe funkcje, np. przypomnienie hasła lub rejestrację konta.

**16.)strona\_z\_podziękowaniem.jsp -** pełni funkcję strony końcowej procesu zakupowego w aplikacji sklepu wędkarskiego. Wyświetlany jest po złożeniu zamówienia i zawiera prosty komunikat potwierdzający zakończenie zakupów oraz link umożliwiający powrót do strony głównej (index.jsp).Na początku pliku znajdują się polecenia JSP służące do oczyszczenia sesji użytkownika z danych koszyka. Usuwane są trzy atrybuty sesyjne:

(cart – zawierający mapę produktów dodanych do koszyka,)

(LastAddedProductName – przechowujący nazwę ostatnio dodanego produktu,)

(LastAddedQuantity – przechowujący ilość ostatnio dodanego produktu.)

Dzięki temu po zakończeniu zamówienia koszyk zostaje zresetowany, a dane pomocnicze są usuwane, co zapobiega ich przypadkowemu wyświetlaniu przy kolejnych wizytach na stronie.Część HTML pliku jest bardzo prosta – zawiera podziękowanie za dokonane zakupy oraz przycisk (link) umożliwiający użytkownikowi powrót na stronę główną sklepu. Brakuje w niej dodatkowej stylizacji czy grafiki, jednak dzięki swojej prostocie stanowi czytelne zakończenie procesu zakupowego.Plik ten może być łatwo rozbudowany o dodatkowe funkcjonalności, takie jak generowanie potwierdzenia zamówienia, numeru transakcji, wiadomości e-mail z podziękowaniem czy ponowne załadowanie oferty promocyjnej. W obecnej postaci spełnia jednak swoją rolę jako potwierdzenie udanego zakupu i logiczne zakończenie ścieżki użytkownika.  
  
  
**17.)UserPanel.jsp -** jest odpowiedzialny za prezentację panelu użytkownika w aplikacji sklepu wędkarskiego. Umożliwia on przegląd oraz edycję podstawowych danych konta zalogowanego użytkownika – takich jak imię, nazwisko, hasło, numer telefonu i adres. Na początku strony za pomocą poleceń JSP pobierany jest obiekt użytkownika (User) z atrybutu request, a jeśli nie został tam przekazany – z atrybutu session. Dzięki temu strona może działać zarówno po bezpośrednim przekierowaniu, jak i po odświeżeniu w ramach aktywnej sesji. W przypadku, gdy użytkownik nie jest zalogowany, przetwarzanie strony zostaje przerwane, a w konsoli wypisywany jest komunikat o nieautoryzowanym dostępie. Strona zawiera prosty formularz z wypełnionymi polami zawierającymi dane aktualnie zalogowanego użytkownika. Login jest wyświetlany w trybie tylko do odczytu (readonly), natomiast hasło, imię, nazwisko, numer telefonu oraz adres mogą zostać zmienione przez użytkownika. Po zatwierdzeniu zmian dane te są przesyłane metodą POST – dalsze ich przetwarzanie odbywa się po stronie serwletu, do którego formularz jest domyślnie skierowany (brak określonego action). Jeśli wcześniej wystąpił jakiś błąd (np. walidacja danych), komunikat zostaje przekazany do strony jako atrybut error i wyświetlony użytkownikowi w kolorze czerwonym nad formularzem. Na dole strony umieszczony jest komunikat oraz link umożliwiający powrót do strony głównej (index.jsp). Styl strony jest bardzo uproszczony – brak tu zaawansowanej estetyki, jednak formularz jest funkcjonalny i zrozumiały dla użytkownika. Plik UserPanel.jsp stanowi kluczowy element systemu zarządzania kontem i może być łatwo rozbudowany o dodatkowe funkcje, takie jak potwierdzanie zmian, walidacja po stronie klienta (JavaScript), czy możliwość zmiany zdjęcia profilowego.