



*Wrocławska Wyższa Szkoła
Informatyki Stosowanej*

Wydział Informatyki

Dominika Maria Rzepka

Nr albumu: 6150

*Oprogramowanie
dla sklepu internetowego
z biżuterią*

Praca: inżynierska

Kierunek: Informatyka

Specjalność: Aplikacje biznesowe Java EE

Praca wykonana pod kierunkiem:
dr inż. Łukasz Jeleń

Wrocław 2022

Spis treści

Wstęp	2
1. Cel i założenia.....	4
2. Metodologia	6
2.1. Wykorzystane technologie	6
2.2. Narzędzia.....	19
2.3. Przegląd istniejących rozwiązań	21
3. Opis projektu.....	25
3.1. Wymagania niefunkcjonalne.....	25
3.2. Wymagania funkcjonalne.....	26
4. Opis implementacji	28
4.1. Baza danych	28
4.2. Struktura projektu.....	34
4.3. Operacje na bazie danych.....	37
4.4. Strona główna.....	39
4.5. Interfejs użytkownika niezalogowanego	41
4.6. Interfejs użytkownika zalogowanego.....	47
4.7. Interfejs użytkownika z rangą	51
5. Testy aplikacji.....	63
5.1. Testy wymagań niefunkcjonalnych.....	63
5.2. Testy wymagań funkcjonalnych	66
5.3. Testy walidacji	71
5.4. Testy responsywności	72
6. Analiza i wnioski	73
6.1. Osiągnięte cele	73
6.2. Możliwości rozwoju aplikacji	73
7. Podsumowanie	74
Bibliografia	75
Spis rysункów	76
Spis tabel.....	77
Załączniki.....	77

Skróty i symbole

Tabela 1 – symbole zastosowane w pracy

Symbol	Objaśnienie
HTTP	(ang. Hypertext Transfer Protocol) Protokół służący do wymiany informacji w aplikacji webowej
IDE	Środowisko programistyczne
Java EE	(ang. Java for Enterprise Edition)
JDK	(ang. Java Development Kit) Pakiet programisty stworzony dla języka Java
JRE	(ang. Java Runtime Environment) Środowisko uruchomieniowe dla języka Java
JVM	(ang. Java Virtual Machine) Maszyna wirtualna Javy
SQL	(ang. Structured Query Language) Strukturalny język zapytań

Tabela 2 – skróty używane w rozdziale testowym

Skrót	Objaśnienie
prz.	Przekierowanie
pw.	Powrót na poprzednią stronę
swl.	Przekierowanie do odpowiedniego serwletu
zmn.	Modyfikacja danych w bazie

Legenda oznaczeń:

Tabela 3 – legenda oznaczeń czcionki

Zapis	Objaśnienie
<code>'id_ord'</code>	Atrybut występujący w tabeli bazy danych
action	Odwółanie do tabeli w bazie danych o podanej nazwie
ControlSrv.java	Odniesienie do nazwy pliku
java	Przytoczenie nazwy katalogu
<code>try{} catch(){}</code>	Przywołanie fragmentu kodu

Wstęp

Człowiek od niepamiętnych czasów ozdabiał swoje ciało na różne sposoby. Zaczynając od farb i naturalnych komponentów, a kończąc na różnego rodzaju biżuterii, która pojawiła się z czasem rozwoju rzemieślnictwa. „Indyjscy archeolodzy twierdzą, na podstawie wykopisk, że w Indiach i Birmie kobiety i mężczyźni ozdabiali się szlachetnymi kamieniami już przed 10 000 lat.”¹ Jednak nie tylko kamienie wartościowe oraz metale szlachetne można nazwać biżuterią, z czym niestety nie zgadza się większość słowników. Gdyby jednak traktować tę definicję jednoznacznie „brązowego naszyjnika nie moglibyśmy wtedy nazywać biżuterią, a przecież dla jego posiadaczki stanowił prawdziwy skarb. Dla ułatwienia nazwiemy biżuterią również inne ozdoby – z kości, korala, pereł, muszli, pazurów, bursztynu czy ptasich piór.”² Trzymając się wyłącznie definicji, ograniczamy punkt widzenia, gdyż nie tylko z metali można wykonać coś wspaniałego. Wyroby biżuteryjne stworzone za pomocą metody makramy³ mogą być równie piękne, a jednocześnie o wiele tańsze. Może też dlatego dzisiejsze media społecznościowe wykreowały trend tworzenia własnej biżuterii. Wiąże się to również z modą na rzeczy rękodzielnicze, które według wielu są cenniejsze niż wyroby fabryczne.

Niezwykle ważne jest również wspomnienie o ogólnie dostępnej informacji, jaką jest przesycony obecny świat. Wiele źródeł oferuje możliwość poznania każdego tematu. Poprzez wynalazek, jakim jest Internet wiedza stała się jeszcze bardziej powszechna niż kiedykolwiek wcześniej. Można powiedzieć, że teraz wszystko dzieje się w sieci globalnej. Ten świat wirtualny przyczynił się do powstania m.in. wyżej wspomnianych portali społecznościowych, gazet internetowych, kursów online oraz jakże ważnych w tej pracy dyplomowej sklepów internetowych, dzięki którym nie trzeba być na miejscu, aby dokonać zakupu danego produktu.

Praca dyplomowa została podzielona na siedem rozdziałów.

Pierwszy opisuje „cel i założenia” całego projektu. Przytaczana jest w nim inspiracja. Dodatkowo omówiony jest proces tworzenia biżuterii dla lepszego zrozumienia problemu. „Metodologia” została podzielona na trzy podrozdziały. W pierwszym opisane są technologie użyte przy tworzeniu aplikacji. Drugi przybliża narzędzia jakimi

¹ Kurlus T., *Kamienie szlachetne. Wszystko o...*, wyd. 1, Krajowa Agencja Wydawnicza 1976, s. 20

² Wdowiak L., *Zarys historii ozdabiania ciała*, wyd. 1, Wydawnictwo Pomorskiego Uniwersytetu Medycznego w Szczecinie 2017, s. 215

³ Technika wiązania sznurków bez użycia narzędzi typu druty, szydełka, igły.

się posługiwano przy pisaniu. Ostatni demonstruje cztery podobne rozwiązania istniejące na rynku.

„Opis projektu” pokazuje wymagania funkcjonalne oraz niefunkcjonalne, które dotyczą aplikacji inżynierskiej przedstawionej w tej pracy.

„Opis implementacji” służy do pokazania projektu inżynierskiego stworzonego na potrzeby pracy dyplomowej. Posiada siedem podrozdziałów. Pierwszym z nich jest opis bazy danych, następnie zostaje przytoczona struktura projektu oraz podstawowe operacje na bazie danych z poziomu aplikacji. Po tych informacjach następuje przejście do interfejsów zaczynających się od strony głównej.

Ostatnim rozdziałem opisującym projekt są „testy aplikacji”. Dowodzą one zgodności z wymaganiami przedstawionymi w rozdziale trzecim.

„Analiza i wnioski”, jak sama nazwa mówi, jest rozdziałem rozważającym wynik ostateczny, jaki udało się osiągnąć. Znajdują się tutaj również możliwości rozwoju, które ulepszyłyby projekt w przypadku wdrożenia.

„Podsumowanie” jest ostatnim rozdziałem kończącym pracę.

1. Cel i założenia

Celem pracy jest rozwiązywanie problemu inżynierskiego, który polega na stworzeniu aplikacji zgodnej z tytułem: *Oprogramowanie dla sklepu internetowego z biżuterią*. Podstawowym czynnikiem motywującym niniejszego projektu jest zatem stworzenie witryny, która byłaby unowocześnioną wersją istniejących na rynku rozwiązań zarówno dla użytkownika sklepu jak i pracowników.

Przed określeniem problemu oraz założeń należy zapoznać się z teorią dotyczącą wyrobu biżuterii. Ten opis ma również za zadanie podkreślić jej znaczenie.

Najważniejszym w tej kwestii elementem jest pomysł. Od niego rodzą się następne kroki, które twórca wykonuje. Sama idea nie jest taka prosta, jak się wydaje. Zdarza się tak, że uderza ona znienacka, kiedy osoba projektująca, się tego nie spodziewa. Bardzo często projekt zmienia się podczas szkiców wstępnych, wykonywanych w następnym kroku. Zmieniają się materiały, a także sposób wykonania danego elementu, dlatego najczęściej projektant trzyma się pewnego motywu. Tej pierwotnej myśli, która kiełkuje stając się w efekcie końcowym arcydziełem.

Praca inżynierska skupia się na osobach nieposiadających odpowiednich maszyn oraz narzędzi do wyrobów jubilerskich, a tworzących wszystko z pasją dla samej sztuki. Tacy artyści są zmuszeni do kupna półfabrykatów, czyli gotowych części do wyrobu biżuterii. Złotnik, zamiast kupować srebrny drut na metry, stopiłby resztki pozostałego srebra i za pomocą odpowiednich przyrządów uzyskałby daną długość oraz średnicę, jaką potrzebuje do projektu.

Przytoczony już drut można połączyć z techniką metaloplastyki (ang. wire-wrapping). Jest to metoda owijania go w taki sposób, aby stworzyć biżuterię. Takich metod jest wiele. Wspomniana we wstępie makrama wykorzystuje sznurki, zazwyczaj nylonowe. Technika sutasu jest natomiast rodzajem haftu, gdzie poszczególne pasma zszywa się odpowiednią nicią. Jeszcze inną metodą jest koralikowanie (ang. beading), w której za pomocą cienkiej nitki nawleka się odpowiednie wzory. Można użyć do tego krosna, szydełka lub zwykłej igły. Zdarza się również, że dane techniki się łączy, co tworzy bardzo unikalne dzieło sztuki. Niestety jest ono bardziej kosztowne niż sam twórca i kupujący by tego chciał.

Problemem, a zarazem inspiracją, jest dostawa materiałów do wyrobu biżuterii. Wiele sklepów oferuje tylko część dostępnych środków, których zazwyczaj brakuje. Nierzadko trzeba zamawiać po jednej rzeczy w kilku sklepach, aby skompletować wszystkie materiały niezbędne do autorskiego projektu. Rodzi to większe koszty, niż jest

to potrzebne, a wartość samego przedmiotu staje się większa, co zniechęcając tym samym przyszłych klientów.

Chcąc połączyć style, bardzo często taka osoba jest zmuszona do korzystania z usług kilku odrębnych dostawców. Nawet przy jednej technice bywają problemy, ponieważ np. dany sklep nie oferuje takiej średnicy sznurka, która jest niezbędna do projektu makramy, ale posiada odpowiednie koraliki, które będą wplecone pomiędzy węzły (rys. 1.1).



Rysunek 1.1. Bransoletka wykonana metodą makramy [wykonanie własne]

Idąc za celem, założeniem tej pracy jest opracowanie aplikacji webowej, która będzie wykorzystywała bazę danych do przechowywania określonych produktów. Całość ma zostać zaimplementowana w formie sklepu internetowego możliwie najbardziej zbliżonego do rzeczywistych rozwiązań.

Aplikacja jest kierowana do osób kreatywnych chcących tworzyć własną biżuterię oraz dla sklepów prowadzących taką działalność. Powinny się w niej znajdować wszelkiego rodzaju półfabrykaty zapewniające wyrób z zastosowaniem różnych technik, również tych, opisanych wyżej.

Jedną z najważniejszych zalet aplikacji będzie jej zakres oraz prawidłowe działanie. Ogólnym pomysłem jest stworzenie witryny, która będzie oparta o rangi, dzięki czemu pracownicy będą mieli inny widok niż zwykły użytkownik sklepu. Zakupy mogą być przeprowadzane zarówno przez osoby chcące tylko raz dokonać zakupu bez przymusu logowania oraz stałych, zalogowanych klientów. Rangi tyczyć się będą pracowników sklepu, którzy zostaną podzieleni na cztery grupy: administratorzy, korektorzy, zaopatrzeniowcy oraz zwykli pracownicy. Każdy z nich zostanie dokładnie rozpisany w rozdziale trzecim.

2. Metodologia

Metodologia, według Wielkiego słownika języka polskiego, to: „nauka badająca wartości poznawcze metod stosowanych w różnych dyscyplinach wiedzy”⁴. W tym przypadku podstawą w tworzeniu projektu inżynierskiego jest przegląd jak i późniejszy wybór dostępnych technologii oraz narzędzi. Niniejszy rozdział jest takim właśnie wprowadzeniem. Ponadto zaprezentowane zostaną istniejące rozwiązania, które przybliżą zrozumienie projektu oraz technologie, jakie posłużyły do stworzenia aplikacji.

2.1. Wykorzystane technologie

Ze względu na wszechobecną digitalizację pojawia się mnóstwo rozwiązań dla jednego, danego zadania. Powstają innowacyjne języki programowania oraz nowe wersje starszych koncepcji. Zaczęło to stwarzać problemy z wyborem formy pisania aplikacji, przez którą po dziś dzień są rozbieżności oraz kłopoty z aktualnianiem lub tworzeniem programów w najnowszych językach, co rodzi za sobą niemałe koszty.

Niniejszy podrozdział jest jednym z najważniejszych aspektów pisania pracy inżynierskiej. Całokształt tej wiedzy pozwala na zrozumienie mechaniki działania konkretnych części projektu, dlatego ważne jest, aby osoba czytająca tę pracę wiedziała, na czym polega funkcjonowanie danych modułów, w tym przypadku poszczególnych języków, struktur, pluginów oraz ogólnie pojętej technologii. Ten opis przybliży powstanie oraz działanie komponentów aplikacji.

Aplikacja internetowa

„Wraz z rozwojem usług WWW pojawiła się potrzeba wprowadzania coraz większej ilości danych od użytkownika. Na ich podstawie strony WWW miały być personalizowane. Zaczęło się od prostych formularzy, takich jak np. te umożliwiające podanie imienia odwiedzającego. Strona zaś wyświetlała powitanie w rodzaju „Witaj, imię”. Tak narodziły się aplikacje internetowe, których część przetwarzania informacji odbywała się na serwerze.”⁵ W ciągu ostatnich lat rynek programów webowych znacznie się powiększył, dając możliwość rozwoju tej dziedziny. Dzięki swojej ogólnodostępności są one coraz bardziej powszechnie. Wystarczy mieć tylko urządzenie z dostępem do Internetu, aby skorzystać z usług np. sklepu internetowego, o którym mowa w tej pracy.

⁴ <https://wsjp.pl/haslo/podglad/32253/metodologia/4058094/nauka> (dostęp: 20.01.2022 r.)

⁵ Ross J., PHP i HTML. Tworzenie dynamicznych stron WWW, Helion 2010, s. 34

Podsumowując: aplikacją internetową nazywamy program Internetowy, który wysyła informacje o podjęciu wyboru przez użytkownika po stronie wizualnej, do serwera danej aplikacji. Za pomocą odpowiednio zaprogramowanych komend jest on w stanie wyświetlić żądanego komunikatu. Całe działanie jest ukryte po stronie serwerowej, do której użytkownik nie ma dostępu.

MVC

W tym miejscu warto wspomnieć o wzorcu projektowym MVC (ang. Model, View, Controller). Jest to skrót, który możemy przetłumaczyć jako Model, Widok, Kontroler. Jest to inna nazwa architektury trójwarstwowej aplikacji internetowej.

Model – obsługuje dane zarówno wejściowe jak i wyjściowe aplikacji. Jest zatem równoważny z warstwą danych w konwencji trójwarstwowej. Przeważnie jest to wcześniej stworzona baza danych oparta na języku SQL, bądź bardziej nowoczesnych rozwiązań typu NoSQL, która używa JavaScript z danymi zapisywanyymi w formacie JSON.

Widok – jak sama nazwa mówi, zarządza warstwą wyświetlania aplikacji, jest jej wizerunkiem, opisywanym jako warstwa prezentacji. Najczęściej tworzony za pomocą kodu HTML połączonym z CSS dla lepszych doznań wizualnych.

Kontroler – odpowiada za logikę całego systemu. „To zestaw klas przetwarzający dane, wykonujący na nich pewne operacje, przetwarzający żądania użytkownika czy też implementującą pewne zależności, warunki bądź dbający o spójność działania całej aplikacji.”⁶ Zatem wszystkie funkcjonalności kryją się właśnie tutaj. Tylko od danego problemu oraz programisty zależy, jak ta warstwa będzie wyglądać, oraz jakie komponenty się w niej znajdą.

„Celem wzorca MVC jest oddzielenie logiki prezentacyjnej od biznesowej, (...) chodzi o utrzymanie wyraźnej granicy między obiektami dziedzinowymi modelującymi problem i prezentacją tej logiki.”⁷ Większość wspomnianych wyżej technologii zostanie opisana w dalszej części tego rozdziału.

⁶ Ross J., PHP i HTML. Tworzenie dynamicznych stron WWW, Helion 2010, s. 38

⁷ Yener M., Theodom A., Java EE. Zaawansowane wzorce projektowe, tłum. Ł. Piwko, Helion 2015, s. 209

SQL

Baza danych oparta na języku SQL (ang. Structured Query Language) jest pamięcią całej aplikacji. W niej zapisywane są wszystkie stałe informacje potrzebne do poprawnego funkcjonowania całego programu. Jest to pewien fragment rzeczywistości, na jakiej opiera się działanie.

Baza danych składa się z tabel, inaczej zwanych encjami. W nich znajdują się kolumny (atrybuty) oraz wiersze (krotki). „Każdy element danych, lub wartość, może być określony jako wynik przecięcia wiersza (elementu poziomego) i kolumny (elementu pionowego).”⁸

Język SQL umożliwia tworzenie zapytań do bazy danych. Jest ich wiele, jednak w tej części zostanie przytoczonych 5 podstawowych:

- CREATE – tworzenie np. tabeli (dopisek TABLE);
- INSERT INTO – wypełnianie encji;
- SELECT – wybór danych z bazy;
- UPDATE – edytowanie elementu;
- DELETE – usuwanie wartości.

Za ich pomocą można zarządzać zawartością. Działa to w ten sam sposób jak w rzeczywistości. Przykładowo zakładając, że bazą danych jest pudło z poszczególnymi mniejszymi pudełkami w środku, w których to znajdują się przedziały. Każdy z poszczególnych pudełek tworzy encje (tabele). Przedziały to nasze kolumny oraz wiersze. Wkładając przedmiot w dane miejsce, zapisujemy go w bazie; wyjmując, usuwamy; poprzez sprawdzenie przedmiotu, wybieramy; a przez kształtowanie na nowo elementu, edytujemy.

Na podstawie modelu relacyjnego, który „został nazwany w ten sposób, (...) dlatego, że poszczególne wartości w kolumnach tabeli występują w ścisłej relacji”⁹, można wykreować wycinek rzeczywistości. Dzięki temu łatwo odnosić do danych zapisanych w bazie. Klucz główny, będący najczęściej indeksem odpowiedniej krotki, może być zastosowany jako klucz obcy w innej tabeli. Taki zabieg umożliwia łatwe przemieszczanie się pomiędzy encjami oraz zapobiega redundancji, czyli nadmiarowości danych, innymi słowy powtarzaniu się ich w bazie.

⁸ Bowman J. S., Darnovsky M., Emerson S. L., *Podręcznik języka SQL*, Wydawnictwa Naukowo Techniczne 2001, s. 3

⁹ Faroult S., Robson P., *SQL. Sztuka programowania*, Helion 2007, s. 17

Tabela 4 -relacje w bazie danych

Symbol	Wyjaśnienie relacji
	Jeden do jednego – każdemu rekordowi z jednej tabeli jest przypisany dokładnie jeden z drugiej.
	Jeden do wielu (stosowane zamiennie) – każdemu rekordowi z pierwszej tabeli odpowiada wiele rekordów z drugiej.
	Wiele do wielu – wielu rekordom z jednej tabeli odpowiada wiele rekordów z drugiej tabeli (tworzy się wtedy tabelę łącznikową pomiędzy).
	Jeden do zera lub jednego – tak jak w pierwszej tylko tam, gdzie znajduje się zero, nie jest wymagane.

W celu efektywnego działania bazę należy poddać normalizacji. „Pierwsze trzy postacie normalne są oparte na czystej logice i dzięki temu doskonale sprawdzają się jako kontrolne listy reguł ułatwiające proces wyciągania danych z chaosu.”¹⁰ Najpierw trzeba się upewnić czy atrybuty mają charakter atomowy. Oznacza to, że muszą zawierać dane, dzięki którym możemy spokojnie przeprowadzić wyszukiwania bez zadawania skomplikowanych zapytań do bazy. Przykładowo: kolumnę adres, która zawiera wszystkie dane adresu (ulicę, numer mieszkania, kod pocztowy, miejscowości), można rozbić na poszczególne atrybuty, po których łatwo się odnieść, szukając tylko jednej wartości, np. ulicy. Następnie należy zoptymalizować bazę danych. Pod tym słowem kryje się wiele operacji jak np. stworzenie tabel słownikowych, w których powtarzające się wartości będą przechowywane. Zapobiegamy tym samym redundancji. Taka encja może się połączyć za pomocą relacyjności z drugą tabelą.

W rozdziale czwartym opisana zostanie baza danych stworzona na potrzeby projektu.

Java oraz Java EE

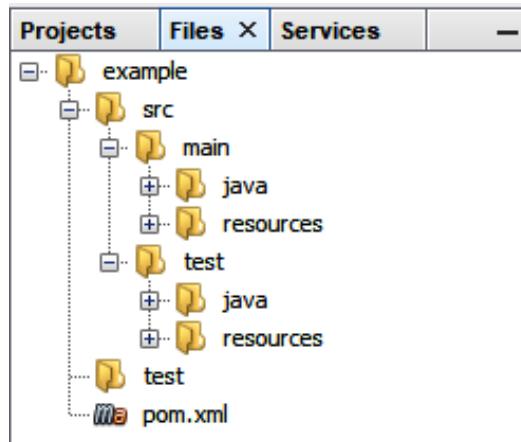
Java EE (ang. Java for Enterprise Edition) jest wersją języka Java, opartego na klasach, dla rozwiązań biznesowych. Posiada wiele wbudowanych metod, które pomagają w implementacji algorytmów, np. opisany wyżej wzorzec projektowy MVC.

Aby móc korzystać z Javy trzeba posiadać specjalny pakiet JDK (ang. Java Development Kit), który jest zestawem zaprojektowanym dla programistów. W jego skład wchodzi pakiet klas samego języka, środowisko uruchomieniowe (JRE, ang. Java Runtime Environment) oraz maszyna wirtualna (JVM, ang. Java Virtual Machine), która wykonuje skompilowany kod.

¹⁰ Faroult S., Robson P., SQL. Sztuka programowania, Helion 2007, s. 20

Maven

Najczęściej do tworzenia aplikacji internetowych w Javie EE stosuje się projekty z obsługą Mavena. Jest to darmowe narzędzie ułatwiające pracę z kodem. Poza automatyzacją niektórych procesów tworzy ono podstawową strukturę katalogową, do której należy się zastosować, aby program mógł się zbudować, a następnie uruchomić.



Rysunek 2.2. Struktura katalogowa Maven

Powyższy rysunek przedstawia drzewo projektu. Jest ono oparte na folderach *src* oraz *target*, ponadto występuje plik *pom.xml*. (ang. Project Object Model). W nim znajdują się wszystkie podstawowe ustawienia jak i dodatkowe zależności oraz pluginy.

Katalog *src* (ang. source, tłum. źródło) jest folderem źródłowym i zawiera podkatalogi *main* (tłum. główne) oraz *test*. W obu znajdują się podfoldery *java*, odpowiadająca za klasy języka i *resources* (tłum. zasoby), gdzie znajdują się źródła widoku użytkownika oraz dokumenty konfiguracyjne dla klas. Katalog *main* odpowiada za źródło projektu, natomiast *test* jak sama nazwa wskazuje służy do testów programu. W podkatalogu *target* Maven zamieszcza skompilowany i przerobiony kod.

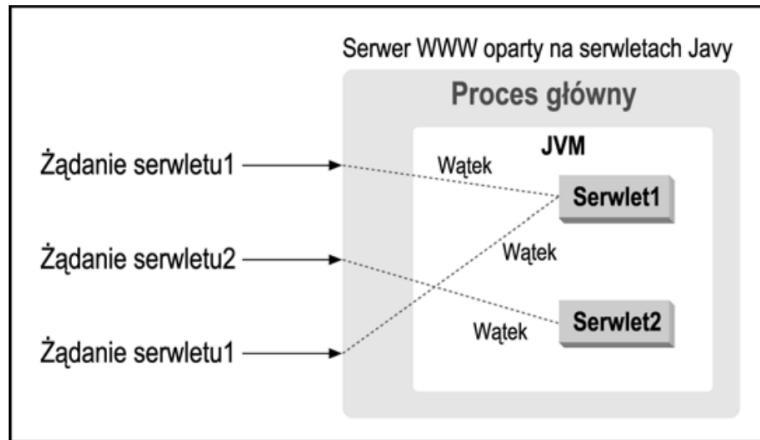
W zależności od tworzonego projektu nazwy poszczególnych katalogów mogą się różnić lecz struktura pozostaje taka sama. Zdarza się również, że nieużywane foldery są tworzone dopiero po wykonaniu odpowiednich kroków.

Serwlety

Serwlet „jest ogólnym rozszerzeniem serwera – klasą Javy, która może być dynamicznie ładowana w celu rozszerzenia funkcjonalności serwera.”¹¹ Działa wewnętrz JVM wyłącznie po stronie serwera aplikacji (zostanie omówiony później). Oznacza to, że jest on swoistym mózgiem, którego kod pozostaje niewidoczny od strony zwykłego użytkownika. Jest jednym z podstawowych, ale starszych elementów Javy EE.

¹¹ Hunter J., Crawford W., *Java Servlet. Programowanie*, wyd. 2, Helion 2002, s. 15

Najczęściej używa się ich do witryn internetowych ze względu na powiązany z nimi model żądanie-odpowiedź (ang. request-response). Za jego pomocą serwlet może obsłużyć każde żądanie użytkownika poprzez wysłanie odpowiedzi na konkretne pytanie. Przykładowe działanie pokazuje poniższy rysunek.



Rysunek 2.3. Okres trwałości serwletu. [7]

Serwlety zapewniają łączność z bazą danych (plugin JDBC), przetwarzanie żądań użytkownika poprzez komunikację klient-serwer, wielowątkowość, kompresję danych i wiele innych. Mogą one „bezpiecznie obsługiwać błędy, dzięki mechanizmowi obsługi wyjątków lub kontrolerowi dostępu Javy.”¹². Zatem „Serwlety kierują sieciowym przepływem sterowania i obsługują interakcje, podczas gdy strony JSP (ang. JavaServer Pages), (...) i biblioteka JSTL (ang. JavaServer Pages Standard Tag Library) przygotowują odpowiedzi dla klienta.”¹³

JSP

Technologia JSP (ang. JavaServer Pages) pozwala na przeplatanie kodu statycznego z kodem dynamicznym zazwyczaj generowanym przez serwlety opisane wyżej. Zwykle współpracują z technologią HTML oraz XML.

Istnieje możliwość umieszczania „fragmentów kodu serwletu (tak zwanych snippetów, ang. snippets) bezpośrednio w dokumencie tekstowym.”¹⁴ Przykładem są:

- Komentarze – zwykle są to wtrącenia autora do opisu kodu:

```
<%-- Przykładowy komentarz --%>
```
- Deklaracje – fragment kodu, który powinien dotyczyć tylko tego pliku:

```
<%! float number = 5.14; %>
```

¹² Hunter J., Crawford W., *Java Servlet. Programowanie*, wyd. 2, Helion 2002, s.19

¹³ Yener M., Theedom A., *Java EE. Zaawansowane wzorce projektowe*, tłum. Ł. Piwko, Helion 2015, s. 42

¹⁴ Jendrock E., Cervera-Navarro R., Evans I., Gollapudi D., Haase K., Markito W., Srivaths C., *Java EE 6. Zaawansowany przewodnik*, tłum. R. Jońca, wyd 4, Helion 2013, s. 42

- Skryptlety – bezpośrednie umieszczenie kodu:

```
<% String str = „Hello”; System.out.println( str ); %>
```

- Wyrażenia – dołączenie fragmentu kodu (konwertuje do String) i wypisanie go:

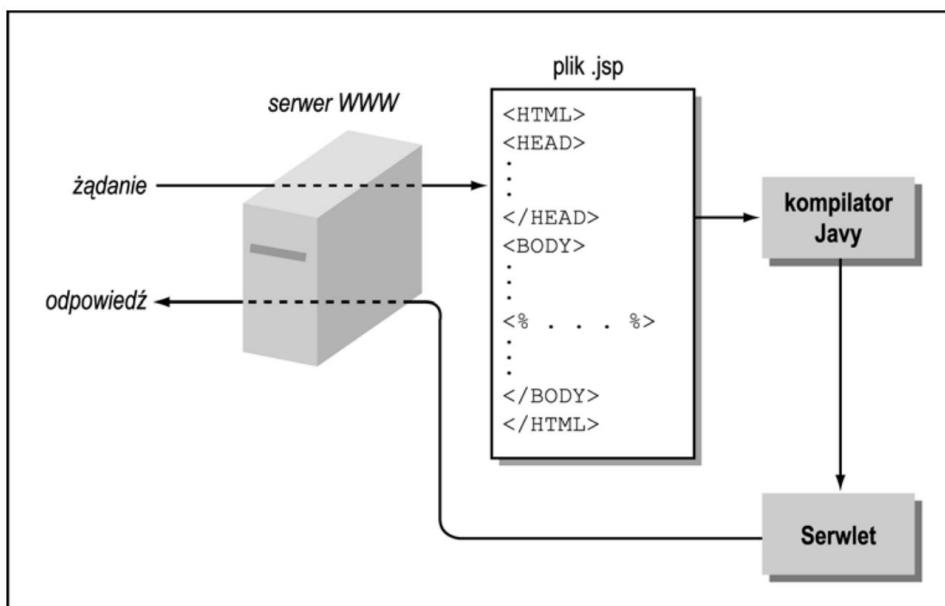
```
<%= new java.util.Date() %>
```

- Instrukcje – sterują zawartością i ustawieniem podstawowym strony:

```
<%@ page contentType="text/plain" %>
```

W tej technologii występują również tagi. Przykładem takiego rozwiązania może być dołączenie innej części kodu do struktury pierwotnej strony napisanej w technologii JSP: `<jsp:include page="other.jsp"/>`.

„Poza obszarem widoczności serwer automatycznie tworzy, kompiluje, ładuje i uruchamia specjalny serwlet tworzący zawartość strony (...). Można myśleć o tym specjalnym serwlecie jak o serwacie roboczym, działającym w tle.”¹⁵ Pierwsze wywołanie niestety zajmie więcej czasu ze względu na brak danych w pamięci podręcznej przeglądarki oraz czas generowania strony JSP. Dzieje się tak, gdyż wszystko musi być skompilowane w czasie rzeczywistym i przesłane bezpośrednio do użytkownika aplikacji. Przy następnych wyświetleniach okres ładowania będzie o wiele krótszy. To działanie wyjaśnia poniższy rysunek.



Rysunek 2.4. Generowanie stron JavaServer Pages. [7]

¹⁵ Hunter J., Crawford W., *Java Servlet. Programowanie*, wyd. 2, Helion 2002, s. 442

JSTL

„Standardowa biblioteka znaczników JavaServer Pages (ang. JavaServer Pages Standard Tag Library, w skrócie JSTL) zawiera podstawową funkcjonalność wspólną dla wielu aplikacji JSP.”¹⁶ Za pomocą prefixów, które używa ta technologia, otrzymamy te same możliwości, co przy pisaniu kodu Javy. Jest to szybki skrót, który wiąże się z optymalnością kodu.

Tagami używanymi w JSTL są:

- Rdzeń – podstawowe funkcjonalności:

```
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>
```

Przykład: za ich pomocą można deklarować zmienne (`<c:set>`), stosować warunki (`<c:if>`), czy też przenieść użytkownika na inną stronę (`<c:redirect>`).

- Formatowanie – wiąże się z przekształcaniem:

```
<%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>
```

Przykład: zarówno liczby (`<fmt:formatNumber>`) jak i daty (`<fmt:formatDate>`) mogą być poddane formatowaniu, jednak można również określić kodowanie znaków (`<fmt:requestEncoding>`).

- SQL – pozwala na zamieszczanie kodu SQL:

```
<%@ taglib prefix = "sql" uri = "http://java.sun.com/jsp/jstl/sql" %>
```

Przykład: podstawowe funkcjonalności jak: zapytanie (`<sql:query>`), aktualizacja (`<sql:update>`) oraz ustawienie parametru (`<sql:param>`).

- Funkcje – są wykonywane na tekście (typ String), dołączanym jako metoda:

```
<%@ taglib prefix = "fn"
```

```
    uri = "http://java.sun.com/jsp/jstl/functions" %>
```

Przykład: sprawdza, czy dany ciąg znaków zaczyna się od jakiejś np. litery (`fn:startsWith()`), dzielenie tekstu (`fn:split()`) oraz łączenie go (`fn:join()`).

Trzeba pamiętać, że każdy tag posiada swoje unikalne atrybuty. Nie każde z nich mogą być pominięte. Np. atrybuty `<c:if>` to: test (warunek), var (nazwa zmiennej, do której możemy się odnieść) oraz scope (zakres zmiennej warunku). Tylko pierwszy atrybut jest wymagany. Resztę można opuścić.

¹⁶ Jendrock E., Cervera-Navarro R., Evans I., Gollapudi D., Haase K., Markito W., Srivaths C., Java EE 6. Zaawansowany przewodnik, tłum. R. Jońca, wyd 4, Helion 2013, s. 42

Serwer aplikacji

Jak już zostało wspomniane Java potrzebuje serwera, aby móc działać poprawnie. Samo słowo serwer może mieć wiele znaczeń potocznych. Najczęściej mówi się o urządzeniach ze specjalistycznym oprogramowaniem służącym do zarządzania siecią wewnętrzną, bądź zewnętrzną. „Serwery WWW to specjalne komputery, które są bezustannie podłączone do internetu i zostały zoptymalizowane pod kątem wysyłania stron WWW do wszystkich, którzy o nie proszą.”¹⁷ Udostępniają one przestarzeń dyskową na potrzeby działania danej aplikacji webowej.

Najprościej można powiedzieć, że jest to miejsce, gdzie jest przechowywana oraz wykonuje się dana aplikacja. Przetworzony kod w następnej kolejności jest wyświetlany po stronie użytkownika w przeglądarce internetowej, która interpretuje kod frontendowy¹⁸ na wizualnie zrozumiałą dla zwykłego człowieka. „Do ogólnych cech serwera można zaliczyć:

- pasywność – serwer czeka na zainicjowanie połączenia z klientem,
- czekanie na żądania klientów – jest tylko realizatorem przysłanych zleceń,
- automatyczność – w momencie otrzymania żądania serwer przetwarza je i wysyła odpowiedź.”¹⁹

Równie dobrze można zainstalować takie oprogramowanie na komputerze. Przykładami są: WebLogic, Glassfish, JBoss/Wildfly, WebSphere oraz Tomcat. Ostatni z przytoczonych jest powszechnie stosowany przez frameworki obsługujące Java takie jak np. Spring Boot. Jest on o wiele lżejszy, lecz nie zawiera pełnych rozwiązań, które niejednokrotnie przydają się przy pracy.

Protokół HTTP

HTTP jest międzynarodowym protokołem stosowanym przy komunikacji klient-serwer w aplikacjach webowych. „Kiedy klient składa zlecenie, pierwszą rzeczą, którą wykonuje, jest komenda HTTP zwana metodą, za pomocą której serwer orientuje się jaki rodzaj zlecenia jest składany. Pierwszy wiersz zlecenia określa adres dokumentu (URL) oraz używaną wersję protokołu HTTP.”²⁰

¹⁷ Duckett J., HTML i CSS. Zaprojektuj i zbuduj witrynę WWW. Podręcznik Front End Developera, Helion 2014, s. 6

¹⁸ część kliencka aplikacji, którą widzi użytkownik

¹⁹ Nowicki T., Wrzosek E., Modelowanie, symulacja i analiza systemów klasy klient-serwer, „Symulacja w Badaniach i Rozwoju”, 2010, Vol. 1, nr 3, s. 267

²⁰ Hunter J., Crawford W., Java Servlet. Programowanie, wyd. 2, Helion 2002, s. 22

Dwie z podstawowych metod, które są fabrycznie generowane przy serwletach Javy to GET oraz POST. Pierwsza z nich jest przeznaczona do uzyskania informacji, przesyłanych za pośrednictwem ścieżki URL, co oznacza, że jest w pełni widoczna. Przykładem jest: `http://aplikacja.pl?i=9&p=2`. Znak ? rozpoczyna ciąg danych, a & zwiastuje następną zmienną. Występują więc dwie zmienne liczbowe i oraz p, których wartości to kolejno 9 i 2. Metoda POST natomiast jest wykorzystywana do przesyłania informacji. Zazwyczaj są to dane formularzy, które nie są widoczne po jego przesłaniu do serwera aplikacji. Oczywiście istnieją inne metody HTTP takie jak DELETE, TRACE, PUT czy też OPTIONS, jednak nie zostały dołączone do stałych metod klas serwletowych, przez co stosuje się je inaczej.

HTML

Dokładna nazwa HTML (ang. HyperText Markup Language) to hipertekstowy język znaczników. Za ich pomocą tworzy się podstawę strony internetowej. Język ten określa się jako szkielet całej części wizualnej.

Każdy znacznik posiada swoje atrybuty, dzięki którym można modyfikować efekt wizualny. „Atrybuty dostarczają dodatkowych informacji o zawartości elementów. Są umieszczane w znaczniku otwierającym elementu i składają się z dwóch części: nazwy oraz wartości, oddzielonych od siebie znakiem równości.”²¹ Przykładem może być hiperłącze `Strona główna`, gdzie znacznik to a, natomiast atrybutem jest `href="index.html"`. Oczywiście trzeba pamiętać, że każdy znacznik, po jego otwarciu (np. `<a>`), trzeba zamknąć (``), dodając ukośnik przed znakiem znacznika. W przeciwnym wypadku strona nie będzie funkcjonować poprawnie.

Podstawą w budowaniu witryny jest deklaracja do wersji kodu: `<!DOCTYPE HTML>` po tym następuje rozpoczęcie strony: `<html>`, którą trzeba zakończyć (`</html>`). Kod posiada dwie podstawowe i nierożłączne części: głowę (`<head>`), w której zamieszcza się ustawienia strony, np. atrybut kodowania (`<meta charset="UTF-8" />`), tytuł (`<title>Tytuł pewnej strony internetowej</title>`), czy też linki do arkuszy stylów (`<link rel="stylesheet" href="styl.css"/>`); oraz ciało (`<body>`), gdzie znajduje się cały kod wizualny. Można to porównać do ludzkiego organizmu, gdzie, w głowie znajduje się podstawowa wiedza. Ciało natomiast jest zbudowane w pewien określony sposób, który każdy może zobaczyć.

²¹ Duckett J., HTML i CSS. Zaprojektuj i zbuduj witrynę WWW. Podręcznik Front End Developera, Helion 2014, s. 24

```

1  <!DOCTYPE HTML>
2  <html lang="pl">
3      <head>
4          <meta charset="UTF-8"/>
5          <title>Strona</title>
6      </head>
7      <body>
8          <h1>Witaj świecie</h1>
9      </body>
10 </html>

```

Rysunek 2.5. Podstawowa struktura kodu HTML

W powyższym przypadku można zauważyc, że na stronie wyświetli się napis „Witaj świecie”. Dokument ten jest kodowany za pomocą UTF-8, czyli polskiego kodowania znaków, podkreślonego również w atrybucie lang znacznika <html>.

CSS

Wyżej przyrównano strukturę HTML do ludzkiego ciała. CSS w takim razie byłby w tym porównaniu odzieżą. W nim nadaje się efekty wizualne konkretnym elementom poprzez umieszczanie atrybutu klas (`class="..."`) lub identyfikatorów (`id="..."`) w języku znaczników. „Działanie kaskadowych arkuszy stylów [(ang. Cascading Style Sheets)] polega na kojarzeniu reguł z elementami HTML. Reguły te określają sposób, w jaki ma być wyświetlana zawartość konkretnych elementów.”²² Taka reguła składa się z: selektora, który można określić po nazwach klas, identyfikatorów lub znaczników; oraz deklaracji, innymi słowy, sposobu wyświetlania elementu. „Każda deklaracja składa się z dwóch części (właściwości i wartości), oddzielonych od siebie znakiem dwukropka.”²³

```

1  h1{
2      color: green;
3      font-family: Arial, sans-serif;
4 }

```

Rysunek 2.6. Przykładowa reguła w języku CSS

Przykład powyżej dotyczy znacznika h1. Napis w tym nagłówku będzie zielony napisany czcionką Arial. Gdyby reguła dotyczyła klasy, przed nazwą pojawiłaby się kropka, a przed identyfikatorem znak #. Istnieje jeszcze wiele oznaczeń dodatkowych wskazujących na dany element z wykorzystaniem wyżej opisanych symboli i znaków specjalnych. „Technologia CSS została tak pomyślana, żeby umożliwić podawanie stylów na różnym poziomie. Istnieją trzy poziomy, na których można je zdefiniować. Każdy kolejny

²² Duckett J., HTML i CSS. Zaprojektuj i zbuduj witrynę WWW. Podręcznik Front End Developera, Helion 2014, s. 230

²³ jw.

dziedziczy wpisy z poprzedniego i może je modyfikować.”²⁴ Chodzi o miejsca, gdzie możemy zamieszczać style: w pliku *.css*, w znaczniku *<head>*, oraz za pomocą atrybutu dla konkretnego znacznika. Najwyższy, a zarazem czwarty poziom, który się bardzo rzadko przytacza to ustawienia domyślne wszystkich projektów.

UIkit

To biblioteka arkuszy stylów udostępniona na licencji MIT. Bazuje na klasach dzięki, którym łatwiej poruszać się w dokumencie stylów. Posiada wiele wbudowanych rozwiązań takich jak animacje, ikony, filtrowanie, powiadomienia, responsywność, itp.

Siatka w tym frameworku²⁵ jest oparta na ułamkach. Możliwe są części ułamkowe od 1 do 6. Daje to sporą możliwość manewru przy układaniu komponentów strony. UIkit obsługuje przeglądarki takie jak: Mozilla Firefox, Safari, Google Chrome, Opera oraz Microsoft Edge do najwyższych wersji. Dodatkowo na stronie producenta znajduje się dopiszek świadczący o tym, że ma on zamiar rozszerzyć wsparcie do każdej możliwej przeglądarki w najbliższym czasie.

JavaScript

Jest to język skryptowy, który pozwala na dynamiczne działanie witryn webowych. „JavaScript zapewnia większą interaktywność stron internetowych, ponieważ umożliwia modyfikowanie ich zawartości oraz kodu znaczników, dzięki którym strony te mogą być wyświetlane w przeglądarkach.”²⁶ Wystarczy odnieść się do danego elementu i zastosować odpowiednią funkcję, aby zmienić zawartość w witrynie na oczach użytkownika. Tym sposobem można przeprowadzać np. proste obliczenia czy też modyfikować wartość przesyłki przy wyborze innej opcji niż podstawowa.

Przykładem wizualnym może być deklaracja zmiennej, do której od razu przypisujemy wartość elementu o danym identyfikatorze „list” (*id="list"*), który jest określony w kodzie HTML: `var lis = document.getElementById('list');`. „Skrypty JS mogą i w przypadku większych fragmentów powinny być umieszczane w osobnych plikach.”²⁷ Wtedy trzeba jednak zamieścić odwołanie do tego pliku (`<script src="scrypt.js"></script>`).

²⁴ Ross J., PHP i HTML. Tworzenie dynamicznych stron WWW, Helion 2010, s. 149

²⁵ narzędzie do budowy aplikacji.

²⁶ Duckett J., JavaScript i jQuery Interaktywne strony WWW dla każdego, Helion 2015, s. 6

²⁷ Ross J., PHP i HTML. Tworzenie dynamicznych stron WWW, Helion 2010, s. 119

jQuery

jQuery to biblioteka JavaScript, która za pomocą selektorów CSS pozwala na wykonywanie odpowiednich żądań. Jego struktura wygląda inaczej niż samego JavaScryptu. „Funkcja o nazwie `jQuery()` pozwala na wyszukanie co najmniej jednego elementu na stronie.”²⁸ Często stosuje się wersję zamienną: `$()`, np. `$('o1')`.

Zaletą tej biblioteki jest jej skrócony zapis oraz inny styl wizualny, dzięki któremu można się łatwiej poruszać po kodzie aplikacji.

jQuery Validation

Jest to plugin bazujący na bibliotece jQuery do szybkiej oraz poprawnej walidacji danych formularza. Istnieje możliwość ukrycia walidacji w osobnym pliku, nie zostawiając go w podstawowym widoku użytkownika. Unika się w ten sposób błędów aplikacji z tym związanych. Zazwyczaj po zatwierdzeniu formularza dane zostają wysłane w dalszą drogę do części serwerowej aplikacji. Dzięki temu rozwiążaniu można wyspecyfikować czego żąda się od użytkownika. Jeśli wpisze on niewłaściwe informacje strona nie przesle wyników i będzie on musiał poprawić swoje dane niepasujące do schematu ustalonego wcześniej. Zostanie również wyświetlony odpowiedni komunikat.

jQuery Mask

Kolejny plugin jQuery służący do autouzupełniania znaków specjalnych. Za jego pomocą można automatycznie ustawić ilość oraz ich rodzaj, jaki jest pożądany w danym polu formularza. Przykładem jest kod pocztowy, który po dwóch pierwszych cyfrach powinien mieć myślnik. Rozwiązaniem takiego problemu w tym pluginie wygląda następująco: `$('#code').mask('00-000');`. Za pomocą takiego zapisu kod wychwyci błąd przy próbie podania liter oraz po pierwszych dwóch liczbach postawi myślnik bez konieczności dodawania go przez użytkownika.

VanillaSelectBox

To biblioteka napisana z wykorzystaniem JavaScrypt do wielokrotnego wybierania danych z listy. Są one wyświetlane w odpowiednim widoku oraz zastępowane ilością wybranych elementów, jeśli nie starczy miejsca na dopisanie ich w taki sposób, aby mieściły się w oknie. Jest to bardzo przydatne narzędzie do wyboru istniejących pól. Posiada również pole wyszukiwania, które można dołączyć do listy, dzięki czemu użytkownik nie musi przewijać do konkretnego elementu.

²⁸ Duckett J., *JavaScript i jQuery Interaktywne strony WWW dla każdego*, Helion 2015, s. 302

2.2.Narzędzia

Jednym z ważnych elementów tworzenia aplikacji jest wybranie właściwych narzędzi. Na rynku znajdują się różne aplikacje oferujące możliwości edytorskie kodu. Dobór odpowiednich i zgodnych z własnymi preferencjami programów bywa ciężki, dlatego warto przybliżyć wybrane opcje.

Środowisko bazodanowe

Edycję baz danych zapewnia zarówno wiersz poleceń jak i bardziej rozbudowane środowisko graficzne. Ze względu na łatwość obsługi oraz wizualną reprezentację encji w pracy zostało użyte narzędzie wchodzące w skład pakietu XAMPP, a mianowicie oprogramowanie phpMyAdmin. Jest to aplikacja napisana w języku PHP do zarządzania bazą danych za pomocą MySQL lub MariaDB stworzona na licencji GPL.

MySQL to ogólnodostępny i wolnoźródłowy system, który jest rozwijany przez firmę Oracle. MariaDB natomiast została stworzona przez pierwotnych twórców powyższego systemu i rozwija się po dziś dzień. Oba systemy stosują język SQL oraz dają możliwość szybkiego wykonywania zapytań.

PhpMyAdmin zapewnia obsługę bazy bez potrzeby stosowania zapytań SQL z linii poleceń, co bywa pomocne przy tworzeniu tabel. Zarówno można przejść w tryb pracy z wykorzystaniem terminala przy kreowaniu, np. większej ilości rekordów ze zmianą szczegółów. W wersji graficznej widoczny jest również model relacyjny, co ułatwia pracę z większą ilością encji. System pozwala m.in. na zarządzanie użytkownikami posługującymi się bazą oraz import i eksport do plików.

Środowisko programistyczne

Podczas tworzenia programu potrzebne jest środowisko, za pomocą którego będzie można napisać, sprawdzić, zmodyfikować oraz przetestować aplikację. Takich właśnie możliwości dostarcza zintegrowane środowisko programistyczne inaczej zwane IDE (ang. Integrated Development Environment).

Zastosowanie języka Java ogranicza wybór dostępnych na rynku aplikacji stworzonych do tego celu. Najpopularniejszymi są: IntelliJ Idea Ultimate, Eclipse oraz Netbeans. Ze względu na przejrzystą strukturę katalogową oraz wolnoźródłowość został wybrany ostatni z nich.

Netbeans podobnie jak inne z wyżej wymienionych posiada zintegrowane rozwiązania, które ułatwiają pracę nad kodem. Przykładem jest możliwość dołączenia bazy danych oraz

serwera do samego IDE dzięki czemu nie trzeba uruchamiać ich osobno, a budując aplikację edytor zadba o wszystko. Dodatkowym atutem jest autosynchronizacja danych przy zapisie plików co oszczędza czas na niepotrzebne akcje ze strony programisty. Debugger jak to przystało na porządne oprogramowanie deweloperskie, również jest wbudowany, przez co szybkiego lokalizowania błędu poprzez prześledzenie wykonywanych operacji nie jest większym problemem.

Serwer aplikacji

Działanie serwera aplikacji opisuje dokładnie podrozdział 2.2 (str. 14). Z wymienionych tam serwerów w pracy został zastosowany WildFly, obecnie rozwijany przez Red Hat. Jego pierwotna nazwa to JBoss, przez którą to firmę został stworzony.

Jest to serwer, który łatwo da się zrozumieć. Poza tym nie trzeba w nim praktycznie nic ustalać poza uruchomieniem serwera oraz założeniem pierwszego konta, którego zazwyczaj ustala się jako root²⁹. Jest lekki (zabiera mało miejsca i zasobów), szybko się uruchamia oraz wykorzystuje najnowsze wersje Javy.

Edytor kodu źródłowego

Stosowany jest do edycji kodu frontendowego. Co prawda można implementować w IDE, jednak dla większego komfortu łatwiej jest użyć innego narzędzia do tych celów. W tym przypadku istnieje jeszcze więcej możliwości niż przy wyborze środowiska programistycznego. Programy takie jak Atom, Visual Studio Code, Brackets, czy nawet Notepad++. Drugi edytor jest tym, który został zastosowany do stworzenia całego szablonu frontendowego aplikacji.

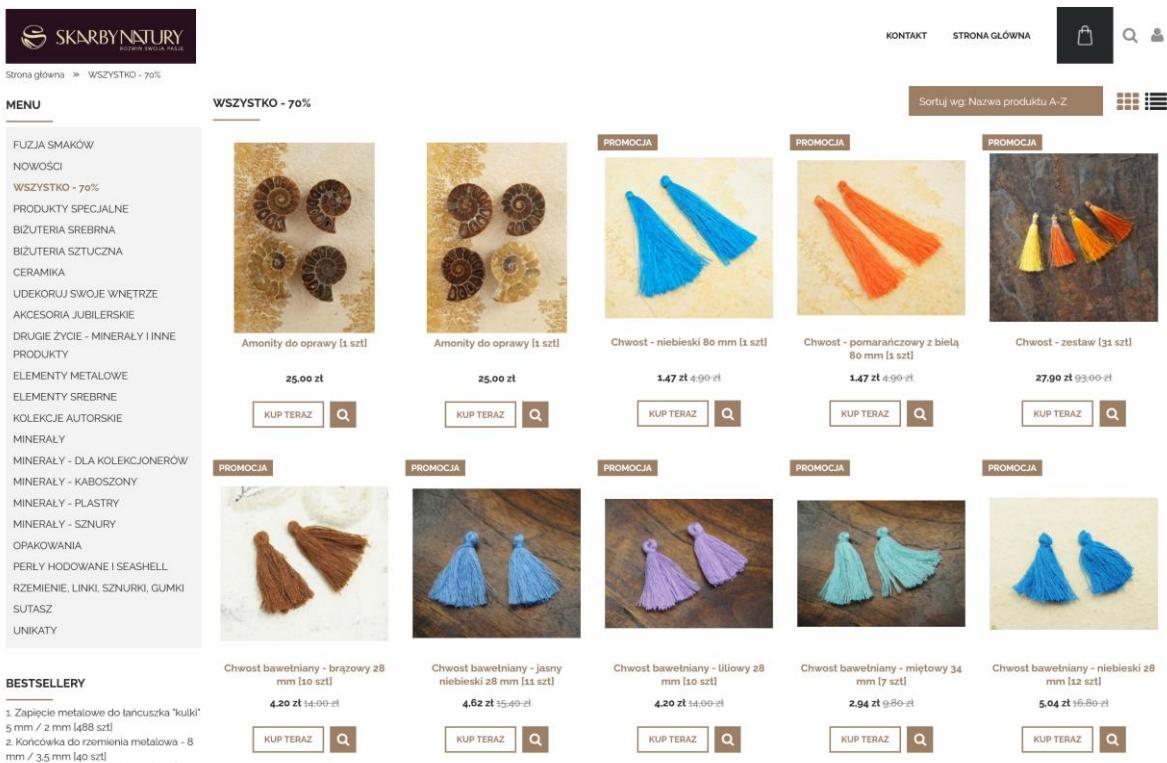
Visual Studio Code jest jednym z najczęściej używanych programów. Posiada wiele wbudowanych funkcji, jakimi są: możliwość zastosowania czarnego motywu, autouzupełnianie kodu, czysta kolorystyka oraz doinstalowanie dodatkowych wtyczek. Na wybór miało również duży wpływ przyzwyczajenie do tego edytora.

²⁹ (ang. korzeń) to użytkownik z najwyższymi uprawnieniami w systemie.

2.3. Przegląd istniejących rozwiązań

Przed przystąpieniem do pracy nad daną aplikacją przeprowadza się badania rynku. Dopiero na podstawie tych działań można wyciągnąć wnioski czego brak istniejącym rynkowym rozwiązaniami lub co jest zrobione dobrze. W tym podrozdziale przedstawione zostaną tylko cztery podobne oraz istniejące strony internetowe uszeregowane w kolejności zasobów sklepu. Oczywiście podobnych metod nie brakuje, lecz wybrane oferują najlepszą jakość produktów.

[skarbynatury.pl \(<https://www.skarbynatury.pl/>\)](https://www.skarbynatury.pl/)



Rysunek 2.7. Widok sklepu skarbynatury.pl [19]

Pierwszym sklepem są skarbynatury.pl. Jeszcze niedawno posiadał bardzo szeroką gamę półfabrykatów wysokiej jakości. W czasie pisania pracy ilość coraz bardziej się zmniejsza, a braki nie są uzupełniane tak jak kiedyś. Prowadząca sklep skupiła się na dodawaniu innych rozwiązań (np. sprzedaży miodów) niż na kontynuowaniu sprowadzania wysokiej jakości produktów jubilerskich. W roku 2020 pojawiła się nawet pogłoska o możliwym zamknięciu sklepu, lecz działa on po dzień pisania tej pracy. Witryna nadal posiada możliwość sprzedaży produktów najlepszej jakości z kategorii elementów i koralików jednak brakuje półfabrykatów do metaloplastyki. W chwili obecnej większość zakładek w menu jest pusta lub wybór jest bardzo skąpy co prowadzi do wrażenia braku towaru. Brak pełnej responsywności aplikacji stwarza problemy przy

składaniu zamówienia z urządzeń mobilnych. Trzeba jednak przyznać, że minimalistyczny design prowadzi do pozytywnych wrażeń wizualnych.

kianit.pl (<https://www.kianit.pl/>)

The screenshot shows the homepage of kianit.pl. At the top right are links for 'Załóż konto' and 'Logowanie'. Below the header is a search bar with placeholder 'Wyszukaj w sklepie' and a 'SZUKAJ' button. A shopping cart icon indicates 'Twój koszyk jest pusty...'. The main navigation menu includes 'STRONA GŁÓWNA', 'NOWOŚCI', 'PROMOCJE', 'AKTUALNOŚCI', and 'PONOWNIE W OFERCIE'. On the left, a sidebar lists various categories: KAMIEŃ NATURALNE, KORAL, PÓŁFABRYKATY SREBRNE 925, PERŁY I MUSZLE, SZKŁO ANTYCZNE, KAMIENIE SYNTETYCZNE, BIZUTERIA SREBRNA 925, STAL SZLACZETNA, TIERRACAST, SWAROVSKI, CERAMIKA, KORALE HANDMADE, and INNE METALE. The 'INNE METALE' section is expanded, listing: Korćowki metalowe, Łączniki z mosiądzu, Przekladiki metalowe, Szpilki z mosiądzu, Zapięcia metalowe, Zawieszki metalowe, KORALIKI Z CYRKONIAМИ, KRYSTALKI, RZEMIENIE, SZNURKI, BRANSOLETKI ZE SKÓRĄ, OPAKOWANIA NA BIŻUTERIĘ, DREWNO, KOKOSOWE, SANDAŁOWE. Below the sidebar are two rows of product cards. The first row contains four items: 'Korćowki do rzemieni 4 mm - 4 szt (Dz 4)' (1,00 zł), 'Zawieszka / charms - muszle - 2 szt (M 24)' (1,40 zł), 'Zapięcie magnesowe 6 mm (MZM f 6)' (2,60 zł), and 'Nakładka na rzemień (R N 10)' (2,45 zł). The second row contains four more items: 'Nakładka na rzemień (R N 14)' (1,45 zł), 'Nakładka na rzemień - serce (R N 9)' (2,10 zł), 'Nakładka na rzemień (N 21)' (1,10 zł), and 'Nakładka na rzemień - kwiat (R N 8)' (1,50 zł). At the bottom of the page are buttons for 'NOWOŚCI' and 'BESTSELLERY'.

Rysunek 2.8. Widok sklepu kianit.pl [13]

Kianit.pl posiada o wiele większy wybór niż powyższy sklep. Co prawda również kuleje w kwestii metaloplastyki oraz sznurków do makramy mniejszej średnicy, to nie brakuje innych elementów, które skuszą kupującego. Dzięki niemu można skompletować zamówienie godne większego projektu jubilerskiego. Design jest oparty na kolorach białym oraz niebieskim, a zdjęcia są wykonywane przez właściciela, zazwyczaj na białym tle, co daje ciekawy całokształt widoczny wyżej. Problemem jest wyświetlanie produktów na szybko. Po najechaniu na obraz przy widoku artykułów dzieli się on na dwie części. Jedna kieruje użytkownika do pełnej karty przedmiotu, a druga, większa, dodaje do koszyka. Przez taki zabieg często, zamiast wyświetlić produkt, dodaje się go do części zamówienia. Na plus można zaliczyć pełną odpowiedzialność strony, dzięki czemu korzystanie z aplikacji jest nieco łatwiejsze.

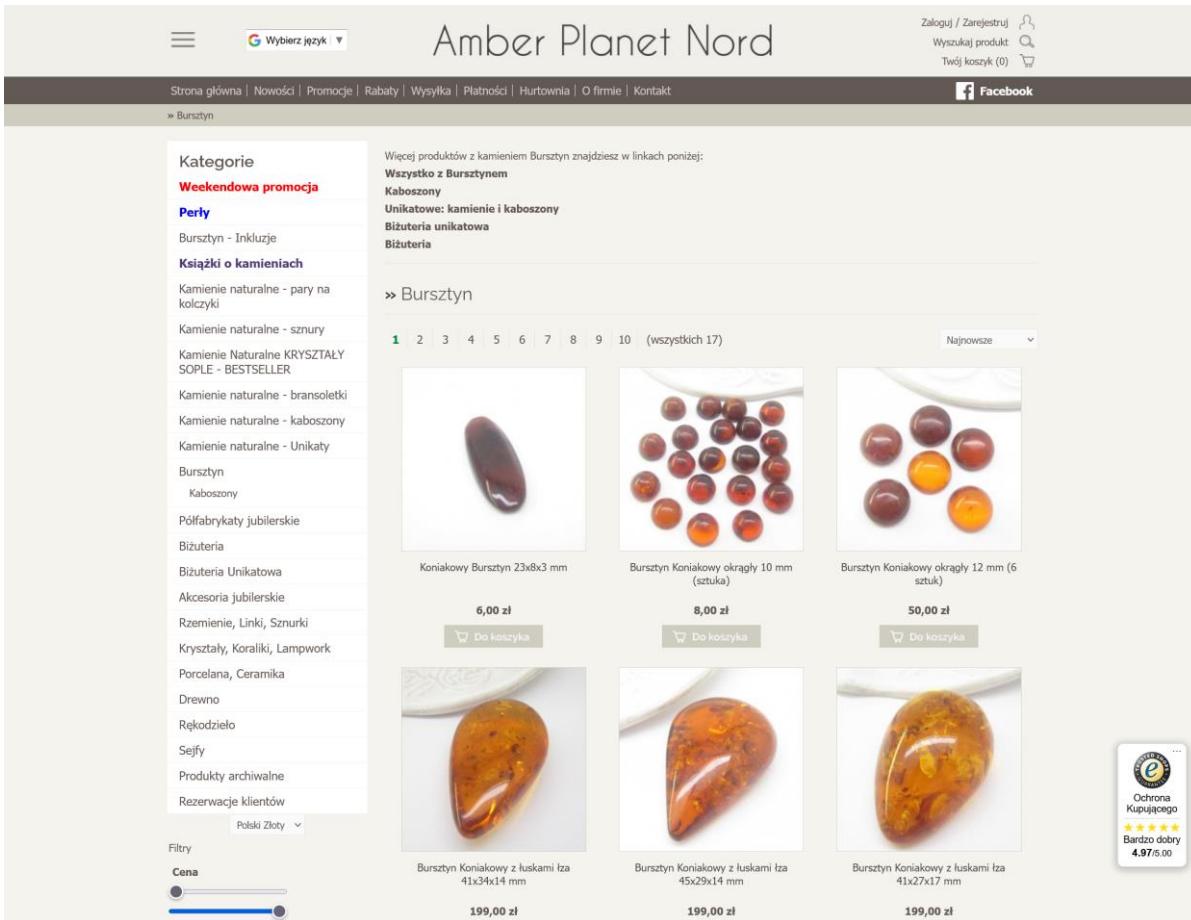
kamieniolomy.pl (<https://kamieniolomy.pl/>)

The screenshot shows the homepage of kamieniolomy.pl. At the top, there's a search bar with a magnifying glass icon and a logo for "KAMIENIOŁOMY.PL" featuring a silhouette of a person working with a pickaxe. To the right of the logo are links for "Zarejestruj się", "Zaloguj się", and a shopping cart labeled "Koszyk: (pusty) do darmowej dostawy: 199.00 zł". Below the header, there's a navigation menu with categories like "STRONA GŁÓWNA", "KAMIENIE NATURALNE SZNURY", "KAMIENIE DROBNE (1-4 MM)", "KAMIENIE BROLETKI", "KAMIENIE MATOWE", "KAMIENIE NA ŁAŃCUZKU", "KAMIENIE W OPRAWIE", "KAMIENIE SYNTETYCZNE", "DREWNO", "PERŁA I MUSZLA", "AKCESORIA I NARZĘDZIA", "EKSPozyTORY I OPAKOWANIA", "CHWOSTY", "PRZEKAŁDKI, ZAWIESZKI I ŁĄCZNIKI", "PÓŁFABRYKATY", "PRZEKAŁDKI, ZAWIESZKI, ŁĄCZNIKI SREBRO 925", "KOLOR", and "KSZTAŁT". A breadcrumb trail indicates the user is at "» KAMIENIE NATURALNE SZNURY". On the left side, there are sections for "WYSZUKIWANIE PO KOLORZE" (with a color palette), "WALUTY" (zł, euro, \$), "AKTUALNE", "NOWOŚCI PROMOCJE", and "MENU" (with a list of mineral types). On the right, there's a "WYSZUKIWANIE ZAAWANSOWANE" section with dropdown menus for "Kategorie: (wybierz)", "Nazwa kamienia: (wybierz)", "Rozmiar: (wybierz)", "Średnica otworu ok.: (wybierz)", "Kolor: (wybierz)", "Cena: (wybierz)", "Promocja: (wybierz)", "Surowiec: (wybierz)", "Kształt: (wybierz)", "Otwór: (wybierz)", "Cena dotyczy: (wybierz)", "Dostępność: (wybierz)", "Nowość: (wybierz)", and "Sortuj wg: NAZWA PRODUKTU A-Z". Below this, there's a section for "KAMIENIE NATURALNE SZNURY" showing three images of agate beaded necklaces: "Agat botswana kula fasetowana ok. 10mm" (80,00 zł), "Agat botswana kula fasetowana ok. 6mm" (55,00 zł), and "Agat botswana kula fasetowana ok. 8mm" (65,00 zł). Each item has a "do koszyka" button.

Rysunek 2.9. Widok sklepu kamieniolomy.pl [12]

Następny sklep (kamieniolomy.pl) skupia się na minerałach. Posiada jedną z największych ofert w internecie. Twórca przemyślał proces filtracji, dzięki czemu można szybko i sprawnie znaleźć to czego się szuka (rys. 2.9). Zarówno w tym sklepie brakuje możliwości zakupu drutów do metaloplastyki. Niestety również sznurki i rzemienie nie są tu dostępne, co uszczupla jego zakres. Oferta półfabrykatów jednak w żadnym razie nie jest określona. Do wyboru są metale zwykłe, stal szlachetna oraz srebro zwykłe i złocone. Menu jest przejrzyste, a dodatkowe opcje pokazują się po wejściu w daną zakładkę. Responsywność jest płynna, co nie sprawia kłopotów w korzystaniu z innych urządzeń.

sklep.amberplanet.pl (<https://www.sklep.amberplanet.pl/>)



Rysunek 2.10. Widok sklepu amberplanet.pl [1]

Ostatni sklep (sklep.amberplanet.pl), jaki zostanie przytoczony w tej pracy, posiada największą gamę zasobów. Półfabrykaty wykonane z różnych metali, minerały w rozmaitych kształtach i wielkościach, sznurki i rzemienie oraz druty srebrne do metaloplastyki. Jest to jeden z najlepiej zaopatrzonych sklepów internetowych w Polsce. Niestety projektując daną biżuterię, nie zawsze można tu znaleźć wszystko. Responsywność również kuleje, ze względów na efekty wizualne. Nieraz ma się wrażenie, że ciężko się po nim poruszać przy szukaniu danych produktów, np. sznurków do makramy.

3. Opis projektu

Każdy projekt programistyczny powinien zostać opisany w postaci wymagań. Jest to pewien zespół warunków oraz cech, które program musi spełnić, aby poprawnie funkcjonować. Na podstawie tego można zacząć tworzyć aplikację będącą odzwierciedleniem tego skrótownego, ale rzeczowego opisu. Zatem przedstawione w tym rozdziale wymagania mają za zadanie przybliżyć zrozumienie podstawowego działania aplikacji.

3.1. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne „opisują te aspekty systemu, które nie są bezpośrednio związane z jego funkcjonalnością. Mogą dotyczyć rozmaitych aspektów systemu, od jego użyteczności do wydajności.”³⁰ Są to zatem funkcje, których zazwyczaj zwykły „zjadacz chleba” nie zauważa na pierwszy rzut oka, ale bez nich nie można prawidłowo utworzyć prawidłowego projektu.

Ograniczenia

Przedstawiony projekt jest aplikacją webową, napisaną w języku Java v13.0.1 z obsługą Maven oraz Java EE 8 Web.

Zależności zastosowane w pliku *pom.xml*: commons-lang3 v3.1, commons-io v2.2, commons-fileupload v1.4 (zarządzanie plikami), maven-shared-utils v0.7, javaee-web-api v8.0, javax.servlet-api v3.1.0, javax.servlet.jsp-api v2.3.1, jstl v1.2, mysql-connector-java v5.1.30 (połączenie z bazą danych), org.jsoup v1.14.2 (konwersja znaków HTML na polskie), javax.mail-api v1.6.2 (obsługa wysyłania wiadomości email).

Pluginy widniejące w dokumencie *pom.xml*: maven-compiler-plugin v3.8.1, maven-war-plugin v2.3, maven-dependency-plugin v2.6, javaee-endorsed-api v7.0.

Strona użytkownika opiera się na: HTML 5, CSS 3, Uikit v3.6.20, JS, jQuery v3.2.1 jQuery Validation v1.12.0 z biblioteką Additional Methods v1.16.0, vanillaSelectBox v0.78, jQuery Mask v1.14.16.

Wykorzystano server WildFly v18.0.1 oraz aplikację phpMyAdmin z pakietu Xampp v7.2.26-0 do tworzenia bazy danych.

Projekt działa poprawnie na przeglądarkach: Google Chrome v66, Opera v45, Edge v17, Firefox v59, co oznacza ich wersje wspierające technologię zastosowaną w pracy.

³⁰ Bruegge B., Dutoit A. H., *Inżynieria oprogramowania w ujęciu obiektowym. UML, wzorce projektowe i Java*, Helion 2011, s. 162

Użyteczność

Każdy, kto korzystał choć raz z usług sklepu internetowego będzie w stanie używać podstawowych funkcji aplikacji. Użytkownicy zalogowani po dwóch pełnych użyciach wszystkich funkcjonalności strony, będą w pełni zapoznani z witryną.

Ponadto aplikacja jest responsywna oraz posiada proces validacji.

Niezawodność

Przesyłanie zapytań SQL odbywa się za pomocą kont odpowiednich użytkowników z ograniczoną funkcjonalnością przeznaczoną do zarządzania przypisanymi im tabelami oraz zastosowano:

- mechanizmy identyfikacji i uwierzytelniania użytkowników;
- spójny i jednolity styl formatowania kodu;
- spójne nazwy plików, zmiennych, klas, funkcji, metod;
- interfejs graficzny w polskiej wersji językowej z obsługą polskich znaków.

Wydajność

W przypadku przerwania sesji system powinien w czasie co najmniej sekundy powrócić do strony głównej z wyświetleniem odpowiedniego komunikatu.

W aplikacji zostały przewidziane podstawowe zabezpieczenia widoków chronionych hasłem. Próba dostania się do nich przez użytkownika nieuprawnionego skutkuje przekierowaniem do strony głównej i pokazaniem stosownej wiadomości.

3.2. Wymagania funkcjonalne

„Wymagania funkcyjne odzwierciedlają interakcje między systemem a jego środowiskiem, w oderwaniu od implementacji tegoż systemu.”³¹ Innymi słowy, jest to wszystko, co widzi użytkownik witryny w postaci środowiska graficznego.

Opisywana aplikacja funkcjonuje jak zwykły sklep internetowy, co oznacza, że działanie podstawowe opiera się na przeglądaniu produktów, dodawaniu ich do koszyka oraz składaniu zamówienia. Poruszanie odbywa się przez odpowiednie menu boczne z kategoriami, przy pomocy ścieżki oraz nawigacji. Klikając ikonę lupy można również wyszukać dany produkt poprzez wpisanie frazy w odpowiednie pole.

³¹ Bruegge B., Dutoit A. H., *Inżynieria oprogramowania w ujęciu obiektowym. UML, wzorce projektowe i Java*, Helion 2011, s. 161

Istnieje również możliwość logowania jako zwykły użytkownik oraz pracownik (tzw. użytkownik z rangą). Zakupów może dokonać każdy z wyżej wymienionych.

Użytkownik zalogowany ma wgląd w swoje zamówienia, które dokonał za pomocą metody logowania. Może również edytować dane konta takie jak: zmiana hasła, danych osobowych, adresów oraz dodania nowego adresu do konta. Osoba taka nie ma ograniczenia w tworzeniu liczby adresów, którymi się posługuje.

Pracownicy dzielą się na cztery kategorie: administratorzy, pracownicy, korektorzy oraz zaopatrzeniowcy. Każdy z nich posiada własne odrębne zakładki oparte o możliwości, jakie posiadają.

Administrator ma możliwość:

- kontroli pracy wszystkich rang poprzez sprawdzanie historii;
- dodawania użytkownika z rangą oraz jego usuwania (jeśli nie wykonał jeszcze żadnych kroków w historii zdarzeń);
- edycji oraz tworzenia kategorii i tagów w menu aplikacji;
- CRUD³² na niestatycznych tabelach słownikowych.

Pracownik:

- zatwierdza zamówienia i zmienia ich status;
- dodaje numer śledzenia przesyłki przy odpowiednim statusie;
- przegląda zamówienia oraz podstawowe dane użytkowników.

Korektor posiada możliwość:

- modyfikacji i dodawania odpowiedzi na recenzje wystawiane przez użytkowników;
- edycji opisu tagu i kategorii;
- wglądu do produktów oraz zmiany ich opisu.

Zaopatrzeniowiec:

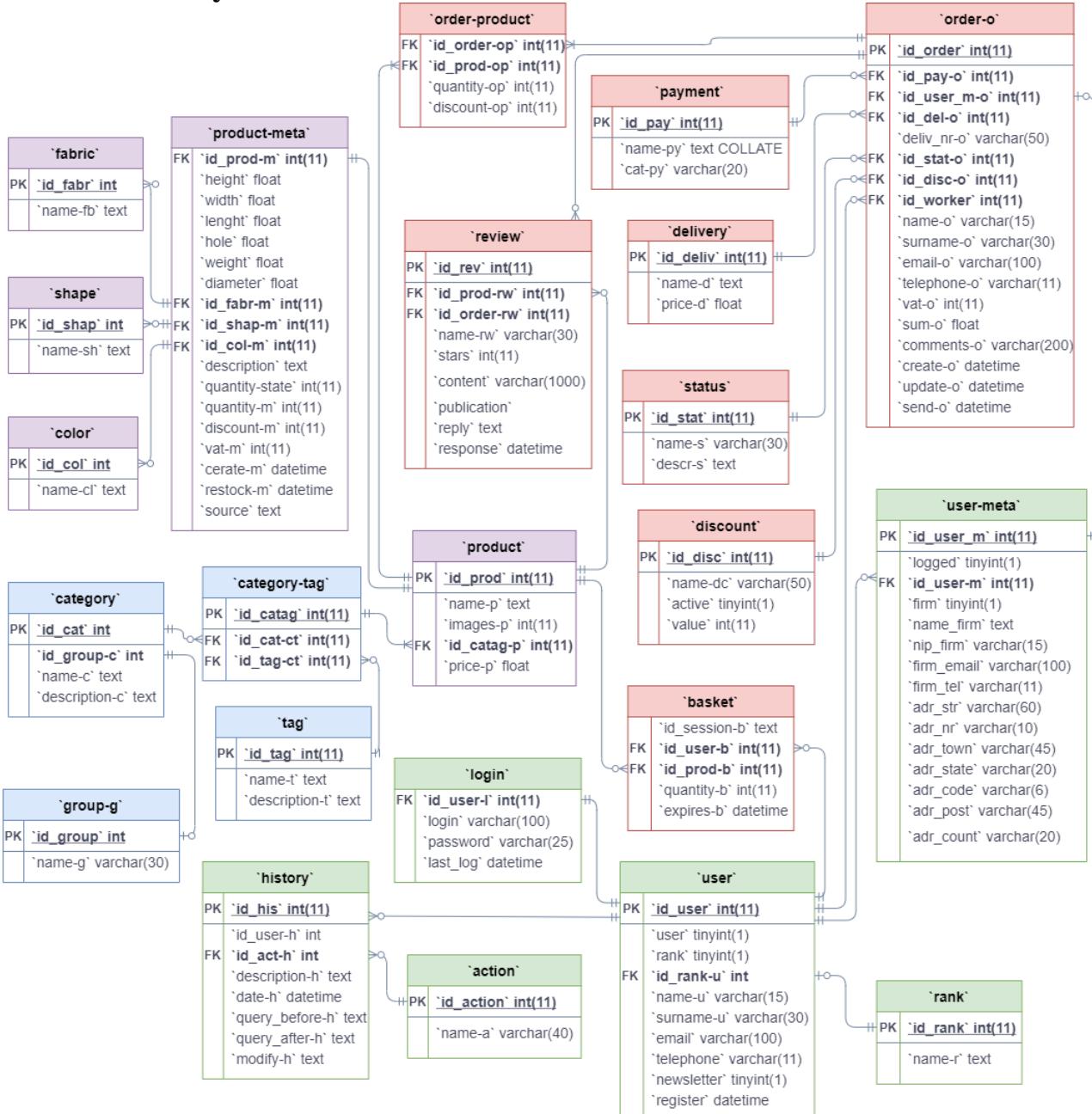
- dodaje nowe przedmioty oraz je edytuje;
- przywraca artykuły, które nie są już na stanie;
- modyfikuje dane w niektórych tabelach słownikowych przypisanych do produktu.

³² (ang. Create, Read, Update, Delete) wykonywanie podstawowych działań tworzenia, czytania, aktualniania i usuwania danych z tabel

4. Opis implementacji

Proces implementacji tak naprawdę oznacza „kodowanie atrybutów i metod każdego obiektu w języku programowania i integrowanie wszystkich obiektów w funkcjonujący pojedynczy system.”³³ Nadszedł zatem czas na pokazanie rzeczywistego działania całej aplikacji. Jest to swoisty wstęp do projektu wraz z zaprezentowanym sposobem postępowania. Kolejność będzie podobna do prezentowanej w rozdziale 2.1.

4.1. Baza danych



Rysunek 4.11. Diagram ERD bazy danych

³³ Bruegge B., Dutoit A. H., *Inżynieria oprogramowania w ujęciu obiektowym. UML, wzorce projektowe i Java*, Helion 2011, s. 53.

Celem poniższej prezentacji jest wyjaśnienie działania oraz wyglądu istniejącej w projekcie bazy danych. Jak widać na rysunku 4.11 baza danych jest dość rozbudowana. Posiada 3 tabele główne (**product**, **order-o**, **user**), 5 dodatkowych (**product-meta**, **review**, **user-meta**, **history**, **login**), 3 tabele łącznikowe (**category-tag**, **order-product**, **basket**) oraz 12 tabel słownikowych (**fabric**, **shape**, **color**, **group-g**, **tag**, **category**, **payment**, **delivery**, **status**, **discount**, **action**, **rank**). Łącznie 23 tabele połączone relacjami. Innym podziałem są encje, które spełniają dane zadanie. W tym przypadku są to 4 grupy: menu (kolor niebieski na rysunku 4.11), produkt (fioletowy), zamówienie (czerwony) oraz użytkownik (zielony).

Ze względu na konwencje pisania przez programistów, baza danych została stworzona w języku angielskim. Jest to łatwiejszy sposób zorientowania się w strukturze projektu. Staje się on bardziej uniwersalny nie tylko dla obcokrajowców. Każda z tabel, dla lepszego zrozumienia, zostanie omówiona poniżej w kolejności grup.

Menu

Zadaniem pierwszej grupy jest wyświetlanie menu, a co się z tym wiąże segregacja produktów na grupy, kategorie i podkategorie nazwane w tym projekcie tagami. Tak samo jak artykuły w sklepie są posortowane według konkretnych wytycznych, tak i w tym przypadku można powiedzieć o pewnej segregacji produktów względem menu.

Encja group-g:

Jest to jedna z tabel słownikowych. Służą one do przechowywania stałych danych, które zazwyczaj się nie zmieniają. Większość z nich bazuje na tylko dwóch atrybutach, jakimi są identyfikator (zwykle zaczynający się od id, a następnie dołączonej skrótowej nazwie tabeli) oraz nazwa (name i skrót zazwyczaj jedno lub dwuliterowy encji). Są połączone z drugą encją za pomocą relacji wiele do jednego. Tabela **group-g** jest tego doskonałym przykładem. Posiada tylko dwa atrybuty `id_group` oraz `name-g`, ponieważ w tym przypadku nie potrzeba więcej.

Encja category:

Następna tabela słownikowa jest bardziej rozbudowana. Oprócz identyfikatora (`id_cat`) oraz nazwy (`name-c`) posiada również opis (`description-c`). Jednak w tym wypadku najważniejszy jest klucz obcy łączący ją z tabelą **group-g** (`id_group-c`). Dzięki temu zabiegowi wiadomo, która kategoria odpowiada grupie.

■ Encja tag:

Ostatnia z tabel słownikowych została stworzona na podstawie bazowego schematu. Posiada swój identyfikator podkategorii (*id_tag*), nazwę (*name-t*) oraz dodatkowy opis (*description-f*). Na potrzeby segregacji nie potrzeba nic więcej.

■ Encja category-tag:

Tabelą łączącą wszystko w całość jest niniejsza encja. Łączy ona identyfikatory kategorii (*id_cat-ct*) oraz tagu (*id_tag-ct*), aby łatwiej było się odnieść do konkretnych produktów po identyfikatorze danego miejsca w menu (*id_catag*). Tabele łącznikowe są połączone z innymi tabelami za pomocą relacji wiele do wielu, co pozwala na dodawanie dużej ilości danych, które mogą się powtarzać.

Produkt

Podobnie jak bez produktów w sklepie nie można dokonać zakupu, tak w tym przypadku bez tej tabeli nie byłoby sensu tworzyć projektu. Zatem jest ona najważniejsza w całej bazie danych.

■ Encja product:

Jak w każdej podstawowej tabeli musi istnieć identyfikator (*id_prod*), po którym inne encje mogą się odnosić dzięki wykorzystaniu klucza obcego. Tak samo jak w rzeczywistości produkt musi mieć nazwę (*name-p*); swój wygląd, czyli w tym przypadku ilość obrazów (*images-p*), jakie znajdują się w odpowiednich folderach projektu; miejsce w katalogu (*id_catag-p*); oraz cenę (*price-p*). Ten przedostatni atrybut jest kluczem obcym do tabeli łącznikowej **category-tag** opisanej wyżej.

■ Encje fabric, shape i color:

Aby przejść dalej, należy opisać pokrótko tabele słownikowe wykorzystywane w tej grupie. Są one stworzone za pomocą podstawowego schematu. Tabela **fabric** (materiał), jak widać na rysunku 4.11, posiada tylko dwa atrybuty, którymi są identyfikator (*id_fabr*) oraz nazwa (*name-fb*). Tą samą metodą opisano dwie następne **shape** (kształt) i **color** (kolor). Tworzą one listę rozwijalną, którą można pobrać oraz przypisać odpowiednią nazwę do produktu.

■ Encja product-meta:

To tabela przechowująca metadane produktu. Odnosi się do bazy **product** po jego identyfikatorze (*id_prod-m*), który w tym przypadku, dla tej encji, jest kluczem obcym. Następne atrybuty nieobowiązkowe są opisem wyglądu określonego produktu: wysokość

(`height`), szerokość (`width`), długość (`length`), wielkość otworu (`hole`), waga (`weight`) oraz średnica (`diameter`). Kolejne są klucze obce, niemogące występować jako wartości puste, odnoszące się do tabel słownikowych odpowiadających za materiał wykonania (`id_fabr-m`), kształt (`id_shap-m`), oraz kolor (`id_col-m`). Ciągiem dalszym wymaganych atrybutów są: opis produktu (`description`), ilość na stanie (`quantity-state`), ilość w opakowaniu (`quantity-m`), przecena produktu (`discount-m`), wartość podatku VAT (`vat-m`), oraz dwie daty utworzenia (`cerate-m`) oraz przywrócenia produktu na stan (`restock-m`). Ostatnim atrybutem jest źródło (`source`), z którego został pobrany produkt. Zabieg ten jest wykonany, aby uniknąć plagiatu.

Zamówienie

Mówiąc o sklepie, pierwsze co nam przychodzi zazwyczaj na myśl, są zakupy. W celu obsługi tej funkcjonalności potrzebne są specjalne tabele opisane poniżej.

■ Encje payment, delivery, status oraz discount:

Pierwszą z czterech tabel słownikowych w tej grupie jest encja **payment**. Opisuje metody płatności, które są możliwe do obsłużenia w sklepie internetowym. Oprócz podstawowych atrybutów (`id_pay`, `name-py`), posiada również kategorię płatności (`cat-py`).

Tabela **delivery** opisuje sposób dostaw. Zapewniona została podstawa struktura odniesienia (`id_deliv`, `name-d`) z atrybutem służącym do zapisu ceny (`price-d`).

Kolejną jest status zamówienia, który prowadzi do skrótowej nazwy (`name-s`) oraz rozwiniętego opisu (`descr-s`), tłumaczącego co się dzieje.

Ostatnią z tabel słownikowych jest kod promocyjny (**discount**). Posiada prócz podstaw, znacznik aktywności (`active`) mówiący o tym, czy dany kod może być użyty oraz jego wartość procentową (`value`).

■ Encja order-o:

Tabela główna zamówień. W niej znajdują się najważniejsze dane dotyczące wyborów zamawiającego. Jak w każdej tabeli głównej musi znaleźć się identyfikator (`id_order`), następnie widnieją klucze obce prowadzące do konkretnej, jednej metody płatności (`id_pay-o`), dostawy (`id_del-o`), statusu (`id_stat-o`) oraz kodu promocyjnego (`id_disc-o`). Pominięty klucz obcy (`id_user_m-o`) prowadzi do adresu klienta. W tabeli znajduje się również numer dostawy (`deliv_nr-o`), który pracownik, przypisany do zamówienia kolejnym kluczem obcym (`id_worker`) dodaje w swoim czasie. Podstawowe dane zamawiającego widnieją w czterech następnych atrybutach takich jak: imię (`name-o`), nazwisko (`surname-o`), adres e-mail (`email-o`) oraz numer telefonu

(`telephone-o`). Tak samo jak na paragonie oraz fakturze znajduje się wielkość podatku VAT (`vat-o`) oraz wartość ostateczna zamówienia (`sum-o`). Ostatnim drobiazgiem od zamawiającego jest komentarz do zamówienia (`comments-o`). Znaczniki czasowe takie jak utworzenie zamówienia (`create-o`), uaktualnienie przez pracownika statusu (`update-o`) oraz wysłanie zamówienia (`send-o`) to ostatnie atrybuty encji.

■ Encja basket oraz order-product:

Każde zamówienie musi posiadać produkty wchodzące w jego skład. Do tego służą niniejsze tabele. Encja **basket** ma za zadanie zapisywać koszyk, który nie został przypisany do zamówienia. Znajdują się tam identyfikatory: sesji (`id_session-b`) dla użytkownika niezalogowanego; zalogowanego (`id_user-b`) oraz produktu (`id_prod-b`). Jednym z ważniejszych atrybutów jest ilość dodanych przedmiotów (`quantity-b`) oraz data wygaśnięcia sesji użytkownika niezalogowanego (`expires-b`).

Podobna, lecz nie taka sama encja **order-product** odpowiada za produkty znajdujące się już na stałe w zamówieniu. Posiada identyfikatory zamówienia (`id_order-op`) oraz produktu (`id_prod-op`). Podobnie jak we wcześniejszej tabeli widnieje tu ilość przedmiotów (`quantity-op`), lecz ze względu na zmieniające się promocje towarów musi mieć również wartość promocji (`discount-op`) przysługującą produktowi w dniu zakupu.

■ Encja review:

Jedną z tabel łączących produkt z zamówieniem jest recenzja. Wystawiana jest przez użytkowników zarówno zalogowanych jak i tych, którzy dokonali ich zakupu. Jak zwykle posiada swój identyfikator (`id_rev`) oraz identyfikatory wspomnianych już produktu (`id_prod-rw`) i zamówienia (`id_order-rw`). Dołączona jest również nazwa recenzji (`name-rw`), ilość wystawionych gwiazdek (`stars`), treść (`content`), jak i datę jej wystawienia (`publication`). Dla korektora przewidziano dwa dodatkowe atrybuty, takie jak odpowiedź na recenzje (`reply`) oraz jej datę (`response`).

Użytkownik

Dla poprawnego działania całego projektu baza danych powinna posiadać miejsca, w których będą przechowywane dane pracowników oraz użytkowników.

■ Encja rank:

Aby odróżnić użytkownika zwykłego od pracownika, dodano tabelę **rank**, która przechowuje możliwe rangi. Jest to jedna z tabel słownikowych o stałych atrybutach (`id_rank`, `name-r`). Rangi tak jak zostało opisane w rozdziale trzecim to administrator, pracownik, korektor oraz zaopatrzeniowiec.

Encja user:

Tabela główna w tej grupie to **user**, która posiada podstawowe dane dotyczące konta użytkownika. Jak każda tabela posiada swój identyfikator (`id_user`) oraz znacznik świadczący o tym, że jest to użytkownik (`user`). W tej encji można również znaleźć znacznik (`rank`) zwiastujący pracownika aplikacji oraz jego identyfikator w postaci klucza obcego (`id_rank-u`). Wspomniane wyżej podstawowe dane to imię (`name-u`), nazwisko (`surname-u`), adres e-mail (`email`), numer telefonu (`telephone`) oraz akceptacja newslettera (`newsletter`). Ostatnim atrybutem jest data rejestracji (`register`).

Encja user-meta:

Niniejsza tabela została już wspomniana w grupie zamówień. Znajdują się tu adresy użytkowników. Zgodnie z wymaganiami mogą oni posiadać więcej niż jeden adres. Identyfikator (`id_user_m`) jest kluczem głównym. Znacznik `logged` pozwala na sprawdzenie, czy jest to osoba zalogowana, dzięki czemu, w następnym kroku, można bezpiecznie sprawdzić identyfikator danego użytkownika (`id_user-m`). Podobnie dzieje się ze znacznikiem `firm`, który mówi, czy jest to adres firmy, czy prywatny. Jeśli jest on w stanie 1, oznacza firmę, której nazwa (`name_firm`), numer NIP (`nip_firm`), adres e-mail (`firm_email`) oraz numer telefonu (`firm_tel`) będzie zapisany w podanych atrybutach. W obu przypadkach będzie wypełniony adres, na który składa się ulica (`adr_str`), numer mieszkania (`adr_nr`), miasto (`adr_town`), województwo (`adr_state`), kod pocztowy (`adr_code`), poczta (`adr_post`) oraz kraj (`adr_count`).

Encja login:

Użytkownik powinien mieć możliwość zalogowania, co zapewnia mu tabela **login**. Kluczem obcym jest identyfikator (`id_user-l`) ustawiony na wartość unikalną, dzięki czemu nie powstaną dwie możliwości logowania dla tej samej osoby. Podobnie z loginem (`login`), który jest adresem e-mail, ma wartość unikalną. Hasło (`password`) nie może być puste, co również zabezpiecza bazę danych. Ostatnim atrybutem jest data ostatniego logowania (`last_log`).

Encja action:

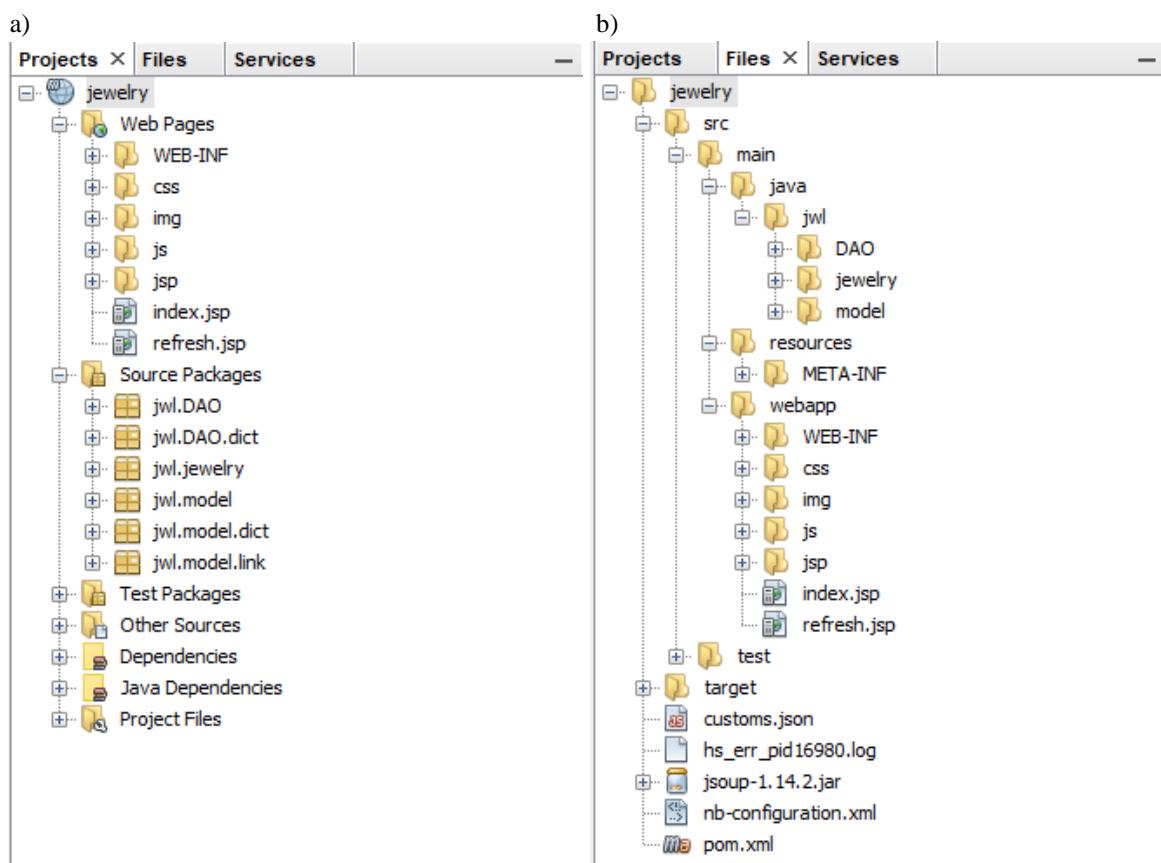
Ostatnia tabela słownikowa jest przygotowana na potrzeby historii zdarzeń, dzięki której użytkownik może prześledzić swoje działania. Posiada wartości podstawowe (`id_action`, `name-a`), które nie mogą być puste.

■ Encja history:

Wyżej wspomniano o historii zdarzeń. Dana zmiana musi dotyczyć użytkownika, do którego tabela odnosi się po kluczu obcym (`id_user-h`) oraz danej czynności w tabeli **action** (`id_act-h`). Oprócz funkcji prześledzenia zdarzeń istnieje możliwość cofania zmian dokonanych na koncie dzięki zapisaniu zapytania SQL przed (`query_before-h`) oraz po (`query_after-h`) zmianie. Oprócz tych atrybutów znajdują się tam: opis (`description-h`), skrót modyfikacji (`modify-h`) oraz data wykonania zmiany (`date-h`).

4.2. Struktura projektu

Struktura jest oparta na wcześniej omawianym Mavenie. W projekcie NetBeans wygląda to jednak trochę inaczej mimo drzewa generowanego automatycznie.



Rysunek 4.12. a) drzewo projektu aplikacji NetBeans, b) drzewo katalogowe projektu

Wyróżnia się część Javy oraz widoku. Jak zostało to wcześniej opisane, w głównym folderze znajdują się pliki, z których korzysta cała aplikacja.

W projekcie podstawowym folderem Javy jest `jwl`, w którym znajdują się trzy podkatalogi: `DAO`, `jewelry` oraz `model`. W ostatnim z wymienionych znajdują się modele wszystkich tabel z bazy danych. Pliki te służą do tworzenia konstruktorów przechowujących odpowiednie dane z encji bazy. Występują w nim dwa podfoldery: `dict` służący do przechowywania tabel słownikowych oraz `link` dla tabel łącznikowych. Niektóre pliki

takie jak *CatTag.java* przechowują więcej niż jedną tabelę, w tym przypadku są to dwie encje słownikowe **category**, **tag** oraz łącząca **category-tag**. W katalogu *DAO* (ang. Data Access Object) przechowywane są operacje na bazie danych dotyczące konkretnej tabeli. Jest to podstawowy CRUD, który pozwala serwletom na wykonywanie modyfikacji. Skoro już wspomniano o serwletach to znajdują się one w folderze *jewelry*. Występuje ich osiem:

- *ControlBasket.java* – obsługuje działanie całego koszyka, zamówienia oraz tabel z nimi związanych;
- *ControlFiles.java* – kontroluje transfer pliku od użytkownika zalogowanego z rangą zaopatrzeniowca, ich usuwanie oraz zmianę ilości zdjęć w bazie danych;
- *ControlLogin.java* – pozwala na logowanie oraz przekierowanie do konkretnej rangi zapisanej w bazie danych po zalogowaniu;
- *ControlProducts.java* – występuje tu działanie całego menu oraz znajdujących się w nim linków, jak i obsługę paska nawigacji z odniesieniami do produktów. Pozwala również wyświetlić dane jednego przedmiotu oraz wystawić recenzje;
- *ControlUser.java* – obsługuje panel użytkownika zalogowanego: jego zamówienia oraz zmianę danych;
- *ControlUserRank.java* – zarządza użytkownikami z rangą oraz wszystkim co się ich tyczy;
- *ControlSrv.java* – podstawowy serwlet kontrolujący funkcje takie jak: historia zdarzeń, wyświetlanie strony głównej oraz obsługa błędów i stron statycznych;
- *RandomPassword.java* oraz *SendMail.java* – klasy pozwalające na dodatkowe funkcje użytkownikom z rangą;

Folder *webapp* obsługuje stronę wizualną całego projektu. To katalog dodatkowy, który przejął działanie *resources*. Znajdują się w nim foldery:

- *WEB-INF* – generowany przez Maven dla stałych danych oraz pliku *web.xml*;
- *css* – gdzie przechowywane są wszystkie arkusze stylów;
- *img* – dla wszystkich obrazów, takich jak logo lub zdjęć produktów;
- *js* – z plikami JavaScript oraz jQuery obsługującymi płynne działanie interfejsu użytkownika;
- *jsp* – ze zmiennymi dostarczonymi przez serwlety. Każdy z nich posiada swój własny katalog;

- *basket* z plikami *Basket.jsp* (wyświetlanie pierwszego kroku w koszyku), *Basket1.jsp* (wybór kontynuowania zakupów jako użytkownik zalogowany lub nie), *BasketNL.jsp* (formularz dla osoby bez logowania), *Order.jsp* (wyświetlanie dokonanego zamówienia) oraz *Summary.jsp* (podsumowanie z akceptacją koszyka i regulaminu);
 - *log* z plikiem *LogIn.jsp* wyświetlającym stronę logowania z przyciskiem możliwej rejestracji;
 - *products* z plikami *ListCT.jsp* (obsługa list produktów z menu oraz nawigacji), *Product.jsp* (wyświetlanie karty przedmiotu), *Review.jsp* (formularz zrecenzji), *Search.jsp* (lista wyszukanych produktów);
 - *static* z plikami stron statycznych;
 - *users* z plikami *RegForm.jsp* (formularz rejestracyjny użytkownika), *UpdateUsr.jsp* (modyfikacja danych takich jak hasło, adresy), *indexUsr.jsp* (strona główna wyświetlająca dokonane zamówienia);
 - *usersRnk* z plikami *Administrator.jsp* (widok główny administratora), *Corrector.jsp* (korektora), *Supplier.jsp* (zaopatrzeniowca), *ViewRnk.jsp* (obsługa poszczególnych danych ze strony głównej rang), *Worker.jsp* (widok główny pracownika);
 - oraz pliki dynamiczne: *aside.jsp* (menu kategorii i tagów), *asideRnk.jsp* (nawigacja boczna użytkownika z rangą), *country.jsp* (lista krajów możliwych do wyboru), *footer.jsp* (stopka), *form.jsp* (pliki .css i .js potrzebne do funkcjonowania walidacji), *head.jsp* (podstawowa konfiguracja <head>), *header.jsp* (nagłówek stron), i nawigacje: *nav.jsp* (bazowa), *navBask.jsp* (w koszyku), *navL.jsp* (użytkownika zalogowanego);
- oraz plik strony głównej *index.jsp* i *refresh.jsp*, który umożliwia przekierowanie po dodaniu i usunięciu zdjęcia.

4.3. Operacje na bazie danych

Do celu ogólnego opisania operacji zostanie użyty plik *ActionDAO.java* z folderu *DAO*. Poniżej przedstawiono cztery podstawowe operacje używane w całej aplikacji, które zostały pokrótko omówione w rozdziale 2.1.

Tworzenie, uaktualnianie i usuwanie

Tworzenie nowych danych w bazie przebiega za pomocą zapytania CREATE definiowanego na początku metody. Są one przygotowywane za pomocą klasy *PreparedStatement*, która zapobiega SQL Injection³⁴. W miejsca znaku ? zostanie wstrzyknięta odpowiednia zmienna. Następnie zapytanie zostaje wykonane po stronie bazy danych przy pomocy metody *executeUpdate()* zwracającej ilość wykonanych wierszy.

```
25  //utworzenie akcji
26  public boolean create(Action act) throws SQLException {
27      String sql = "INSERT INTO `action`(`name-a`) "
28      | | | | + "VALUES (?)";
29      connect();
30
31      boolean inserted;
32      try (PreparedStatement statement = jdbcConnection.prepareStatement(sql)) {
33          statement.setString(1, act.getName());
34          inserted = statement.executeUpdate() > 0;
35      }
36      disconnect();
37      return inserted;
38  }
```

Rysunek 4.13. Utworzenie danych pliku *ActionDAO.java*

Podobnie dzieje się przy uaktualnianiu wierszy. Różnicą w tym wypadku jest zapytanie UPDATE dla modyfikacji wybranych danych.

```
104 //uaktualnienie nazwy akcji
105 public boolean update(Action act) throws SQLException {
106     String sql = "UPDATE `action` SET `name-a` = ? "
107     | | | | + " WHERE `id_action` = ?";
108     connect();
109
110     boolean updated;
111     try (PreparedStatement statement = jdbcConnection.prepareStatement(sql)) {
112         statement.setString(1, act.getName());
113         statement.setInt(2, act.getId());
114         updated = statement.executeUpdate() > 0;
115     }
116     disconnect();
117     return updated;
118 }
```

Rysunek 4.14. Uaktualnienie danych pliku *ActionDAO.java*

³⁴ Metoda ataku na bazę danych polegająca na wstrzykiwaniu fragmentu kodu SQL, dzięki czemu może wyłuskać dane z bazy.

Sytuacja się powtarza gdy usuwamy je przy pomocy zapytania DELETE. Zmieniają się tylko dane, które trzeba wstrzyknąć oraz sprecyzowane zapytania.

```
122     //usuwanie akcji
123     public boolean delete(int id) throws SQLException {
124         String sql = "DELETE FROM `action` WHERE `id_action` = ?";
125         connect();
126
127         boolean deleted;
128         try (PreparedStatement statement = jdbcConnection.prepareStatement(sql)) {
129             statement.setInt(1, id);
130             deleted = statement.executeUpdate() > 0;
131         }
132         disconnect();
133         return deleted;
134     }
135 }
```

Rysunek 4.15. Usuwanie danych pliku ActionDAO.java

Czytanie

Trochę inaczej to wygląda przy pobieraniu, inaczej czytaniu. Również w tym miejscu występuje przygotowanie zapytania, lecz rezultat powinien zostać pobrany do odpowiednio utworzonej zmiennej. W tym przypadku jest to nazwa danej akcji (name), która została pobrana jako wynik zapytania SELECT.

```
63     //pobranie jednej akcji
64     public Action read(int id) throws SQLException {
65         Action act = null;
66         String sql = "SELECT `name-a` FROM `action` WHERE `id_action` = ?";
67         connect();
68
69         try (PreparedStatement statement = jdbcConnection.prepareStatement(sql)) {
70             statement.setInt(1, id);
71
72             try (ResultSet resultSet = statement.executeQuery()) {
73                 if (resultSet.next()) {
74                     String name = resultSet.getString("name-a");
75                     act = new Action(id, name);
76                 }
77             }
78         }
79         disconnect();
80         return act;
81     }
```

Rysunek 4.16. Czytanie danych pliku ActionDAO.java

4.4. Strona główna

The screenshot shows the homepage of the Amethyst website. At the top, there is a purple header with the brand name "Amethyst" in a stylized font. Below the header, a navigation bar includes links for "NOWOŚCI", "PROMOCJE", "PONOWIONE", a shopping cart icon, a user profile icon, and a search icon.

On the left side, a sidebar contains a tree menu with categories and sub-categories:

- KORALIKI
 - NATURALNE MINERAŁY <
 - MINERAŁY SYNTETYCZNE <
 - PERŁY <
 - KORAL <
 - BURSZTYN <
 - SZKŁO <
 - KRYSTAŁKI <
 - CERAMIKA <
 - DREWNO <
- PÓŁFABRYKANTY
 - STAL SZLACZETNA <
 - SREBRO <
 - MIEDŹ <
 - INNE <
- BIŻUTERIA
 - BIŻUTERIA <
- PASMANTERIA
 - RZEMIEŃ <
 - SZNURKI <
 - WSTĄŻKI <
 - JEDWAB <
 - SUTASZ <
- INNE
 - OPAKOWANIA <
 - NARZĘDZIA <

The main content area features a welcome message "Witaj wędrowcze!" and a message of thanks from the store owner. It also includes a section titled "Proponowane produkty" (Recommended products) displaying various jewelry items with their names and prices:

Produkt	Opis	Cena
AMETYST ZIELONY FASETOW. BRIOLLETTE MIGDAŁ 10-9 MM	5.2 zł	AMETYST FASETOWANE KULE 10 MM 2.4 zł
KYANIT FASETOWANE BRIOLLETTE 9 - 8 MM (IN 2)	6.9 zł	GRANAT KULE 6 MM - SZNUR 29.0 zł
OPALIT FIOLETOWY KULA GŁADKA OK. 6MM	12.0 zł	Dumortieryt Fasetowany kropla 5.0 zł
AGAT FUKSJA FASETOWANE KULE 6 MM - SZNUR	16.98 zł	AMETYST KULE 4 MM - 10 SZT 3.8 zł

At the bottom, there are three footer sections: "O FIRMIE" (About the company), "GWARANCJA I ZWROTY" (Guarantee and Returns), and "MOJE KONTO" (My Account). Each section contains links to specific pages like "O nas", "Polityka prywatności", and "Twoje zamówienia".

Rysunek 4.17. Strona główna aplikacji

Interfejs graficzny

Kolorystyka pełni bardzo ważną rolę w tworzeniu strony internetowej. Ze względu na logo witryny, którym jest ametyst, postanowiono zastosować odcień fioletu jako motyw przewodni. Dodatkowo jako kolory dopełniające zostały użyte odcienie czerni i szarości dla łatwego rozróżnienia większości przycisków. Sklep wykorzystuje przeźroczystości oraz cienie dla lepszego efektu wizualnego.

Komunikaty:

- Zielony: proces przebiegł pomyślnie;
- Czerwony: wystąpił błąd w trakcie wykonywania procesu.

Frontend:

Wchodząc na stronę z poziomu wyszukiwarki internetowej, ląduje się na stronie głównej. Na niej znajdują się: nazwa, pasek nawigacyjny (Nowości, Promocje, Ponowione, logo prowadzące do strony głównej, koszyk, logowanie, wyszukiwanie), menu z kategoriami rozwijalnymi, krótki opis sklepu, proponowane produkty oraz stopka ze stronami statycznymi. Z tego miejsca można przeglądać przedmioty poprzez poruszanie się w menu bocznym, z poziomu panelu nawigacyjnego lub klikając jeden z losowo wyświetlonych poniżej artykułów.

Backend:

Jak już zostało wspomniane skryplety wykonują się po stronie serwerowej. Dla poprawności obsługi danych bezpośrednio na stronie *index.jsp* został zamieszczony fragment kodu wywołujący losowe produkty:

```
13 <%  
14 | ProductMDAO prodMDAO = new ProductMDAO(application.getInitParameter("jdbcURL"),  
15 | | | | | | | | | | application.getInitParameter("jdbcUserNL"),  
16 | | | | | | | | | | application.getInitParameter("jdbcUserNLPassw"));  
17 | List<ProductMeta> listProdIndex = prodMDAO.readIndexRand();  
18 | request.setAttribute("listProdI", listProdIndex);  
19 %>
```

Rysunek 4.18. Skryplet z pliku *index.jsp*

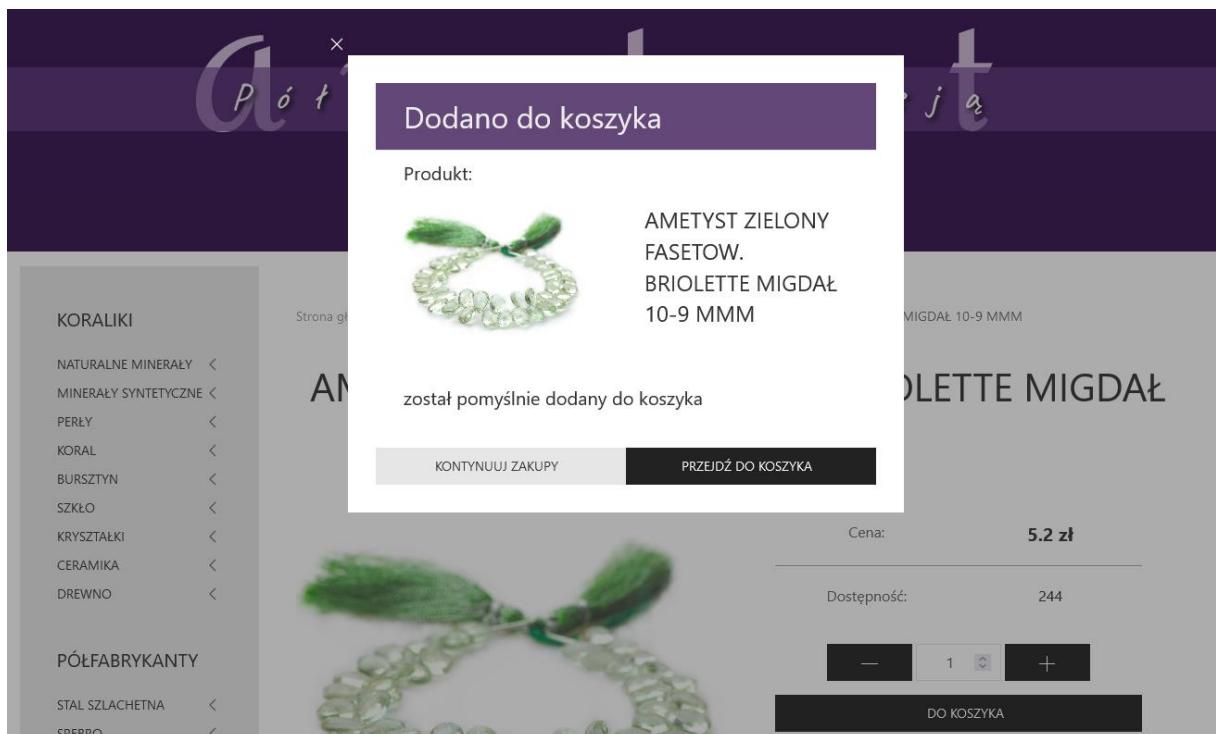
Wyświetlanie ich odbywa się za pośrednictwem konta użytkownika niezalogowanego, którego login ("jdbcUserNL"), i hasło ("jdbcUserNLPassw") są zapisane w pliku *web.xml*. Następnie do listy (*listProdIndex*) pobrane zostają dane według metody z pliku *ProductMDAO.java*. Ta metoda pobiera za pomocą języka SQL dziewięć losowych produktów, które następnie są wyświetlane w postaci kart na stronie głównej.

4.5. Interfejs użytkownika niezalogowanego

Użytkownik niezalogowany ma ograniczoną możliwość korzystania z usług sklepu internetowego do podstawowej jego funkcjonalności tj. składania zamówień. Może bez przeszkód przeglądać produkty, dodawać je do koszyka, a następnie na podstawie formularza dla użytkowników niezalogowanych złożyć jednorazowe zamówienie.

Frontend:

Rozpoczęcie kupna danej rzeczy następuje poprzez kliknięcie przycisku „DO KOSZYKA” w karcie produktu. Po tej czynności wyświetla się komunikat o dodaniu artykułu do części zamówienia z dwoma przyciskami umożliwiającymi pozostanie na stronie oraz przejście do odpowiedniej karty.



Rysunek 4.19. Komunikat po dodaniu produktu do koszyka

Do koszyka prowadzi również przycisk widniejący na pasku nawigacyjnym. Przez najechanie na ikonę koszyka można podglądać znajdujące się tam przedmioty, ich ilość oraz wartość koszyka. Klikając na ikonę przechodzi się do nowej strony, gdzie widnieje pełna lista produktów wyłącznie z wyborem dostawy i płatności. Możliwe jest usuwanie produktu, edycja ich ilości oraz opróżnienie całego koszyka. W tym miejscu można również dodać kod promocyjny, który odejmuję odpowiednią wartość procentową od zamówienia jeśli kod jest aktywny.

Krok 1/4
Akceptacja koszyka

Krok 2/4
Możliwość zalogowania

Krok 3/4
Dane osobowe

Krok 4/4
Podsumowanie

PRODUKT	ILOŚĆ	CENA	WARTOŚĆ	USUŃ
 AMETYST ZIELONY FASETOW. BRIALETTE MIGDAŁ 10-9 MMM	<input type="button" value="—"/> <input type="text" value="3"/> <input type="button" value="▼"/> <input type="button" value="+"/> <input checked="" type="button" value="✓"/>	5,20 zł	15,60 zł	

 Opróżnij koszyk

Suma: **15,6 zł** 7,80 zł

Dostawa: 8,00 zł

Rabat: 50 %

Łączna wartość zakupów: **15,80 zł**



PŁATNOŚĆ:

- + Platforma płatnicza:
- + Karta płatnicza:
- Inne:

- Przelew na rachunek bankowy;
- Za pobraniem;
- Gotówka na miejscu;
- Przedpłata na konto;

DOSTAWA:

- Poczta Polska - list polecony (8,00 zł)
- InPost - paczkomaty (9,50 zł)
- InPost - przesyłka kurierska (14,90 zł)
- Kurier DPD - przesyłka kurierska (14,00 zł)
- Kurier DPD - przesyłka za pobraniem (18,00 zł)
- Poczta Polska (zagraniczna) (26,00 zł)
- Kurier DPD (zagraniczna) - Europa środkowa (40,00 zł)
- Kurier DPD (zagraniczna) - Europa Zachodnia (60,00 zł)
- Kurier DPD (zagraniczna) - Afryka (70,00 zł)
- Kurier DPD (zagraniczna) (120,00 zł)
- Odbiór osobisty w siedzibie firmy (0,00 zł)
- FedEx (30,20 zł)

 DALEJ...

Rysunek 4.20. Widoku koszyka

Po zatwierdzeniu zamówienia przechodzi się na stronę z możliwością logowania, rejestracji lub kontynuowania jako użytkownik niezalogowany. Ze względu na tytuł sekcji osoba kontynuuje, jako gość („BEZ KONTA”).

Krok 1/4
Akceptacja koszyka

Krok 2/4
Możliwość zalogowania

Krok 3/4
Dane osobowe

Krok 4/4
Podsumowanie

Zaloguj się

Email *

example@example.com

Hasek *

ANULUJ

 ZALOGUJ

Masz też inne opcje:

Nie restrowałę się?

Nie robisz u nas zakupów?

 ZAREJESTRUJ SIĘ

LUB

Zrób zakupy jednorazowe:

 BEZ KONTA

Rysunek 4.21. Widok wyboru kontynuowania zakupu

Przycisk ten przekierowuje do strony z podstawowymi danymi osobowymi oraz adresem prywatnym z opcją zakupu na firmę. W tym przypadku należy zaznaczyć odpowiednie pole wyboru na górze strony i wypełnić dodatkowe dane. Jak widać na rysunku 4.22, formularz posiada validację.

Krok 1/4 Akceptacja koszyka	Krok 2/4 Możliwość zalogowania	Krok 3/4 Dane osobowe	Krok 4/4 Podsumowanie
--------------------------------	-----------------------------------	--------------------------	--------------------------

Dane osobowe

Firma

Imię *

Nazwisko *

Telefon *

Email *

Adres *

Nazwa firmy *

Nip *

Telefon firmy *

Email firmy*

Rysunek 4.22. Formularz danych użytkownika niezalogowanego

Po wypełnieniu tych danych i zaakceptowaniu („PRZEJDŹ DALEJ...”) aplikacja przekieruje użytkownika do strony akceptacji zamówienia. Widok ten posiada listę produktów bez możliwości zmiany. Może jednak za pomocą odpowiedniego przycisku wrócić do kroku pierwszego. Na tym etapie osoba może się jeszcze wycofać z zakupu lub zmienić adres wpisany w poprzednim kroku. Produkty będą czekać w koszyku do wygaśnięcia ciasteczka. Wymagane jest zaakceptowanie regulaminu sklepu, aby złożyć zamówienie. Gdy to się stanie nastąpi przekierowanie do strony wyświetlenia pełnego zamówienia wraz ze statusem realizacji.

Taki użytkownik może również w każdej chwili sprawdzić status poprzez odpowiedni formularz znajdujący się w stopce strony pod nazwą „Czytaj zamówienie”. Należy w nim wpisać imię, nazwisko, adres e-mail, oraz identyfikator jaki został wyświetlony po akceptacji. Poprawne dane pozwolą na wyświetlenie widoku identycznego z tym, co po złożeniu zamówienia wraz z odpowiednim, zaktualizowanym statusem.

Wypełnij formularz aby wyświetlić zamówienie:

Bożena	Wysocka
wysocb@gmail.com	65
POTWIERDŹ	

Rysunek 4.23. Formularz sprawdzania zamówienia przez użytkownika niezalogowanego

W tym miejscu można również dokonać wystawienia recenzji produktów przypisanych do zamówienia. Jest to bardzo prosta metoda oparta na ilości przyznanych gwiazdek, tytuły oraz treści. Po jej dodaniu dla np. dwóch przedmiotów z pięciu, przycisk nadal będzie aktywny, a po jego kliknięciu pojawią się tylko produkty niezrecenzowane. Zniknie dopiero, gdy użytkownik dokona recenzji wszystkich artykułów w zamówieniu.

Recenzje w wybranym zamówieniu:

– 1  AMETYST ZIELONY FASETOW. BRIOLETTE MIGDAŁ 10-9 MMM
Link do produktu
★ ★ ★ ★ ★
Piękny okaz
Nic dodać nic ująć. Jakość wyśmienita, kolor delikatny. Taki jak chciałem. Polecam
POTWIERDŹ

Rysunek 4.24. Formularz recenzji produktu

Backend:

W momencie dodania produktu do koszyka przedmiot zostaje zapisany do tabeli **basket** w bazie danych na podstawie klucza sesji, który jest ustalony w ciasteczku na określony czas. Są to unikalne informacje użytkownika niezalogowanego oparte na jego sesji. Po pobraniu odpowiednich zmiennych następuje sprawdzenie, czy osoba jest zalogowana. Jeśli tak, następuje uaktualnienie koszyka, w przeciwnym razie produkt zostaje dopisany do użytkownika sesjnego, inaczej nazywanym niezalogowanym. Następuje sprawdzenie,

czy artykuł istnieje już w koszyku, jeśli jest to prawda, zostaje dodana przesłana ilość, w przeciwnym razie rozpoczyna się proces tworzenia nowego produktu w koszyku z identyfikatorem sesji. W kolejnym kroku program przekieruje użytkownika do strony poprzedniej, gdzie wyświetlany jest odpowiedni komunikat o dodaniu przedmiotu.

```

117     //dodawanie produktu do koszyka + dodawanie sesji jeśli nie istnieje
118     private void insertBasket(HttpServletRequest request, HttpServletResponse response)
119             throws SQLException, ServletException, IOException {
120
121         sess = request.getSession();
122
123     //sprawdzanie czy wszystkie wymagane dane zostały przesłane
124     //pobranie sesji i daty jej wygaśnięcia
125     //pobranie daty wygaśnięcia (przybliżony czas) musi być osobno!
126     //sprawdzanie czy użytkownik jest zalogowany
127     //dodawanie/uaktualnianie stanu produktu do koszyka
128
129         boolean acc = false;
130     if((id_usr!=0)&&(id_usr!=NULL)){
131     }
132     else{
133         Basket bask = new Basket(session, NULL, id_prod, quant, exp);
134         boolean exist = baskDAO.checkSess(bask);
135         if(exist==true){
136             boolean q = baskDAO.updateBaskSess(bask);
137             if(q){ acc = true; }
138             else{ sess.setAttribute("err_disc","Błąd polecenia."); }
139         }
140         else{
141             boolean q = baskDAO.create(bask);
142             if(q){ acc = true; }
143             else{ sess.setAttribute("err_disc","Błąd polecenia."); }
144         }
145     }
146     sess.setAttribute("acc", acc);
147
148     //przekierowanie do poprzedniej strony
149     String url = request.getHeader("referer");
150     response.sendRedirect(url);
151 }
```

Rysunek 4.25. Metoda dodawania produktów do koszyka

Za pomocą odpowiednich metod koszyk jest usuwany (`deleteBasket(request, response)`) opróżniany (`deleteAllBasket(request, response)`), aktualizowany (`updateBasket(request, response)`) oraz dodawany jest kod promocyjny (`checkDisc(request, response)`). Przy każdym kroku sprawdzane są zmienne sesyjne dodane w poprzednich miejscach, aby nikt niepożądany nie dostał się do nich bez wcześniejszej akceptacji.

Podczas dodawania adresu odpowiednia metoda (`usermDAO.read(userm)`) sprawdza, czy istnieje on w bazie danych. Jeśli tak, to pobiera jego identyfikator oraz przesłane informacje. Zapobiega to namnażaniu rekordów o takich samych wartościach.

```

627     int id = usermDAO.read(userm);
628     if(id==NULL){
629         if(firm_m){
630             boolean q = usermDAO.createFirmN(userm);
631             if(q){ sess.setAttribute("succ_disc","Dane zostały zapisane.");}
632             else{ sess.setAttribute("err_disc","Błąd polecenia.");}
633         }
634         else{
635             boolean q = usermDAO.createAdrN(userm);
636             if(q){ sess.setAttribute("succ_disc","Dane zostały zapisane.");}
637             else{ sess.setAttribute("err_disc","Błąd polecenia.");}
638         }
639         id = usermDAO.read(userm);
640     }
641     sess.setAttribute("userMeta_id",id);

```

Rysunek 4.26. Sprawdzanie oraz tworzenie adresu użytkownika niezalogowanego

Podczas dodawania zamówienia do bazy danych sprawdzane są wszystkie zmienne sesyjne zapisane do tej pory. Na ich podstawie będzie rezerwacja produktów (status „Poczekalnia”). Następuje obliczanie sumy bez kodu promocyjnego sprawdzanego w kolejnym kroku. Po tych czynnościach tworzony jest konstruktor dla tabeli **order-o** z pierwszym statusem w bazie. Jeśli wystąpi błąd, aplikacja odeśle użytkownika do wcześniejszej strony z odpowiednią informacją, a następne kroki metody nie będą wykonywane. Jeśli jednak dodanie się powiedzie, odczytany zostanie identyfikator zlecenia a wszystkie przedmioty z encji **basket** zostaną przesłane do stałej tabeli łącznikowej **order-product**.

Czytanie zamówienia oraz widok po akceptacji obsługuje ta sama metoda. Sprawdza ona zmienne: sesyjne zapisane przez wcześniejsze kroki oraz te przesłane metodą GET. Na tej podstawie aplikacja wyświetli zamówienie dla zarówno strony po akceptacji oraz formularza. Brak przesłania odpowiednich danych skutkować będzie powrotem do wcześniejszej strony z komunikatem wystąpienia błędu.

Recenzje obsługują dwie metody w pliku *ControlProduct.java*. Wyświetlanie (*showRew(request, response)*) odbywa się po odpowiednim identyfikatorze, natomiast dodawanie (*addRew(request, response)*) jest sprawdzane po ilości produktów. Ta opcja zostaje wykonana tylko dla wypełnionych danych, dzięki czemu wiadomo, które produkty zostały zrecenzowane.

4.6. Interfejs użytkownika zalogowanego

Użytkownik zalogowany w odróżnieniu od niezalogowanego ma do dyspozycji kilka udogodnień w postaci zapisywania wszystkich zamówień bez konieczności sprawdzania ich pojedynczo w formularzu opisanym wcześniej. Posiada również możliwość edycji danych osobowych, adresu, zmiany hasła oraz prześledzenia historii zdarzeń, dzięki której może ponawiać i cofać konkretne modyfikacje.

Frontend:

W celu zalogowania użytkownik musi kliknąć ikonę logowania i wpisać prawidłowy e-mail oraz hasło pasujące do znajdującego się w bazie danych. Jeżeli nie jest on jeszcze zalogowany, może się zarejestrować poprzez formularz rejestracyjny, przypominający dodawanie adresu przez użytkownika niezalogowanego, potwierdzając hasło (walidacja w czasie rzeczywistym z możliwością podejrzenia wartości). Zalogować można się również podczas składania zamówienia w kroku drugim (rysunek 4.21).

Po poprawnym logowaniu ląduje on na swojej stronie głównej, gdzie znajdują się zamówienia na różnym stopniu realizacji. Po rozwinięciu zamówienia pokazują się szczegółowe informacje identyczne jak przy czytaniu jednego zamówienia.

Witaj ponownie Elżbieta Oluś!

Twoje zamówienia:

+ Zamówienie nr. 64

Rysunek 4.27. Skrócony widok strony głównej użytkownika zalogowanego

Użytkownik ma możliwość zapisu koszyka do własnego identyfikatora dzięki czemu jego termin ważności nie wygasza.

Jednym z podstawowych funkcjonalności jest możliwość zmiany hasła, danych osobowych, czy też adresu prywatnego lub firmy.

W przypadku zmiany hasła należy podać teraźniejsze, zapisane w bazie danych, następnie wpisać nowe i potwierdzić je w kolejnym polu. Ikona w środku pól hasła odkrywa je dla sprawdzenia znaków.

Zmiana danych osobowych tyczy się imienia, nazwiska, numeru telefonu oraz akceptacji newslettera. Adresu e-mail nie można zmienić, ponieważ jest traktowany, jako unikalny, zapasowy klucz główny z poziomu bazy.

Następnymi zakładkami są edycja oraz dodanie adresu prywatnego lub firmowego. W każdym momencie można zmienić dane, które się tam znajdują. Za pomocą historii zdarzeń można usunąć adres, ale tylko w wypadku, gdy nie był on wykorzystany w żadnym istniejącym zamówieniu.

Zmień co chcesz:

– Hasło:

Poprzednie hasło *

••••••••

Nowe hasło *

HasłoToMasło

Potwierdź hasło *

••••••••••••

ANULUJ ZMIEN

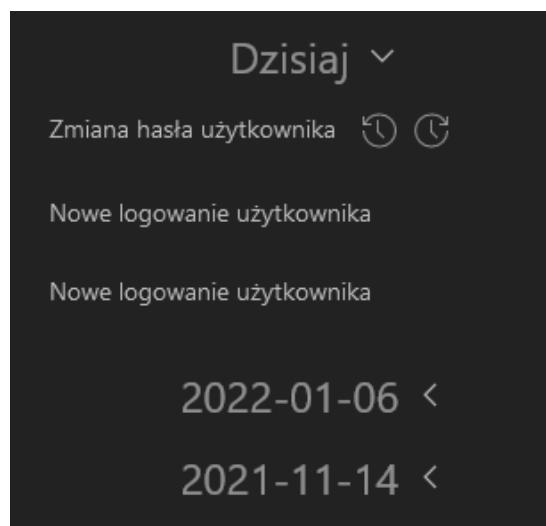
+ Dane osobowe:

+ Adres 1:

+ Dodaj nowy adres:

Rysunek 4.28. Widok ustawień użytkownika zalogowanego

Historia użytkownika znajduje się na pasku nawigacji przed ikoną wyszukiwania. Po kliknięciu wysuwa się boczny panel z możliwościami cofania oraz ponawiania operacji, które zostały wykonane w konkretnym czasie.



Rysunek 4.29. Historia użytkownika zalogowanego

Backend:

Logowanie rozpoczyna się pobraniem danych loginu i hasła z formularza. Następnie zaciągane są dane z bazy po loginie, czyli po adresie e-mail. Na podstawie tego obiektu algorytm porównuje hasła. Jeśli się zgadzają, następuje zmiana daty ostatniego logowania i ustawienie odpowiedniej historii zdarzeń oraz podstawowych danych do sesji m.in. identyfikator użytkownika, adresu, imię i nazwisko.

```
135     Login log = loginDAO.read(login);
136     if((log!=null)){
137         if(log.getPassw().equals(passw)){
138             //aktualnianie daty logowania
139             String lastLog = loginDAO.getLastLog(login);
140             String last_log = formatLocalDate.format(LocalDateTime.now());
141             Login logging = new Login(log.getId(), login, passw, last_log);
142             boolean q = loginDAO.updateLastLog(logging);
143             if(q){ sess.setAttribute("succ_disc","Logowanie przebiegło pomyślnie.");}
144             else{
145                 sess.setAttribute("err_disc","Błąd logowania.");
146                 String url = request.getHeader("referer");
147                 response.sendRedirect(url);
148                 return;
149             }
150         }
151     }
```

Rysunek 4.30. Fragment kodu logowania użytkownika

Podczas rejestracji użytkownika następuje sprawdzenie, czy osoba o tym samym adresie e-mail występuje w bazie danych. W przeciwnym wypadku tworzony jest rekord w tabeli **user** oraz w encji **login**. Po poprawnym dodaniu przypisana zostaje historia zdarzeń. Następnie aplikacja tworzy pierwszy adres przesyłany przez ten sam formularz.

```
210     //tworzenie użytkownika
211     String reg = formatLocalDate.format(LocalDateTime.now());
212     User user = new User(0, true, false, NULL, name, surname, mail, tel, true, reg);
213     boolean q1 = userDAO.create(user);
214     if(q1){ sess.setAttribute("succ_disc","Pomyślnie dodano użytkownika.");}
215     else{
216         sess.setAttribute("err_disc","Wystąpił błąd podczas dodawania użytkownika.");
217         String url = request.getHeader("referer");
218         response.sendRedirect(url);
219         return;
220     }
221     id_usr = userDAO.readId(mail);
222     //tworzenie loginu
223     Login log = new Login(id_usr, mail, passw, reg);
224     boolean q2 = loginDAO.create(log);
225     if(q2){ sess.setAttribute("succ_disc","Pomyślnie dodano login.");}
226     else{
227 }
```

Rysunek 4.31. Fragment metody rejestracji użytkownika

Strona główna polega na wyświetleniu zamówień przypisanych do użytkownika po adresach. Aby to nastąpiło, tworzone są listy, do których pobierane zostają informacje dla odpowiedniego adresu.

Po sprawdzeniu, czy użytkownik jest zalogowany oraz pobraniu odpowiednich danych dla formularzy istnieje możliwość zmiany hasła (`updatePassw(request, response)`), danych osobowych (`updateInform(request, response)`), adresu prywatnego lub firmowego (`updateAddr(request, response)`) oraz dodanie nowego adresu (`insertAddr(request, response)`). Wszystkie z wyżej wymienionych bazują na operacjach CRUD przedstawionych w rozdziale 4.3.

Historia znajduje się w pliku `ControlSrv.java`. Metoda `history(request, response)` po wczytaniu identyfikatora oraz numeru akcji przechodzi do wyboru cofnięcia lub ponowienia zmiany. Cofnięcie odwraca kolejność zapytań SQL oraz ustawia odpowiednie dane. Ponowienie zostawia je tak jak były. Po poprawnym wykonaniu operacji następuje usunięcie lub nadpisanie zmiennych sesyjnych.

```

211     switch(action){
212         case 1: // = cofnięcie zmiany
213             //obsługa historii modyfikacji
214             sql = hist.getQuery_b();
215             beforeQuery = hist.getQuery_a();
216             afterQuery = hist.getQuery_b();
217             description = "Cofnięcie zmiany";    modify = hist.getModify();
218             his = new History(NULL, id_usr, id_act, description, now,
219                             beforeQuery, afterQuery, modify);
220             break;
221         case 2: // = ponowne wykonanie zmiany
222             //obsługa historii modyfikacji
223             sql = hist.getQuery_a();
224             id_act = 6;
225             beforeQuery = hist.getQuery_b();
226             afterQuery = hist.getQuery_a();
227             description = "Ponowne wykonanie zmiany";    modify = hist.getModify();
228             his = new History(NULL, id_usr, id_act, description, now,
229                             beforeQuery, afterQuery, modify);
230             break;
231     }
232
233     //Obsługa zapytań
234     histDAO.connect();
235     try{
236         PreparedStatement statement = histDAO.jdbcConnection.prepareStatement(sql);
237         q = statement.executeUpdate() > 0;
238     >     if(q){      //wymuszanie przeładowania sesji
239     }
240     else{ sess.setAttribute("err_disc","Wystąpił błąd podczas zmiany informacji");}
241     histDAO.disconnect();
242
243         q = histDAO.create(his);
244         if(q){sess.setAttribute("succ_disc","Pomyślnie wykonano zapis w historii."); }
245         else{ sess.setAttribute("err_disc","Wystąpił błąd podczas zmiany informacji.");}
246     }
247     catch (SQLException se){
248         sess=request.getSession(true);
249         sess.setAttribute("err_disc","Nie można wykonać operacji.");
250     }

```

Rysunek 4.32. Fragment kodu modyfikacji kolejności zapytań w historii zdarzeń

4.7. Interfejs użytkownika z rangą

W aplikacji istnieją cztery rodzaje użytkowników z rangą. Każdy z nich musi się zalogować. Po tej czynności aplikacja przekieruje do właściwej strony głównej odpowiadającej im randze. Osoby te posiadają również wszystkie opcje wymienione w poprzednich podrozdziałach.

Administrator

Sprawdza i koordynuje pracę reszty rang. Posiada możliwość modyfikowania niektórych tabel słownikowych; dodawania innych użytkowników z rangą w tym względ w ich historię zdarzeń; edycji menu z kategoriami i tagami.

Frontend:

Po zalogowaniu administrator ląduje na swoim pierwszym widoku, którym jest zarządzanie niestatycznymi tabelami słownikowymi. Są nimi encje: **color** (kolor), **shape** (kształt), **fabric** (materiał), **delivery** (dostawa), **payment** (płatność) oraz **discount** (kody promocyjne). W każdej takiej tabeli administrator ma możliwość przeprowadzania pełnych operacji CRUD. Po wejściu do każdej z zakładek w bocznym menu ukazuje się pełna lista wartości znajdujących się w bazie danych. Po najechaniu na ikony ołówka pokaże się opis „Edytuj”, oznacza to, że po naciśnięciu na tę ikonę pojawi się okienko modalne z odpowiednimi, wypełnionymi zmiennymi, które można modyfikować. Podobnie się stanie po kliknięciu przycisku plusa u góry ekranu. Za pomocą tego okienka modalnego użytkownik ma możliwość dodania nowej zmiennej. Ikona **×** usuwa zmienne.

The screenshot shows the Ametyst application's main administrator interface. On the left is a sidebar with navigation links: 'TABELY SŁOWNIKOWE' (with sub-links for Kolor, Kształt, Materiał, Dostawa, Płatność, Promocje), 'UŻYTKOWNICY', 'MENU', and 'HISTORIA'. Below these is a user status section: 'None None'. The main area features a search bar at the top right and a table below it. The table has columns: 'ID.', 'NAZWA', and 'AKCJE'. The 'NAZWA' column is sorted by name. The table contains 10 rows of color names: biały, żółty, pomarańczowy, czerwony, różowy, fioletowy, niebieski, granatowy, turkusowy, and zielony. Each row has edit ('Edytuj') and delete ('usuń') icons in the 'AKCJE' column. A '+' icon is located at the top right of the table area.

ID.	NAZWA	AKCJE
1.	biały	
2.	żółty	
3.	pomarańczowy	
4.	czerwony	
5.	różowy	
6.	fioletowy	
7.	niebieski	
8.	granatowy	
9.	turkusowy	
10.	zielony	

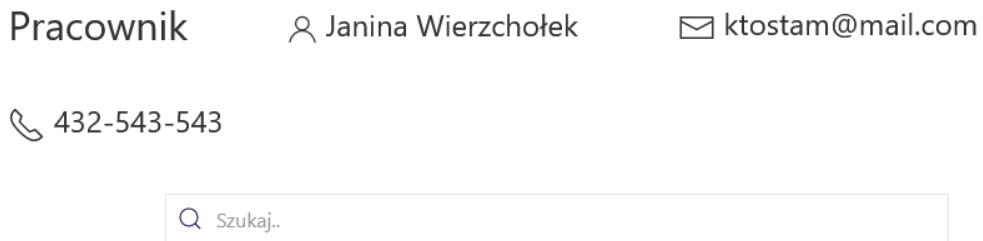
Rysunek 4.33. Widok strony głównej administratora

Następną zakładką jest widok listy użytkowników z rangą. Administrator posiada możliwość usunięcia użytkownika, który nie posiada jeszcze historii zdarzeń, tzn. nie zalogował się ani razu. Podczas tworzenia nowego konta osoba otrzymuje wiadomość e-mail na wpisany przez administratora adres z wiadomością o loginie, haśle oraz przyznanej randze. Otrzymuje również wiadomość informującą o usunięciu konta.

Każdy z identyfikatorów użytkowników prowadzi do jego historii zdarzeń. Dzięki takiemu zabiegowi administrator będzie wiedział kogo obwinić za konkretne błędy, ponieważ historii z poziomu aplikacji nie da się usunąć. Jedyna możliwość to usunięcie rekordu z panelu zarządzania bazą danych, do którego zwykli użytkownicy jak i zarówno administrator nie posiadają dostępu.

5	Pracownik	Wierzchołek Janina	ktostam@mail.com	432-543-543
6	Pracownik	Habdżibadło Dawid	habad@gmail.com	452-563-235 

Rysunek 4.34. Fragment widoku użytkowników ze strony administratora



Pracownik  Janina Wierzchołek  ktostam@mail.com

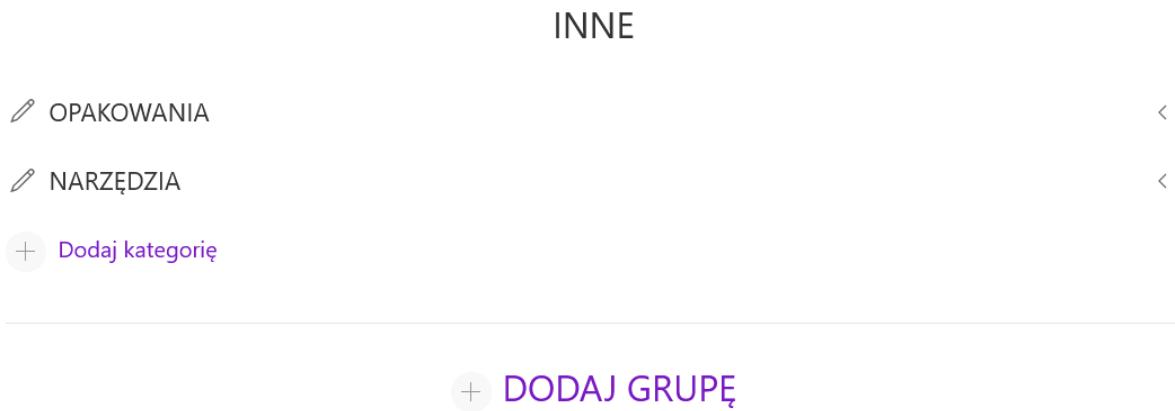
 432-543-543

 Szukaj..

ID. ↓ ↑	AKCJA ↓ ↑	OPIS	DATA ↓ ↑	MODYFIKACJA
315.	modyfikacja	Zmaina statusu	2021-10-29 19:04:36.0	
314.	modyfikacja	Zmaina statusu	2021-10-29 19:04:27.0	
313.	nowe logowanie	Nowe logowanie użytkownika	2021-10-29 19:03:40.0	
274.	nowe logowanie	Nowe logowanie użytkownika	2021-10-26 17:41:31.0	

Rysunek 4.35. Fragment historii zaopatrzeniowca od strony administratora

Ostatnią zakładką jest modyfikacja menu z kategoriami i tagami. Zmiany od strony użytkowników są widoczne dopiero po dodaniu tagów do kategorii. Podobnie jak w przypadku tabel słownikowych również tutaj występuje możliwość edycji oraz dodawania. Dodatkowo można wprowadzić nową grupę z jej pierwszą kategorią. Usuwanie odbywa się tylko i wyłącznie za pomocą historii w przypadku braku powiązanych produktów, gdzie najpierw usuwa się kategorię a następnie grupę.



Rysunek 4.36. Fragment strony menu administratora

Backend:

Modyfikacja tabel słownikowych jest dzielona przez dwóch użytkowników z rangą: administratora oraz zaopatrzeniowca. Posiadają oni możliwość:

- tworzenia (`insertDict(request, response)`);
- aktualizacji (`updateDict(request, response)`);
- usuwania (`deleteDict(request, response)`)

poszczególnych tabel im przypisanych.

Wszystkie z powyższych metod posiadają zabezpieczenia przed dostaniem się nieuprawnionych osób w postaci sprawdzania, czy posiada on odpowiednią rangę, a następnie sprawdzane są przesłane przez formularz dane. Jeśli ich brak, aplikacja powiadomi użytkownika o błędzie oraz przerwie wykonywanie metody.

Podczas dodawania odpowiednia tabela zostaje sprawdzona pod kątem istnienia podanej zmiennej. Obrywa się to przez odwołanie do bazy danych sprawdzającej jej istnienie po nazwie. Jeśli taki rekord istnieje, użytkownik zostaje powiadomiony o sytuacji, a zmienna nie jest dodawana do bazy. Taki sam zabieg został zastosowany przy aktualnianiu.

W przypadku usuwania pojawia się potrzeba sprawdzenia, czy usunięcie nie zagraża funkcjonowaniu całej aplikacji. Do tego celu wykorzystano `try{} catch(){}`, które informuje o wykrytych błędach przy próbie usunięcia danych.

Na koniec zmiany zostają zapisane w historii użytkownika oraz następuje przeładowanie zmiennych sesyjnych dla aktualnego widoku bazy.

```

3126     Color col = colDDAO.getName(id);
3127     try{
3128         q = colDDAO.delete(id);
3129         if(q){ sess.setAttribute("succ_disc","Pomyślnie usunięto kolor.");}
3130         else{
3131             sess.setAttribute("err_disc","Wystąpił błąd podczas usuwania koloru.");
3132             String url = request.getHeader("referer");
3133             response.sendRedirect(url);
3134             return;
3135         }
3136     }
3137     catch (SQLException | NullPointerException ex){
3138         sess.setAttribute("err_disc","Nie można usunąć koloru! Jest on używany przez produkty.");
3139         if(rnk.getId_rank()==1){ response.sendRedirect("administrator?t=1");}
3140         if(rnk.getId_rank()==4){ response.sendRedirect("supplier?t=1");}
3141         return;
3142     }
3143 }
3144 > //ustawianie historii modyfikacji
3153 > //aktualnienie zmiennej sesyjnej

```

Rysunek 4.37. Fragment usuwania rekordu z tabeli color

Podstawowy widok zakładki użytkownika obsługuje ścieżka administrator?t=2. Ta zakładka gwarantuje filtrowanie jedynie użytkowników z rangą. Administrator nie ma wglądu do osób zwykłych, opisanych w podrozdziale 4.6.

Widok jednej rangi natomiast jest przetwarzany i przygotowywany przez metodę viewUsersHistory(request, response). Pobierane są w niej odpowiednie zmienne dla wybranego użytkownika widoczne na rysunku 4.34. Następnie pobrana zostaje pełna lista historii modyfikacji przypisana do wybranej osoby.

Dodawanie odbywa się za pomocą metody addUserAdmin(request, response), która w przypadku użytkownika z rangą jest sprawdzana pod kątem odpowiedniej osoby przesyłającej żądanie. Po sprawdzeniu przesłanych danych użytkownik oraz jego hasło jest tworzone. Za pomocą odpowiedniej klasy (*SendMail.java*) wysyłana jest wspomniana wcześniej wiadomość. Wszystkie dane zostają zapisane do historii zdarzeń. Administrator może również usunąć użytkownika za pomocą metody deleteUserAdmin(request, response), gdzie również wysyłana jest wiadomość.

```

532     //wysłanie emaila powiadamiającego
533     String to = mailu;
534     String subject = "Dane logowania do wytryny amethyst";
535     String message = "<h2>Administrator aplikacji właśnie założył Pani/Panu "
536     + "konto z rangą: <i>" + rankDAO.getName(id_rank) + "</i></h2> "
537     + "<h3>Poniżej znajdują się dane logowania:</h3>"
538     + "<h2>Login: <b>" + mailu + "</b></h2>"
539     + "<h2>Hasło: <b>" + pass + "</b></h2>"
540     + "<h4>Życzymy udanego korzystania z aplikacji</h4>";
541     SendMail.send(to, subject, message);

```

Rysunek 4.38. Fragment wysyłania wiadomości e-mail o założeniu konta pracowniczego

Ostatnia zakładka funkcjonuje podobnie do wcześniej opisanych tabel słownikowych.
Dodawanie:

- kategorii (addCategoryAdmin);
- grupy (addGroupAdmin (request, response))

odbywa się jednakowo ze zmianą odniesienia. Sprawdzana jest osoba, zmienne oraz wykonywane jest żądanie dodania. Następnie tworzona jest historia zdarzeń, a na końcu użytkownik zostaje przekierowany do odpowiedniej strony. Jedyna różnica występuje w dodawaniu tagu (addTagAdmin (request, response)). Ta metoda obsługuje zarówno tworzenie pojedynczego tagu jak i zespół istniejących. W tym celu sprawdzana zostaje przesłana tablica. Każdy element z osobna jest przypisany do wybranej kategorii.

Aktualizacja odbywa się podobnie do podstawowej z różnicą żądania. Modyfikować można tylko tagi oraz kategorie.

```
701 //sprawdzanie tagów
702 String[] tags = null;
703 try{ tags = request.getParameterValues("multi_tags"); }
704 catch (NumberFormatException | NullPointerException ex){ }
705 CatTag CT = null; CatTag CTnames = null;
706 if(tags!=null){
707     modify="nazwa: ";
708     for (String tag1 : tags) {
709         //dodawanie istniejących tagów
710         CT = new CatTag(0, id, Integer.parseInt(tag1));
711         q = ctADAO.createCatTag(CT);
712         if(q){ sess.setAttribute("succ_disc","Pomyślnie dodano tagi.");}
713     >     else{ //w przeciwnym wypadku...
714     }
715
716     CTnames = ctADAO.readTag(Integer.parseInt(tag1));
717     int id_cattag = ctADAO.getIdCatTag(id, Integer.parseInt(tag1));
718
719
720 >     //ustawianie wstępnej historii modyfikacji
721     }
722
723 > }
724 }
```

Rysunek 4.39. Fragment dodawania wielu tagów do kategorii

Pracownik

Posiada możliwość zmiany statusu, numeru przesyłki oraz ustala datę modyfikacji i wysłania zamówienia. Jego panel główny jest oparty na statusie.

Frontend:

Jak już zostało powiedziane, panel główny pracownika opiera się na dziewięciu statusach. Pierwszy widzą wszyscy pracownicy. Dzięki temu każdy może wybrać, nad którym będzie w tej chwili pracował. Następne są wyświetlane po identyfikatorze pracownika, które prowadzi. Zapobiega to nakładaniu się zamówień oraz pomyłek.

The screenshot shows the main worker page of the Ametyst software. At the top right is a search bar with the placeholder "Szukaj..". On the left, there's a sidebar with a logo and navigation links: "ZAMÓWIENIA" (with sub-links: Poczekalnia, Akceptacja, Płatność, Przygotowanie, W realizacji, Wyśiano, W drodze, Zakończono, Anulowano), "HISTORIA" (with sub-link: Karol Worecki), and a user profile icon.

The main content area displays a table of orders:

ID.	NAZWISKO I IMIĘ	ILOŚĆ PRODUKTÓW	DATA UTWORZENIA	DATA AKTUALIZACJI
53	Pozorny Sylwia	4	2021-10-26 15:00:55.0	
56	Florianski Jacek	3	2021-11-01 20:25:20.0	
59	Radzikowski Leonard	6	2021-11-02 13:18:16.0	
60	Florianski Jacek	3	2021-11-02 14:37:18.0	
61	Grudka Ludwik	11	2021-12-30 14:09:39.0	
62	Kowalski Jan	8	2021-12-30 14:46:58.0	
63	Wielmożna Karolina	2	2022-01-06 12:24:45.0	

Rysunek 4.40. Strona główna pracownika

Edycję statusu można przeprowadzić poprzez stronę widoku zamówienia. Znajdują się tam podstawowe dane z tabeli **order-o**. Zmiany dokonuje się za pomocą przycisku „EDYTUJ”. Po pojawieniu się okienka modalnego pracownik może zmieniać status oraz przypisywać numer przewozu przesyłki.

Zamówienie nr 48

Status:	Płatność:	Dostawa:								
Poczekalnia Zamówienie czeka na akceptację	Karta płatnicza: MasterCard Suma: 81.2 zł	Kurier DPD - przesyłka kurierska								
Dane osobowe: Ludmiła Gardziołek tel: 214-624-437 mail: cokolwiek1@jgyr.pl		Adres: ul. Katolickich Świętych 209 56-400 Oleśnica Dolnośląskie, Polska	Firma: Wywrotowcy NIP. 4121434235 mail: Ludnila@wywrot.pl tel: 432-523-523	EDYTUJ						
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 5px;">LP.</th> <th style="text-align: left; padding: 5px;">PRODUKT</th> <th style="text-align: left; padding: 5px;">ILOŚĆ</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">1.</td> <td style="padding: 5px; text-align: center;">  AGAT FUKSJA FASETOWANE KULE 6 MM - SZNUR </td> <td style="padding: 5px; text-align: center;">4</td> </tr> </tbody> </table>					LP.	PRODUKT	ILOŚĆ	1.	 AGAT FUKSJA FASETOWANE KULE 6 MM - SZNUR	4
LP.	PRODUKT	ILOŚĆ								
1.	 AGAT FUKSJA FASETOWANE KULE 6 MM - SZNUR	4								

Komentarz: brak komentarza

Rysunek 4.41. Strona widoku zamówienia z poziomu pracownika

Backend:

Pracownik posiada tylko trzy przypisane mu strony: "/worker" (strona główna), "/viewOrder" (widok zamówienia) oraz "/modifyOrder" (edytacja zamówienia).

```
1130 //sprawdzenie i przypisanie zmiennych sesyjnych (jeśli brak)
1131 orders = (List<Order>) sess.getAttribute("listOrder");
1132 if(orders==null){
1133     //wypisanie zamówień
1134     List<Order> ord = ordWDAO.readWorker(id_usr);
1135     for (Order order : ord) {
1136         ||| orde.add(order);
1137     }
1138     List<OrderProd> ordp= new ArrayList<>();
1139     int sum = 0;
1140     for (Order order : orde) {
1141         sum=0;
1142         ordp= ordpDAO.read(order.getId());
1143         for (OrderProd orderp : ordp) {
1144             ||| sum += orderp.getQuantity();
1145         }
1146         quant.add(sum);
1147     }
1148     usr = usermDAO.readForWorker();
1149     //przypisanie zmiennych sesyjnych
1150     sess.setAttribute("listOrder", orde);
1151     sess.setAttribute("listUser", usr);
1152     sess.setAttribute("listQuantity", quant);
1153 }
```

Rysunek 4.42. Przypisanie zmiennych sesyjnych na stronie głównej pracownika

Zasada działania wstępnego jest taka sama jak u administratora. Po sprawdzeniu zostaje wykonana podstawowa funkcjonalność dla określonej metody. W przypadku strony głównej jest to pobranie odpowiednich zmiennych do sesji.

Strona widoku, jak sama nazwa wskazuje, powinna wyświetlać odpowiednie dane. W tym przypadku również pobierane są zmienne jednorazowe niezakłócające pracy sesji widoczne na rysunku 4.41.

Najważniejsza metoda (`updateViewWorker(request, response)`) pozwala na edycje danych. Na początku następuje sprawdzenie, czy status jest większy lub równy od żądanego. Jeżeli ten warunek zostanie spełniony, sprawdzone zostają następne kroki. Tylko w przypadku statusu równego 2 (Akceptacja) stan magazynowy produktów zostaje uszczuplony o znajdujące się w zamówieniu przedmioty. Zmiany numeru zamówienia można dokonać dopiero od statusu nr 6 (Wysłano), w którym to ustawiana jest data wysłania. Jeśli są to inne identyfikatory statusu, zostają zaktualizowane z odpowiednią datą modyfikacji bez żadnych dodatkowych opcji.

```

1281 Order order = ordWDAO.read(id);
1282 if((order.getIdStat()<=stat)){
1283     String del_nr = null;
1284     if(stat>=6){ del_nr = Jsoup.parse(request.getParameter("delivery_nr")).text(); }
1285     String update = formatLocalDate.format(LocalDateTime.now());
1286     String send = null;
1287     if(stat==6){ send = formatLocalDate.format(LocalDateTime.now()); }
1288     if(stat==2){
1289         List<OrderProd> orderp = ordpDAO.read(id);
1290         for (OrderProd ordp : orderp) {
1291             prodmWDAO.updateQuant(ordp.getIdProd(), ordp.getQuantity());
1292         }
1293     }
1294     Order ord = new Order(id, del_nr, stat, update, send, id_usr);
1295     q = ordWDAO.updateForWorker(ord);
1296     if(q){ sess.setAttribute("succ_disc","Pomyślnie edytowano zamówienie.");}
1297     else{
1298         sess.setAttribute("err_disc","Wystąpił błąd podczas edycji zamówienia.");
1299         String url = request.getHeader("referer");
1300         response.sendRedirect(url);
1301     }
1302 }

```

Rysunek 4.43. Fragment aktualizacji statusu dla pracownika

Korektor

Kontroluje błędy popełnione w aplikacji. Jego podstawową funkcją jest odpowiedź na negatywne opinie. Posiada również możliwość dodania opisu kategorii oraz tagu jak i zmiany opisu produktu w przypadku wykrycia błędu.

Frontend:

Strona główna korektora to lista wystawionych recenzji z datami, ilością gwiazdek oraz odnośnikiem do produktu. Aby dokonać zmiany, należy kliknąć identyfikator recenzji. Pojawi się pełny zapis recenzji, do której może się odnieść w komentarzu zwrotnym. Należy wpisać odpowiedni tekst w okienku modalnym i zatwierdzić go przyciskiem. Odpowiedź pojawi się po powrocie i odświeżeniu strony. Za pomocą tego samego okienka można ją poprawić.

Recenzja nr 6

Osoba wystawiająca:

Wiesław Jastrząb

Data wystawienia recenzji:

2021-10-23 18:50:42.0

Gwiazdki:



Recenzja:

Nic dodać nic ująć

Dotyczy produktu:



AMONIT - 31 MM / 22 MM

Odpowiedź z dnia 2021-10-29 19:05:52.0

Dziękujemy bardzo

EDYTUJ ODPOWIEDŹ

Rysunek 4.44. Strona wystawiania odpowiedzi na recenzje użytkowników

W celu poprawy opisu produktu należy się udać do odpowiedniej zakładki i wybrać dany przedmiot do odczytu. Zostanie on wyświetlony w następnej karcie, jak każdy odnośnik dla wygody pracownika. Następnie wystarczy przez kliknięcie przycisku edycji poprawić błędne zdanie i zatwierdzić. W taki sam sposób można dodać oraz edytować opis tagu i kategorii.

Produkt nr 7



OPIS PRODUKTU

Ametystowe naturalne kule.

EDYTUJ OPIS PRODUKTU

Nazwa przedmiotu:

AMETYST KULE 4 MM - 10 SZT

Dane produktu

Materiał: Ametyst
Wielkość otworu: 0.9 mm
Kształt: kula
Średnica: 4.0 mm
Kolor: fioletowy
Ilość: 10
Źródło: <https://www.kianit.pl/amethyst-kule-4-mm-10-szt-a--id-269>

Rysunek 4.45. Widok produktu od strony korektora

Backend:

Tak jak przy innych rangach korektor posiada swoją stronę główną, która po sprawdzeniu użytkownika pobiera odpowiednie dane.

Widok korektora opiera się na czterech możliwościach. Wyboru recenzji, produktu, kategorii oraz tagu. W zależności, którą zakładkę wybierze użytkownik metoda `viewForCorrector(request, response)` przekieruje go do żądanej z pobraniem odpowiednich danych.

Podobnie dzieje się w metodzie `updateViewCorrector(request, response)`, gdzie aktualizacja jest oparta na właściwej zakładce. Dzieje się to za pomocą metod odnoszących się do konkretnych tabel.

```
1583 //aktualnienie recenzji
1584 if(!reply.isEmpty()){
1585     String create = formatLocalDate.format(LocalDateTime.now());
1586     Review revi = rewDAO.readOne(id);
1587     Review rev = new Review(id, reply, create);
1588     q = rewDAO.updateForCensor(rev);
1589     if(q){ sess.setAttribute("succ_disc","Pomyślnie ustawiono odpowiedź na recenzję.");}
1590 > else{
1595 }
```

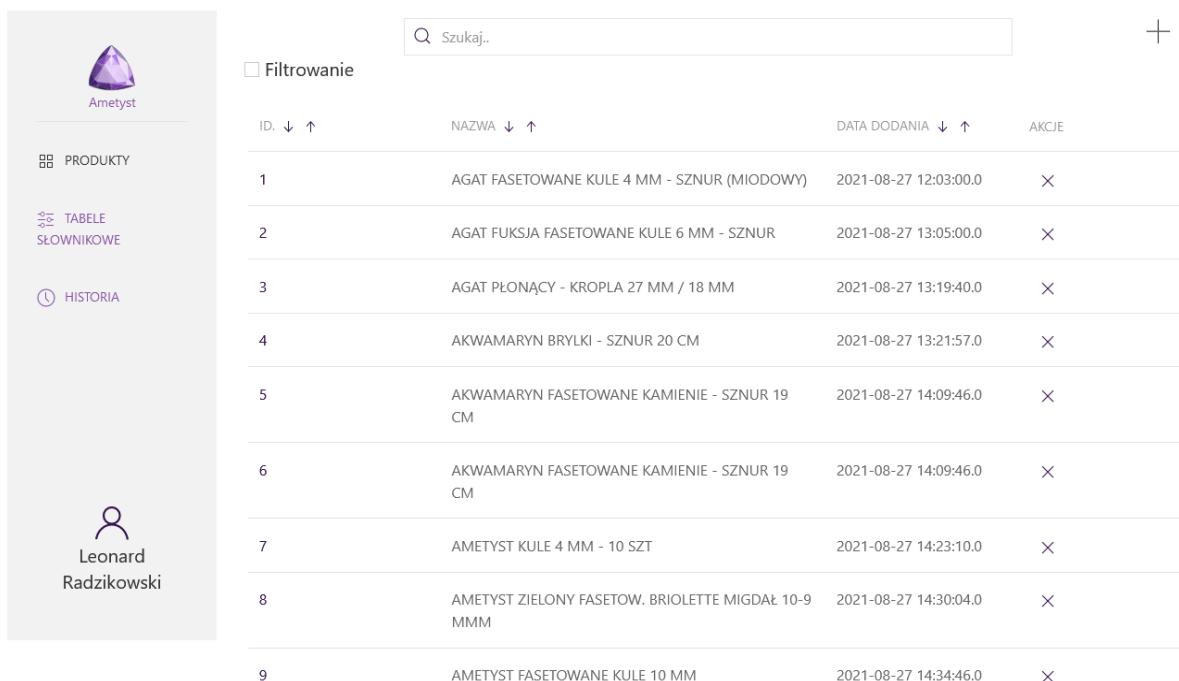
Rysunek 4.46. Uaktualnianie odpowiedzi na recenzje dla korektora

Zaopatrzeniowiec

Jak sama nazwa mówi, zajmuje się zaopatrzeniem. Ma wgląd w pełną edycję produktów: ich dodawanie, modyfikacje, usuwanie (ustawienie stanu na zero lub pełne usunięcie za pomocą historii zdarzeń jeśli nie jest już używany), dodawanie obrazów oraz odnawianie ilości na stanie magazynu. Posiada również CRUD tabel słownikowych przypisanych do produktu, opisanych przy omawianiu tabel słownikowych administratora.

Frontend:

Strona główna wyświetla listę produktów z wybranej grupy. Wygląd listy opiera się na identyfikatorze, nazwie, dacie dodania oraz akcjach dodatkowych. Filtrowanie przebiega po wybranej grupie, kategorii, tagu, polu wyszukiwania lub po zmiennych widniejących na górze tabeli. W tym widoku można usuwać produkty, dodawać nowe oraz odnawiać ich ilość na stanie magazynowym. Dodawanie produktów jest oparte na formularzu modalnym.



The screenshot shows the main interface of the Zaopatrzeniowiec application. On the left is a sidebar with icons for Ametyst (purple gemstone), PRODUKTY (products), TABELE SŁOWNIKOWE (dictionary tables), and HISTORIA (history). Below these is a user profile for Leonard Radzikowski. The main area has a search bar labeled 'Szukaj..'. A 'Filtrowanie' section allows sorting by ID, NAME, ADD_DATE, and ACTIONS. A table lists 9 products with columns for ID, NAME, ADD_DATE, and ACTIONS (marked with an 'X').

ID.	NAZWA	DATA DODANIA	AKCJE
1	AGAT FASETOWANE KULE 4 MM - SZNUR (MIODOWY)	2021-08-27 12:03:00.0	X
2	AGAT FUKSJA FASETOWANE KULE 6 MM - SZNUR	2021-08-27 13:05:00.0	X
3	AGAT PŁONĄCY - KROPLA 27 MM / 18 MM	2021-08-27 13:19:40.0	X
4	AKWAMARYN BRYLKI - SZNUR 20 CM	2021-08-27 13:21:57.0	X
5	AKWAMARYN FASETOWANE KAMIENIE - SZNUR 19 CM	2021-08-27 14:09:46.0	X
6	AKWAMARYN FASETOWANE KAMIENIE - SZNUR 19 CM	2021-08-27 14:09:46.0	X
7	AMETYST KULE 4 MM - 10 SZT	2021-08-27 14:23:10.0	X
8	AMETYST ZIELONY FASETOW. BRIOLETTE MIGDAŁ 10-9 MMM	2021-08-27 14:30:04.0	X
9	AMETYST FASETOWANE KULE 10 MM	2021-08-27 14:34:46.0	X

Rysunek 4.47. Strona główna zaopatrzeniowca

Po przejściu do widoku produktu można dokonać modyfikacji oraz dodać lub usunąć zdjęcie. Dodawanie odbywa się na podstawie metody drag&drop, co oznacza przenieś i upuść. Jest to najłatwiejsza metoda. Istnieje również możliwość skorzystania z linka odnoszącego się do ręcznego wyboru plików ze ścieżki. Aplikacja obsługuje możliwość przesyłania wielu obrazów jednocześnie. Program przekieruje użytkownika na 5 sekund do strony przejściowej, gdzie zaczeka na dodanie wszystkiego do w projektu.

Wybierz obraz

DODAWANIE

USUWANIE

Upuść pliki tutaj lub skorzystaj z [tego linka](#)

ZATWIERDŹ PLIKI

Rysunek 4.48. Okienko modalne dodawania nowego zdjęcia produktu

W celu usunięcia należy przejść na odpowiednią zakładkę w okienku modalnym i wybrać niechciane zdjęcia, a następnie zatwierdzić wybór. Po doczekaniu pięciu sekund na stronie przejściowej system przekieruje użytkownika do widoku produktu.

Wybierz obraz

DODAWANIE

USUWANIE



USUŃ ZAZNACZONE

Rysunek 4.49. Okienko modalne usuwania zdjęcia produktu

Uaktualnianie produktów jest oparte na podobnym okienku modalnym co dodawanie przedmiotów, z tą różnicą, że w przypadku edycji pola są już wypełnione.

Backend:

Strona główna tak samo jak strona widoku jest oparta na pobieraniu odpowiednich danych do sesji wyświetlających listy produktów z grupy oraz kategorii.

Aktualizację produktu obsługuje metoda `updateProductForSupplier`. Po sprawdzeniu użytkownika oraz jego rangi uaktualniane są odpowiednie dane z pominięciem ilości obrazów, które odbywa się automatycznie z innego miejsca. W przypadku dodawania produktu (`addProductForSupplier`) użytkownik może pominąć niektóre wartości, takie jak: wysokość, szerokość, długość, wielkość otworu, wagę oraz średnicę. Algorytm wpisze w ich miejsca wartości zerowe, które w formularzu edycji można poprawić.

Wspomniane wcześniej usuwanie produktów za pomocą znaku \times na stronie głównej ustawia ilość na stanie magazynowym na wartość zerową. Usunięcie z bazy danych nie jest

możliwe ze względu na ewentualne powiązania z zamówieniami użytkowników. Jedyna opcja pełnego usunięcia istnieje z poziomu historii zaraz po dodaniu pliku, gdzie użytkownik nie ma do niego jeszcze pełnego dostępu.

Obsługa plików znajduje się w serwecie *ControlFiles.java* oraz odbywa się za pomocą metod: `uploadImg(request, response)`, `deleteImg(request, response)` oraz `updateImg(request, response)`, czyli kolejno dodawanie, usuwanie i zmiana ilości zdjęć w bazie danych, które odbywają się za pomocą metod w klasie *ProductDAO.java*.

W przypadku dodawania zdjęcia czytane są przesłane pliki, których kopie tworzone są w dwóch lokalizacjach: folderze projektu oraz lokalizacji serwerowej. Wszystkie pliki, które mają wsparcie ze strony aplikacji (`.png`, `.jpg`, `.gif`, `.jfif`, `.jpeg`) są konwertowane na dokumenty `.jpg`, które są używane jako rozszerzenie statyczne dla całego projektu. Są one dodawane jako kolejne liczny całkowite, np.: `1.jpg`, `2.jpg`, itd.

```
148 //Konwersja pliku do .jpg
149     BufferedImage originalImage = ImageIO.read(storeFile);
150     try{
151         BufferedImage newBufferedImage = new BufferedImage(
152             originalImage.getWidth(),
153             originalImage.getHeight(),
154             BufferedImage.TYPE_INT_RGB);
155         newBufferedImage.createGraphics().drawImage(originalImage,0,0,Color.WHITE,null);
156         boolean next = true;
157         String uploadImagePath = PATH + "\\\" + id;
158         File targetDir = new File(uploadImagePath);
159         if (!targetDir.exists()) { targetDir.mkdir(); }
160         do{
161             String targetImagePath = uploadImagePath + File.separator + img + ".jpg";
162             File targetImg = new File(targetImagePath);
163             if (targetImg.exists()){
164                 img++;
165             }
166             else{
167                 ImageIO.write(newBufferedImage, "jpg", targetImg);
168                 next = false;
169             }
170         }while(next);
171     }
172     catch (IOException ex){
173         sess.setAttribute("err_disc","Błąd konwersji obrazu: " + ex.getMessage());
174     }
175     storeFile.delete();
```

Rysunek 4.50. Konwersja przesyłanych zdjęć do .jpg

Podobna metoda zostaje zastosowana przy usuwaniu. W tym przypadku jednak konwersja nie ma miejsca. Trzeba jednak zastąpić „dziury” spowodowane usuniętymi dokumentami. Kopiowane są zatem pliki o większej liczbie niż usunięta, zastępując tym samym ubytki.

Ze względu na opisywanie edycji tabel słownikowych w sekcji administratora zostanie to pominięte.

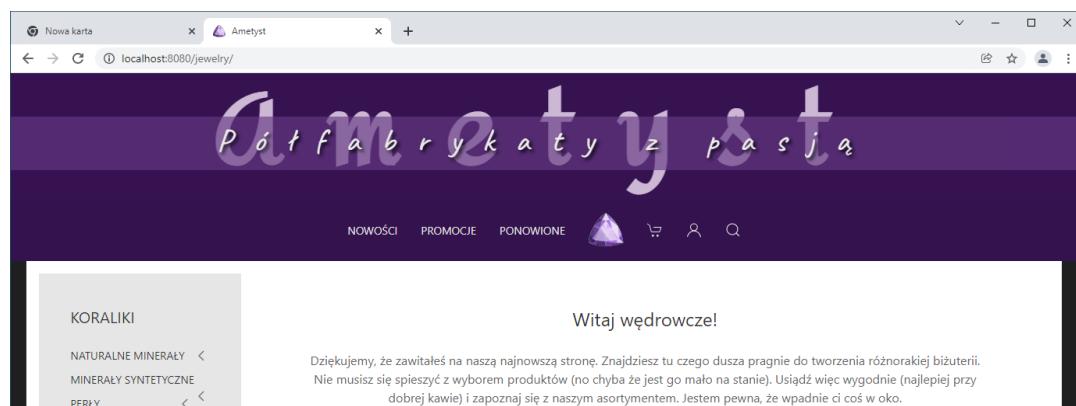
5. Testy aplikacji

„Celem testowania systemu z punktu widzenia modelowania jest wykazanie niezgodności ze stawianymi mu wymaganiami lub niespójności z jego modelami, czyli w praktyce wcześnie wykrywanie usterek, zanim okażą się źródłem poważnych problemów.”³⁵ Zostanie ono przeprowadzone pod kątem braku wykrytych błędów podczas przeprowadzania podstawowych testów.

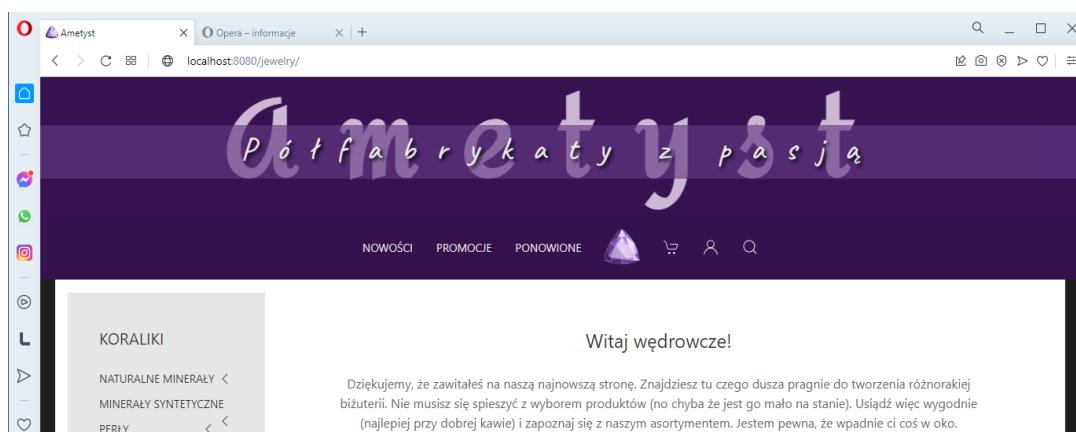
5.1. Testy wymagań niefunkcjonalnych

Testy przeprowadzono manualnie przy użyciu większości wersji opisanych w rozdziale 3.1. Zauważono, że do funkcjonowania aplikacji można użyć serwera Wildfly o wyższej wersji, tyczy się to również pakietu Xampp. Jej zmiana w przypadku Javy skutkuje nieprawidłowym budowaniem pliku projektu, co uniemożliwia jego prawidłowe działanie.

Test przeglądarek przeprowadzono na wypisanych w wymaganiach aplikacjach: Google Chrome v96.0.4664.110, Opera v82.0.4227.43, Microsoft Edge v44.18362.449.0, Mozilla Firefox v95.0.2.

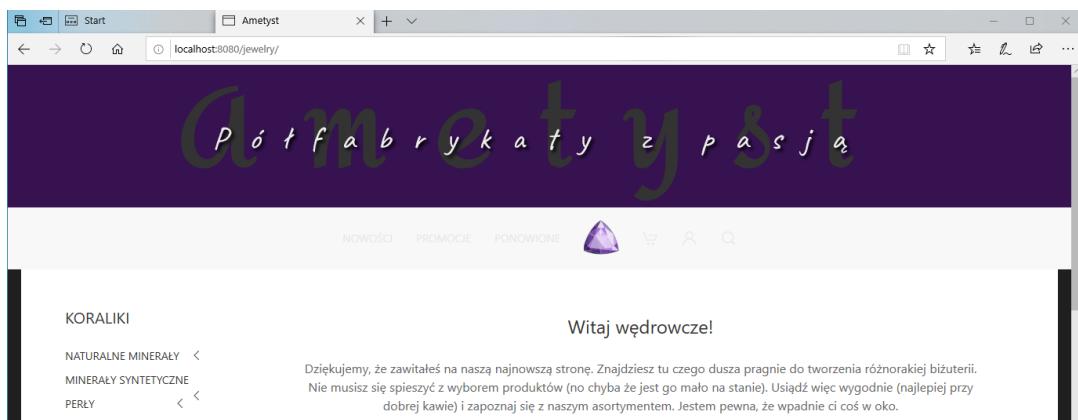


Rysunek 5.51. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Chrome

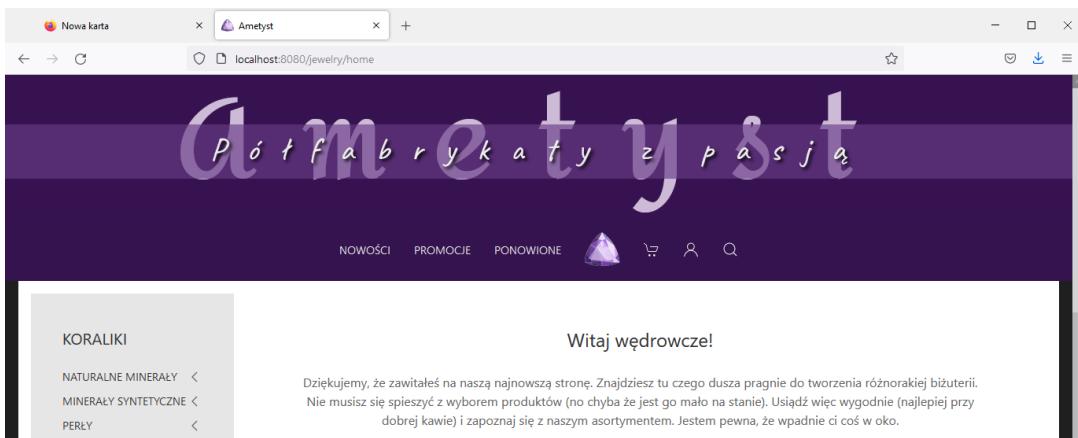


Rysunek 5.52. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Opera

³⁵ Bruegge B., Dutoit A. H., *Inżynieria oprogramowania w ujęciu obiektowym. UML, wzorce projektowe i Java*, Helion 2011, s. 491.



Rysunek 5.53. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Edge



Rysunek 5.54. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Firefox

Jak widać na powyższych zrzutach ekranów, aplikacja działa prawidłowo na przytoczonych przeglądarkach. Jedynie Edge ma mały problem z poprawnym rozpoznawaniem kolorów.

Aplikacje opisano w poprzednich rozdziałach. Została stworzona z myślą o łatwości użytkowania oraz zgodności z istniejącymi rozwiązaniami. Interfejs użytkownika obsługuje polskie znaki oraz poprawnie je wyświetla, jak widać na powyższych zrzutach ekranu oraz formularzach w rozdziale czwartym, np. rysunku 4.22.

Bezpieczeństwo zapewniają odpowiednie funkcje wprowadzone zarówno w projekcie jak i w bazie danych. Żaden użytkownik (włącznie z administratorem) nie posiada wglądu do listy haseł ze strony aplikacji. Pojedyncza dana jest wyciągana jedynie podczas logowania na potrzeby uwierzytelnienia. Stworzone zostały konta użytkowników z odpowiednimi ograniczeniami korzystania z bazy. Np. użytkownik „products” posiada wgląd w tabele z trzech pierwszych grup, opisanych w rozdziale 4.1, przy czym żadną z nich nie może edytować (przysługuje mu tylko wyświetlanie odpowiednich atrybutów). Wszystkie konta widoczne są na poniższym rysunku 5.55. Zaznaczone dane utworzone są na potrzeby projektu.

Nazwa użytkownika	Nazwa hosta	Hasło	Globalne uprawnienia	Grupa użytkownika	Nadawanie	Działanie
<input type="checkbox"/> Dowolny	%	Nie	USAGE	Nie	Edytuj uprawnienia Eksport	
<input type="checkbox"/> admin	localhost	Tak	ALL PRIVILEGES	Tak	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> administrator	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> corrector	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> dictDelete	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> dictInsert	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> dictSelect	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> dictUpdate	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> history	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> historyAll	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> login	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> menu	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input type="checkbox"/> pma	localhost	Nie	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> products	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input type="checkbox"/> root	127.0.0.1	Nie	ALL PRIVILEGES	Tak	Edytuj uprawnienia Eksport	
<input type="checkbox"/> root	::1	Nie	ALL PRIVILEGES	Tak	Edytuj uprawnienia Eksport	
<input type="checkbox"/> root	localhost	Nie	ALL PRIVILEGES	Tak	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> supplier	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> user	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input checked="" type="checkbox"/> userNL	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	
<input type="checkbox"/> worker	localhost	Tak	USAGE	Nie	Edytuj uprawnienia Eksport	

Rysunek 5.55. Widok użytkowników bazy danych w aplikacji phpMyAdmin

Hasła nie zostały dodatkowo zahaszowane ze strony bazy danych ze względu na czas wykonywania polecień oraz porównania i haszowania hasła podanego przez użytkownika w formularzu. Istnieje również ryzyko pomyłki podczas wpisywania hasła przy zmianie lub rejestracji na błędne, które później ciężko byłoby odzyskać.

Inną formą zabezpieczeń jest uwierzytelnianie danych sesyjnych Javy, na których opiera się działanie aplikacji. Opisane w rozdziale czwartym metody pomagają wychwytywać osoby niepożądane.

```

276 //sprawdzanie czy użytkownik jest zalogowany
277     int id_usr = 0;
278     try { id_usr = (int) sess.getAttribute("id_user_logged"); }
279     catch (NullPointerException e) {
280         sess.setAttribute("err_disc","Brak dostępu!");
281         response.sendRedirect("home");
282         return;
283     }
284
285 //sprawdzanie czy użytkownik posiada rangę
286     User rnk = null;
287     try { rnk = userADA0.checkRankId(id_usr); }
288     catch (NullPointerException e) {
289         sess.setAttribute("err_disc","Brak dostępu!");
290         response.sendRedirect("home");
291         return;
292     }

```

Rysunek 5.56. Fragment uwierzytelniania użytkownika

5.2. Testy wymagań funkcjonalnych

Poniżej zostaną przedstawione podstawowe testy wymagań funkcjonalnych wykonywane manualnie. W tym kroku nie bierze się pod uwagę walidacji. Aplikacja została zaprojektowana w taki sposób, aby nie uznawała błędnych danych w przypadku dysfunkcji systemu walidacyjnego.

Wyświetlanie produktów

Tabela 5 – testy produktów

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Poruszanie poprzez menu boczne z kategorią	identyfikator kategorii	Wyświetlenie listy wybranych produktów	Zaliczony
2	Poruszanie poprzez menu boczne z tagiem	identyfikator tagu		Zaliczony
3	Poruszanie poprzez nawigację	identyfikator zakładki		Zaliczony
4	Wyszukiwanie produktów	szukana fraza		Zaliczony
5	Wyświetlenie pojedynczego produktu	identyfikator produktu	Wyświetlenie odpowiedniego produktu	Zaliczony
6	Zmiana identyfikatora w pasku URL	istniejące, aktualne dane		Zaliczony
7		istniejące, archiwalne dane	Wyświetlenie archiwalnego produktu bez możliwości kupna	Zaliczony
8		błędne dane	Przekierowanie (pw.) do strony głównej z komunikatem o błędzie	Zaliczony
9	Dodanie produktu do koszyka z podaną ilością	identyfikator oraz ilość produktu	Przekierowanie do serwletu (swl.) ze zmianą danych (zmn.), powrót na stronę (pw.) i wyświetlenie okienka modalnego	Zaliczony

Składanie zamówienia

Tabela 6 – testy zamówienia

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Przejście do koszyka z poziomu okienka modalnego na stronie produktu	_____	prz. do pierwszego kroku zamówienia	Zaliczony
2	Przejście do koszyka z poziomu nawigacji	_____		Zaliczony
3	Kontynuowanie zamówienia z poziomu koszyka	zmienne do zapisania w sesji aplikacji	prz. do drugiego kroku zamówienia	Zaliczony
4	Wybór kontynuowania jako użytkownik bez konta	_____	prz. do trzeciego kroku zamówienia	Zaliczony
5	Wybór kontynuowania jako użytkownik z kontem	poprawne dane loginu i hasła użytkownika	Przypisanie aktualnego koszyka do użytkownika zalogowanego i przekierowanie do czwartego kroku zamówienia	Zaliczony
6		błędne dane loginu i hasła użytkownika	swl. i powrót na stronę pw. z informacją o błędzie	Zaliczony
7	Wybór kontynuowania po rejestracji	_____	prz. do strony rejestracji	Zaliczony
8	Kontynuowanie z poziomu użytkownika niezalogowanego w kroku trzecim	poprawne dane adresu prywatnego lub firmy	prz. do czwartego kroku zamówienia	Zaliczony
9		błędne dane adresu prywatnego lub firmy	swl. pw. z informacją o błędzie	Zaliczony
10	Akceptacja zamówienia z poziomu kroku czwartego	akceptacja regulaminu i przekazanie komentarza	swl. zmn., migracja produktów z koszyka i prz. do strony podsumowującej	Zaliczony

Rejestracja i logowanie

Tabela 7 – testy rejestracji i logowania

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Rejestracja za pomocą formularza	poprawne dane osobowe, hasła oraz adresu	prz. do strony logowania	Zaliczony
2		błędne dane osobowe, hasła oraz adresu	swl. pw. i wyświetlenie błędu	Zaliczony

3	Rejestracja za pomocą formularza w trakcie składania zamówienia	poprawne dane osobowe, hasła oraz adresu	Migracja koszyka i prz. do kroku czwartego zamówienia	Zaliczony
4		błędne dane osobowe, hasła oraz adresu	swl. pw. i wyświetlenie błędu	Zaliczony
5	Logowanie do aplikacji	poprawne dane logowania	Prz. do strony głównej użytkownika	Zaliczony
6		błędne dane logowania	swl. pw. i wyświetlenie błędu	Zaliczony

Użytkownik zalogowany

Tabela 8 – testy użytkownika zalogowanego

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Widok strony głównej po zalogowaniu	_____	Wyświetlenie listy zamówień dokonanych w sklepie	Zaliczony
2	Wybór przejścia na stronę edycji danych	_____	Wyświetlenie odpowiedniej strony z danymi użytkownika	Zaliczony
3	Zmiana hasła	nowe hasło z potwierdzeniem	swl. zmn. pw. i wyświetlenie komunikatu o sukcesie	Zaliczony
4		brak hasła lub potwierdzenia	swl. pw. i wyświetlenie komunikatu o błędzie	Zaliczony
5	Zamiana danych osobowych	nowe dane	swl. zmn. pw. i wyświetlenie komunikatu o sukcesie	Zaliczony
6	Zamiana danych istniejącego adresu	dane formularza		Zaliczony
7	Dodanie nowego adresu	dane adresu		Zaliczony
8		błędne dane adresu	swl. pw. i wyświetlenie komunikatu o błędzie	Zaliczony

Użytkownik z rangą administratora

Tabela 9 – testy użytkownika z rangą administratora

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Widok strony głównej po zalogowaniu	_____	Wyświetlenie panelu edycyjnego tabel słownikowych	Zaliczony
2	Kliknięcie ikony (plus, ołówek)	dane rekordu	Wyświetlenie okienka modalnego z danymi odpowiadającymi wybranym	Zaliczony
3	Dodanie rekordu za pomocą ikony plusa	nowe dane	swl. zmn. pw. i wyświetlenie komunikatu o sukcesie	Zaliczony

4	Edycja rekordu za pomocą ikony ołówka	nowe dane	swl. zmn. pw. i wyświetlenie komunikatu o sukcesie	Zaliczony
5	Usunięcie rekordu za pomocą ikony ×	identyfikator rekordu		Zaliczony
6	Kliknięcie zakładki "użytkownicy"	_____	prz. do panelu widoku użytkowników	Zaliczony
7	Dodanie użytkownika za pomocą ikony plusa	wszystkie wymagane dane	swl. zmn. pw. i wyświetlenie komunikatu o sukcesie	Zaliczony
8	Kliknięcie identyfikatora użytkownika	identyfikator	prz. do listy historii użytkownika	Zaliczony
9	Kliknięcie zakładki "menu"	_____	prz. do widoku edycji menu	Zaliczony

Użytkownik z rangą pracownika

Tabela 10 – testy użytkownika z rangą pracownika

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Widok strony głównej po zalogowaniu	_____	Wyświetlenie panelu zamówień	Zaliczony
2	Kliknięcie identyfikatora zamówienia	identyfikator	prz. do pełnego widoku zamówienia	Zaliczony
3	Naciśnięcie na przycisk "Edytuj"	_____	Wyświetlenie okienka modalnego	Zaliczony
4	Wybranie nowego statusu	status większy niż obecny	swl. zmn. pw. i powrót z komunikatem o sukcesie	Zaliczony
5	Dodanie numeru przesyłki	status większy lub równy 6 („Wysłano”)		Zaliczony
6		status mniejszy niż 6	swl. pw. i wyświetlenie komunikatu o błędzie	Zaliczony

Użytkownik z rangą korektora

Tabela 11 – testy użytkownika z rangą korektora

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Widok strony głównej po zalogowaniu	_____	Wyświetlenie panelu recenzji	Zaliczony
2	Kliknięcie identyfikatora recenzji	identyfikator	prz. do pełnego widoku recenzji	Zaliczony
3	Edycja odpowiedzi poprzez okienko modalne	treść odpowiedzi	swl. zmn. pw. i powrót z komunikatem o sukcesie	Zaliczony
4	Kliknięcie zakładki "produkt"	_____	prz. do panelu produktu	Zaliczony

5	Edycja opisu produktu w widoku poprzez okienko modalne	treść opisu	swl. zmn. pw. i powrót z komunikatem o sukcesie	Zaliczony
6	Kliknięcie zakładki "kategoria"	_____	prz. do panelu tagu	Zaliczony
7	Edycja opisu kategorii w widoku poprzez okienko modalne	treść opisu	swl. zmn. pw. i powrót z komunikatem o sukcesie	Zaliczony
8	Kliknięcie zakładki "tag"	_____	prz. do panelu tagu	Zaliczony
9	Edycja opisu tagu i widoku poprzez okienko modalne	treść opisu	swl. zmn. pw. i powrót z komunikatem o sukcesie	Zaliczony

Użytkownik z rangą zaopatrzeniowca

Tabela 12 – testy użytkownik z rangą zaopatrzeniowca

Lp.	Krok	Dane wejściowe	Oczekiwany rezultat	Wynik testu
1	Widok strony głównej po zalogowaniu	_____	Wyświetlenie panelu produktów	Zaliczony
2	Usunięcie produktu przez ikonę ×	identyfikator produktu	swl. zmn. pw. i powrót z komunikatem o sukcesie	Zaliczony
3	Odnowienie produktu na stanie za pomocą okienka modalnego	identyfikator i nowa ilość produktu		Zaliczony
4	Dodanie produktu za pomocą formularza modalnego	potrzebne dane		Zaliczony
5	Kliknięcie identyfikatora produktu	identyfikator	prz. do pełnego widoku produktu	Zaliczony
6	Edycja za pomocą okna modalnego	nowe dane	swl. zmn. pw. i powrót z komunikatem o sukcesie	Zaliczony
7	Dodanie nowego zdjęcia	przesłane pliki		Zaliczony
8	Usunięcie zdjęcia	nazwa zdjęcia		Zaliczony
9	Kliknięcie zakładki "tabele słownikowe"	_____	prz. do panelu tabel słownikowych	Zaliczony
10	Dodanie rekordu za pomocą ikony plusa	nowe dane	swl. zmn. pw. i wyświetlenie komunikatu o sukcesie	Zaliczony
11	Edycja rekordu za pomocą ikony ołówka	nowe dane		Zaliczony
12	Usunięcie rekordu za pomocą ikony ×	identyfikator rekordu		Zaliczony

5.3. Testy walidacji

Walidację można zobaczyć między innymi na rysunkach 4.22 oraz 4.23. Większość formularzy się powtarza jak na przykład dane adresu, które występują w trzech miejscach: podawania danych przez użytkownika niezalogowanego przy trzecim kroku koszyka, rejestracji oraz w panelu użytkownika zalogowanego. To tam można zarówno zmieniać istniejące adresy jak i dodać nowe.

Poniższy przykład odzwierciedla prawidłowe działanie formularza z funkcją walidacji. Po wpisaniu poprawnych danych można przejść do następnego kroku potwierdzenia ich po stronie serwerowej aplikacji.

The screenshot shows a password change form with three fields: 'Poprzednie hasło *' (Previous password) containing '*****', 'Nowe hasło *' (New password) containing 'Nie Mam', and 'Potwierdź hasło *' (Confirm password) containing 'NiePosiadamHasła'. Below the 'Nowe hasło' field is an error message: 'Musi mieć przynajmniej 8 znaków!' (Must have at least 8 characters!). Below the 'Potwierdź hasło' field is another error message: 'W obu polach musi być to samo hasło!' (Both fields must contain the same password!). At the bottom are two buttons: 'ANULUJ' (Cancel) and a dark 'ZMIEN' (Change) button.

Rysunek 5.57. Pokazanie działanie walidacji na przykładzie zmiany hasła

Jest ona przeprowadzana za pomocą biblioteki jQuery Validation, która umożliwia szybkie i płynne działanie w tej kwestii.

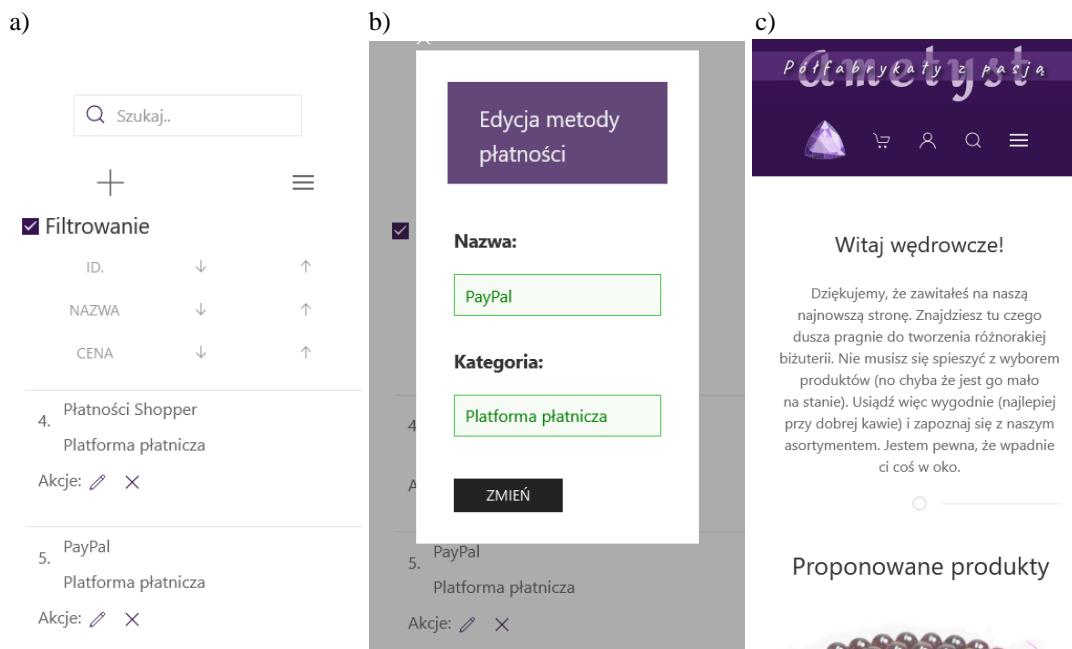
Testy były prowadzone analogicznie do rozdziału sprawdzającego wymagania funkcjonalne. W tym przypadku pominięto większość kroków na rzecz zbadania poprawnej reakcji formularzy z dołączoną do nich walidacją.

Sprawdzane były strony: `/basketNL` (formularz użytkownika niezalogowanego w koszyku), `/summaryBask` (formularz akceptacji zamówienia), `/register` (rejestracja), `/login` (logowanie) oraz `/basketI` (logowanie ze strony koszyka), `/settings` (zmiana danych konta użytkownika zalogowanego), `/administrator` (okienka modalne administratora), `/worker` (okienka modalne pracownika), `/corrector` (okienka modalne korektora), `/supplier` (okienka modalne zaopatrzeniowca).

Wszystkie z wyżej wymienionych stron działają poprawnie w oryginalnym zamыśle. Mają zapobiegać pomyłkom ludzkim oraz maszynowym.

5.4. Testy responsywności

Responsywność to zdolność aplikacji dostosowania się do różnych urządzeń mobilnych w formie czytelnej oraz pozwalającej na pełne użytkowanie. Jak zostało już wspomniane w wymaganiach, niniejsza aplikacja posiada tę funkcję. Poniżej przedstawiono przykładowe widoki ze strony mniejszych urządzeń mobilnych.



Rysunek 5.58. a) responsywna strona administratora wyświetlająca metody płatności, b) responsywna strona z oknem edycji metody płatności, c) responsywna strona główna.

Tabele są w pełni responsywne, ale ich wygląd nieco zmienia się w zależności od widoku użytkowników. Przykładem są kroki w koszyku. Krok pierwszy umożliwia edycje produktów, które się w nim znajdują. Musi mieć zatem inny układ niż strona podsumowująca, gdzie znajduje się mniej elementów w tabeli.



Rysunek 5.59. a) widok kroku pierwszego koszyka, b) widok kroku czwartego

Testy przeprowadzono na trybie responsywnym przeglądarki Firefox oraz Brave (opartej o Chromium). Jedynym problemem jest brak trybu asynchronicznego aplikacji, przez co należy ją odświeżyć po otwarzeniu widoku responsywnego. W przeciwnym wypadku nie zostanie ona wyświetlona poprawnie oraz niektóre funkcje nie będą dostępne.

6. Analiza i wnioski

Ostatni rozdział bezpośrednio odnoszący się do aplikacji i jej działania. Jest to wprowadzenie do podsumowania końcowego opisanego w ostatnim rozdziale.

6.1.Osiągnięte cele

Aplikacja działa zgodnie z wymaganiami przedstawionym i w rozdziale pierwszym oraz trzecim. Zostały zastosowane odpowiednie metody rozwiązuające podany problem.

Cel aplikacji został osiągnięty dzięki stworzeniu jej w postaci witryny internetowej, której działanie ukryte jest po stronie serwerowej Javy. Bazuje ona na istniejących, realnych rozwiązaniach.

Aplikacja opiera się o użytkowników sesyjnych oraz zalogowanych włącznie z czterema rangami zgodnie z zamysłem początkowym. Jak zostało to opisane wcześniej, wszyscy użytkownicy posiadają dostęp do podstawowej funkcjonalności sklepu.

Istnieje możliwość rozwoju menu po stronie administratora, dzięki czemu mogą być dodane nowe techniki tworzenia biżuterii. Tym sposobem zostanie rozwiązany problem w postaci braku materiałów z danej metody wyrobu w sklepie.

6.2.Możliwości rozwoju aplikacji

Projekt sam w sobie na obecnym etapie jest rozbudowywany, jednak jak w przypadku każdej aplikacji możliwe są poprawki oraz rozbudowania niektórych funkcji.

W celu poprawnego zabezpieczenia w przyszłości można dodać pytanie pomocnicze do tabeli **login**, które pomogłoby w odzyskaniu hasła; oraz zastosowanie szyfrowania.

Dodatkowo można rozbudować system wysyłania wiadomości e-mail do użytkowników składających zamówienia i informować ich w ten sposób o zmianie statusu. Na podobnej zasadzie można dodać odpowiedni panel dla korektora, który zajmowałby się cykliczną wysyłką newslettera.

Inną opcją jest dodanie możliwości projektowania biżuterii opartej na zapisie współrzędnych danego elementu ułożonego na kanwie przez użytkownika. Mogłoby to być coś w postaci pomniejszego edytora grafiki wektorowej opartej na JavaScript oraz CSS z możliwościami wyginania elementów niestatycznych według własnego upodobania. Do tego pomysłu konieczne byłoby rozbudowanie panelu pracownika o osobne opcje dla takiego projektanta oraz utworzenie odpowiednich tabel obsługujących te funkcje.

7. Podsumowanie

Realizując pracę zaczęto, od jasnego określenia celu i założeń, jakie projekt powinien spełniać w efekcie końcowym. Dzięki rzetelnej pracy oraz nieustannej kontroli jakości podczas pisania aplikacji udało się w pełni dostosować całokształt do wymagań.

W czasie implementacji projekt zmieniał kilkakrotnie wygląd oraz swoje funkcjonalności. Zaczęto od panelu roota, który posiadał możliwość operacji CRUD na każdej tabeli. W ostateczności usunięto tę możliwość ze względów bezpieczeństwa aplikacji. Zostawiono jedynie metody mogące się przydać w przyszłości w rozwijaniu projektu. Odpowiednie funkcje były dodawane na różnych etapach, a pomysły na dodatkowe rozwiązania mnożyły się z każdym dniem, dzięki czemu ostateczny wygląd jest rozwinięty oraz dopracowany.

Testy pozwoliły wykryć błędy na różnych etapach projektowania i w ostateczności wszystkie przeszły pozytywnie. Nie świadczy to jednak o idealnej funkcjonalności aplikacji. Zawsze znajdzie się błąd, którego nie zauważała osoba projektująca lub też nie wzięła go pod uwagę. Po ukończeniu projektu wykonano ostateczne sprawdzenie aplikacji pod kątem funkcjonowania oraz poprawiono drobne niezgodności.

Istnieje wiele możliwości, o które można by rozwinąć ten projekt, takie jak przykłady przedstawione we wcześniejszym rozdziale. Do tego celu jednak potrzeba więcej czasu oraz pomysłów. Cały proces tworzenia aplikacji z poziomu jednej osoby jest oczywiście inny niż w przypadku specjalistycznej grupy, gdzie jedna osoba pisze tylko fragment kodu. Miało to jednak na celu pokazanie umiejętności piszącej oraz jej tok myślenia podczas rozwiązywania problemu.

Zrealizowany projekt pozwolił na zapoznanie się z procesem wytwarzania aplikacji biznesowych, diagnozowania i testowania ich pod kątem postawionych wymagań. Proces przedstawiony w pracy pozwolił na rozwiązanie postawionego w tytule pracy problemu inżynierskiego.

Bibliografia

- [1] *amberplanet.pl, Bursztyn*, <https://www.sklep.amberplanet.pl/k11,bursztyn.html> (dostęp: 20.12.2021 r.)
- [2] Bowman J. S., Darnovsky M., Emerson S. L., *Podręcznik języka SQL*, Wydawnictwa Naukowo Techniczne 2001.
- [3] Bruegge B., Dutoit A. H., *Inżynieria oprogramowania w ujęciu obiektowym. UML, wzorce projektowe i Java*, Helion 2011.
- [4] Duckett J., *HTML i CSS. Zaprojektuj i zbuduj witrynę WWW. Podręcznik Front End Developera*, Helion 2014.
- [5] Duckett J., *JavaScript i jQuery Interaktywne strony WWW dla każdego*, Helion 2015
- [6] Faroult S., Robson P., *SQL. Sztuka programowania*, Helion 2007.
- [7] Hunter J., Crawford W., *Java Servlet. Programowanie*, wyd. 2, Helion 2002.
- [8] Jendrock E., Cervera-Navarro R., Evans I., Gollapudi D., Haase K., Markito W., Srivaths C., *Java EE 6. Zaawansowany przewodnik*, tłum. R. Jońca, wyd 4, Helion 2013.
- [9] *jQuery Api*, <https://api.jquery.com/> (dostęp: 09.12.2021 r.)
- [10] *jQuery Mask*, <https://igorescobar.github.io/jQuery-Mask-Plugin/docs.html> (dostęp: 14.12.2021 r.)
- [11] *jQuery Validation*, <https://jqueryvalidation.org/documentation/> (dostęp: 02.12.2021 r.)
- [12] *kamieniolomy.pl, kamienie naturalne sznury*, <https://kamieniolomy.pl/kamienie-naturalne-sznury> (dostęp: 20.12.2021 r.)
- [13] *kianit.pl, inne metale*, <https://www.kianit.pl/inne-metale-cat-18> (dostęp: 20.12.2021 r.)
- [14] Kurlus T., *Kamienie szlachetne. Wszystko o...*, wyd. 1, Krajowa Agencja Wydawnicza 1976.
- [15] *Metodologia*, <https://wsjp.pl/haslo/podglad/32253/metodologia/4058094/nauka> (dostęp: 20.01.2022 r.)
- [16] Nowicki T., Wrzosek E., *Modelowanie, symulacja i analiza systemów klasycznych klient-serwer*, „Symulacja w Badaniach i Rozwoju”, 2010, Vol. 1, nr 3.
- [17] *phpMyAdmin*, <https://docs.phpmyadmin.net/en/latest/intro.html> (dostęp: 16.12.2021)
- [18] Ross J., *PHP i HTML. Tworzenie dynamicznych stron WWW*, Helion 2010.
- [19] *SkarbynaNature.pl, wszystko -70%*, <https://www.skarbynaNature.pl/pl/c/WSZYSTKO-70/5121> (dostęp: 20.12.2021 r.)
- [20] *UIkit*, <https://getuikit.com/docs/introduction/> (dostęp: 02.12.2021 r.)
- [21] *VanillaSelectBox*, <https://www.cssscript.com/single-multi-select-vanilla/> (dostęp: 14.12.2021 r.)
- [22] Wdowiak L., *Zarys historii ozdabiania ciała*, wyd. 1, Wydawnictwo Pomorskiego Uniwersytetu Medycznego w Szczecinie 2017.
- [23] *WildFly*, <https://docs.wildfly.org/24/> (dostęp: 16.12.2021)
- [24] Yener M., Theedom A., *Java EE. Zaawansowane wzorce projektowe*, tłum. Ł. Piwko, Helion 2015.

Spis rysków

Rysunek 1.1. Brązowa broszka wykonana metodą makramy [wykonanie własne]	5
Rysunek 2.2. Struktura katalogowa Maven	10
Rysunek 2.3. Okres trwałości serwletu. [7]	11
Rysunek 2.4. Generowanie stron JavaServer Pages. [7]	12
Rysunek 2.5. Podstawowa struktura kodu HTML	16
Rysunek 2.6. Przykładowa reguła w języku CSS	16
Rysunek 2.7. Widok sklepu skarbynatury.pl [18].....	21
Rysunek 2.8. Widok sklepu kianit.pl [13]	22
Rysunek 2.9. Widok sklepu kamieniolomy.pl [12].....	23
Rysunek 2.10. Widok sklepu amberplanet.pl [1].....	24
Rysunek 4.11. Diagram ERD bazy danych.....	28
Rysunek 4.12. a) drzewo projektu aplikacji NetBeans, b) drzewo katalogowe projektu	34
Rysunek 4.13. Utworzenie danych pliku ActionDAO.java	37
Rysunek 4.14. Uaktualnienie danych pliku ActionDAO.java	37
Rysunek 4.15. Usuwanie danych pliku ActionDAO.java	38
Rysunek 4.16. Czytanie danych pliku ActionDAO.java.....	38
Rysunek 4.17. Strona główna aplikacji.....	39
Rysunek 4.18. Skryplet z pliku index.jsp.....	40
Rysunek 4.19. Komunikat po dodaniu produktu do koszyka	41
Rysunek 4.20. Widoku koszyka.....	42
Rysunek 4.21. Widok wyboru kontynuowania zakupu.....	42
Rysunek 4.22. Formularz danych użytkownika niezalogowanego	43
Rysunek 4.23. Formularz sprawdzania zamówienia przez użytkownika niezalogowanego	44
Rysunek 4.24. Formularz recenzji produktu	44
Rysunek 4.25. Metoda dodawania produktów do koszyka	45
Rysunek 4.26. Sprawdzanie oraz tworzenie adresu użytkownika niezalogowanego	46
Rysunek 4.27. Skrócony widok strony głównej użytkownika zalogowanego	47
Rysunek 4.28. Widok ustawień użytkownika zalogowanego	48
Rysunek 4.29. Historia użytkownika zalogowanego	48
Rysunek 4.30. Fragment kodu logowania użytkownika	49
Rysunek 4.31. Fragment metody rejestracji użytkownika	49
Rysunek 4.32. Fragment kodu modyfikacji kolejności zapytań w historii zdarzeń	50
Rysunek 4.33. Widok strony głównej administratora	51
Rysunek 4.34. Fragment widoku użytkowników ze strony administratora	52
Rysunek 4.35. Fragment historii zaopatrzeniowca od strony administratora	52
Rysunek 4.36. Fragment strony menu administratora	53
Rysunek 4.37. Fragment usuwania rekordu z tabeli color	54
Rysunek 4.38. Fragment wysyłania wiadomości e-mail o założeniu konta pracowniczego.....	54
Rysunek 4.39. Fragment dodawania wielu tagów do kategorii	55
Rysunek 4.40. Strona główna pracownika	56

Rysunek 4.41. Strona widoku zamówienia z poziomu pracownika	56
Rysunek 4.42. Przypisanie zmiennych sesyjnych na stronie głównej pracownika	57
Rysunek 4.43. Fragment aktualizacji statusu dla pracownika	58
Rysunek 4.44. Strona wystawiania odpowiedzi na recenzje użytkowników	58
Rysunek 4.45. Widok produktu od strony korektora	59
Rysunek 4.46. Uaktualnianie odpowiedzi na recenzje dla korektora.....	59
Rysunek 4.47. Strona główna zaopatrzeniowca.....	60
Rysunek 4.48. Okienko modalne dodawania nowego zdjęcia produktu.....	61
Rysunek 4.49. Okienko modalne usuwania zdjęcia produktu	61
Rysunek 4.50. Konwersja przesłanych zdjęć do .jpg	62
Rysunek 5.51. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Chrome.....	63
Rysunek 5.52. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Opera.....	63
Rysunek 5.53. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Egde	64
Rysunek 5.54. Zrzut ekranu z funkcjonującą aplikacją na przeglądarce Firefox.....	64
Rysunek 5.55. Widok użytkowników bazy danych w aplikacji phpMyAdmin	65
Rysunek 5.56. Fragment uwierzytelniania użytkownika	65
Rysunek 5.57. Pokazanie działanie walidacji na przykładzie zmiany hasła	71
Rysunek 5.58. a) responsywna strona administratora wyświetlająca metody płatności, b) responsywna strona z oknem edycji metody płatności, c) responsywna strona główna.	72
Rysunek 5.59. a) widok kroku pierwszego koszyka, b) widok kroku czwartego	72

Spis tabel

Tabela 1 – symbole zastosowane w pracy	1
Tabela 2 – skróty używane w rozdziale testowym.....	1
Tabela 3 – legenda oznaczeń czcionki	1
Tabela 4 – testy produktów.....	66
Tabela 5 – testy zamówienia.....	67
Tabela 6 – testy rejestracji i logowania.....	67
Tabela 7 – testy użytkownika zalogowanego.....	68
Tabela 8 – testy użytkownika z rangą administratora	68
Tabela 9 – testy użytkownika z rangą pracownika	69
Tabela 10 – testy użytkownika z rangą korektora.....	69
Tabela 11 – testy użytkownika z rangą zaopatrzeniowca	70

Załączniki

Płyta CD z aplikacją