

**WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI**
POLITECHNIKI RZESZOWSKIEJ

Mykola Nykolaichuk

System pośrednictwa usług mechaników samochodowych

Projekt inżynierski

Opiekun pracy:

dr. inż. Grzegorz Dec

Rzeszów, 2023

Spis treści

1.	WPROWADZENIE	6
2.	UZASADNIENIE WYBORU NARZĘDZI PROGRAMISTYCZNYCH UŻYTYCH DO REALIZACJI SERWISU.....	8
2.1.	Język programowania Java	10
2.2.	Spring Framework	13
2.2.1.	Moduł Spring Core Container	16
2.2.2.	Moduł Spring WEB	16
2.2.3.	Spring Data JPA	19
2.3.	Spring Security	21
2.4.	Spring Boot	23
2.4.1.	Spring Boot starters	23
2.4.2.	Konfiguracja automatyczna	23
2.4.3.	Zalety Spring Boot.....	24
2.5.	Relacyjne bazy danych	25
2.6.	Technologia Thymeleaf	27
3.	SPECYFIKACJA SYSTEMU	28
3.1.	Użytkownicy oraz ich uprawnienia	28
3.1.1.	Specyfikacja aplikacji.....	28
3.2.	Diagramy przypadków użycia	33
3.2.1.	System pośrednictwa usług mechaników samochodowych	33
3.2.2.	Zarządzanie kontem.....	34
3.2.3.	Zarządzanie zleceniami użytkownik niezalogowany	35
3.2.4.	Zarządzanie zleceniami klient	36
3.2.5.	Tworzenie zleceń.....	38
3.2.6.	Zarządzanie samochodami	39
3.2.7.	Zarządzanie warsztatem samochodowym	40
3.2.8.	Obsługa zleceń warsztat samochodowy	40
3.2.9.	Zarządzanie systemem pośrednictwa usług mechaników samochodowych	41
4.	BAZA DANYCH.....	43

4.1.	SQL skrypt	43
4.2.	Zaprojektowanie i tworzenie relacyjnej bazy danych	44
4.2.1.	Klasy encyjne (Entity)	44
4.2.2.	Diagram bazy danych	46
4.2.3.	Tabela car	47
4.2.4.	Tabela car_model	48
4.2.5.	Tabela car_make	48
4.2.6.	Tabela customer	48
4.2.7.	Tabela customer_detail	49
4.2.8.	Tabela customer_car	49
4.2.9.	Tabela ordering	49
4.2.10.	Tabela city	50
4.2.11.	Tabela workshop	50
4.2.12.	Tabela order_answer	50
4.2.13.	Tabela security token	51
4.2.14.	Tabela authority	51
4.2.15.	Tabela customer_authority	51
4.2.16.	Tabela workshop_authority	52
4.2.17.	Tabela persistent_logins	52
4.3.	Macierz CRUD	52
4.3.1.	Macierz CRUD dla operacji dotyczących użytkownika	52
4.3.2.	Macierz CRUD dla operacji dotyczących zlecenia	53
4.3.3.	Macierz CRUD dla operacji dotyczących samochodów	53
5.	IMPLEMENTACJA SYSTEMU	55
5.1.	Struktura projektu	55
5.1.1.	Komunikacja z bazą danych	55
5.1.2.	Warstwa biznes logiki serwisu	56
5.1.3.	Warstwa kontrolerów	59
5.1.4.	Warstwa widoków serwisu	61
6.	PRZEJRZENIE I OPIS SERWISU INTERNETOWEGO	63
6.1.	Widok serwisu ze strony niezalogowanego użytkownika	63
6.2.	Widok serwisu ze strony klienta	71
6.3.	Widok serwisu ze strony autoserwisu	76
6.4.	Widok serwisu ze strony administratora	78
7.	PODSUMOWANIE	80

ZAŁĄCZNIK A. INSTRUKCJA INSTALACJI	81
ZAŁĄCZNIK B. ZAWARTOŚĆ PŁYTY CD	82
LITERATURA.....	83

1. Wprowadzenie

W nasz czas do wyboru kompanii, która by zrealizowała nasze potrzeby w zakresie nadania usług, coraz częściej wykorzystują się online serwisy. Oni pozwalają, w jednym miejscu, ocenić większość propozycji do rozstrzygnięcia postawionego problemu.

Niestety obecnie nie istnieje online serwisu, który pozwalałby wybrać optymalny warsztat samochodowy na podstawie propozycji dla konkretnie stworzonego zlecenia.

Naprawa samochodów – zakres, gdzie technologizacja tylko zaczyna przenikać. Zwykliśmy tracić dzień albo nawet dwa, kiedy w samochodzie coś się psuje. Jeździmy po dużych oraz małych warsztatach samochodowych, porównujemy ceny, w końcu decydujemy oddać samochód na naprawę i w wyniku otrzymujemy wcale nie tę cenę, która była umówiona. Żeby na następny raz nie oddawać samochód na naprawę nieznajomym ludziom z niejasnymi cenowymi prognozami, został stworzony serwis Iremont.pl.

Pomysł na tworzenie takiego serwisu był oczywisty. Z jednej strony – w samochodziarza zawsze jest zapotrzebowanie na dobór warsztatów samochodowych do naprawy samochodu. Zazwyczaj to odbywa się tak: człowiek sam szuka po całych dostępnych zasobach, po znajomych, w internecie, obdzwaniania dziesiątki warsztatów samochodowych. Z innej strony – są tysiące autoserwisów, które poszukują klientów. Oni zmuszeni wkładać się do reklamy, tracić siły na poszukiwania i dołączenia klientów. Iremont.pl – serwis, co rozwiązuje wszystkie zagadnienia dla obu stron.

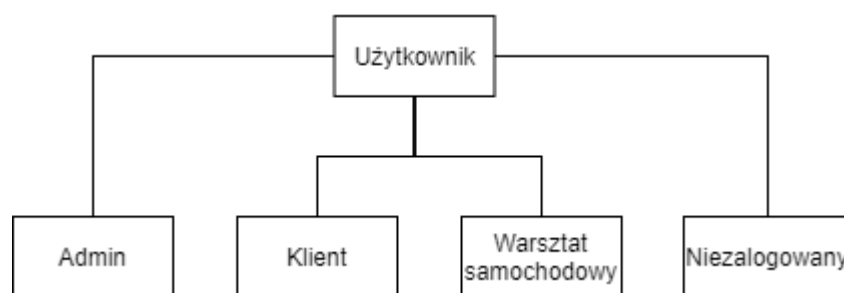
Na przedstawionej stronie internetowej to odbywa się w prosty sposób: klient wypełnia krótki formularz, gdzie ma wpisać swoje dane, a także główne techniczne charakterystyki samochodu – markę, model, datę produkcji, typ silnika, numer rejestracyjny, numer nadwozia i opisuje, co trzeba zrobić. Jednym kliknięciem zlecenie wysyła się do wszystkich zarejestrowanych w wybranym mieście warsztatów samochodowych. Partnerzy autoserwisu mają możliwość skontaktować się z klientem na podstawie jego danych kontaktowych.

Dla zarejestrowanego klienta strona internetowa proponuje szersze spektrum możliwości do współdziałania z autoserwisami. Istnieje możliwość dodania listy swoich samochodów, co pozwala podczas stworzenia zlecenia wybierać samochód i jego dane będą przeniesione do formularza. Dane klienta będą automatycznie przypisane do zlecenia. Warsztaty samochodowe gotowe zrealizować zlecenie wysyłają swoją odpowiedź ze wskazaniem dokładnego kosztu naprawy oraz datę prognozowanego zakończenia robót. Na podstawie odpowiedzi autoserwisów klient ma możliwość wyboru optymalnej propozycji, o czym

autoserwis będzie poinformowany. W przypadku użytkownika niezarejestrowanego wybór autoserwisu odbywa się za pośrednictwem poczty elektronicznej, natomiast w przypadku klienta zarejestrowanego odbywa się to przez funkcjonalność serwisu internetowego iremont.pl. Po zakończeniu prac strona internetowa nadaje możliwość autoserwisu poinformować klienta o zrealizowaniu zlecenia.

W wyniku samochodziarz otrzymuje na wybór kilka optymalnych propozycji, a warsztat samochodowy – klienta.

Dlatego celem tej pracy jest tworzenie systemu pośrednictwa między właścicielem samochodu a warsztatem samochodowym umożliwiającym wybór najlepszej oferty od strony klienta oraz znalezieniu większej liczby klientów od strony autoserwisu. System informatyczny realizujący takiego typu zadanie techniczne wymaga wykorzystania różnych praw dostępu do danych dla każdego użytkownika. Na rysunku 1.1 przedstawiono schemat podziału ról dla użytkowników.



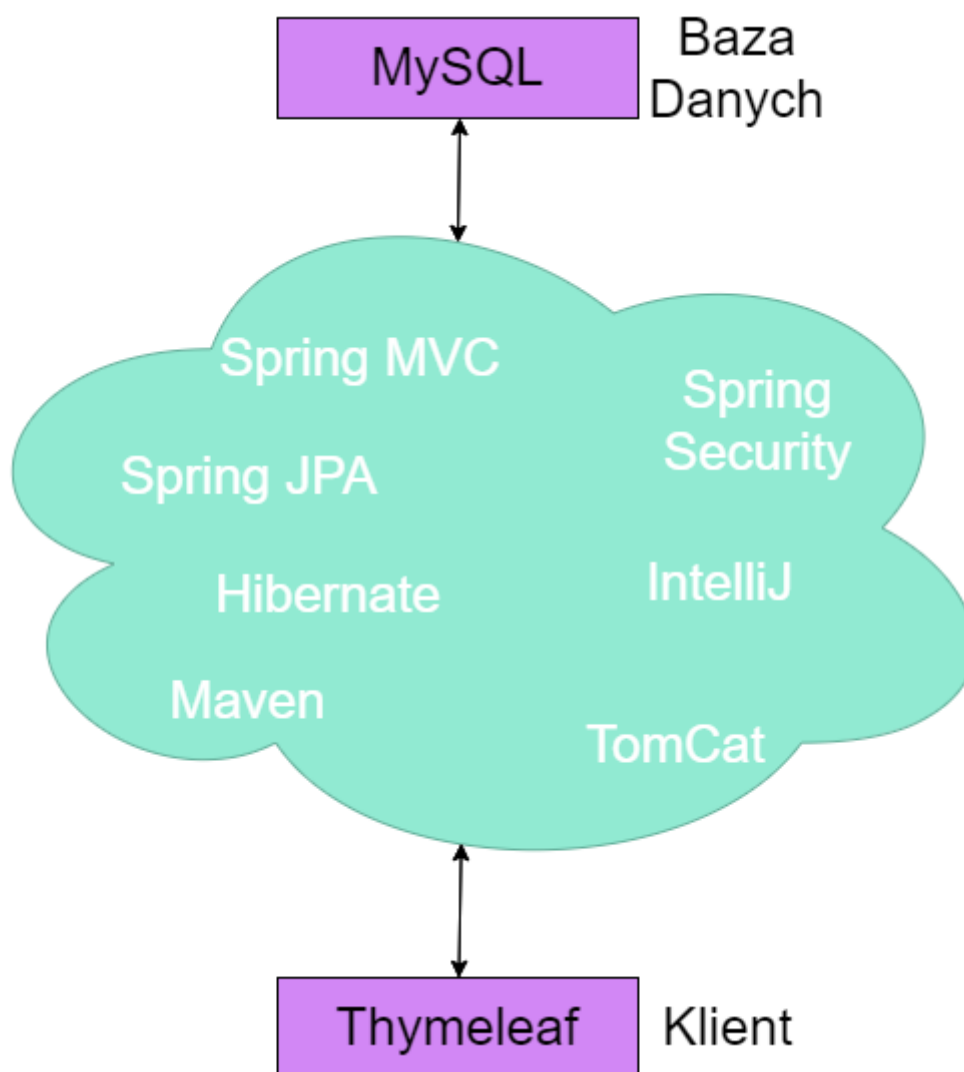
Rys. 1.1. Schemat podziału ról dla użytkowników

Praca składa się z wprowadzenia, pięciu rozdziałów, podsumowania oraz bibliografii. We wstępie został określony cel pracy oraz podano motywację do jej napisania. Kolejne rozdziały zawierają opis technologii informatycznych zastosowanych do implementacji systemu informatycznego, specyfikację funkcjonalną systemu, specyfikację techniczną bazy danych systemu, opis implementacji oraz instrukcję użytkownika.

Literatura zawiera 26 pozycji, z czego 24 to odwołania do stron w sieci Internet, a dwie po pozycje książkowe. Literatura poświęcona jest przeglądowi technologii informatycznych zastosowanych do tworzenia oprogramowania. Znajdują się pozycje opisujące język Java, frameworki Spring i Thymeleaf, technologię ORM i bibliotekę Hibernate, uwierzytelnianie użytkowników w technologii Spring.

2. Uzasadnienie wyboru narzędzi programistycznych użytych do realizacji serwisu

Jednym z najważniejszych zadań przy tworzeniu systemów informatycznych jest wybór takich narzędzi, które ułatwią pracę programisty, oferując wygodne instrumenty do realizacji postawionego zadania oraz umożliwiają zdobyć wynik, który w całości zadowala użytkownika. Przy tworzeniu systemu informatycznego było wykorzystano narzędzia programistyczne przedstawione na rysunku 2.1.



Rys. 2.1. Narzędzia programistyczne użyte do realizacji serwisu

Jak przedstawiono na rysunku 2.1, przy realizacji serwisu były wykorzystane takie narzędzia programistyczne:

- środowisko programistyczne JetBrains IntelliJ IDEA, dlatego że ma najszerszą funkcjonalność wśród konkurentów;

- język programowania Java do napisania części serwerowej;
- framework Spring, a mianowicie jego moduły – Core Container, Web oraz Data Access/Integration do organizacji architektury i współdziałania części serwerowej z częścią klienta (interfejsem użytkownika) oraz bazą danych;
- silnik szablonów Thymeleaf. Umożliwia budowę elementów interfejsu użytkownika po stronie serwera (generuje HTML);
- język arkuszy stylów CSS wraz z biblioteką Bootstrap do opracowania graficznego interfejsu użytkownika;
- narzędzie Maven do automatyzacji podłączenia całych modułów oraz bibliotek projektu;
- kontener serwletów Tomcat co pozwala zmniejszyć czas na rozwijanie projektu na hostingu albo w sieci lokalnej;
- relacyjną bazę danych MySQL jest ona darmowa, ma intuitywny interfejs i w prosty sposób integruje się ze środowiskiem programistycznym IntelliJ IDEA.

Przedstawione wyżej narzędzia programistyczne są niezależne od systemu operacyjnego – Java, HTML, Maven i MySQL, dzięki czemu uruchomienie serwisu oraz jego stabilna praca możliwa na dowolnym urządzeniu.

Językiem programowania wysokiego poziomu został wybrany język Java oprócz już wymienionej wieloplatformowości, zawiera wiele bibliotek, co pomaga ułatwić proces tworzenia serwisu. Dzięki silnemu ukierunkowaniu na programowanie obiektowe architektura oraz styl napisania kodu są standaryzowane, co jest ważnym do możliwego rozszerzenia funkcjonalności przez innych deweloperów [1].

Bazą danych została wybrana MySQL. Zaletami MySQL nad innymi SQL bazami danych z otwartym kodem źródłowym takimi jak SQLite, PostgreSQL są [2]:

- bezpieczeństwo – duża ilość funkcji bezpieczeństwa, które są domyślnie dostępne;
- skalowalność – MySQL łatwo pracuje z dużą ilością danych i łatwo się skaluje;
- szybkość – rezygnacja z niektórych standardów SQL pozwala MySQL znacznie zwiększyć wydajność;
- MySQL rekomendowany do wykorzystania do aplikacji internetowych [2][3];
- większość aplikacji internetowych, korzystają z MySQL mimo niektórych ograniczeń. Będąc lekką w konfiguracji i skalowalną bazą danych MySQL jest sprawdzona czasem.

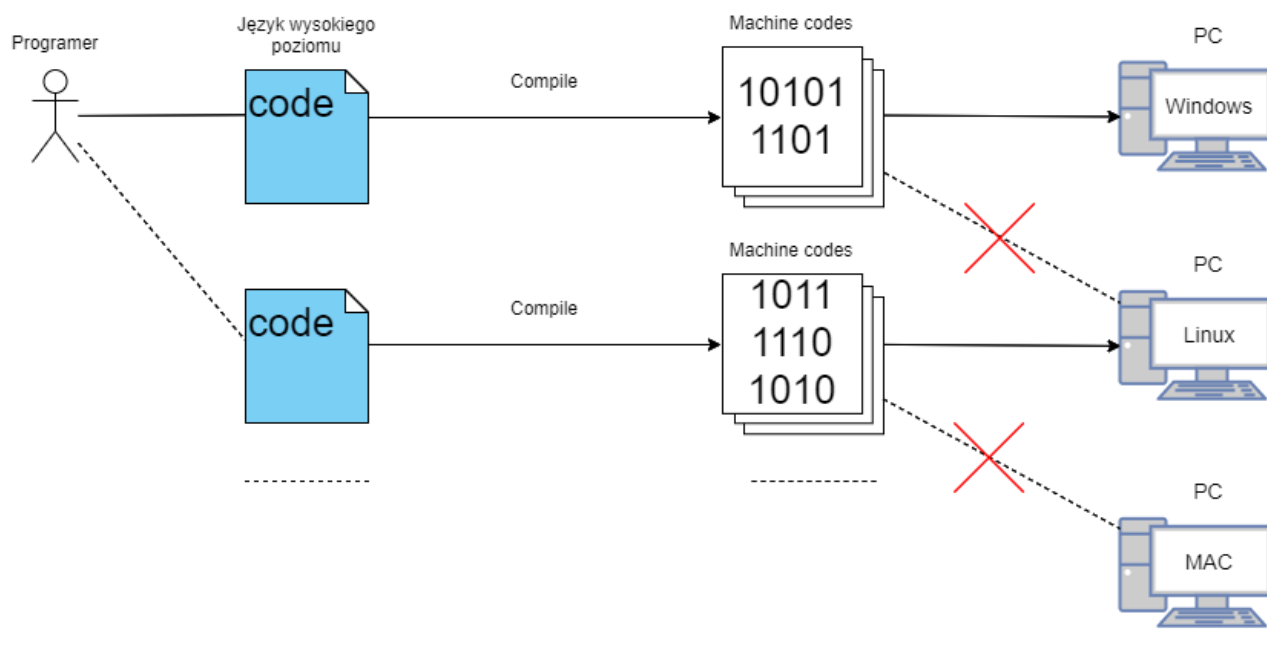
Decydującą przewagą została popularność użycia tej bazy danych.

2.1. Język programowania Java

Cała logika po stronie serwera jest napisana na języku wysokiego poziomu Java. Przede wszystkim Java – język do tworzenia dużych systemów informatycznych [4]. Możliwe to głównie dzięki niżej wymienionym cechom:

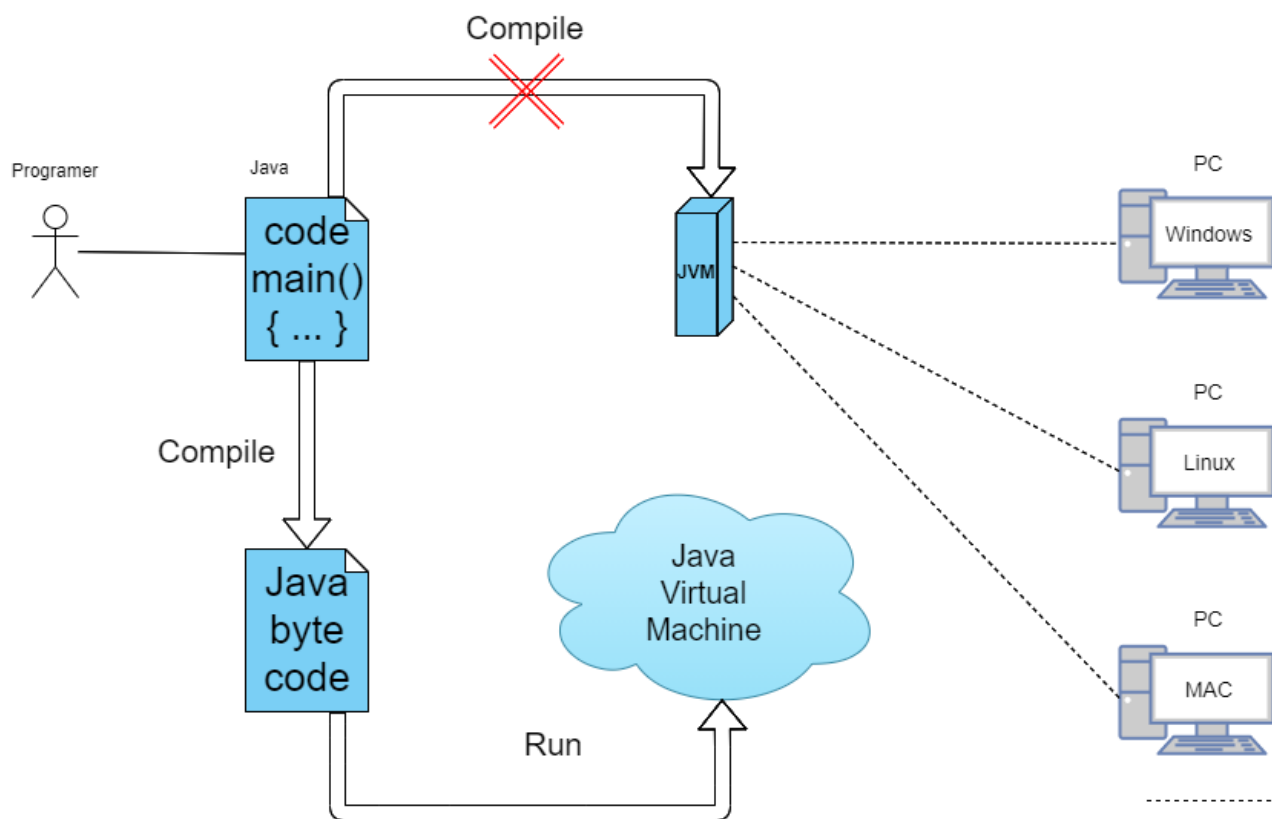
- akcent na czytelność kodu oraz jego efektywność;
- silnie ukierunkowany na programowanie obiektowy, silnie typowany język wysokiego poziomu;
- Garbage Collector służący do automatycznego zwolnienia pamięci;
- rozwinięte frameworki;
- duża ilość bibliotek, które służą do ułatwienia zrealizowania szerokiego spektrum zadań.

Inną swego czasu rewolucyjną przewagą języka Java jest wirtualna maszyna, co zabezpiecza niezależności od architektury [5]. Na rysunku 2.2 przedstawiono, jak pracują kompilowane języki programowania wysokiego poziomu [6].



Rys. 2.2. Schemat algorytmu pracy kompilowanych języków programowania

Wadą takiego podejścia jest konieczność napisania różnego kodu na języku programowania wysokiego poziomu dla różnych platform. Natomiast projektanci Java popularyzowały użycie wirtualnej maszyny, jak to przedstawiono na rysunku 2.3.



Rys. 2.3. Schemat algorytmu pracy interpretowanych języków programowania

Dzięki takiemu podejściu udało się osiągnąć slogan WORA (Write Once Run Anywhere). To oznacza, że programista może opracowywać kod Java w jednym systemie operacyjnym i spodziewać się, że on będzie pracował w dowolnym innym systemie ze wsparciem Java bez jakichkolwiek dodatkowych zmian czy konfiguracji. Jest to możliwe dzięki JVM (Java Virtual Machine).

Ważną zaletą języka we współczesnym świecie wielordzeniowych procesorów jest wielowątkowość. Za pomocą wielowątkowości możemy wydzielić w aplikacji kilka wątków, które będą wykonywały różne zadania jednocześnie. Dzięki wątkom możemy wydzielić wysyłkę żądania albo dowolne inne zadanie, które potrzebuje dużo czasu na wykonanie do osobnego wątku [7]. Dlatego nowoczesną web aplikację ciężko wyobrazić bez stosowania wielowątkowości.

W celu zapobiegania niejednoznaczności i uproszczenia rozumienia kodu z języka zostało wykluczone wielokrotne dziedziczenie. Zamiast tego wprowadzono pojęcie interfejsu, co przedstawia zestaw metod bez ich implementacji. Właściwa implementacja metod danego interfejsu znajduje się w klasie lub klasach implementujących dany interfejs [8].

Innym czynnikiem wysokiej niezawodności tego języka programowania jest system wyjątków oraz ich obsługi, które są podzielone na dwa rodzaje:

- obsługiwane wyjątki, które mogą powstać podczas pracy programy, które obowiązkowo programista musi obsłużyć (checked exceptions), wymaga tego kompilator [9]. Na przykład: nieudana próba odczytu lub zapisu do pliku związana z brakiem odpowiednich praw dostępu [10];
- nieobsługiwane wyjątki (unchecked exception) są one rzucane przez maszynę wirtualną i aplikacja nie powinna samodzielnie ich rzucać. Oznacza to, że nie ma obowiązku ich obsługi w miejscach, w których mogą wystąpić. Na przykład: próba odwołania się do pola obiektu lub metody poprzez niezainicjowaną zmienną.

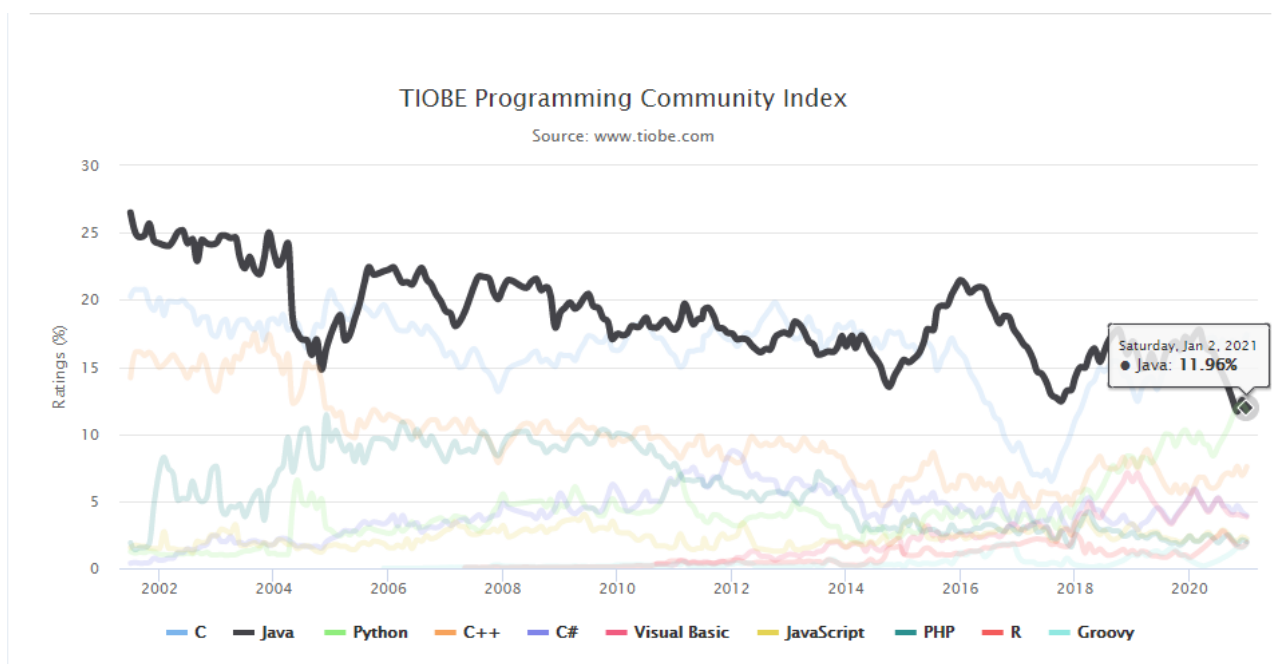
Programista Javy w zasadzie nie musi martwić się zarządzaniem pamięcią, tak jak automatyczne odświeżanie GC (garbage collection) zarządza pamięcią automatycznie [11]. GC uruchamia się wirtualną maszyną Java (JVM). Automatyczne odświeżanie – to proces w tle, który uruchamia się okresowo i zwalnia pamięć wykorzystaną obiektami, które są więcej niepotrzebne. Obiekt może podlegać utylizacji w różnych przypadkach:

- jeżeli zmienna referencyjna, która przechowuje odwołanie do obiektu równa "null";
- obiekt podlega utylizacji w tym przypadku, jeżeli do niego nie ma odwoływań;
- obiekty stworzone lokalnie w metodzie podlegają utylizacji, kiedy metod kończy pracę, jeżeli oni nie są eksportowane z tej metody (czyli metod ich zwraca albo generują się jako wyjątek);
- obiekty, które odwołują się jeden na drugiego, mogą podlegać pod utylizację, jeżeli żaden z nich niedostępny wątkowi, który został uruchomiony, ale nie został jeszcze zakończony.

Różne JVM mają odmienne algorytmy odświeżania. Można wyróżnić dwie zasadnicze grupy: skalarne i wektorowe.

Obiektowe podejście do napisania kodu pozwala operować pojęciami, co spotykają się w codziennym życiu z pewnym poziomem abstrakcji. Paradygmat OOP (object-oriented programming) przyniosła w język skalowalność, co daje możliwość nieraz rozszerzać opracowany system informatyczny. Rozszerzenie systemu polega na tym, że w system można dodawać nowe komponenty bez przemiany już istniejących. Inną zaletą tego podejścia jest wielokrotne użycie napisanego kodu, co znacznie skraca ilość napisanego kodu.

Jak widać na rysunku 2.4, stanem na styczeń 2021 roku język Java ma drugie miejsce w popularności według autorytatywnego rankingu Index TIOBE [12].



Rys. 2.4. Index popularności TIOBE języków programowania

2.2. Spring Framework

Do stworzenia architektury systemów informatycznych z użyciem interfejsu internetowego wykorzystują współczesne biblioteki czy platformę programistyczne (framework), które ułatwiają proces napisania kodu.

Platforma programistyczna (ang. software framework) – to złożenie przygotowanych do użycia rozwiązań programistycznych, które zawierają architekturę projektu, logikę oraz podstawową funkcjonalność systemu albo podsystemu.

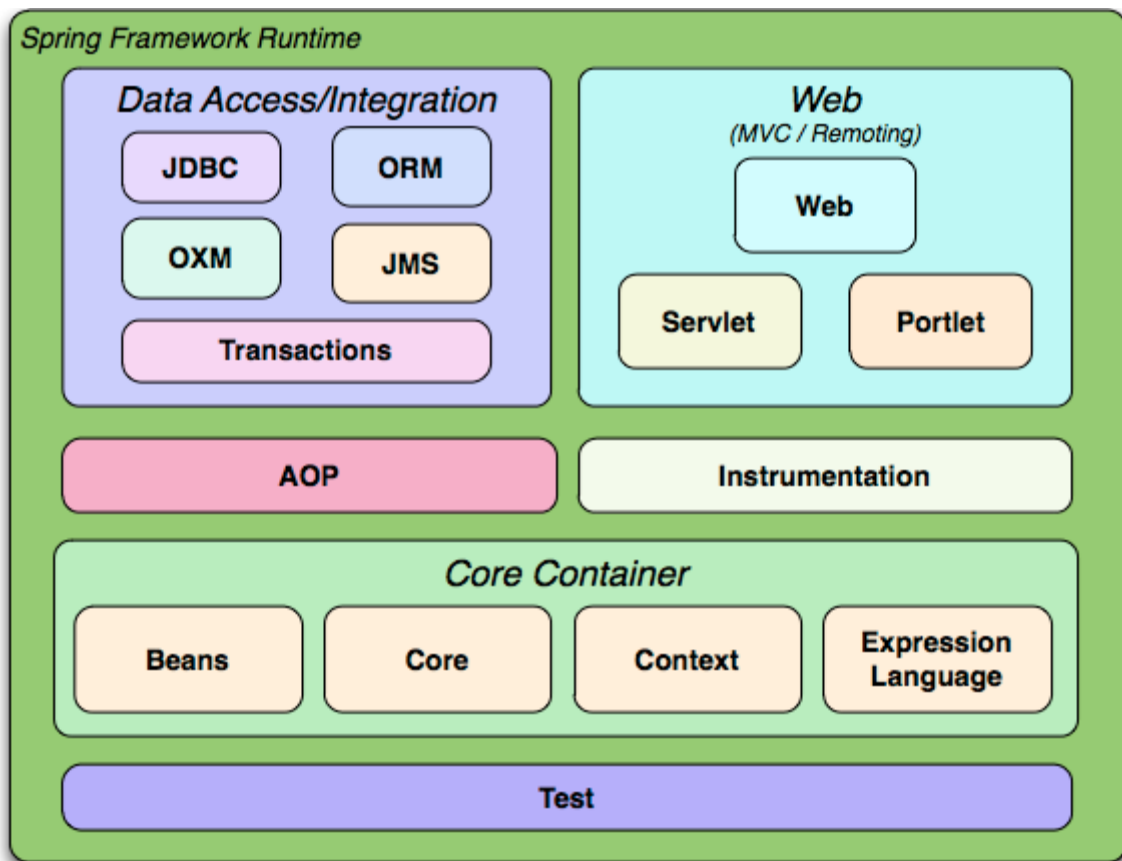
Framework zawiera zestaw bibliotek, które udostępniają gotowy zestaw decyzji, również może zawierać programy pomocnicze, skrypty i w ogóle wszystko do ułatwienia tworzenia i połączenia różnych komponentów dużego systemu informatycznego lub do szybkiego tworzenia gotowego i nieobowiązkowo objętościowego produktu programistycznego. Jedną z głównych zalet użycia frameworków jest standaryzacja struktury projektu.

Spring jest ogólnodostępnym frameworkiem z otwartym kodem źródłowym, został stworzony w celu uproszczenia tworzenia systemów informatycznych, zabezpiecza kompleksowy model opracowania i konfiguracji do tworzenia współczesnych aplikacji na Java. Zasadniczy element Spring – to wsparcie infrastruktury na poziomie programu: programista może skupić się na implementacji logiki biznesowej bez potrzeby pisania dodatkowego kodu do konfiguracji środowiska [6]. Spring można wykorzystywać do budowy dowolnego programu na języku Java, co pozytywnie wyróżnia Spring Framework spośród innych.

W celu uproszczenia tworzenia projektów w języku Java, Spring zawiera następujące podstawowe cechy:

- **wstrzykiwanie zależności (ang. Dependency Injection, DI)** – słabe związkiwanie (bez bezpośrednich zależności) przez iniekcję zależności przez konstruktora albo settery (ang. setter) i organizacja współdziałania przez interfejsy, to podejście pozwala łatwiej testować aplikacje oraz koordynować pracę każdego obiektu w systemie od trzeciej strony;
- **odwrócenie sterowania (ang. Inversion of Control, IoC)** – co pozwala pisać niezależne komponenty, jest to zaletą przy tworzeniu aplikacji w zespole, przenoszeniu kodu oraz zamianie modułów;
- **programowanie aspektowe (aspect-oriented programming, AOP)** – paradygmat programowania oparty na idei separacji głównej (biznes) oraz służbowej funkcjonalności. Służbowa funkcjonalność zapisuje się do Aspekt klasy;
- **POJO (Plain Old Java Object)** – użycie prostych Java obiektów bez obowiązku realizować czy dziedziczyć specyficzne dla frameworku interfejsy czy klasy;
- Spring występuje w postaci kontenera dla obiektów, więc zarządza ich cyklem życia.

Spring składa się z modułów (rys. 2.5), które są niezależne oprócz modułu Core Container, który jest jądrem i do niego dołączają inne moduły.



Rys. 2.5. Moduły Spring Framework [13]

Moduł Data Access/Integration składa się z komponentów, co odpowiadają za wsparcie z bazą danych. Zawiera pięć podmodułów, spośród których trzy są w naszym przypadku kluczowe do współpracy z bazą danych:

- ORM (Object-relational mapping) – mapowanie obiektowo-relacyjne, zapewnia warstwę realizacji współdziałania z popularnymi bibliotekami do odwzorowania obiektowej architektury systemu informatycznego na bazę danych o relacyjnym charakterze [14]. Na przykład z: JPA, JDO, Hibernate oraz iBatus.
- JDBC (Java DataBase Connectivity) stwarza warstwę realizacji standardu łączy do baz danych w języku Java (JDBC).
- Transaction jak wynika z jego nazwy – za wsparcie i obróbkę transakcji.

Moduł AOP dodaje wsparcie programowania aspektowego, co zapewnia możliwość jeszcze bardziej elastycznie skonfigurować aplikacje.

Moduł Instrumentation zapewnia możliwość kontrolować wydajność aplikacji.

Moduł Test zawiera biblioteki do napisania modułowych oraz testów integracyjnych.

2.2.1. Moduł Spring Core Container

Spring Core Container moduł składa się z czterech składowych: Beans, Core, Context, SpEL.

Beans i Core zapewniają najbardziej fundamentalne części frameworku – wstrzykiwanie zależności oraz odwrócenie sterowania oprócz tego zapewniają realizację wzorca projektowego fabryka, co usuwa konieczność realizować wzorzec projektowy Singleton i wydobywa konfigurację ze specyfikacją od biznes logiki programu.

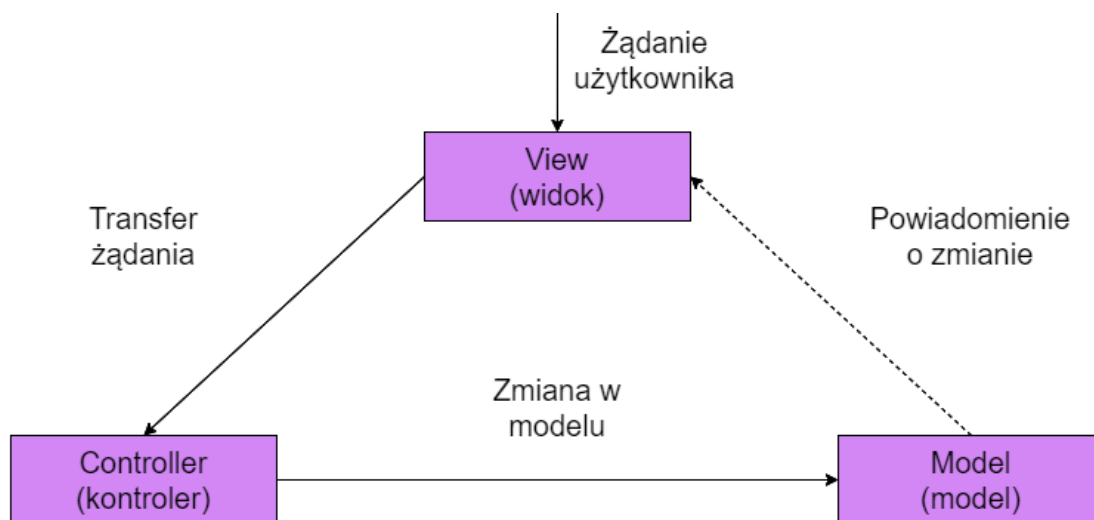
Wstrzykiwanie zależności (ang. Dependency Injection, DI) – wzorzec projektowania oraz model architektoniczny, który opisuje sytuację, kiedy jeden obiekt realizuje swoją funkcjonalność przez inny obiekt. Obiekt przekazuje sterowanie tworzeniem koniecznych jemu zależności zewnętrznemu specjalnie do tego przeznaczonemu mechanizmu. Na przykład łączenie z bazą danych udziela się konstruktorowi obiektu przez argument zamiast tego, żeby konstruktor samodzielnie nawiązywał połączenie. Zaletami tego podejścia jest to, że on pozwala oddzielić obiekty od realizacji mechanizmów, które on wykorzystuje, skutkiem czego jest elastyczność w tworzeniu oprogramowania.

Context zapewnia możliwość otrzymywać dostęp do obiektów i dodaje wsparcie internacjonalizacji.

SpEL (ang. Spring Expression Language) zapewnia wsparcie języków zapytań, mianowicie: ustalenie oraz otrzymanie wartości właściwości, odczyt z plików konfiguracyjnych, wezwanie metody, dostęp do kontekstu tablic, kolekcji oraz indeksów, logicznych i arytmetycznych operacji, poszukiwanie obiektów po imieniu.

2.2.2. Moduł Spring WEB

Moduł Web udostępnia główną funkcjonalność aplikacji webowych. Komponent Web-Servlet zapewnia realizację wzorca architektonicznego służącego do organizowania struktury aplikacji MVC (Model-View-Controller – model-widok-kontroler) i HTTP protokołu do aplikacji webowej. Na rysunku 2.6 przedstawiono architekturę wzorca MVC.



Rys.2.6. Wzorzec architektoniczny MVC

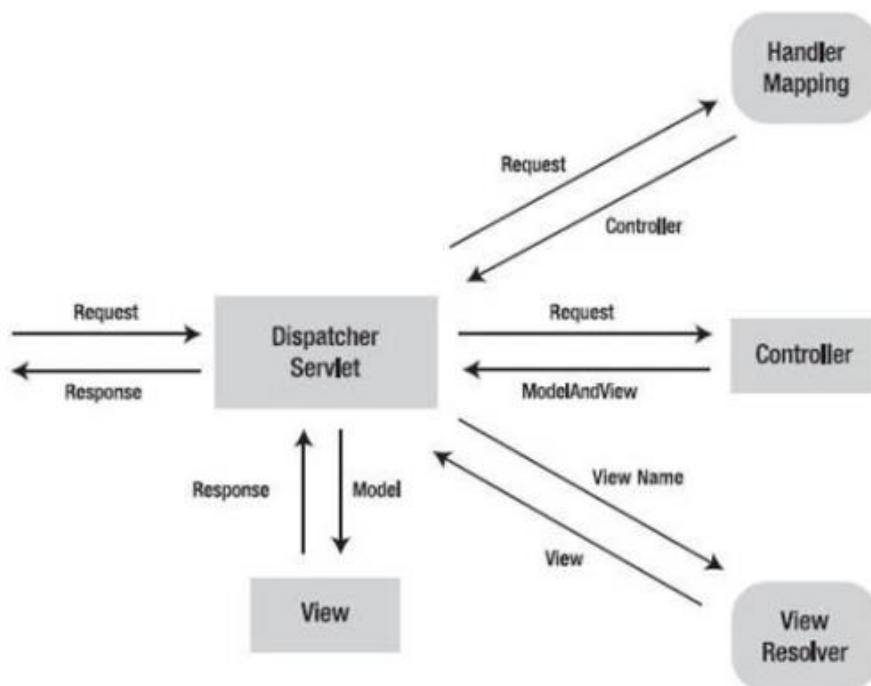
- Model (model) zawiera dane, które programista chce wyświetlić za pomocą widoków lub dane odpowiedzi od klienta aplikacji webowej. Dane przedstawione w postaci POJO (Plain Old Java Object – zwykły stary obiekt Java).
- View (widok) HTML szablon (szkielet) strony napisanej za pośrednictwem biblioteki silnika szablonów. Takie biblioteki umożliwiają dostęp do danych modelu.
- Controller (kontroler) – odpowiada na HTTP żądanie. Przetwarza http żądanie do obiektu Java oraz obiekty Java w HTML odpowiedź [15].

Dzięki temu wzorcu architektonicznemu możliwa wyraźna separacja logiki biznesowej a form webowych. Web Socket – moduł co umożliwia poparcie dwustronnej komunikacji pomiędzy klientem i serwerem.

W Spring Web MVC jest możliwość wykorzystywać dowolny obiekt w postaci rozkazu albo obiektu ze wstecznym związkiem; nie ma konieczności implementacji żadnych specjalnych interfejsów frameworka albo klasy bazowej. Dzięki elastyczności związywanie danych w Spring niezgodność typów rozpatruje się jako błędy walidacji, dlatego mogą one być obsługiwane w programie, a nie w jakości błędów systemowych. Znaczy to, że nie ma potrzeby dublować właściwości biznes-obiektów w postaci prostych nietypizowanych pól do obiektów form.

Ten moduł Spring jest zbudowany wokół głównego serwleta, który nazywa się DispatcherServlet, jego funkcją jest rozdzielenie żądań między kontrolerami. W nim konfiguruje się obsługa żądań, lokalizacja oraz strefy czasowe. DispatcherServlet jest serwletem realizującym protokół HTTP. Na rysunku 2.7 przedstawiony schemat współpracy tego serwletu z aplikacją. Najpierw żądanie skierowane na DispatcherServlet potem on przekazuje

jego w HandlerMapping, który przegląda wszystkie istniejące kontrolerzy i wybiera ten, który wie, jak obsłużyć to żądanie oraz zwraca jego imię z powrotem na DispatcherServlet.



Rys.2.7. Schemat algorytmu działania DispatcherServlet [16]

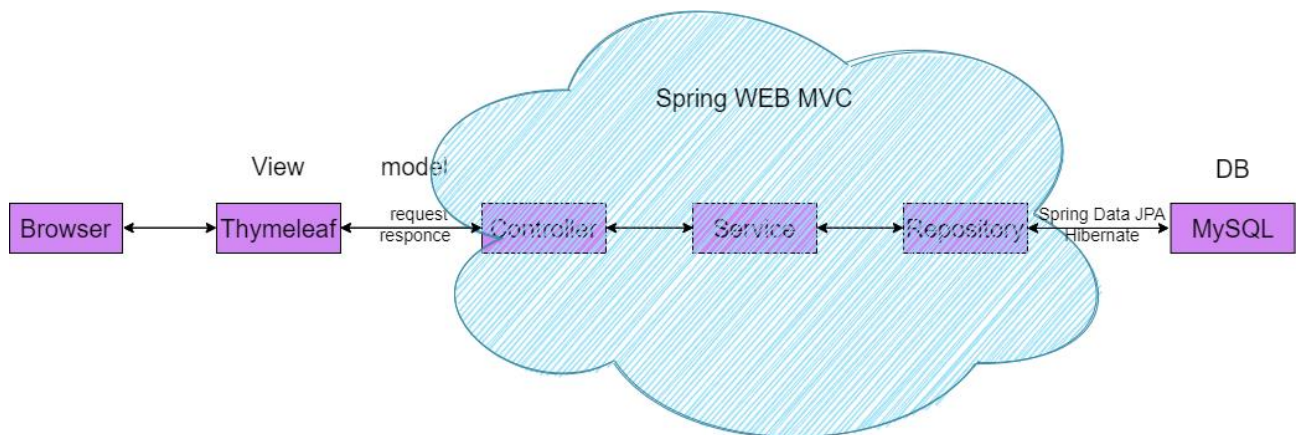
W przypadku znalezienia dwóch kontrolerów, które mają konieczną funkcjonalność, obsługa żądania niemożliwa, na etapie uruchomienia projektu programiście będzie zwrócony błąd. Po otrzymaniu imienia kontrolera żądanie udziela się jemu do obsługi. W kontrolerze odbywa się proces obsługi żądania i zwraca się obiekt ModelAndView, który zawiera dane oraz sposób ich wyświetlenia. DispatcherServlet na podstawie tego obiektu szuka odpowiedniego widoku (imię obiekt typu View) za pomocą ViewResolver. Dzięki czemu DispatcherServlet dysponuje informacją, w który widok przesłać model (obiekt typu Model) zawierający dane. Od widoku DispatcherServlet otrzymuje odpowiedź, która potem zostaje wysłana na stronę serwerową aplikacji, jeżeli jest potrzebna.

Spring organizuje architekturę projektu przez zobowiązanie programisty wskazywać poprzez adnotacje, do którego typu klas należy dana klasa, jeżeli bowiem nie wskazać typ to framework nie będzie umieszczać dany klas w swój kontener, a więc w nim nie będą pracować metody z bibliotek, które oferuje Spring. Istnieją 4 adnotacje tworzące architekturę projektu:

- @Component wskazuje przynależność klasy do frameworku;
- @Controller wskazuje na klasę, w której DispatcherServlet będzie szukał metodę do obsługi żądania;

- `@Service` wskazuje się klasa, w której zrealizowana biznes logika aplikacji nie różni się od `@Component`, lecz pozwala wskazać miejsce klasy w architekturze projektu;
- `@Repository` wskazuje na klasę będącą wykorzystywaną się do pracy z poszukiwaniem, otrzymaniem oraz przechowywaniem danych. Klasy z taką adnotacją wykorzystują się w celu organizacji współdziałania z bazą danych.

Wynikiem wykorzystania wyżej przedstawionego frameworku jest przedstawiona na rysunku 2.8 architektura projektu.



Rys.2.8. Architektura projektu z wykorzystaniem Spring Framework

2.2.3. Spring Data JPA

Spring Data JPA jest implementacją oficjalnego standardu Java Persistence API. Spring Data JPA realizuje warstwę dostępu do danych, która może być dość sporą. Zbyt wiele szablonowego kodu pisze się do realizacji takich zadań jak paginacja oraz audyt. Spring Data JPA w znacznym stopnie ulepszy realizację warstwy dostępu do danych, zezwoli skupić się na biznes funkcjonalności. Programista pisze repozytorium interfejs zawierający własne metody poszukiwania, a Spring zabezpiecza ich automatyczną realizację [17]. Spring Data JPA zapewnia szereg interfejsów, w których już opisane najbardziej często wykorzystywane metody pracy z bazą danych. Będę rozpatrzone na przykładzie jednego z głównych interfejsów – `JpaRepository` (rys. 2.9).

```

@NoRepositoryBean
public interface JpaRepository<T, ID> extends PagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T> {
    List<T> findAll();

    List<T> findAll(Sort var1);

    List<T> findById(Iterable<ID> var1);

    <S extends T> List<S> saveAll(Iterable<S> var1);

    void flush();

    <S extends T> S saveAndFlush(S var1);

    void deleteInBatch(Iterable<T> var1);

    void deleteAllInBatch();

    T getOne(ID var1);

    <S extends T> List<S> findAll(Example<S> var1);

    <S extends T> List<S> findAll(Example<S> var1, Sort var2);
}

```

Rys. 2.9. Interfejs JpaRepository

Dziedzicząc od tego interfejsu, programista otrzymuje gotową realizację metod CRUD (Create, Read, Update, Delete) – utwórz, aktualizuj, usuń dane z bazy oraz metody poszukiwania całych tabel elementami, oraz poszukiwanie elementu przez Id z bazy danych. W celu tworzenia zapytania wbudowanego (metody w interfejsie dziedziczącym interfejs Spring Data JPA) sygnatura metody musi składać się ze specjalnych słów kluczowych. Umożliwiają one nam tworzenie zapytań na podstawie mechanizmu składania zapytań z nazwy akcji (find...By, read...By, query...By, count...By, get...By, delete...By) oraz części własnej w postaci nazw pól encji, jak przedstawiono na rysunku 2.10 [18].

```

User findByToken(String token);

Long countByLastname(String lastname);

Stream<User> readByEmail(String email);

Long deleteByLastname(String lastname);

List<Item> findFirst10ByLastname(String lastname, Sort sort);

List<Item> findTop3ByLastname(String lastname, Sort sort);

```

Rys. 2.10. Przykład zapytań wbudowanych [18]

Na rysunku 2.7 też przedstawione wyrażenia limitujące takie jak Top and First. Przy ich wykorzystaniu należy podać liczbę rekordów, jakie chcemy otrzymać oraz **kierunek sortowania** [18].

Jeżeli podczas tworzenia kodu aplikacji zdarza się tak, że zapytań wbudowanych nie wystarcza do rozwiązywania naszych zadań, możemy tworzyć zapytania własne. Treść zapytania zostanie przekazana do wykonania poprzez użycie adnotacji `@Query`. Na rysunku 2.11 przedstawiono przykład tworzenia zapytania własnego.

```

public interface ItemRepository extends JpaRepository<Item, Long> {

    @Query("select ai from Item ai where ai.name LIKE :searchValue "
        + "OR cast(ai.id as string) LIKE :searchValue "
        + "OR FORMATDATETIME(ai.dateTimeFrom, 'dd/MM/yyyy') LIKE :searchValue "
        + "OR FORMATDATETIME(ai.dateTimeTo, 'dd/MM/yyyy') LIKE :searchValue")
    Page<Item> findAll(@Param("searchValue") String searchValue, Pageable pageable);

    ...
}

```

Rys. 2.11. Przykład zapytania własnego [19]

2.3. Spring Security

Spring Security jest frameworkiem, który udostępnia głównie funkcjonalność weryfikacji oraz autoryzacji w aplikacjach Java. Jak i w przypadku wszystkich Spring projektów główną zaletą Spring Security jest to, że on może być łatwo uzupełniony potrzebną funkcjonalnością. Główne możliwości frameworku Spring Security [20]:

- kompleksowe i rozszerzane wsparcie jak weryfikacji tak i autoryzacji;
- ochrona od ataków typu zafiksowanie sesji, clickjacking, CSRF (ang. Cross-site request forgery) itd;
- integracja z Servlet API;
- integracja ze Spring Web MVC.

Podstawowym obiektem jest `SecurityContextHolder`. W nim przechowuje się informacja o bieżącym kontekście bezpieczeństwa aplikacji, on zawiera dokładną informację o użytkowniku pracującym z aplikacją. Spring Security wykorzystuje obiekt `Authentication` użytkownika autoryzowanej sesji.

`User` jest obiektem klasy `Object`. W większości przypadków on może być doprowadzony do klasy `UserDetails`. `UserDetails` można przedstawić jako adapter pomiędzy bazą danych użytkowników i tym, co wymaga Spring Security wewnątrz `SecurityContextHolder`.

W celu tworzenia `UserDetails` wykorzystuje się interfejs `UserDetailsService` z jedyną metodą przedstawioną na rysunku 2.12.

```
UserDetails loadUserByUsername(String username) throws UsernameNotFoundException
```

Rys. 2.12. Metoda `loadUserByUsername` [21]

Kluczowe obiekty Spring Security [21]:

- `SecurityContextHolder` zawiera informacje o bieżącym kontekście bezpieczeństwa programu, dokładną informację o użytkowniku, który obecnie pracuje z aplikacją. Domyślnie `SecurityContextHolder` wykorzystuje `ThreadLocal` do przechowywania takiej informacji, oznacza to, że kontekst bezpieczeństwa jest zawsze dostępny metodom dlatego, że przechowuje się w bieżącym wątku;
- `SecurityContext` zawiera obiekt `Authentication` oraz w przypadku konieczności informację systemu bezpieczeństwa powiązaną z żądaniem od użytkownika;
- `Authentication` przedstawia użytkownika z punktu widzenia Spring Security;
- `GrantedAuthority` zawiera zezwolenia wydane użytkownikowi pod względem całej aplikacji, takie zezwolenia z reguły nazywają się "role" na przykład `ROLE _ ANONYMOUS`, `ROLE _ USER`, `ROLE _ ADMIN`;

- UserDetails nadaje konieczną do budowy obiektu Authentication informację z obiektów klas Repository. Obiekt UserDetails zawiera imię użytkownika, hasło oraz flagi: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled oraz Collection – ról użytkownika;
- UserDetailsService wykorzystuje się w celu stworzenia UserDetails obiektu przez realizację jedynej metody tego interfejsu.

2.4. Spring Boot

Spring Boot jest projektem, który zapewnia ustalony zestaw frameworków w celu zmniejszenia szablonej konfiguracji (boilerplate configuration) oraz uruchomienia aplikacji Spring z minimalną ilością kodu. Takie podejście pozbawia dewelopera od trwałej konfiguracji i podłączenia niezbędnych zależności. W wyniku to wszystko ulepsza proces testowania oraz integracji.

2.4.1. Spring Boot starters

Startery są ważną częścią Spring Boot, która wykorzystuje się do ograniczenia ilości ręcznie konfigurowanych zależności, które są potrzebne do funkcjonowania projektu. Starter jest zestawem zależności (w naszym przypadku Maven POM) specyficznych dla typu programu, którą przedstawia starter [22]. Poniżej przedstawiono listę popularnych starterów Spring Boot:

- spring-boot-starter-web wykorzystuje się w celu tworzenia RESTful aplikacji webowych, wykorzystując Spring MVC oraz Tomcat jako wbudowany kontener serwerów;
- spring-boot-starter-jersey jest alternatywą springboot-starter-web, która wykorzystuje Apache Jersey zamiast Spring MVC;
- spring-boot-starter-jdbc wykorzystuje się w celu łączenia połączeń JDBC. Polega on na wprowadzeniu wyciągu połączeń JDBC od Tomcat.

2.4.2. Konfiguracja automatyczna

Jedną z największych zalet Spring Boot jest konfiguracja automatyczna. Ona zaczyna się z dodania adnotacji `@EnableAutoConfiguration`. Podstawą konfiguracji automatycznej są JAR-pliki znajdujące się w classpath dewelopera oraz na klasach oznaczonych przez adnotacje, z których powinny być stworzone beany.

2.4.3. Zalety Spring Boot

Na rysunku 2.13 przedstawiony schemat działania oraz konfiguracji Spring Framework. Zalety Spring Boot:

- tworzenia aplikacji bazujących na Spring za pomocą Java albo Groovy o wiele łatwiejsze;
- zmniejsza ilość czasu na tworzenie oraz zwiększa wydajność aplikacji;
- zapewnia możliwość uniknąć napisania dużej ilości kodu, adnotacji oraz XML-konfiguracji;
- integrować aplikacje Spring Boot bardzo łatwo z jego ekosystemem Spring na przykład Spring JDBC, Spring ORM, Spring Data, Spring Security;
- przestrzega podejścia "Opinionated Defaults Configuration" w celu ułatwienia pracy programisty [22];
- zapewnia wbudowane HTTP-serwery takie jak domyślnie Tomcat, Jetty w celu ułatwienia tworzenia oraz testowania aplikacji webowych;
- zapewnia mnóstwo pluginów do tworzenia oraz testowania aplikacji Spring za pomocą narzędzi do automatyzacji takich jak Maven i Gradle;
- zapewnia mnóstwo pluginów do pracy ze wbudowanymi oraz in-memory bazami danych.



Rys. 2.13. Schemat działania oraz konfiguracji Spring Framework

Głównym celem Spring Boot Framework jest skrócenie czasu na tworzenie, testowanie, integrację oraz ułatwienie opracowania aplikacji webowych. Możliwości zapewnione przez Spring Boot do realizacji tego celu:

- w całości uniknąć XML konfiguracji;
- nie pisać dużo importów;
- zapewnia konfiguracje oraz komponenty domyślne w celu szybkiego uruchomienia nowych projektów w ciągu krótkiego czasu.

Czyli Spring Boot Framework skraca czas tworzenia, trud programisty oraz podwyższa wydajność.

2.5. Relacyjne bazy danych

Relacja baza danych (Relational DataBase) zawiera zbiór elementów z wyraźnie określonymi relacjami między nimi [23]. Te elementy są zorganizowane jako zestaw tablic zawierających kolumny oraz wierszy.

Relacyjne bazy danych są oparte na modelu relacyjnym — jest to prosty i intuicyjny sposób przedstawiania danych w tabelach. W relacyjnej bazie danych każdy wiersz tabeli jest rekordem z unikatowym identyfikatorem nazywanym kluczem. Kolumny tabeli zawierają atrybuty danych, a każdy rekord zawiera zwykle wartość dla każdego atrybutu, co ułatwia ustalenie relacji między poszczególnymi elementami rekordu [24].

Tablice w bazie danych są logicznie powiązane, co ułatwia poszukiwanie danych, organizację i sprawozdawczość. RDB wykorzystują SQL, która jest uniwersalnym językiem dla różnych systemów zarządu bazami danych, co umożliwia prosty interfejs programowania dla współdziałania z bazami danych [25].

Aby istniała możliwość utworzenia z tabel relacyjnego modelu danych, przynajmniej w jednej z nich musi występować klucz główny (zwany też podstawowym) – kolumna służąca do identyfikacji poszczególnych rekordów tabeli. Wartości w kluczu podstawowym muszą być unikalne, aby istniała możliwość przypisania jednego wiersza tabeli do jednej wartości klucza, co przedstawiono na rysunku 2.14, gdzie kluczem podstawowym jest kolumna "id".

	id	description	creation_date	car_id	customer_detail_id	city_id
	1	Wymiana klocków hamulcowych.	2021-02-23 15:24:00	2	2	1
	2	Usługi lakiernicze. Malowanie zderzaka tył.	2021-02-23 15:24:00	2	2	1
	3	Wymiana oleju.	2021-02-23 15:26:00	4	3	1
	4	Problemy z pompą wysokiego ciśnienia.	2021-02-23 15:26:00	4	3	1
	5	Bez gwiazdy nie ma jazdy.	2021-02-23 15:27:00	6	4	1
▶*		NULL	NULL	NULL	NULL	NULL

Rys. 2.14. Wierszy z przykładowej tabeli

W powyższym przykładzie kluczem podstawowym jest klucz jednopolowy – identyfikacja rekordu odbywa się przy pomocy jednego pola w wierszu. Istnieją także klucze złożone (wielopolowe) – w ich przypadku identyfikacja odbywa się przy pomocy więcej niż jednej kolumny. Unikalne jest zestawienie komórek tworzących klucz w wierszu (rys. 2.15).

	customer_id	car_id	car_verified
▶	2	1	1
	2	3	3
	3	3	1
	4	3	2
	4	5	1
*	NULL	NULL	NULL

Rys. 2.15. Wierszy z przykładowej tabeli z kluczem złożonym

Na rysunku 2.15 możemy zauważyć, że ani kolumna "customer_id", ani "car_id" nie zawiera unikalnych wartości, jednakże zestawienie wartości w tych dwóch komórkach jest unikalne dla każdego wiersza.

Relację ustanawiamy pomiędzy dwoma tabelami na podstawie wartości klucza podstawowego w jednej tabeli i kolumny w drugiej tabeli zawierającej wartości klucza podstawowego z tabeli pierwszej. Wyróżniamy cztery rodzaje relacji:

- jeden do jednego (One to One): jeden rekord tablicy dotyczy innego rekordu w innej tablicy;
- jeden do wielu (One to Many): jeden rekord tablicy jest związany z wieloma rekordami w innej tablicy;
- wiele do jednego (Many to One): więcej niż jeden rekord tablicy dotyczy innego jednego rekordu w innej tablicy;
- wiele do wielu (Many to Many): więcej niż jeden rekord tablicy dotyczy więcej niż jednego rekordu w innej tablicy.

RDB wykonuje operacje nad bazą danych "select", "update" i "join", gdzie "select" wykorzystuje się dla poszukiwania danych, "update" zmienia dane a "join" łączy ich kombinacje.

RDB mają wiele innych zalet na przykład:

- łatwo rozszerza się, ponieważ nowe dane mogą być dodane bez zmiany istniejących zapisów (skalowalność);
- wydajność i elastyczność;
- bezpieczeństwo danych, które są krytyczne, kiedy wymiana danymi jest konfiden-
cjonalna.

Na przykład kierownictwo może podzielać pewne przywileje danych i zablokować do-stęp pracownikom do danych, takich jak konfidencjonalna informacja o pensję albo pomoc.

2.6. Technologia Thymeleaf

Thymeleaf to biblioteka, która umożliwia przechowywanie html szablonów na stronie serwera oraz ich zwrócenie za wezwaniem java klasów kontrolerów.

Klient w przeglądarce widzi html widok odpowiedni do jego żądania zwrócony stroną serwerową. Cała widoczna dla klienta zawartość strony internetowej przechowywana jest w postaci pliku z rozszerzeniem html, który składa się z html tegów. Taki właśnie plik przechowuje na stronie serwerowej silnik szablonów Thymeleaf. On nie tylko umożliwia przechowanie takiego pliku, a również zawiera funkcjonalność do zarządzania jego zawartością.

Thymeleaf umożliwia jak przeglądanie statycznej strony internetowej w przeglądarce tak i może pozwolić programiście przeglądać dynamiczny efekt strony z danymi na serwerze.

Ważną zaletą biblioteki Thymeleaf jest to, że jej twórcy zapewniają integrację ze Spring MVC i Spring Security dzięki czemu programista może w transparentny sposób przekazywać dane od części serwerowej do części klienta i na odwrót.

3. Specyfikacja systemu

3.1. Użytkownicy oraz ich uprawnienia

System zawiera następujące typy użytkowników:

- niezalogowany – użytkownik ma dostęp do przeglądania strony domowej serwisu, tworzenia zleceń, logowania oraz rejestracji;
- klient – użytkownik ma dostęp do zarządzania własnym kontem, tworzenia oraz zarządzania zleceniami oraz historią zleceń, dodania oraz usuwania samochodów;
- warsztat samochodowy – użytkownik ma dostęp do zarządzania własnym kontem, zarządzania zleceniami oraz historią zleceń;
- administrator – użytkownik ma dostęp do zarządzania kontami innych użytkowników, zarządzania listami dostępnych do wyboru miast oraz samochodów.

3.1.1. Specyfikacja aplikacji

Opis funkcjonalności systemu:

- 1) Rejestracja – umożliwia założenie konta klienta lub warsztatu samochodowego w aplikacji, wymaga wprowadzenia następujących danych:
 - nazwa użytkownika;
 - hasło;
 - dane personalne:
 - W przypadku rejestracji nowego użytkownika „Klient”:
 - imię;
 - nazwisko;
 - W przypadku rejestracji nowego użytkownika „Warsztat samochodowy”:
 - nazwa firmy;
 - dane kontaktowe:
 - W przypadku rejestracji nowego użytkownika „Klient”:
 - adres e-mail, wymagana weryfikacja;
 - numer telefonu;
 - W przypadku rejestracji nowego użytkownika „Warsztat samochodowy”:
 - adres;
 - adres e-mail, wymagana weryfikacja;

- numer telefonu.

Dostęp: użytkownik niezalogowany.

- 2) Edytowanie danych personalnych oraz danych kontaktowych.

Dostęp: klient oraz warsztat samochodowy.

- 3) Zmiana adresu e-mail – pozwala na zmianę dotychczasowego adresu e-mail, wymagany jest potwierdzenie nowego adresu e-mail.

Dostęp: klient oraz warsztat samochodowy.

- 4) Zmiana hasła – pozwala na zmianę hasła po wpisaniu dotychczasowego hasła.

Dostęp: klient oraz warsztat samochodowy.

- 5) Zapamiętaj mnie – umożliwia automatyczne logowanie bez widoku strony logowania.

Dostęp: klient, warsztat samochody oraz administrator.

- 6) Resetowanie hasła – możliwość zresetowania hasła po wpisaniu nazwy użytkownika przez wysłanie formy podania nowego hasła na adres e-mail użytkownika.

Dostęp: klient, warsztat samochodowy.

- 7) Tworzenie zlecenia – możliwość tworzenia nowego zlecenia na naprawę samochodu.

Dostęp: użytkownik niezalogowany, klient.

- 8) Oglądanie listy aktualnych zleceń – pozwala na obejrzenie listy nowostworzonych zleceń wraz ze szczegółami każdego zlecenia.

Dostęp: klient, warsztat samochodowy.

- 9) Wysyłanie informacji o nowym zleceniu na adres e-mail – użytkownik otrzymuje na swój adres e-mail pismo zawierające wszystkie szczegóły dotyczące nowostworzonego zlecenia.

Dostęp: warsztat samochodowy.

- 10) Usuwanie aktualnego zlecenia – funkcjonalność umożliwia usuwanie jeszcze niezrealizowanego zlecenia.

Dostęp: klient.

- 11) Oferowanie warunków zrealizowania zlecenia – funkcjonalność umożliwia oferowanie terminu oraz ceny realizacji zlecenia.

Dostęp: warsztat samochodowy.

- 12) Wybór oferty – możliwość zapoznania się ze wszystkimi ofertami realizacji zlecenia oraz wyboru optymalnej.

Dostęp: klient.

- 13) Wysłanie oferty na adres e-mail – użytkownik otrzymuje na swój adres e-mail pismo zawierające wszystkie szczegóły dotyczące oferty zaproponowanej przez warsztat samochodowy oraz link umożliwiający wybór warsztatu samochodowego do realizacji zlecenia.
Dostęp: użytkownik niezalogowany, klient.
- 14) Oglądanie bieżących zleceń – możliwość oglądania zleceń do realizacji których został wybrany aktualnie zalogowany użytkownik wraz ze szczegółami każdego zlecenia.
Dostęp: warsztat samochodowy.
- 15) Oznaczenie zlecenia jako zrealizowane – możliwość oznaczenia przez użytkownika zlecenia jako zrealizowane.
Dostęp: warsztat samochodowy.
- 16) Wysłanie informacji o zrealizowaniu zlecenia na adres e-mail – użytkownik otrzymuje na swój adres e-mail pismo zawierające informację o zrealizowaniu zlecenia.
Dostęp: użytkownik niezalogowany, klient.
- 17) Oglądanie historii zleceń – umożliwia obejrzenie już zrealizowanych zleceń wraz ze szczegółami każdego zlecenia.
Dostęp: klient, warsztat samochodowy.
- 18) Usuwanie zarchiwizowanego zlecenia – funkcjonalność do usuwania już zrealizowanego zlecenia z listy.
Dostęp: klient, warsztat samochodowy.
- 19) Dodanie samochodu – funkcjonalność umożliwia wybranie dodanego samochodu przy tworzeniu zlecenia.
Dostęp: klient.
- 20) Oglądanie moich samochodów – możliwość oglądania listy moich samochodów oraz informacji dotyczącej każdego pojazdu.
Dostęp: klient.
- 21) Usuwanie samochodu – możliwość usunąć samochód z listy moich samochodów.
Dostęp: klient.
- 22) Usuwanie konta – funkcjonalność umożliwia usuwanie konta.
Dostęp: klient, warsztat samochodowy.
- 23) Oglądanie listy zarejestrowanych użytkowników – możliwość oglądania wszystkich zarejestrowanych na serwisie użytkowników wraz z możliwością usuwania ich kont.
Dostęp: administrator.

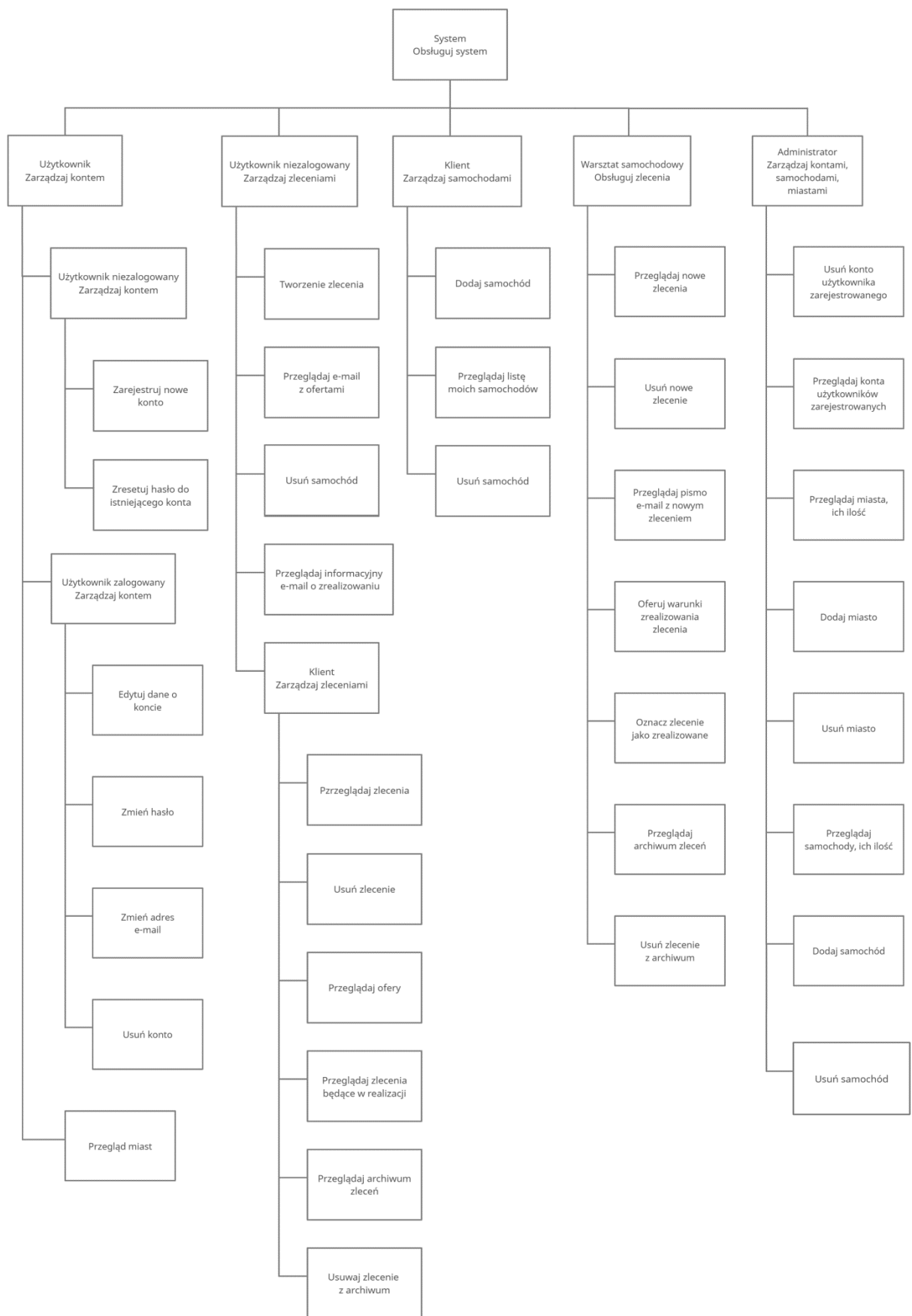
24) Oglądanie listy miast oraz liczby autoserwisów – możliwość oglądania wszystkich dostępnych do wyboru miast ich dodania oraz usuwania, liczby zarejestrowanych na ich terenie autoserwisów.

Dostęp: administrator.

25) Oglądanie listy samochodów oraz ich ilości w systemie – możliwość oglądania wszystkich dostępnych do wyboru marek i modeli samochodów, ich dodanie, usuwanie oraz ich liczby w systemie.

Dostęp: administrator.

Wyżej wymieniona funkcjonalność została zgrupowana w pięć modułów przedstawionych na diagramie hierarchii funkcji, rysunek 3.1.



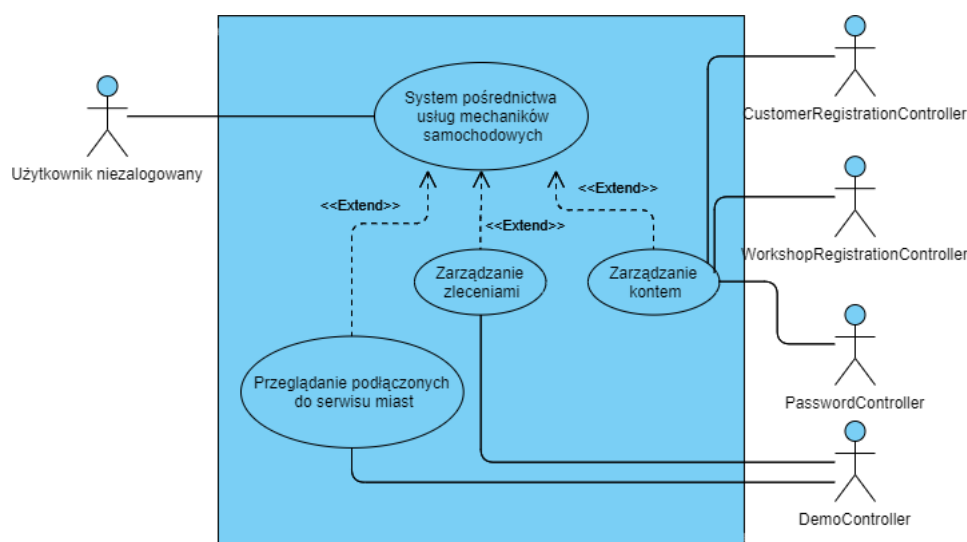
Rys. 3.1. Diagram hierarchii funkcji systemu

3.2. Diagramy przypadków użycia

Diagram przypadków użycia – to graficzna reprezentacja zachowania systemu, kiedy on współdziała z obiektem z zewnątrz. Obiektem może być użytkownik systemu, części systemu czy urządzenia. Ten obiekt nazywa się aktorem, a diagram przypadków użycia jest kolejnością kroków, które opisują działania aktora oraz reakcję systemu na nich.

3.2.1. System pośrednictwa usług mechaników samochodowych

Przypadek na rysunku 3.2 przedstawia funkcjonalność dostępną dla użytkownika niezalogowanego, jest to przeglądanie podłączonych do serwisu miast, zarządzanie zleceniami oraz zarządzanie kontem. Diagram został szczegółowo opisany w tabeli 3.1.



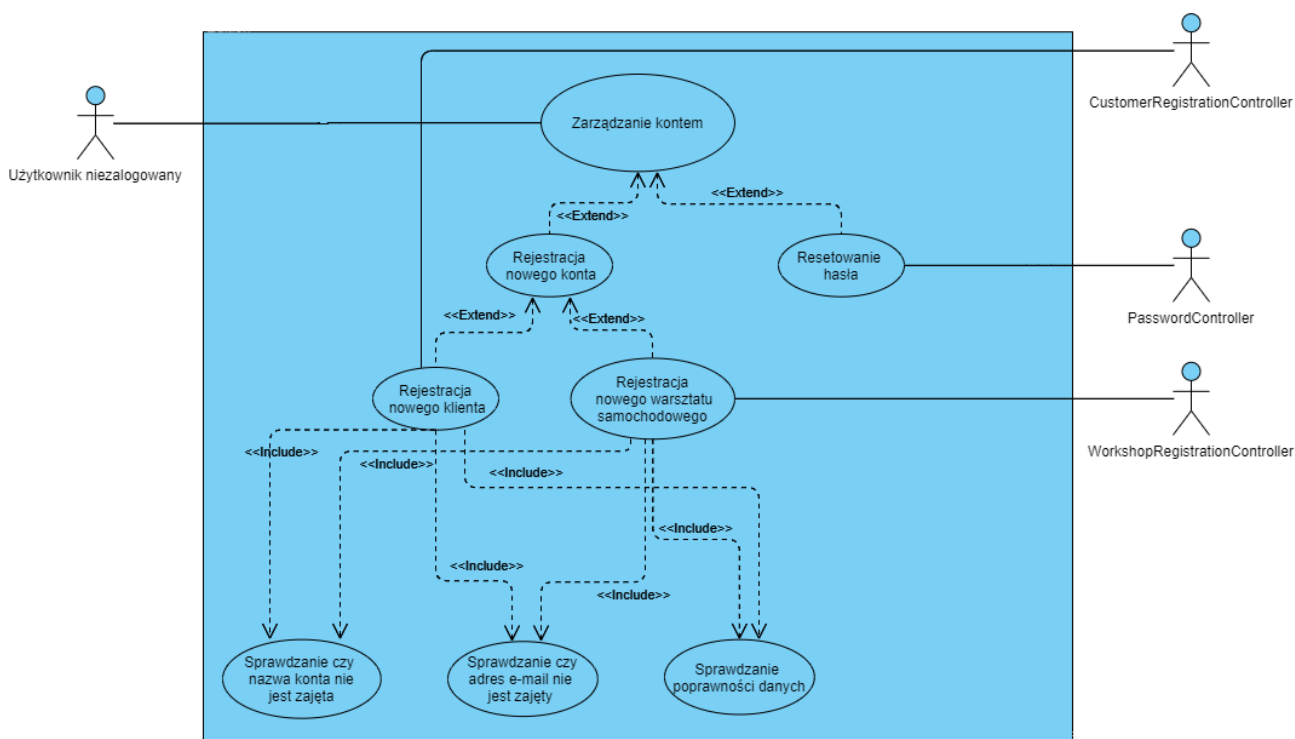
Rys. 3.2. Przypadek użycia: system pośrednictwa usług mechaników samochodowych

Tabela 3.1. Przypadek użycia: system pośrednictwa usług mechaników samochodowych

Nazwa przypadku	System pośrednictwa usług mechaników samochodowych
Aktorzy	Użytkownik niezalogowany
Krótki opis	Przypadek przedstawia funkcjonalności udostępnione niezalogowanemu użytkownikowi
Warunki wstępne	Użytkownik jest niezalogowany
Warunki końcowe	Brak
Główny przepływ zdarzeń	<ol style="list-style-type: none">1) Użytkownik niezalogowany przechodzi do systemu pośrednictwa usług mechaników samochodowych2) Użytkownik zarządza zleceniami3) Użytkownik przegląda miasta zawierające partnerskie warsztaty samochodowe4) Użytkownik zarządza swoim kontem
Alternatywne przepływy danych	3a) aktualnie brak dostępnych partnerskich warsztatów samochodowych

3.2.2. Zarządzanie kontem

Przypadek na rysunku 3.3 przedstawia funkcjonalność dostępną dla użytkownika niezalogowanego, związaną z zarządzaniem kontem. System umożliwia tworzenie nowego konta dwóch typów, klienta albo warsztatu samochodowego oraz resetowanie hasła do istniejącego konta. Diagram został szczegółowo opisany w tabeli 3.2.



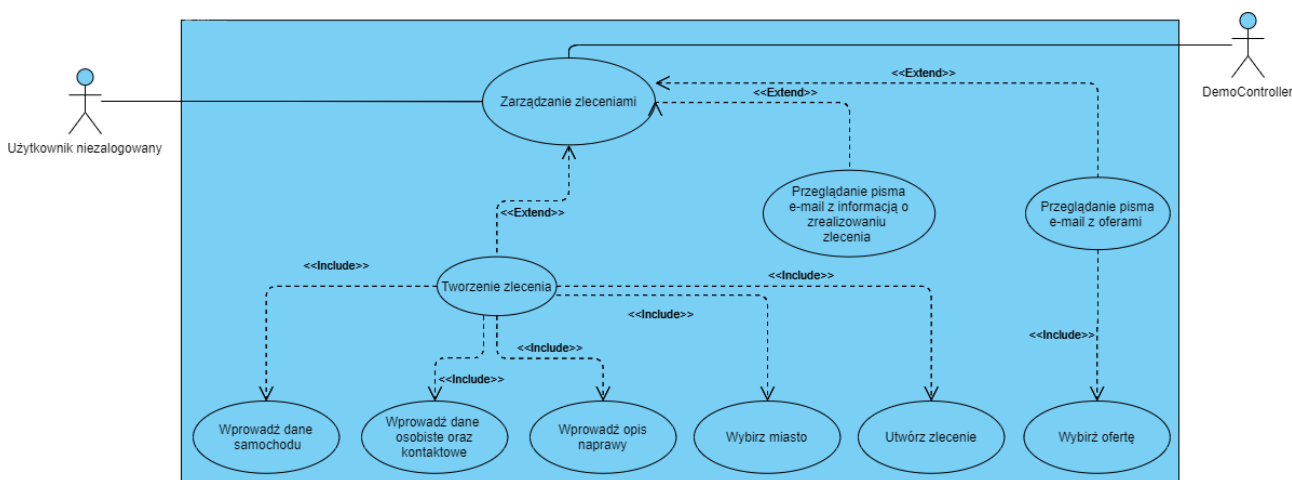
Rys. 3.3. Przypadek użycia: zarządzanie kontem

Tabela 3.2. Przypadek użycia: zarządzanie kontem

Nazwa przypadku	Zarządzanie kontem
Aktorzy	Użytkownik niezalogowany
Krótki opis	Przypadek przedstawia funkcjonalność rejestracji nowego konta klienta albo warsztatu samochodowego oraz resetowanie hasła istniejącego konta
Warunki wstępne	Użytkownik jest niezalogowany
Warunki końcowe	Brak
Główny przepływ zdarzeń	<ol style="list-style-type: none"> 1) Użytkownik niezalogowany przechodzi do formularza rejestracji konta klienta lub warsztatu samochodowego 2) Użytkownik wypełnia dane i podejmuje próbę rejestracji 3) System sprawdza poprawność wprowadzonych danych, wysyła na podany adres e-mail pismo weryfikacyjne i rejestruje nowe, jeszcze nie aktywowane konto 4) Użytkownik weryfikuje swój adres e-mail 5) System aktywuje nowostworzone konto
Alternatywne przepływy danych	<ol style="list-style-type: none"> 2a) system zgłasza informację o nieprawidłowo wprowadzonych danych 3a) system zgłasza informację, że konto o podanej nazwie użytkownika już istnieje 3a) system zgłasza, że konto z podanym adresem e-mail istnieje 3c) system zgłasza, że podano nieprawidłowe dane

3.2.3. Zarządzanie zleceniami użytkownik niezalogowany

Przypadek na rysunku 3.4 przedstawia funkcjonalność dotyczącą zarządzania zleceniami dostępną dla użytkownika niezalogowanego. System umożliwia tworzenie nowego zlecenia, otrzymanie pisma na adres elektroniczny zawierającego informację o ofertach zrealizowania zlecenia od warsztatów samochodowych z możliwością wyboru warsztatu samochodowego w celu realizacji zlecenia oraz pisma z informacją o realizacji zlecenia. Diagram został opisany w tabeli 3.3.



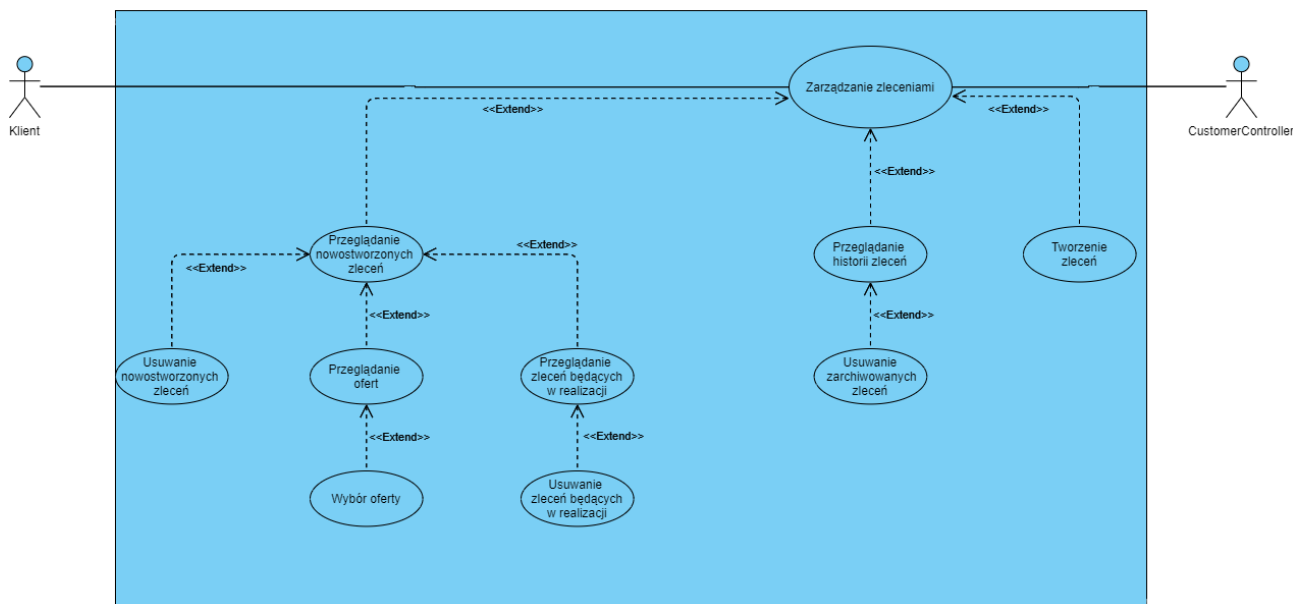
Rys. 3.4. Przypadek użycia: zarządzanie zleceniami użytkownik niezalogowany

Tabela 3.3. Przypadek użycia: zarządzanie zleceniami użytkownik niezalogowany

Nazwa przypadku	Zarządzanie zleceniami
Aktorzy	Użytkownik niezalogowany
Krótki opis	Przypadek przedstawia funkcjonalność zarządzania zleceniami oraz tworzenia zleceń
Warunki wstępne	Użytkownik jest niezalogowany
Warunki końcowe	Po tworzeniu nowego zlecenia zostaje ono wysyłane do wszystkich warsztatów samochodowych znajdujących się w mieście wybranym przez użytkownika
Główny przepływ zdarzeń	<ol style="list-style-type: none"> 1) Tworzenie zlecenia 2) Wprowadzenie danych o samochodzie 3) Wprowadzenie danych osobistych oraz kontaktowych użytkownika 4) Wprowadzenie opisu żądanej naprawy 5) Wybór miasta 6) Utworzenie zlecenia 7) Przeglądanie pism e-mail z ofertami od warsztatów 8) Wybór warsztatu samochodowego do zrealizowania zlecenia. 9) Przeglądanie pisma e-mail z informacją o zrealizowaniu zlecenia
Alternatywne przepływy danych	<ol style="list-style-type: none"> 2a) system informuje, że nie wypełniono wszystkich wymaganych danych 2b) system informuje, że dane wprowadzone w niepoprawnej postaci 3a) system informuje, że nie wypełniono wszystkich wymaganych danych 3b) system informuje, że dane wprowadzone w niepoprawnej postaci 4a) system informuje, że nie wypełniono wszystkich wymaganych danych 6a) system informuje, że zlecenie nie zostało stworzone 6b) system informuje, że w wybranym mieście nie ma warsztatów samochodowych partnerów serwisu

3.2.4. Zarządzanie zleceniami klient

Przypadek na rysunku 3.5 przedstawia funkcjonalność dotyczącą zarządzania zleceniami dostępną dla klienta. System umożliwia tworzenie nowego zlecenia, otrzymanie pisma na adres elektroniczny zawierającego informację o ofertach zrealizowania zlecenia od warsztatów samochodowych oraz pisma z informacją o realizacji zlecenia, przeglądanie nowostworzonych zleceń z możliwością ich usuwania, przeglądanie ofert od warsztatów samochodowych z możliwością wyboru warsztatu samochodowego w celu realizacji zlecenia, przeglądanie zleceń będących w realizacji z możliwością ich usuwania, przeglądanie historii zleceń z możliwością ich usuwania. Diagram został szczegółowo opisany w tabeli 3.4.



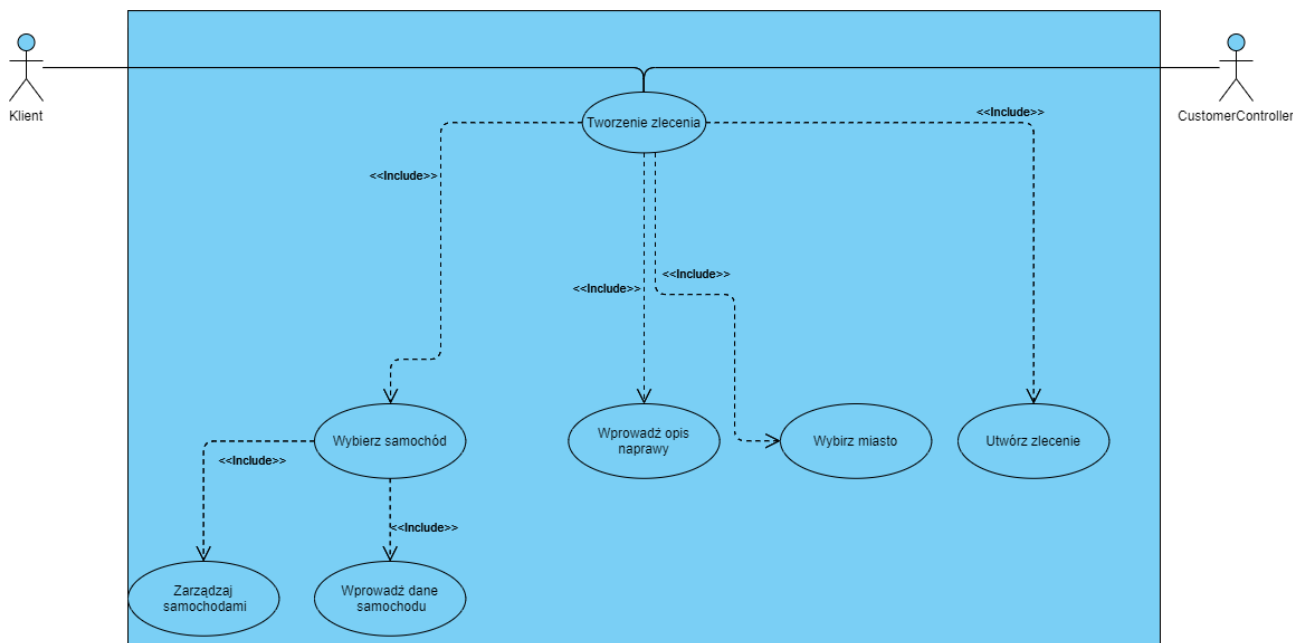
Rys. 3.5. Przypadek użycia: zarządzanie zleceniami klient

Tabela 3.4. Przypadek użycia: zarządzanie zleceniami klient

Nazwa przypadku	Zarządzanie zleceniami
Aktorzy	Klient
Krótki opis	Przypadek przedstawia funkcjonalność zarządzania zleceniami oraz tworzenia zleceń
Warunki wstępne	Użytkownik jest zalogowany, posiada uprawnienie klienta
Warunki końcowe	Po tworzeniu nowego zlecenia zostaje ono wysłane do wszystkich warsztatów samochodowych znajdujących się w mieście klienta
Główny przepływ zdarzeń	<ol style="list-style-type: none"> 1) Przeglądanie własnych nowostworzonych zleceń 2) Usunięcie własnych nowostworzonych zleceń 3) Przeglądanie listy ofert 4) Wybór oferty 5) Przeglądanie zleceń będących w realizacji 6) Usunięcie zleceń będących w realizacji 7) Przeglądanie historii zleceń 8) Usunięcie zarchiwowanych zleceń 9) Tworzenie zlecenia
Alternatywne przepływy danych	<ol style="list-style-type: none"> 2a) system informuje, że zlecenie nie zostało usunięte 3a) system informuje, że dla wybranego zlecenia jeszcze nie zostały zaproponowane oferty od warsztatów samochodowych 5a) system informuje, że na chwilę obecną nie ma zleceń w realizacji 6a) system informuje, że zlecenie nie zostało usunięte 7a) system informuje, że na chwilę obecną nie ma zarchiwowanych zleceń 8a) system informuje, że zlecenie nie zostało usunięte

3.2.5. Tworzenie zleceń

Przypadek na rysunku 3.6 przedstawia funkcjonalność dotyczącą tworzenia zlecenia przez klienta. Proces tworzenia zlecenia składa się z kilku etapów, szczegółowo opisanych w tabeli 3.5.



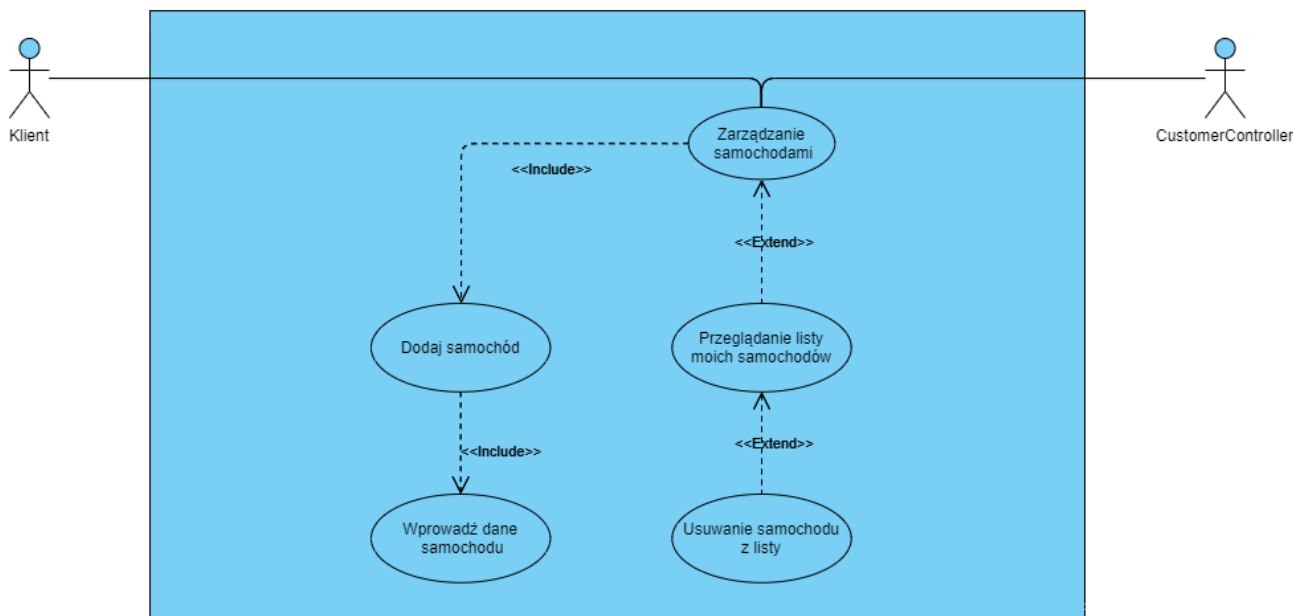
Rys. 3.6. Przypadek użycia: tworzenie zlecenia

Tabela 3.5. Przypadek użycia: tworzenie zlecenia

Nazwa przypadku	Tworzenie zlecenia
Aktorzy	Klient
Krótki opis	Przypadek przedstawia funkcjonalność tworzenia zlecenia
Warunki wstępne	Użytkownik jest zalogowany, posiada uprawnienie klienta
Warunki końcowe	Po utworzeniu nowego zlecenia zostaje ono wysłane do wszystkich warsztatów samochodowych znajdujących się w mieście wybranym przez klienta
Główny przepływ zdarzeń	<ol style="list-style-type: none"> 1) Wybór samochodu 2) Zarządzanie samochodami 3) Wprowadzenie danych o samochodzie 4) Wprowadzenie opisu żądanej użytkownikiem naprawy 5) Wybór miasta 6) Utworzenie zlecenia
Alternatywne przepływy danych	<ol style="list-style-type: none"> 1a) system informuje, że lista moich samochodów jest pusta i trzeba dodać nowy samochód do listy lub wprowadzić dane samochodu jednorazowo 3a) system informuje, że nie wypełniono wszystkich wymaganych danych 3b) system informuje, że dane wprowadzone w nieprawidłowej postaci 4a) system informuje, że nie wypełniono wszystkich wymaganych danych 6a) system informuje, że zlecenie nie zostało stworzone 6b) system informuje, że w wybranym mieście nie ma warsztatów samochodowych partnerów serwisu

3.2.6. Zarządzanie samochodami

Przypadek na rysunku 3.7 przedstawia funkcjonalność dotyczącą zarządzania samochodami w tym dodanie samochodu do listy, przeglądanie listy dodanych samochodów oraz ich usuwanie. Diagram został szczegółowo opisany w tabeli 3.6.



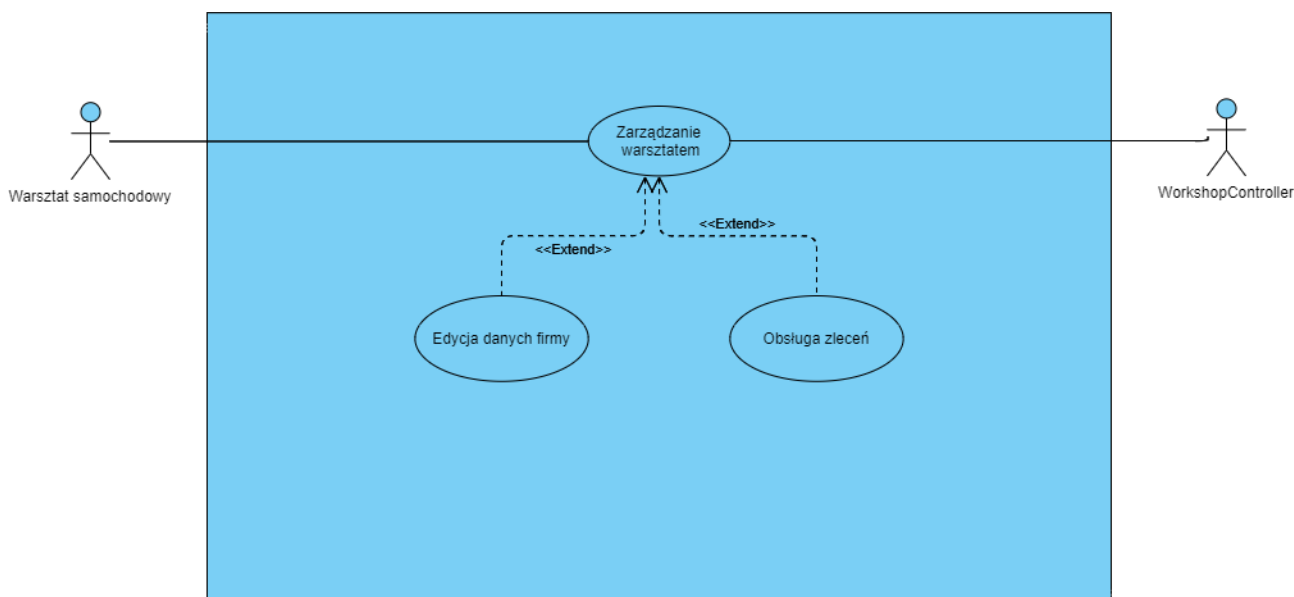
Rys. 3.7. Przypadek użycia: zarządzanie samochodami

Tabela 3.6. Przypadek użycia: zarządzanie samochodami

Nazwa przypadku	Zarządzanie samochodami
Aktorzy	Klient
Krótki opis	Przypadek przedstawia funkcjonalność zarządzania samochodami
Warunki wstępne	Użytkownik jest zalogowany, posiada uprawnienie klienta
Warunki końcowe	Po dodaniu samochodu do listy będzie on dostępny do przeglądania w zakładce moje samochody oraz będzie dostępny do wyboru przy tworzeniu zlecenia
Główny przepływ zdarzeń	<ol style="list-style-type: none"> 1) Przeglądanie listy własnych samochodów 2) Usuwanie samochodu z listy 3) Wprowadzenie danych o samochodzie 4) Dodanie samochodu
Alternatywne przepływy danych	<ol style="list-style-type: none"> 1a) system informuje, że lista moich samochodów jest pusta 2a) system informuje, że samochód nie został usunięty 3a) system informuje, że nie wypełniono wszystkich wymaganych danych 3b) system informuje, że dane wprowadzone w niepoprawnej postaci 4a) system informuje, że samochód nie został dodany z powodu tego, że on już jest w liście innego użytkownika serwisu. Samochód zostanie dodany po potwierdzeniu dodania przez innych właścicieli samochodu

3.2.7. Zarządzanie warsztatem samochodowym

Przypadek na rysunku 3.8 przedstawia funkcjonalność dostępną w celu zarządzania warsztatem samochodowym. Diagram został szczegółowo opisany w tabeli 3.7.



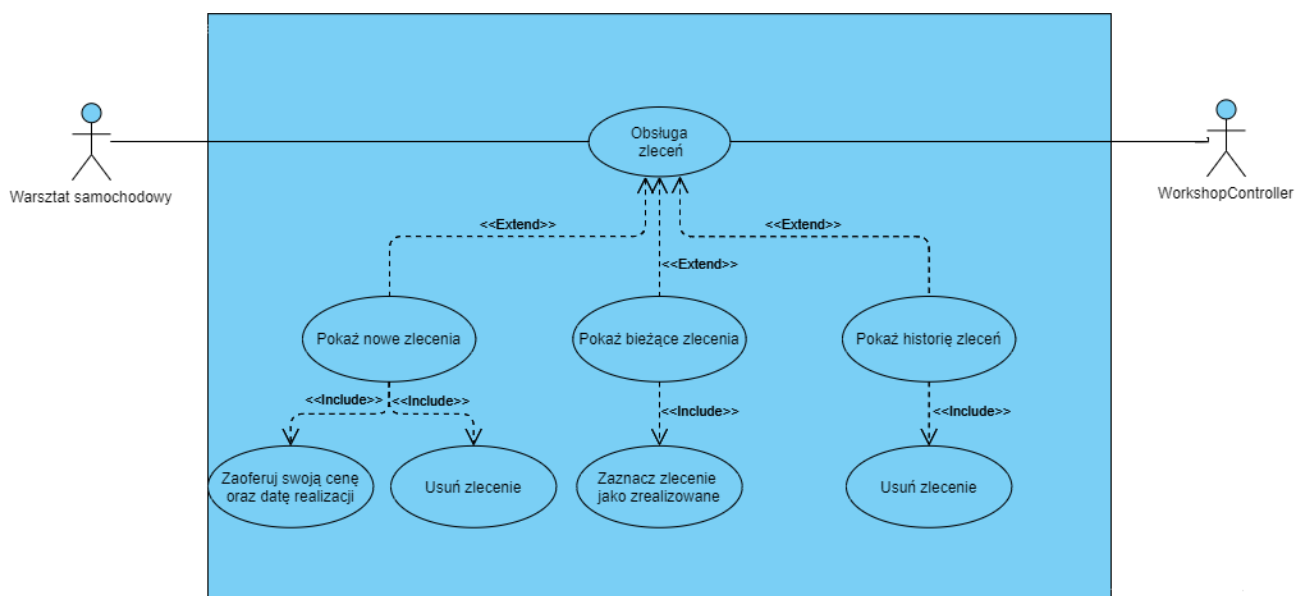
Rys. 3.8. Przypadek użycia: zarządzanie warsztatem samochodowym

Tabela 3.7. Przypadek użycia: zarządzanie warsztatem samochodowym

Nazwa przypadku	Zarządzanie warsztatem
Aktorzy	Warsztat samochodowy
Krótki opis	Przypadek przedstawia funkcjonalności udostępnione do zarządzania warsztatem samochodowym
Warunki wstępne	Użytkownik jest zalogowany, posiada uprawnienie warsztatu samochodowego
Warunki końcowe	Brak
Główny przepływ zdarzeń	1) Zarządzanie warsztatem samochodowym 2) Warsztat samochodowy edytuje dane o firmie 3) Warsztat samochodowy obsługuje zlecenia
Alternatywne przepływy danych	2a) system informuje, że nowe dane wprowadzone w nieprawidłowej postaci

3.2.8. Obsługa zleceń warsztat samochodowy

Przypadek na rysunku 3.9 przedstawia funkcjonalność dotyczącą obsługi zleceń dostępną dla warsztatów samochodowych w tym przeglądanie nowych zleceń z możliwością oferowania warunków realizacji oraz usuwania zleceń, przeglądanie będących w realizacji zleceń z możliwością oznaczenia zlecenia zrealizowanym, przeglądanie historii zleceń z możliwością usunięcia. Diagram został szczegółowo opisany w tabeli 3.8.



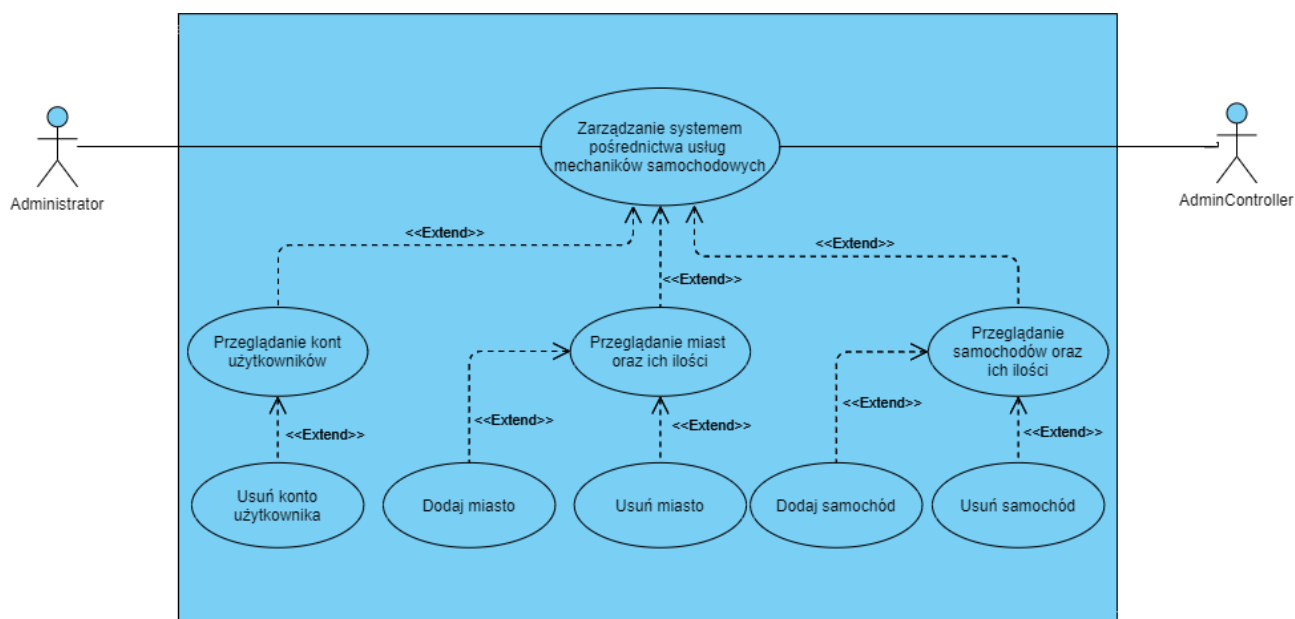
Rys. 3.9. Przypadek użycia: obsługa zleceń warsztat samochodowy

Tabela 3.8. Przypadek użycia: obsługa zleceń warsztat samochodowy

Nazwa przypadku	Obsługa zleceń
Aktorzy	Warsztat samochodowy
Krótki opis	Przypadek przedstawia funkcjonalności udostępnione do obsługi zleceń
Warunki wstępne	Użytkownik jest zalogowany, posiada uprawnienie warsztatu samochodowego
Warunki końcowe	Po zaoferowaniu swojej ceny oraz daty realizacji zlecenia oferta zostaje wysłana do klienta. Po wyborze warsztatu samochodowego do realizacji zlecenia przez klienta, zostaje ono dostępne pod wkładką bieżące zlecenia. Po zaznaczeniu zlecenia jako zrealizowanego klient zostaje o tym poinformowany natomiast zlecenie będzie dostępne pod wkładką „Archiwum zleceń”
Główny przepływ zdarzeń	<ol style="list-style-type: none"> 1) Przeglądanie nowostworzonych zleceń 2) Usuwanie nowostworzonych zleceń 3) Oferowanie swoich warunków realizacji zlecenia 4) Przeglądanie bieżących zleceń 5) Zaznaczenie zlecenia jako zrealizowanego 6) Przeglądanie historii zleceń 7) Usuwanie zarchiwizowanych zleceń
Alternatywne przepływy danych	<ol style="list-style-type: none"> 1a) system informuje, że lista nowostworzonych zleceń jest pusta 2a) system informuje, że zlecenie nie zostało usunięte 3a) system informuje, że dane wprowadzone w niepoprawnej postaci 3b) system informuje, że nie wypełniono wszystkich wymaganych danych 4a) system informuje, że lista bieżących zleceń jest pusta 6a) system informuje, że archiwum zleceń jest puste 7a) system informuje, że zlecenie nie zostało usunięte

3.2.9. Zarządzanie systemem pośrednictwa usług mechaników samochodowych

Przypadek na rysunku 3.10 przedstawia funkcjonalności udostępnione administratorowi w ramach zarządzania systemem. Diagram został szczegółowo opisany w tabeli 3.9.



Rys. 3.10. Przypadek użycia: zarządzanie systemem pośrednictwa usług mechaników samochodowych

Tabela 3.9. Przypadek użycia: zarządzanie systemem pośrednictwa usług mechaników samochodowych

Nazwa przypadku	Zarządzanie systemem pośrednictwa usług mechaników samochodowych
Aktorzy	Administrator
Krótki opis	Przypadek przedstawia funkcjonalności udostępnione do zarządzania systemem
Warunki wstępne	Użytkownik jest zalogowany, posiada uprawnienie administratora
Warunki końcowe	Brak
Główny przepływ zdarzeń	<ol style="list-style-type: none"> 1) Przeglądanie kont użytkowników 2) Usuwanie konta użytkownika 3) Przeglądanie miast dostępnych do wyboru przy tworzeniu konta warsztatu samochodowego jako adres lokalizacji oraz przy tworzeniu zlecenia 4) Dodanie nowego miasta 5) Usuwanie miasta 6) Przeglądanie dostępnych do wyboru marek oraz modeli samochodów 7) Dodanie nowych marek lub modeli samochodów 8) Usuwanie dostępnych do wyboru marek i modeli samochodów
Alternatywne przepływy danych	<ol style="list-style-type: none"> 4a) system informuje, że dodane miasto już istnieje 7a) system informuje, że dodana marka lub model samochodu już istnieje

4. Baza danych

W tym rozdziale rozpatrzony zostanie sposób zaprojektowania oraz tworzenia relacyjnej bazy danych. W poniższych podrozdziałach zostaną przedstawione wszystkie potrzebne tabele bazy danych, utworzonej przez Hibernate(ORM) na podstawie klas encyjnych w bazie MySQL.

4.1. SQL skrypt

W celu tworzenia bazy danych zostały wykorzystane polecenia w języku SQL. SQL skrypt jest dokumentem tekstowym z rozszerzeniem *.sql zawierającym szereg poleceń służących do tworzenia obiektów łącznie z instrukcjami napełnienia ich danymi. Na rysunku 4.1 przykładowo przedstawiona część poleceń odpowiadająca za tworzenie tabel *customer* oraz *customer_detail*.

```
CREATE TABLE customer_detail (  
    id bigint(20) NOT NULL AUTO_INCREMENT,  
    first_name varchar(45) NOT NULL,  
    last_name varchar(45) NOT NULL,  
    email varchar(45) DEFAULT NULL,  
    phone_number varchar(15) NOT NULL,  
    account_verified bit(1),  
  
    UNIQUE (id),  
    PRIMARY KEY (id)  
);  
  
CREATE TABLE customer (  
    id bigint(20) NOT NULL AUTO_INCREMENT,  
    username varchar(45),  
    password varchar(68),  
  
    customer_detail_id bigint(20),  
  
    UNIQUE (id),  
    UNIQUE(username),  
    PRIMARY KEY (id),  
    FOREIGN KEY (customer_detail_id) REFERENCES customer_detail(id)  
);
```

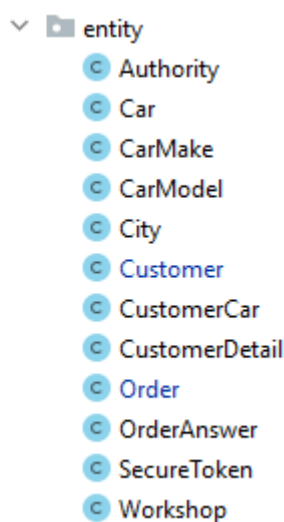
Rys. 4.1. SQL skrypt tworzenie tabel *customer*, *customer_detail*

4.2. Zaprojektowanie i tworzenie relacyjnej bazy danych

4.2.1. Klasy encyjne (Entity)

Encja (Entity) – jest kluczowym elementem mapowania obiektowo-relacyjnego (ORM – Object-relational mapping), realizowanego przez bibliotekę Hibernate, zgodnie z Java EE specyfikacją JPA (Java Persistence API). Główną ideą ORM (JPA) jest zestawienie modelu danych, zrealizowanego z użyciem Java kodu oraz obiektów (tablic) po stronie relacyjnej bazy danych.

Właśnie encja jest elementem Java kodu, który odzwierciedla się na tablicę relacyjnej bazy danych. Encja jest to zwykły POJO (Plain Old Java Object) Java klasa właściwości, której odzwierciedlają się na pola tablicy bazy danych, czyli obiekt klasy encyjnej przedstawia zapis w tablicy bazy danych [26].



Rys. 4.2. Klasy encyjne w systemie

W celu przedstawienia tablicy bazy danych za pomocą POJO Java klasy oraz rekordów bazy danych za pomocą pól tej klasy są wykorzystywane adnotacje standardu Java Persistence API (JPA). Klasa encyjna „City” przykładowo przedstawiona na rysunku 4.3.

```

@Entity
@Table(name = "city")
public class City {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    @Column(name = "name", unique = true)
    private String cityName;

    @OneToMany(mappedBy = "city",
        fetch = FetchType.LAZY,
        cascade = CascadeType.ALL)
    private List<Workshop> workshops;

    @OneToMany(mappedBy = "city",
        fetch = FetchType.LAZY,
        cascade = CascadeType.PERSIST)
    private List<Order> orders;
}

```

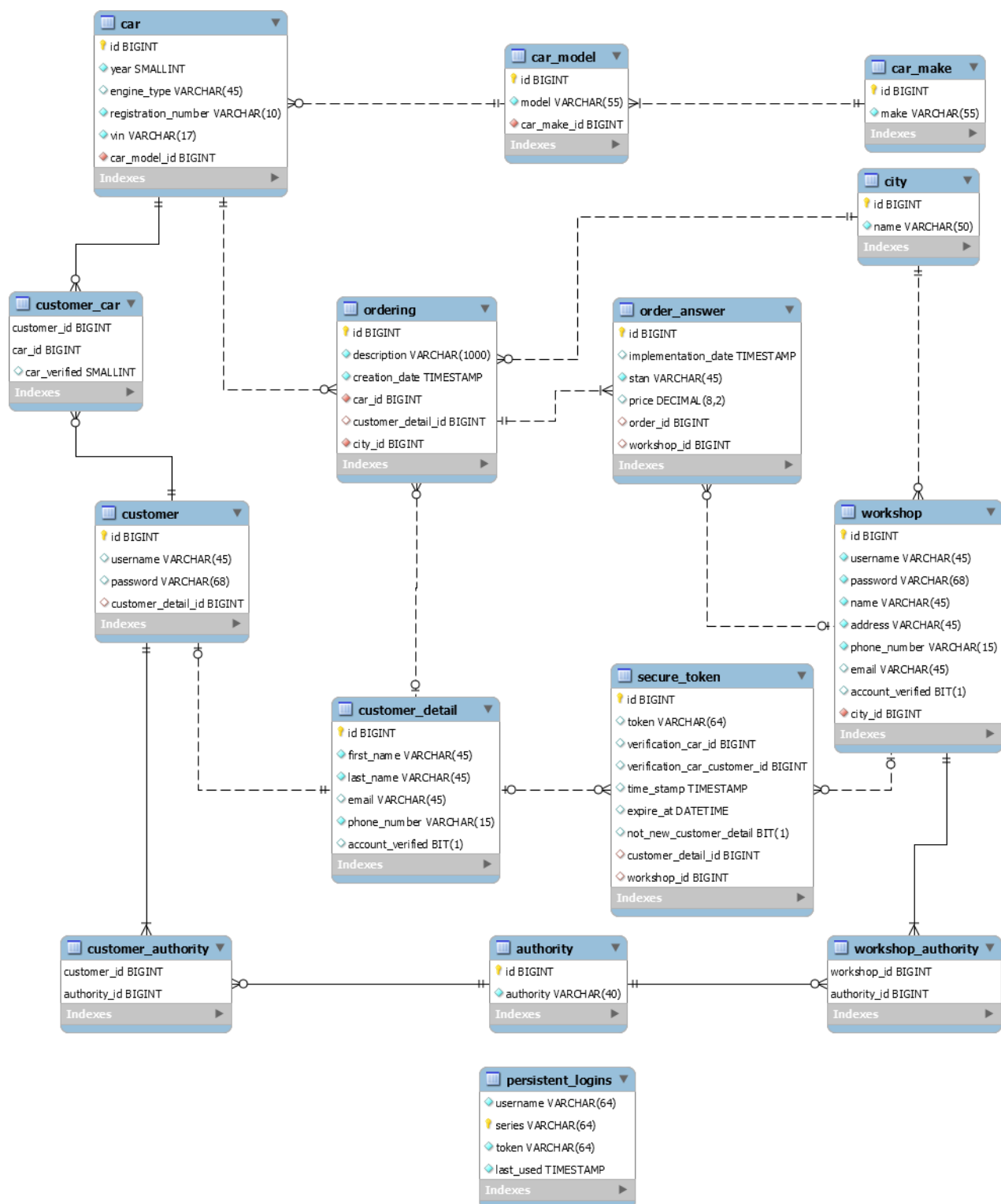
Rys. 4.3. Klasy encyjne w systemie

Opis adnotacji

- @Entity – informuje o tym, że klasa jest encyjną, musi zawierać odwzorowanie w bazie danych;
- @Table – informuje o tym do której tablicy w bazie danych jest przywiązana dana klasa;
- @Id – informuje o tym, że w tablicy bazy danych kolumna związana z danym polem występuje jako klucz główny (Primary Key);
- @GeneratedValue – opisuje strategię generacji wartości kolumny będącej kluczem głównym. W przykładowo rozpatrywanej klasie encyjnej stosuje się strategia GenerationType.IDENTITY, która polega na automatycznym zwiększeniu wartości kolumny zgodnie z zasadami określonymi w bazie danych;
- @Column – informuje o tym do której kolumny w bazie danych jest przywiązane dane pole;
- @OneToMany – wskazuje na typ relacji pomiędzy klasami encyjnymi, w tym przypadku jeden do wielu.

4.2.2. Diagram bazy danych

W tym rozdziale baza danych zostanie graficznie przedstawiona za pomocą diagramu związków encji ERD (entity relationship diagram). ERD pozwala na graficzne przedstawienie encji, które są tabelami bazy danych, atrybutów, które są elementami służącymi do określenia stanu encji oraz związków między encjami. Rysunek 4.4 przedstawia ERD stworzony za pomocą programu MySQL Workbench i graficznie reprezentuje strukturę bazy danych systemu pośrednictwa usług mechaników samochodowych.



Rys. 4.4. Diagram związków encji

4.2.3. Tabela car

W tabeli bazy danych *car* zapisane są samochody jak dodane użytkownikami do listy swoich samochodów tak i dodane przy tworzeniu zlecenia. W tabeli 4.1 zamieszczony opis wszystkich pól tablicy.

Tabela 4.1. Tabela *car*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
year	smallint	rok produkcji
engine_type	varchar(45)	rodzaj paliwa
registration_number	varchar(10)	numer rejestracyjny
vin	varchar(17)	numer nadwozia
car_model_id	bigint	klucz obcy z tabeli <i>car_model</i>

4.2.4. Tabela *car_model*

W tabeli bazy danych *car_model* zapisane są dostępne do wyboru modeli samochodów. W tabeli 4.2 zamieszczony opis wszystkich pól tablicy.

Tabela 4.2. Tabela *car_model*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
model	varchar(55)	model
car_make_id	bigint	klucz obcy z tabeli <i>car_make</i>

4.2.5. Tabela *car_make*

W tabeli bazy danych *car_make* zapisane są dostępne do wyboru marki samochodów. W tabeli 4.3 zamieszczony opis wszystkich pól tablicy.

Tabela 4.3. Tabela *car_make*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
make	varchar(55)	marka

4.2.6. Tabela *customer*

W tabeli bazy danych *customer* zapisane są dane do logowania zarejestrowanych użytkowników. W tabeli 4.4 zamieszczony opis wszystkich pól tablicy.

Tabela 4.4. Tabela *customer*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
username	varchar(45)	nazwa konta użytkownika
password	varchar(68)	hasło po szyfrowaniu
customer_detail_id	bigint	klucz obcy z tabeli <i>customer_detail</i>

4.2.7. Tabela *customer_detail*

W tabeli bazy danych *customer_detail* zapisane są dane kontaktowe, dane personalne oraz flaga weryfikacji adresy poczty elektronicznej użytkowników typu klient oraz użytkownik niezarejestrowany. W tabeli 4.5 zamieszczony opis wszystkich pól tablicy.

Tabela 4.5. Tabela *customer_detail*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
first_name	varchar(45)	imię
last_name	varchar(45)	nazwisko
email	varchar(45)	adres poczty elektronicznej
phone_number	varchar(15)	numer telefonu
account_verified	bit(1)	określa czy adres e-mail jest zweryfikowany

4.2.8. Tabela *customer_car*

Tabela bazy danych *customer_car* powstała w celu przedstawienia związku wiele do wielu (Join Table) pomiędzy tabelami *customer* i *car*. Kolumny Join Table – to są klucze obce, które odnoszą się do kluczy głównych połączonych tabel. W tabeli 4.6 zamieszczony opis wszystkich pól tablicy.

Tabela 4.6. Tabela *customer_car*

Nazwa pola	Typ danych	Opis
customer_id	bigint	klucz obcy z tabeli <i>customer</i>
car_id	bigint	klucz obcy z tabeli <i>car</i>
car_verified	smallint	w przypadku dodania samochodu już istniejącego w systemie, określa czy wszystkie dotychczasowe właściciele potwierdzili dodanie

4.2.9. Tabela *ordering*

W tabeli bazy danych *ordering* zapisane są dane dotyczące zlecenia. W tabeli 4.7 zamieszczony opis wszystkich pól tablicy.

Tabela 4.7. Tabela *ordering*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
description	varchar(1000)	opis wymaganej przez klienta naprawy samochodu
creation_date	timestamp	data tworzenia zlecenia
car_id	bigint	klucz obcy tabeli <i>car</i>
customer_detail_id	bigint	klucz obcy tabeli <i>customer_detail</i>
city_id	bigint	klucz obcy tabeli <i>city</i>

4.2.10. Tabela *city*

Tabela bazy danych *city* zawiera dostępne do wyboru miasta. W tabeli 4.8 zamieszczony opis wszystkich pól tablicy.

Tabela 4.7. Tabela *city*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
name	varchar(50)	nazwa miasta

4.2.11. Tabela *workshop*

W tabeli bazy danych *workshop* zapisane są dane do logowania, informacja o formie, dane kontaktowe oraz flaga weryfikacji adresy poczty elektronicznej użytkowników typu warsztat samochodowy. W tabeli 4.9 zamieszczony opis wszystkich pól tablicy.

Tabela 4.9. Tabela *workshop*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
username	varchar(45)	nazwa konta użytkownika
password	varchar(68)	hasło po szyfrowaniu
name	varchar(45)	nazwa firmy
address	varchar(45)	adres firmy
email	varchar(45)	adres poczty elektronicznej
phone_number	varchar(15)	numer telefonu
account_verified	bit(1)	określa czy adres e-mail jest zweryfikowany
city_id	bigint	klucz obcy do tabeli <i>city</i>

4.2.12. Tabela *order_answer*

W tabeli bazy danych *order_answer* zapisane są unikatowe dla każdego użytkownika typu warsztat samochodowy dane dotyczące zlecenia. W tabeli 4.10 zamieszczony opis wszystkich pól tablicy.

Tabela 4.10. Tabela *order_answer*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
implementation_date	timestamp	oferowana data realizacji
stan	varchar(45)	status zlecenia
price	decimal(8,2)	oferowana cena realizacji
order_id	bigint	klucz obcy do tabeli <i>ordering</i>
workshop_id	bigint	klucz obcy do tabeli <i>workshop</i>

4.2.13. Tabela *security_token*

W tabeli bazy danych *security_token* przechowywane są informacje służące do prawidłowej weryfikacji adresu poczty elektronicznej użytkownika biorąc pod uwagę token i datę jego tworzenia (na weryfikację użytkownik ma osiem godzin) oraz do potwierdzenia dodania samochodu nowym użytkownikiem w przypadku istnienia w tego pojazdu innych właścicieli. Token jest zaszyfrowaną za pomocą algorytmu Base64 wartością generowaną za pomocą klasy *KeyGenerators*. W tabeli 4.11 zamieszczony opis wszystkich pól tablicy.

Tabela 4.11. Tabela *security_token*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
token	varchar(64)	token
verification_car_id	bigint	klucz główny weryfikowanego samochodu
verification_car_customer_id	bigint	klucz główny użytkownika dodającego samochód
time_stamp	timestamp	data oraz godzina tworzenia tokena
expire_at	datetime	data oraz godzina wygasania ważności tokena
not_new_customer_detail	bit(1)	określa czy użytkownik ma historię zleceń
customer_detail_id	bigint	klucz obcy do tabeli <i>customer_detail</i>
workshop_id	bigint	klucz obcy do tabeli <i>workshop</i>

4.2.14. Tabela *authority*

Tabela bazy danych *authority* zawiera określone serwisem rolę użytkowników. W tabeli 4.12 zamieszczony opis wszystkich pól tablicy.

Tabela 4.12. Tabela *authority*

Nazwa pola	Typ danych	Opis
id	bigint	klucz główny
authority	varchar(40)	role użytkowników

4.2.15. Tabela *customer_authority*

Tabela bazy danych *customer_authority* powstała w celu przedstawienia związku wiele do wielu (Join Table) pomiędzy tabelami *customer* i *authority*. W tabeli 4.13 zamieszczony opis wszystkich pól tablicy.

Tabela 4.13. Tabela *customer_authority*

Nazwa pola	Typ danych	Opis
customer_id	bigint	klucz obcy związany z tabelą <i>customer</i>
authority_id	bigibt	klucz obcy związany z tabelą <i>authority</i>

4.2.16. Tabela *workshop_authority*

Tabela bazy danych *workshop_authority* powstała w celu przedstawienia związku wiele do wielu (Join Table) pomiędzy tabelami *workshop* i *authority*. W tabeli 4.14 zamieszczony opis wszystkich pól tablicy.

Tabela 4.14. Tabela *workshop_authority*

Nazwa pola	Typ danych	Opis
<i>workshop_id</i>	bigint	klucz obcy związany z tabelą <i>workshop</i>
<i>authority_id</i>	bigint	klucz obcy związany z tabelą <i>authority</i>

4.2.17. Tabela *persistent_logins*

Tabela bazy danych *persistent_logins* zawiera informacje niezbędne do funkcjonowania opcji zapamiętaj mnie, która realizowana na podstawie zapisania tokena do bazy danych. Spring Security biorąc za podstawę plik „cookies” szyfruje go i przechowuje w dwóch polach tabeli *persistent_logins*. W przypadku kiedy użytkownik ponownie otwiera stronę internetową serwisu Spring Security szuka w bazie danych token wykorzystując wartość zapisaną w polu *series*, deszyfruje token oraz porównuje z nowostworzonym w przeglądarce. Jeżeli porównanie skończyło się sukcesem logowanie odbędzie się automatycznie. W tabeli 4.15 zamieszczony opis wszystkich pól tablicy.

Tabela 4.15. Tabela *workshop_authority*

Nazwa pola	Typ danych	Opis
<i>username</i>	varchar(64)	nazwa konta użytkownika
<i>series</i>	varchar(64)	klucz główny
<i>token</i>	varchar(64)	token
<i>last_used</i>	timestamp	data oraz godzina tworzenia albo ostatniego logowania się

4.3. Macierz CRUD

CRUD – to zestaw głównych funkcji dla pracy z bazą danych. Żeby przy komunikacji z bazą danych nie pisać maszynowych zapytań w języku SQL, wykorzystuje się CRUD – utwórz (create), odczytaj (read), aktualizuj (update), usuń (delete).

4.3.1. Macierz CRUD dla operacji dotyczących użytkownika

W tabeli 4.16 została przedstawiona macierz CRUD dotycząca operacji na użytkownikach.

Tabela 4.16. Macierz CRUD operacji na użytkownikach

	car	car_model	car_make	customer	customer_detail	customer_car	ordering	city	workshop	order_answer	security_token	authority	customer_authority	workshop_authority	persistent_logins
rejestracja użytkownika				C	CU				C		CR		C	C	
logowanie użytkownika				R					R				R	R	R
edycja danych użytkownika				RU	RU				RU						
usuwanie konta użytkownika	RD			RD	RD	RD			RD		RD		RD	RD	RD
zapamiętaj mnie				R					R						C

4.3.2. Macierz CRUD dla operacji dotyczących zlecenia

W tabeli 4.17 została przedstawiona macierz CRUD dotycząca operacji ze zleceniami.

Tabela 4.17. Macierz CRUD operacji ze zleceniami

	car	car_model	car_make	customer	customer_detail	customer_car	ordering	city	workshop	order_answer	security_token	authority	customer_authority	workshop_authority	persistent_logins
tworzenie zlecenia	CR	R	R		C		C	R	R	C	CR				
usuwanie zlecenia	RD						RD			RD					
oferowanie warunków realizacji zlecenia	R	R	R	R	R		R	R	R	RU					
wybór użytkownika do realizacji zlecenia	R	R	R	R	R		R	R	R	RUD					
oznaczenie zlecenia jako zrealizowane	R	R	R	R	R		R	R	R	RU					

4.3.3. Macierz CRUD dla operacji dotyczących samochodów

W tabeli 4.18 została przedstawiona macierz CRUD dotycząca operacji nad samochodami.

Tabela 4.18. Macierz CRUD operacji nad samochodami

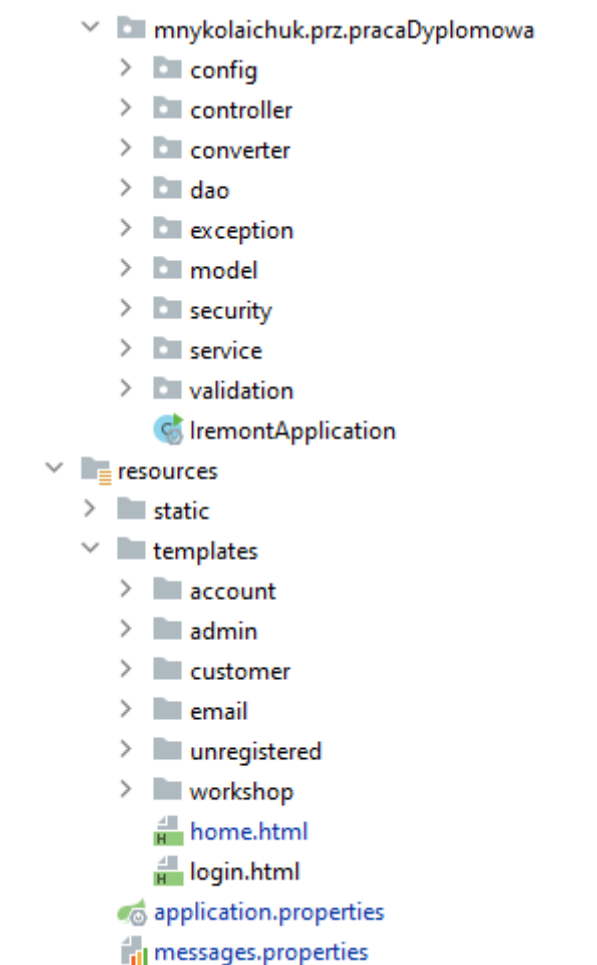
	car	car_model	car_make	customer	customer_detail	customer_car	ordering	city	workshop	order_answer	security_token	authority	customer_authority	workshop_authority	persistent_logins
dodanie samochodu	C	R	R	R	R	C					CR				
usuwanie samochodu	RD			R	R	RD									

5. Implementacja systemu

Rozdział zawiera opis stworzenia serwisu na podstawie architektury MVC za pomocą opisanych wyżej narzędzi programistycznych. Implementacja dotyczy jak back-endu tak i również części klienta – widok dla użytkownika systemu.

5.1. Struktura projektu

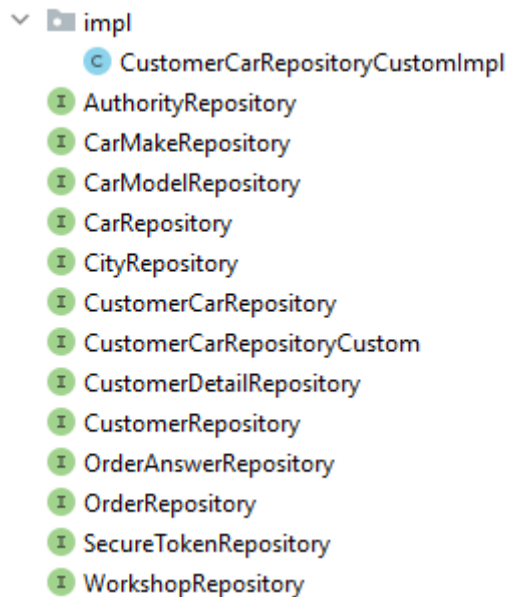
Na rysunku 5.1 przedstawiona hierarchia pakietów serwisu, natomiast w poniższych podrozdziałach zostaną opisane główne warstwy przepływu informacji od części klienta do bazy danych i na odwrót.



Rys. 5.1. Hierarchia pakietów serwisu

5.1.1. Komunikacja z bazą danych

W celu realizacji warstwy komunikacji z bazą danych została wybrana biblioteka Spring Data JPA, realizacja, której zamieszczona do pakietu Java o nazwie „dao”. Na rysunku 5.2 przedstawiona zawartość tego pakietu.



Rys. 5.2. Java pakiet „dao”

Spring Data JPA jest szczegółowo opisany w rozdziale uzasadnienie wyboru narzędzi programistycznych do realizacji serwisu. Na rysunku 5.3 przedstawiony przykładowe repozytorium *CustomerRepository*.

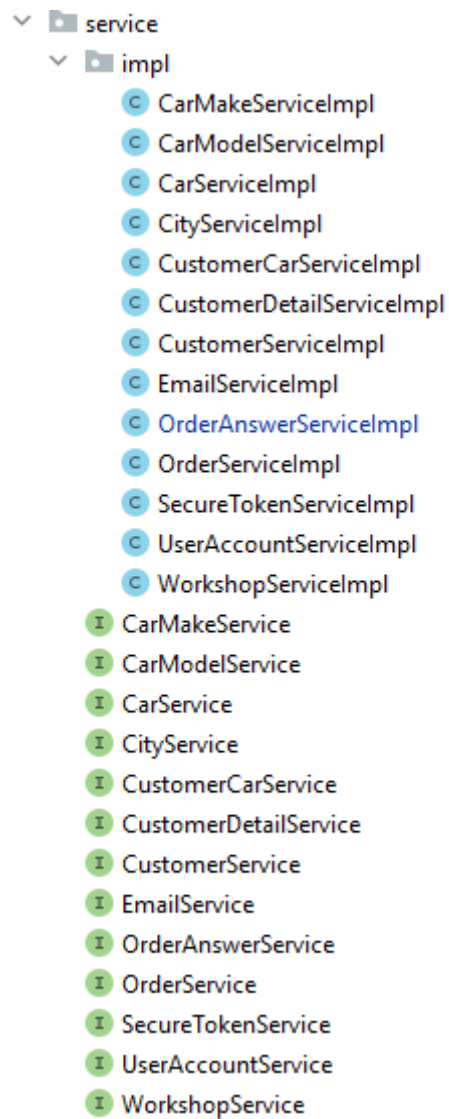
```
public interface CustomerRepository extends JpaRepository<Customer, Integer> {
    @Transactional
    @Modifying
    @Query("delete from Customer c where c.id=:id")
    void deleteCustomerById(Integer id);

    Customer findCustomerByUsername(String username);
    Customer findCustomerById(Integer id);
    List<Customer> findAll();
}
```

Rys. 5.3. Zawartość klasy *CustomerRepository*

5.1.2. Warstwa biznes logiki serwisu

W tej warstwie zamieszczona realizacja głównej logiki serwisu. Warstwa komunikacji z bazą danych zapewnia narzędzia do zmiany lub odczytu danych wykorzystując które warstwa biznes logiki zapewnia sposób przetwarzania tych danych w celu realizacji zadań serwisu przed zapisaniem do bazy danych lub przed wysyłaniem do warstwy kontrolerów. W architekturze MVC przyjęto realizować biznes logikę aplikacji w Java pakiecie „service”. Na rysunku 5.4 przedstawiony Java pakiet „service”, zawierający Java interfejsy oraz klasy implementujące tę interfejsy w pakiecie „impl”.



Rys. 5.4. Java pakiet „service”

Na rysunku 5.5 przedstawiony fragment implementacji warstwy serwisowej na przykładzie klasy *CustomerServiceImpl* zawierający metod realizujący logikę rejestracji.

```

@Service
public class CustomerServiceImpl implements CustomerService {

    @Autowired
    private CustomerRepository customerRepository;

    @Autowired
    private WorkshopService workshopService;

    @Autowired
    private CarService carService;

    @Autowired
    private AuthorityRepository authorityRepository;

    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    @Autowired
    private CustomerDetailService customerDetailService;

    @Autowired
    private OrderService orderService;

    @Autowired
    private MessageSource messageSource;

    @Override
    public void register(CustomerData customerData) throws UserAlreadyExistException, EmailAlreadyExistException {
        boolean isCustomerDetailNotNew = false;
        //sprawdza czy istnieje klient albo warsztat z takim username
        if(checkIfUsernameExist(customerData.getUsername())) {
            //zeruje pole username obiektu który będzie wysłany na Front End
            customerData.setUsername(null);
            throw new UserAlreadyExistException(messageSource.getMessage( s: "user.already.exist.exception"
                , objects: null, LocaleContextHolder.getLocale()));
        }
        //sprawdza czy istnieje klient z takim email
        if(checkIfEmailExist(customerData.getEmail())) {
            if (customerDetailService.isCustomerInCustomerDetail(customerDetailService.findByEmail(customerData.getEmail()
                customerData.setEmail(null);
                throw new EmailAlreadyExistException(messageSource.getMessage( s: "email.already.exist.exception"
                    , objects: null, LocaleContextHolder.getLocale()));
            } else {
                isCustomerDetailNotNew = true;
            }
        }
    }

    Customer customer = new Customer();
    BeanUtils.copyProperties(customerData, customer);
    CustomerDetail customerDetail = new CustomerDetail();
    BeanUtils.copyProperties(customerData, customerDetail);
    encodePassword(customerData, customer);
    customer.setAuthorities
        (Stream.of(authorityRepository.findByAuthority(AuthorityEnum.ROLE_CUSTOMER))
            .collect(Collectors.toSet()));
    customer.setCustomerDetail(customerDetail);
    customerDetailService.save(customerDetail);
    customerRepository.save(customer);
    if(isCustomerDetailNotNew) {
        customerDetailService.sendNotNewCustomerDetailEmailVerificationEmail(customerDetail);
    }
    else {
        customerDetailService.sendCustomerDetailEmailVerificationEmail(customerDetail);
    }
}

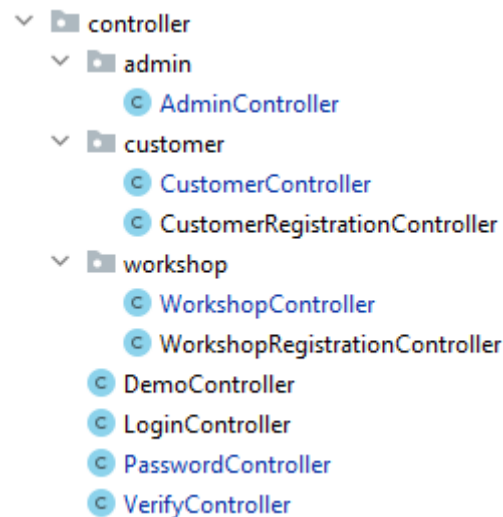
```

Rys. 5.5. Część zawartości klasy *CustomerServiceImpl*

5.1.3. Warstwa kontrolerów

Java klasa oznaczona adnotacją „@Controller” w architekturze MVC nazywa się kontrolerem oraz odpowiada za obsługę żądań klienta, tworzy odpowiedni model danych oraz przekazuje go w celu wyświetlenia dla klienta na warstwę widoków systemu.

W celu realizacji wymaganej funkcjonalności systemu został stworzony java pakiet „controller” zawierający oddzielne klasy kontrolerów dla każdego typu użytkownika zgrupowane w jednoimienne podpakiety (rys. 5.6).



Rys. 5.6. Java pakiet „controller”

Na przykładzie java klasy kontrolera *CustomerController* przedstawionego na rysunku 5.7 została opisana zawartość warstwy kontrolerów.

```

@Controller
@RequestMapping("/customer")
public class CustomerController {

    @Autowired
    private CustomerService customerService;

    @Autowired
    private CityService cityService;

    @Autowired
    private CarService carService;

    @Autowired
    private CustomerDetailService customerDetailService;

    @Autowired
    private OrderService orderService;

    @Autowired
    private OrderAnswerService orderAnswerService;

    @Autowired
    private CarMakeService carMakeService;

    @Autowired
    private CarModelService carModelService;

    @Autowired
    private UserAccountService userAccountService;

    @Autowired
    private MessageSource messageSource;

    @GetMapping("/dashboard")
    public String showCustomerDashboard(Model model
        , @CurrentSecurityContext(expression = "authentication.name") String username) {

        model.addAttribute("customerDetail", customerDetailService.findByCustomerUsername(username));
        model.addAttribute("cityList", cityService.loadCities());
        return "customer/dashboard";
    }
}

```

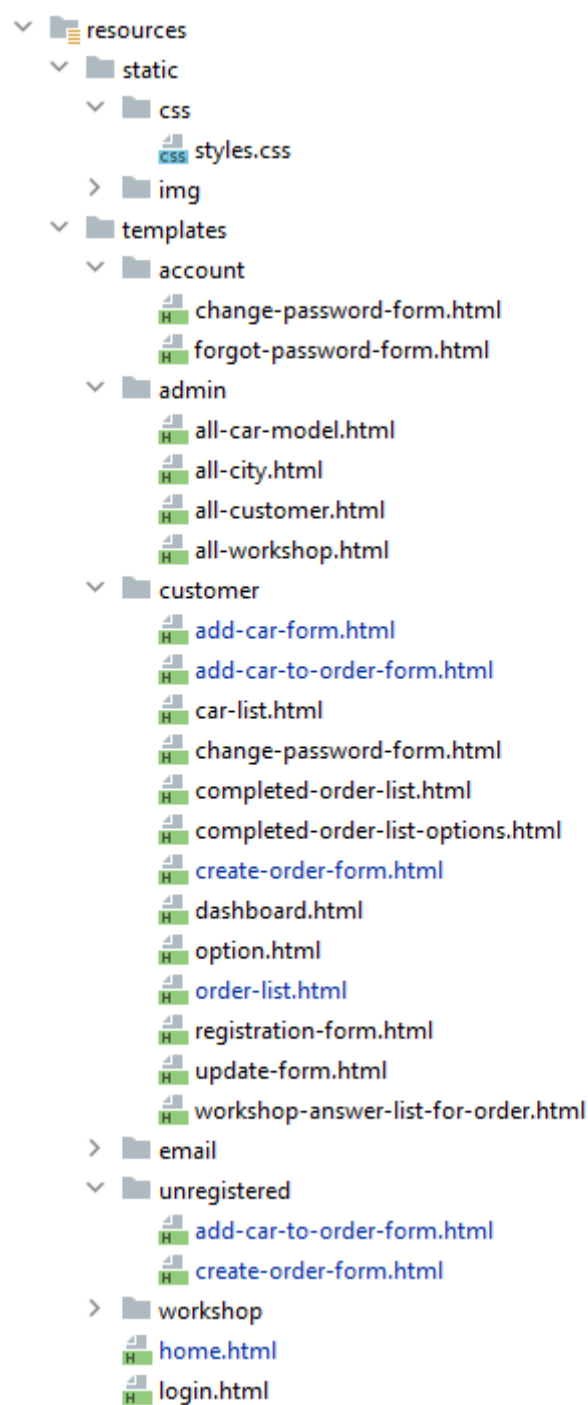
Rys. 5.7. Część zawartości klasy *CustomerController*

Adnotacja „`@RequestMapping`” wskazuje na to, że wszystkie metody danego kontrolera są dostępne pod adresem URL „/customer”. Adnotacja „`@GetMapping`” przed sygnaturą metody wskazuje na to, że dany metod wykorzystuje się do przetwarzanie żądań typu GET, natomiast „/dashbord” w nawiasach adnotacji wskazuje na URL pod którym dana metoda będzie dostępna. Wykorzystanie adnotacji „`@CurrentSecurityContext`” umożliwia wykorzystanie nazwy obecnie zalogowanego użytkownika wewnątrz metody. W celu umożliwienia dostępu do danych uzyskanych dzięki biznes logice wewnątrz metod kontrolerów od strony widoków wykorzystuje się klasa „`Model`”. Dane wewnątrz obiektu klasy „`Model`”

są zapisane w postaci klucz-obiekt oraz dostępne do pobierania poprzez widok o nazwie zwróconej metodą kontrolera za pomocą klucza.

5.1.4. Warstwa widoków serwisu

Widoki serwisu stworzone na podstawie plików HTML z wykorzystaniem atrybutów Thymeleaf oraz podłączeniu do nich plików CSS. Wszystkie pliki z wyżej wymienionym rozszerzeniem zostały dodane do katalogu „resources”. Na rysunku 5.8 przedstawiono wygląd części hierarchii plików odpowiedzialnych za część klienta systemu w drzewie projektu.



Rys. 5.8. Hierarchia plików i katalogów, odpowiadających za część klienta serwisu

6. Przejrzanie i opis serwisu internetowego

Poniżej przedstawiono oraz opisano zaimplementowaną funkcjonalność serwisu internetowego do pośrednictwa między klientem, oraz autoserwisem. Przedstawiono, jak wygląda cała funkcjonalność od strony użytkownika opis implementacji, której został opisany we wcześniejszym rozdziale. Ten rozdział będzie podzielony na cztery części w celu demonstracji funkcjonalności dla niezalogowanego użytkownika, zalogowanego klienta, warsztatu samochodowego oraz administratora.

6.1. Widok serwisu ze strony niezalogowanego użytkownika

Strona domowa dostępna dla niezarejestrowanych użytkowników przedstawiona na rysunku 6.1. Ona zawiera funkcjonalność tworzenia nowego zlecenia, logowania do serwisu, tworzenie konta klienta, tworzenie konta autoserwisu, informacyjny blok oraz na bieżąco aktualizującą się listę miast zawierających warsztaty samochodowe partery serwisu.

Wybierz Twój autoserwis!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ea, laboriosam!

NOWE ZLECENIE



Welcome

Lorem, ipsum dolor sit amet consectetur adipisicing elit. A, iure saepe eaque delectus sunt voluptates deserunt accusantium sed ea dignissimos amet harum asperiores laboriosam perspiciatis ratione nam similique provident quos enim architecto.

Lorem, ipsum dolor sit amet consectetur adipisicing elit. A, iure saepe eaque delectus sunt voluptates deserunt accusantium sed ea dignissimos amet harum asperiores laboriosam perspiciatis ratione nam similique provident quos enim architecto.

Lorem, ipsum dolor sit amet consectetur adipisicing elit. A, iure saepe eaque delectus sunt voluptates deserunt accusantium sed ea dignissimos amet harum asperiores laboriosam perspiciatis ratione nam similique provident quos enim architecto.

AUTOSERWISY W MIASTACH POLSKI

Rzeszów

Warszawa

Gdańsk

Rys. 6.1. Widok strony domowej

Dalej zostaną rozpatrzone formularze rejestracji. Na rysunku 6.2 przedstawiony formularz rejestracji klienta. Z formy jest możliwość przejścia z powrotem do strony domowej, do strony rejestracji warsztatu samochodowego oraz na stronę logowania się do serwisu. Tutaj należy wprowadzić unikatową nazwę użytkownika, hasło, powtórzyć hasło, imię, nazwisko, adres poczty elektronicznej oraz numer telefonu klienta.

Również na rysunku przedstawione są możliwe przy wypełnieniu błędy polegające na niewypełnieniu niezbędnych pól formularza albo wypełnienie przez dane w nieprawidłowej postaci. Błędy opisują, w jaki sposób mają być przedstawione dane do prawidłowej rejestracji konta. W przypadku podania danych w prawidłowy sposób, ale już istniejących w serwisie nazwy użytkownika lub adresu poczty elektronicznej również będzie zwrócony błąd rejestracji.

IREMONT Utwórz konto klienta | Utwórz konto autoserwisu Zaloguj się

Tworzenie konta

Nazwa użytkownika

Nazwa użytkownika (*)

Nazwa użytkownika *

Hasło

Hasło (*)

Hasło *

Powtórz hasło

Powtórz hasło (*)

Hasło *

Imię

mykola

Imię musi zaczynać się od dużej litery oraz zawierać tylko łacińskie litery

Nazwisko

nykolaichuk

Nazwisko musi zaczynać się od dużej litery oraz zawierać tylko łacińskie litery

Email

email (*)

Email *

Numer telefonu

777

Numer telefonu musie mieć format +48-XXX-XXX-XXX lub +48XXXXXXXX lub +48 XXX XXX XXX lub XXX-XXX-XXX lub XXXXXXXX XXX XXX XXX

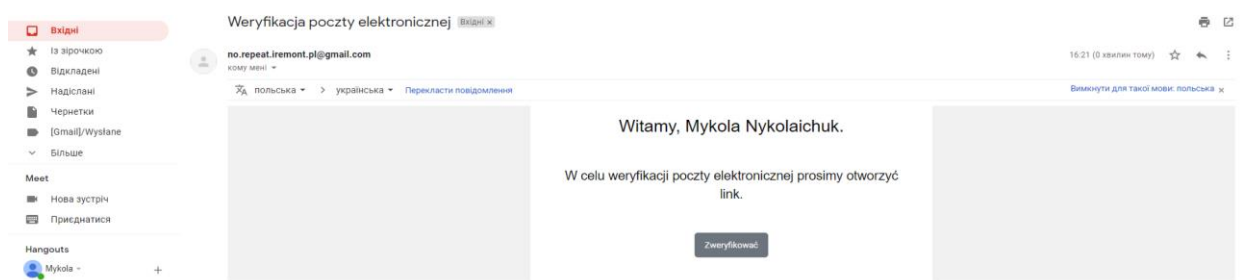
Rys. 6.2. Widok strony rejestracji klienta zawierający błędy przy podaniu danych

Po podaniu prawidłowych danych użytkownik zostanie przekierowany na widok strony logowania i będzie mu wyświetlona wiadomość o udanym utworzeniu konta oraz

niezbędności potwierdzenia adresu poczty elektronicznej w celu aktywacji konta. Na rysunku 6.3 przedstawiony widok strony logowania zawierający opisaną wyżej wiadomość.

Rys. 6.3. Widok strony logowanie przy udanej rejestracji

Na adres poczty elektronicznej zostało wysłane pismo zawierające link weryfikacji. Pismo przedstawiono na rysunku 6.4.



Rys. 6.4. Widok pisma zawierającego link weryfikacji adresu poczty elektronicznej

Po wciśnięciu przycisku weryfikacji klient zostanie przekierowany na widok strony logowania i będzie mu wyświetlona w taki sam sposób jak w przypadku udanej rejestracji wiadomość o udanej aktywacji konta.

W przypadku, jeżeli token zawierający się w URL-adresie linku weryfikacji będzie równy null lub będzie zmieniony, lub będzie już nieaktualny (po 24 godzinach następuje wygaśnięcie aktualności) klient będzie przekierowany na widok strony logowania i będzie mu wyświetlona wiadomość o wystąpieniu błędu przy weryfikacji adresu poczty elektronicznej (rys. 6.5).

The screenshot shows the login page of the IREMONT service. At the top, there is a dark navigation bar with the text "IREMONT" on the left, "Utwórz konto klienta | Utwórz konto autoserwisu" in the center, and "Zaloguj się" on the right. The main content area has a light gray background. In the center, there is a white box titled "Zaloguj się". Inside this box, there is a pink message box that says: "Wygląda, że adres URL linku weryfikacji stracił ważność lub został zmieniony. Prosimy spróbować ponownie." Below this message, there are two input fields: "Nazwa użytkownika" and "Hasło". Below the "Hasło" field is a "Dalej" button. At the bottom of the white box, there is a checkbox labeled "Zapamiętaj mnie" and a link "Nie pamiętasz hasła?".

Rys. 6.5. Widok strony logowania przy nieudanej weryfikacji adresu poczty elektronicznej

Na rysunku 6.6 przedstawiona forma rejestracji autoserwisu. Ona zawiera po analogii z formą rejestracji klienta pola: nazwa użytkownika, hasło, powtórz hasło, adres poczty elektronicznej oraz numer telefonu. Różnicą przed poprzednio rozpatrywaną formą są pola: nazwa firmy, adres oraz wybór miasta spośród dostępnych. Sprawdzenie prawidłowości postaci wprowadzonych danych oraz weryfikacja adresu poczty elektronicznej realizowane są w taki sam sposób jak w przypadku rejestracji klienta.

The screenshot shows the registration page of the IREMONT service. At the top, there is a dark navigation bar with the text "IREMONT" on the left, "Utwórz konto klienta | Utwórz konto autoserwisu" in the center, and "Zaloguj się" on the right. The main content area has a light gray background. In the center, there is a white box titled "Tworzenie konta". Inside this box, there are several input fields: "Nazwa użytkownika", "Hasło", "Powtórz hasło", "Nazwa firmy", "Adres", "Email", "Numer telefonu", and "Miasto". The "Miasto" field is a dropdown menu with "Reszów" selected. Below the "Miasto" field is an "Utwórz" button.

Rys. 6.6. Widok strony rejestracji warsztatu samochodowego

Dla wszystkich użytkowników również dostępna forma logowania się do serwisu, która oprócz formularza logowania zawierającego pola: nazwa użytkownika oraz hasło, zawiera opcję: zapamiętaj mnie oraz nie pamiętasz hasła.

Zaznaczenie opcji zapamiętaj mnie, powoduje, że po zamknięciu i ponownym otwarciu przeglądarki użytkownik nawet po kilku dniach jest zalogowany do serwisu internetowego.

Opcja nie pamiętasz hasło, po naciśnięciu przekieruje do formy podania nazwy użytkownika przedstawionej na rysunku 6.7.

Rys. 6.7. Widok formularza podania nazwy użytkownika do resetowania hasła

Po tym, jak użytkownik wypełni pole, zostanie on przekierowany na widok strony logowania i będzie mu wyświetlona wiadomość o tym, że jeżeli podana nazwa użytkownika istnieje, to na adres poczty elektronicznej zostanie wysłane pismo z instrukcjami resetowania hasła.

Widok pisma z instrukcjami do wznowienia hasła zrealizowany w podobny sposób do pisma z weryfikacją adresu poczty elektronicznej.

W przypadku, jeżeli link będzie nieaktualny lub niepoprawny, użytkownik zobaczy ten sam widok co przy nieudanej rejestracji. Natomiast w przeciwnym przypadku użytkownik zostanie przekierowany na widok zawierający formę do podania nowego hasła.

Nowe hasło będzie sprawdzone w taki sam sposób jak w przypadku rejestracji. Po podaniu nowego poprawnego hasła i kliknięciu przycisku „Dalej” użytkownik zostanie przekierowany na widok strony logowania zawierającej wiadomość o udanej zmianie hasła.

Dla niezarejestrowanego użytkownika również dostępna funkcjonalność tworzenia zlecenia. Po naciśnięciu zakładki „NOWE ZLECENIE” użytkownik będzie przekierowany do widoku formularza podania danych samochodu. Widok formularza przedstawiony na rysunkach 6.8 oraz 6.9.

Rys. 6.8. Widok formularza dodania samochodu, wybór marek

Na rysunku 6.8 widać, że najpierw trzeba wybrać markę samochodu i potwierdzić wybór. Na rysunku 6.9 przedstawiona forma dodania samochodu przy już wybranej marce. Tutaj należy podać model, rok produkcji, numer nadwozia, numer rejestracyjny pojazdu oraz rodzaj paliwa.

Rys. 6.9. Widok formularza dodania samochodu

Formularz dodania samochodu do zlecenia sprawdza dane i zwraca błędy w przypadku niewypełniania pól obowiązkowych. Rok musi być z okresu (1900 – bieżący rok), numer nadwozia (VIN) musi zawierać siedemnaście symbolów cyfr lub łacińskich liter, numer rejestracyjny musi zawierać tylko cyfry lub litery łacińskie. Błędy opisują, w jaki sposób mają być przedstawione dane do prawidłowej operacji dodania samochodu.

Po dodaniu samochodu użytkownik zostanie przekierowany na formę tworzenia zlecenia. Zawiera ona dane wprowadzone przez użytkownika w poprzedniej formie oraz pola do wprowadzenia: imiona, nazwiska, adresy poczty elektronicznej, numeru telefonu, które sprawdzają się na błędy w taki sam sposób jak przy rejestracji klienta, pola „Opis naprawy” będącego obowiązkowym oraz wybór miasta (rys.6.10).

Nowe zlecenie

Renault

Master

2020

Diesel

RZE80745

VF1MAF4SE53894520

Imię
Mykola

Nazwisko
Nykolaichuk

Email
mnykolaichuk90@gmail.com

Numer telefonu
+48 791-863-388

Opis naprawy
Wymiana klocków hamulcowych

Reszów

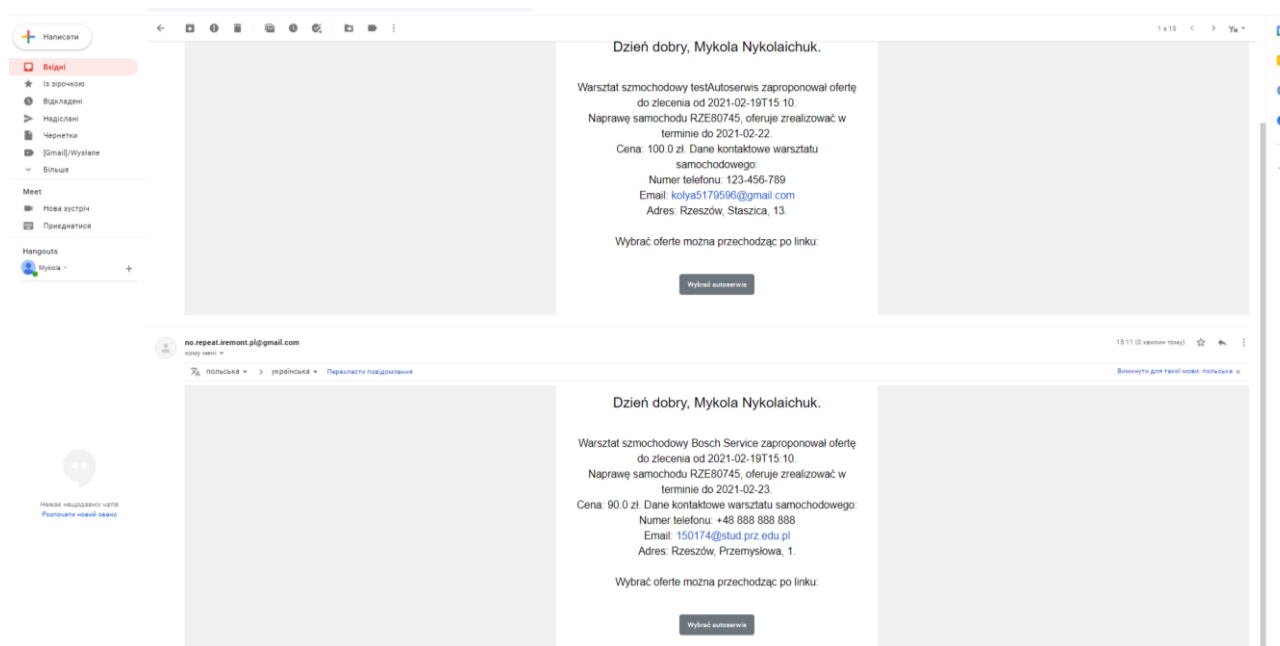
Utwórz

Rys. 6.10. Widok formularza tworzenia zlecenia

Przy kliknięciu przycisku „Utwórz”, jeżeli na podany przez użytkownika adres poczty elektronicznej już jest zarejestrowane konto, zostanie on przekierowany na widok strony domowej zawierającej wiadomość o nieudanym tworzeniu zlecenia w związku z istniejącym kontem użytkownika oraz instrukcją o zalogowaniu i kontynuacji tworzenia zlecenia. Za dodanie oraz usunięcie dostępnych do wyboru miast odpowiada administrator serwisu internetowego, ale w przypadku, jeżeli miasto już albo jeszcze będzie dostępne, ale bez dostępnych partnerów autoserwisów użytkownik będzie również przekierowany na widok strony domowej zawierającej wiadomość, że niestety w tej chwili w wybranym mieście nie ma dostępnych autoserwisów partnerów. Jeżeli wymienione wyżej sytuacje nie wystąpią, użytkownik będzie przekierowany na widok strony domowej zawierającej wiadomość o udanym tworzeniu zlecenia oraz niezbędności potwierdzenia adresu poczty elektronicznej w celu aktywacji zlecenia.

Na pocztę elektroniczną klient otrzymuje takie samo pismo jak przy rejestracji i po udanej weryfikacji adresu poczty elektronicznej zlecenie zostanie dostępne dla wszystkich partnerów autoserwisów znajdujących się w wybranym przez użytkownika mieście.

Warsztaty samochodowe oferują swoją cenę oraz termin wykonania zlecenia. O nowych ofertach użytkownik jest informowany za pomocą pisma na adres poczty elektronicznej (rys. 6.11). Pismo zawiera link do wyboru warsztatu samochodowego będącego realizować zlecenie.



Rys. 6.11. Widok pisma zawierającego oferty od warsztatów samochodowych

Kiedy wybrany przez użytkownika warsztat samochodowy zrealizuje zlecenie, użytkownik zostanie o tym poinformowany pismem na pocztę elektroniczną.

6.2. Widok serwisu ze strony klienta

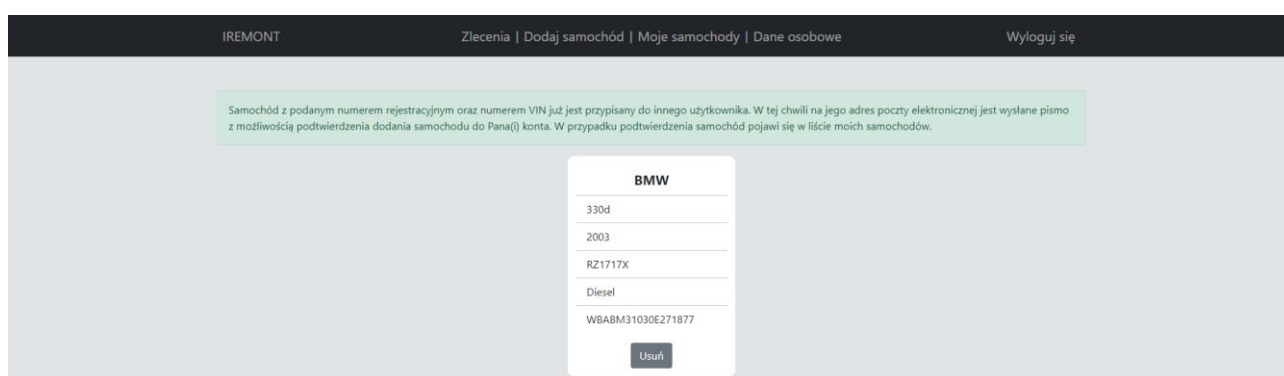
Po zalogowaniu klient zostaje przekierowany na widok strony domowej klienta, która w porównaniu do strony domowej użytkownika niezalogowanego rozszerzona funkcjonalnością przedstawioną na rysunku 6.12.



Rys. 6.12. Widok strony domowej klienta

Klient ma możliwość dodania samochodów do listy swoich samochodów, dzięki czemu przy tworzeniu zlecenia klient nie będzie musiał za każdym razem ręcznie wpisywać dane samochodu, a po prostu wybiera z listy swoich samochodów. Widok formy dodania samochodu jest analogiczny do widoku dodania samochodu do zlecenia w przypadku tworzenia zlecenia użytkownikiem niezarejestrowanym.

Samochód jest unikatowy, jeżeli w innych klientów serwisu w liście „Moje samochody” nie ma pojazdu z tym samym numerem nadwozia (VIN) i numerem rejestracyjnym. W przypadku, jeżeli wprowadzony samochód już jest we własności innego klienta lub klientów na ich adresy poczty elektronicznej będą wysłane pisma z informacją, że inny klient serwisu chce dodać ich samochód do listy „Moje samochody”. Natomiast klient, który aktualnie dodaje auto, będzie przekierowany na widok strony „Moje samochody” zawierający wiadomość o tym, że samochód będzie dodany do jego listy po potwierdzeniu przez innych właścicieli. Na rysunku 6.13 przedstawiony widok strony „Moje samochody” zawierający już wcześniej dodany pojazd o numerze rejestracyjnym „RZ1717X” oraz opisaną wyżej wiadomość spowodowaną próbą dodania nowego pojazdu do listy „Moje samochody”.



Rys. 6.13. Widok strony „Moje samochody” zawierającej wiadomość o dodaniu samochodu zawierającego właściciela lub listy właścicieli

Pismo zawiera link potwierdzający dodanie pojazdu przez innego klienta i jego dane osobowe oraz kontaktowe.

Wciśnięcie zakładki „NOWE ZLECENIE” przekieruje klienta na widok dodania samochodu z możliwością: wybory pojazdu spośród listy „Moje samochody”, podania danych ręcznie bez zapisywania samochodu do listy dodanych, przekierowania do formy dodania samochodu do listy „Moje samochody” (rys. 6.14).

Rys. 6.14. Widok formularza dodania samochodu do zlecenia

Po dodaniu samochodu w dowolny sposób klient zostanie przekierowany do formularza tworzenia zlecenia (rys. 6.15).

Rys. 6.15. Widok formularza tworzenia zlecenia

Jak i w przypadku tworzenia zlecenia przez niezarejestrowanego użytkownika, jeżeli w wybranym mieście nie ma autoserwisów partnerów, zlecenie nie będzie stworzone. Po udanym stworzeniu klient zostanie przekierowany na widok strony „Zlecenia” (rys.6.16).

IREMONT

Nowe Zlecenie | Zlecenia | Archiwum zleceń

Wyloguj się

Zlecenie zostało stworzone.

#	Numer rejestracyjny	Data stworzenia	Opis zlecenia	Nazwa autoserwisu	Miasto	Numer telefonu	Data realizacji	Cena
1	RZE80745	2021-02-15T14:35	Wymiana klocków hamu...					<div>Oferty</div> <div>Usuń</div>
2	RZ9445U	2021-02-16T20:36	Wymiana oleju.					<div>Oferty</div> <div>Usuń</div>

Rys. 6.16. Widok strony „Zlecenia” zawierającej wiadomość o udanym stworzeniu nowego zlecenia

Na rysunku 6.16 są dwa zlecenia, jedno stworzone zalogowanym klientem dla samochodu z numerem rejestracyjnym „RZ9445U” oraz drugie zlecenie stworzone za pomocą adresu poczty elektronicznej klienta jeszcze przed rejestracją. Będąc użytkownikiem niezalogowanym, klient nie wybrał autoserwis do realizacji stworzonego zlecenia do naprawy samochodu o numerze rejestracyjnym „RZE80754”, dlatego zostało ono przeniesione do widoku strony „Zlecenia” oraz już zaproponowane przez autoserwisy oferty są dostępne po kliknięciu przycisku „Oferty”.

Pisma na adres poczty elektronicznej informujące o nowych ofertach od warsztatów samochodowych wyglądają w taki sam sposób jak w przypadku zlecenia stworzonego użytkownikiem niezarejestrowanym za wyłączeniem tego, że link przekierowuje na zakładkę „Zlecenia”.

Po kliknięciu przycisku „Oferty” klient zostanie przekierowany na widok strony z ofertami od warsztatów samochodowych (rys. 6.17).

IREMONT		Wstecz					Wyloguj się	
#	Nazwa autoserwisu	Miasto	adres	Numer telefonu	Data realizacji	Cena		
1	testAutoserwis	Reszów	Staszica, 13	123-456-789	2021-02-15	90.0	Wybrać	
2	Bosch Service	Reszów	ul. Przemysłowa, 1	+48 888-888-888	2021-02-15	100.0	Wybrać	

Rys. 6.17. Widok strony „Oferty”

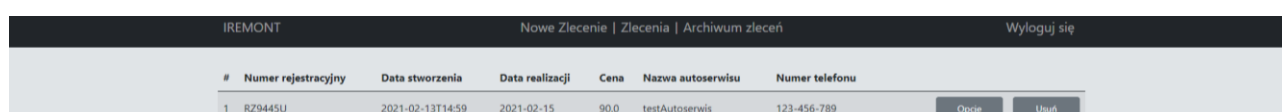
Po wybraniu optymalnej dla klienta oferty zostanie on przekierowany z powrotem na widok strony „Zlecenia”, gdzie w informacji o zlecenie pojawi się informacja o autoserwisie realizującym zlecenie (rys. 6.18).

IREMONT		Nowe Zlecenie Zlecenia Archiwum zleceń					Wyloguj się	
#	Numer rejestracyjny	Data stworzenia	Opis zlecenia	Nazwa autoserwisu	Miasto	Numer telefonu	Data realizacji	Cena
1	RZE80745	2021-02-15T14:35	Wymiana klocków hamu...	testAutoserwis	Rzeszów	123-456-789	2021-02-16	100.0
2	RZ9445U	2021-02-16T20:36	Wymiana oleju.					Oferty Usun

Rys. 6.18. Widok strony „Zlecenia” zawierającej zlecenie w procesie realizacji

Klient ma możliwość usunięcia zlecenia znajdującego się na etapie realizacji. W tym przypadku autoserwis będzie poinformowany o usunięciu pismem na adres poczty elektronicznej zawierającym dane kontaktowe klienta.

Kiedy autoserwis zaznaczy zlecenie jak zrealizowane, klient będzie poinformowany pismem na adres poczty elektronicznej. Dane o zleceniu będą przemieszczone na zakładkę „Archiwum zleceń” (rys. 6.19). Jeżeli klient już miał zrealizowane zlecenia, będąc użytkownikiem niezarejestrowanym, informacja o nich też będzie dostępna na zakładce „Archiwum zleceń”.

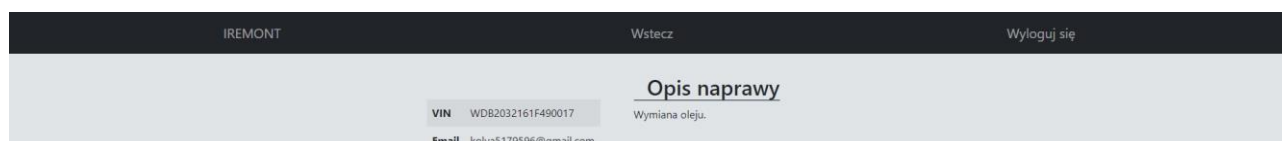


IREMONT							Nowe Zlecenie Zlecenia Archiwum zleceń		Wyloguj się
#	Numer rejestracyjny	Data stworzenia	Data realizacji	Cena	Nazwa autoserwisu	Numer telefonu			
1	RZ9445U	2021-02-13T14:59	2021-02-15	90.0	testAutoserwis	123-456-789			

Rys. 6.19. Widok strony „Archiwum zleceń”

W przypadku, jeżeli wszystkie autoserwisy partnerzy, zamiast zaoferować swoją propozycję, usuną zlecenie, klient zostanie poinformowany o tym, że żaden autoserwis nie zaoferował realizacji zlecenia pismem na adres poczty elektronicznej. Zlecenie będzie przemieszczone do zakładki „Archiwum zleceń”.

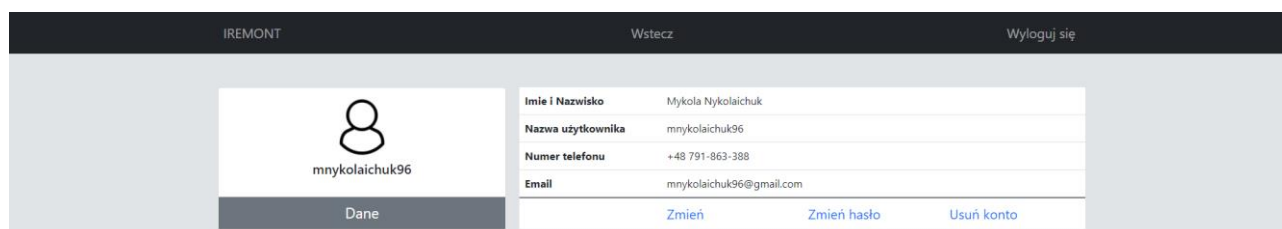
Oprócz możliwości usunięcia zlecenia dla klienta są dostępne szczegóły zlecenia po kliknięciu przycisku „Opcje” (rys. 6.20).




IREMONT		Wstecz	Wyloguj się
		<u>Opis naprawy</u>	
VIN	WDB2032161F490017	Wymiana oleju.	
Email	kolya5179596@gmail.com		

Rys. 6.20. Widok strony „Opcje” zlecenia w archiwum zleceń

Dane osobowe oraz kontaktowe klienta są dostępne do obejrzenia, oraz zmiany po wciśnięciu zakładki „Dane osobowe” (rys. 6.21).

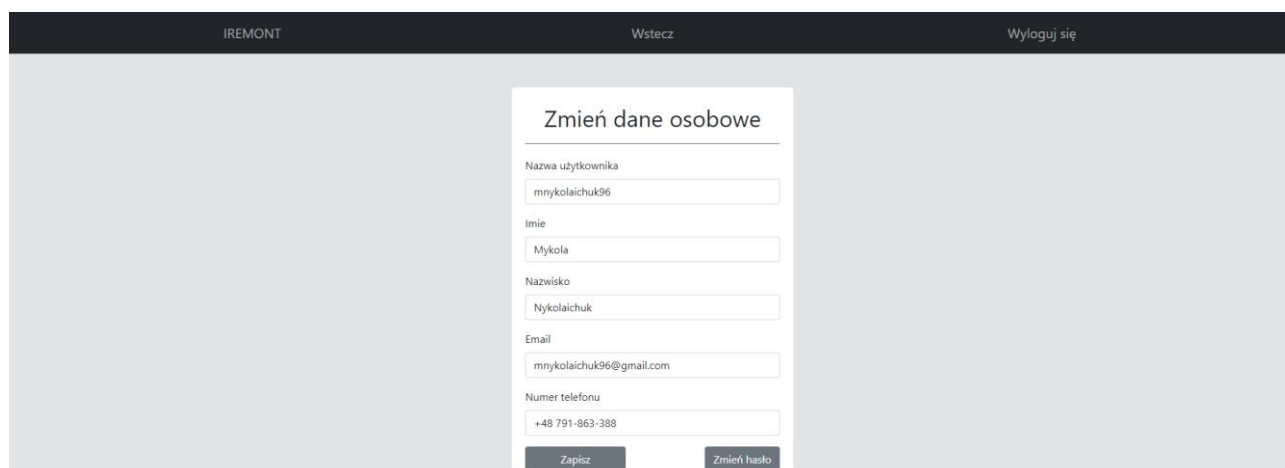


IREMONT		Wstecz	Wyloguj się
 mnykolaichuk96 Dane			
Imię i Nazwisko		Mykola Nykolaichuk	
Nazwa użytkownika		mnykolaichuk96	
Numer telefonu		+48 791-863-388	
Email		mnykolaichuk96@gmail.com	
		Zmień	Zmień hasło
		Usuń konto	

Rys. 6.21. Widok strony „Dane osobowe”

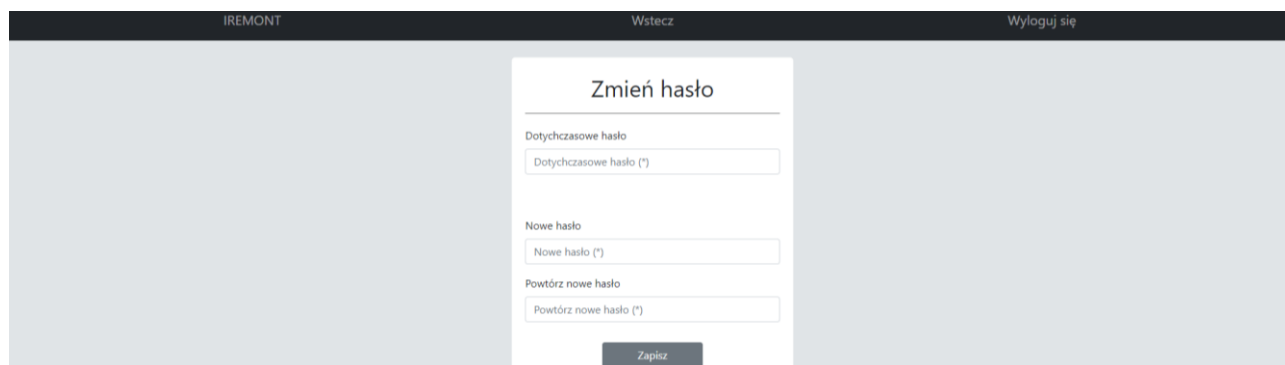
Formularz zmiany danych osobowych dostępny po kliknięciu przycisku „Zmień” (rys. 6.22). Sprawdzanie wprowadzonych danych realizowane w taki sam sposób jak w przypadku rejestracji. Klient też może zmienić nazwę użytkownika oraz adres poczty

elektronicznej. W obu przypadkach po zapisaniu nowych danych następuje automatyczne wylogowanie z serwisu. W przypadku zmiany adresu poczty elektronicznej konto klienta będzie ponownie aktywowane dopiero po weryfikacji nowego adresu poczty elektronicznej.



Rys. 6.22. Widok formularza zmiany danych osobowych

Po kliknięciu przycisku „Zmień hasło” na formularzu zmiany danych osobowych (rys.6.22) albo na widoku strony „Dane osobowe” (rys.6.21) klient zostanie przekierowany na formularz zmiany hasła (rys. 6.23). Sprawdzanie wprowadzonych danych realizowane w taki sam sposób jak w przypadku rejestracji.

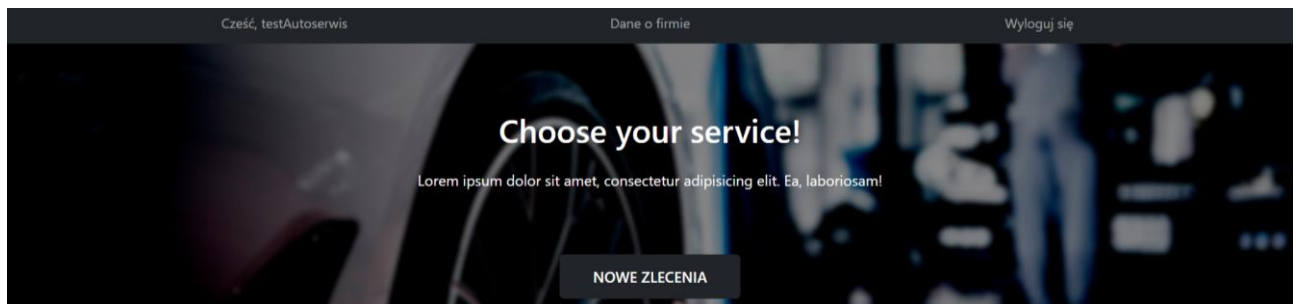


Rys. 6.23. Widok formularza zmiany hasła

Dla klienta dostępna możliwość usunięcia konta, po czym klient będzie przekierowany na widok strony domowej użytkownika (rys. 6.1) zawierającej wiadomość o udanym usunięciu konta.

6.3. Widok serwisu ze strony autoserwisu

Po zalogowaniu autoserwis zostaje przekierowany na widok strony domowej auto-serwisu (rys.6.24).



Rys. 6.24. Widok strony domowej autoserwisu

Zakładka „Dane o firmie” realizowana w taki sam sposób i z tą samą funkcjonalnością jak w przypadku zakładki „Dane osobowe” konta klienta z jednym wyłączeniem, autoserwis nie ma możliwości usunięcia konta, dopóki ma bieżące zlecenia. Dopiero po realizacji wszystkich aktualnych zleceń umożliwia się funkcja usunięcie konta.

Kiedy użytkownik niezalogowany lub zalogowany klient tworzy nowe zlecenie, warsztat samochodowy otrzymuje pismo na adres poczty elektronicznej zawierające dane o nowoutworzonym zleceniu oraz link, który przekieruje autoserwis na widok zakładki „NOWE ZLECENIA” (rys. 6.25).

IREMONT		Nowe zlecenia Bieżące zlecenia Archiwum zleceń						Wyloguj się	
#	Data stworzenia	Marka	Model	Rok	Imię i Nazwisko	Numer telefonu	Data Realizacji	Cena	
1	2021-02-16T20:36	Mercedes-Benz	C270	2003	Mykola Nikolaichuk	+48 791-863-388	dd.mm.pppp	<input type="text" value="Cena (*)"/>	<input type="button" value="Zaoferować"/> <input type="button" value="Opcje"/> <input type="button" value="Usuń"/>

Rys. 6.25. Widok strony „NOWE ZLECENIA”

Warsztat samochodowy ma możliwość obejrzeć szczegóły zlecenia przez kliknięcie przycisku „Opcje”. Widok strony „Opcje” jest realizowany w taki sam sposób, jak wyżej przedstawiono dla klienta (rys. 6.20). Jest możliwość usunięcia zlecenia oraz zaoferowania swojej ceny i daty realizacji. Te dwa pola są obowiązkowe do wypełnienia, jeżeli warsztat samochodowy chce zaoferować zrealizowanie zlecenia dla klienta. Data realizacji nie może być przeszła, inaczej wystąpi błąd. Cena nie może być ujemną.

Kiedy warsztat samochodowy zaoferował swoje warunki realizacji zlecenia i został wybrany przez klienta, na adres poczty elektronicznej autoserwisu zostaje wysłane pismo z informacją o wybraniu go do realizacji zlecenia oraz linkiem do zakładki „Bieżące zlecenia”, do której zostają przeniesione zlecenia realizujące się warsztatem samochodowym (rys. 6.26).

IREMONT		Nowe zlecenia Bieżące zlecenia Archiwum zleceń						Wyloguj się	
#	Numer rejestracyjny	Data stworzenia	Data realizacji	Cena	Imię	Nazwisko	Numer telefonu		
1	RZE80745	2021-02-15T14:35	2021-02-16	100.0	Mykola	Nykolaichuk	+48 791-863-388	Opcje	Zrealizowany

Rys. 6.26. Widok strony „Bieżące zlecenia”

Jak i w przypadku nowych zleceń istnieje możliwość obejrzenia szczegół zlecenia za pomocą przycisku „Opcje”.

Po realizacji napraw wymienionych w opisie zlecenia warsztat samochodowy w celu poinformowania klienta o realizacji zlecenia może wcisnąć przycisk „Zrealizowany”. Do klienta zostanie wysłane pismo informacyjne na adres poczty elektronicznej, natomiast zlecenie będzie przenieszone do widoku zakładki „Archiwum zleceń” (rys. 6.27).

IREMONT		Nowe zlecenia Bieżące zlecenia Archiwum zleceń						Wyloguj się	
#	Numer rejestracyjny	Data stworzenia	Data realizacji	Cena	Imię	Nazwisko	Numer telefonu		
1	RZE80745	2021-02-15T14:35	2021-02-16	100.0	Mykola	Nykolaichuk	+48 791-863-388	Opcje	Usuw

Rys. 6.27. Widok strony „Archiwum zleceń”

6.4. Widok serwisu ze strony administratora

Po zalogowaniu administrator serwisu internetowego zostaje przekierowany na widok strony „Klienci” gdzie przedstawione wszystkie klienci z możliwością usunięcia (rys. 6.28).

IREMONT		Klienci Autoserwis Miasta Samochody				Wyloguj się	
#	Username	Imię	Nazwisko	Email	Numer telefonu		
1	mnykolaichuk96	Mykola	Nykolaichuk	mnykolaichuk96@gmail.com	+48 791-863-388	Usuw	

Rys. 6.28. Widok strony „Klienci”

Wciśnięcie zakładki „Autoserwis” przekierowuje na widok strony „Autoserwis” (rys. 6.29) zawierającej taką samą funkcjonalność jak widok strony „Klienci” (rys. 6.28).

IREMONT		Klienci Autoserwis Miasta Samochody				Wyloguj się	
#	Username	Nazwa firmy	Miasto	Adres	Email	Numer telefonu	
1	workshop	testAutoserwis	Rzeszów	Staszica, 13	kolya5179596@gmail.com	123-456-789	Usuw
2	workshop96	Bosch Service	Rzeszów	Przemysłowa, 1	150174@stud.prz.edu.pl	+48 888 888 888	Usuw

Rys. 6.29. Widok strony „Autoserwis”

Dodanie oraz usuwanie dostępnych do wyboru miast odbywa się w zakładce „Miasta” (rys. 6.30).

IREMONT			Klienci Autoserwisy Miasta Samochody	Wyloguj się
#	Miasto	Liczba autoserwisów		
#	<input type="text" value="Miasto"/>	<input type="button" value="Dodaj"/>		
1	Rzeszów	2	<input type="button" value="Usuń"/>	
2	Warszawa	0	<input type="button" value="Usuń"/>	
3	Gdańsk	0	<input type="button" value="Usuń"/>	

Rys. 6.30. Widok strony „Miasta”

Dodanie oraz usuwanie dostępnych do wyboru morek, oraz modeli samochodów odbywa się w zakładce „Samochody” (rys. 6.31).

IREMONT			Klienci Autoserwisy Miasta Samochody	Wyloguj się
#	Marka	Model	Liczba samochodów w systemie	
#	<input type="text" value="Marka"/>	<input type="text" value="Model"/>	<input type="button" value="Dodaj"/>	
1	Acura	2.2CL	0	<input type="button" value="Usuń"/>
2	Acura	2.3CL	0	<input type="button" value="Usuń"/>
3	Acura	2.5TL	0	<input type="button" value="Usuń"/>
4	Acura	3.0CL	0	<input type="button" value="Usuń"/>
5	Acura	3.2CL	0	<input type="button" value="Usuń"/>
6	Acura	3.2TL	0	<input type="button" value="Usuń"/>
7	Acura	3.5 RL	0	<input type="button" value="Usuń"/>
8	Acura	CL Models	0	<input type="button" value="Usuń"/>
9	Acura	ILX	0	<input type="button" value="Usuń"/>
10	Acura	Integra	0	<input type="button" value="Usuń"/>
11	Acura	Legend	0	<input type="button" value="Usuń"/>
12	Acura	MDX	0	<input type="button" value="Usuń"/>
13	Acura	NSX	0	<input type="button" value="Usuń"/>
14	Acura	Other Acura Models	0	<input type="button" value="Usuń"/>
15	Acura	RDX	0	<input type="button" value="Usuń"/>
16	Acura	RL	0	<input type="button" value="Usuń"/>
17	Acura	RL Models	0	<input type="button" value="Usuń"/>

Rys. 6.31. Widok strony „Samochody”

7. Podsumowanie

Celem pracy było wytworzenie systemu pośrednictwa między właścicielem samochodu a warsztatem samochodowym. System miał umożliwiać wybór najlepszej oferty dla klienta oraz znalezienie większej liczby klientów dla autoserwisu.

Cel pracy został osiągnięty. Autor za wkład własny uważa:

1. Sporządzenie specyfikacji systemu w postaci słownego opisu realizowanych funkcji oraz identyfikacja typów użytkowników.
2. Utworzenie hierarchii funkcji systemu.
3. Utworzenie diagramów przypadków użycia.
4. Zaprojektowanie relacyjnej bazy danych w postaci diagramu bazy danych.
5. Zaprojektowanie zależności między funkcjami a encjami w formie macierzy CRUD.
6. Dobór i przegląd technologii informatycznych (język Java, framework Spring i Thymeleaf).
7. Implementacja systemu informatycznego z użyciem wspomnianych technologii.
8. Utworzenie dokumentacji użytkownika systemu oraz dokumentacji technicznej.

Załącznik A. Instrukcja instalacji

W celu uruchomienia aplikacji na komputerze należy wykonać następujące kroki:

- skopiować plik zasoby/Docker Desktop Installer.exe na dysk;
- zainstalować Docker;
- skopiować plik zasoby/docker-compose.yml na dysk;
- uruchomić wiersz poleceń;
- przejść do foldera zawierającego skopiowane zasoby;
- uruchomić serwis internetowy polecenie docker-compose up.

Po wykonaniu opisanych kroków serwis internetowy będzie dostępny pod adresem localhost:8080, panel phpMyAdmin pod adresem localhost:8082.

Załącznik B. Zawartość płyty CD

Płyta CD dołączona do projektu inżynierskiego zawiera następujące pliki:

- MNYkolaichukDyplom.pdf – wersja projektu inżynierskiego w formacie PDF;
- src/ – katalog z kodem źródłowym serwisu internetowego;
- zasoby/ – pliki niezbędne do uruchomienia aplikacji:
 - Docker Desktop Installer.exe – plik do instalacji Docker;
 - docker-compose.yml – plik do uruchomienia kontenerów Docker.

Literatura

- [1] <https://pl.wikipedia.org/wiki/Java>. Dostęp 7.07.2022
- [2] <https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql/>. Dostęp 9.07.2022
- [3] <https://pl.euro-linux.com/blog/postgresql-vs-mysql-ktora-baza-open-source-jest-lepsza/>. Dostęp 9.07.2022
- [4] <https://habr.com/ru/company/yandex/blog/481688/>. Dostęp 7.07.2022
- [5] <https://java-ru-blog.blogspot.com/2019/12/how-works-jvm.html>. Dostęp 7.07.2022
- [6] <https://www.youtube.com/watch?v=AnRKrnbiVzU>. Dostęp 7.07.2022
- [7] <https://www.samouczekprogramisty.pl/interfejsy-w-jezyku-java/#fn:domyslne>. Dostęp 9.07.2022
- [8] <https://www.samouczekprogramisty.pl/wyjatki-w-jezyku-java/>. Dostęp 9.07.2022
- [9] <https://javastart.pl/baza-wiedzy/wyjatki/ioexception>. Dostęp 10.07.2022
- [10] <https://edu.pjwstk.edu.pl/wyklady/ppj/scb/PrgJav/PrgJav.html>. Dostęp 10.07.2022
- [11] <http://java-online.ru/garbage-collection.xhtml>. Dostęp 7.07.2022
- [12] <https://www.tiobe.com/tiobe-index/>. Dostęp 7.07.2022
- [13] <https://docs.spring.io/spring-framework/docs/3.0.0.M4/reference/html/ch01s02.html>. Dostęp 15.07.2022
- [14] https://pl.wikipedia.org/wiki/Mapowanie_obiektowo-relacyjne. Dostęp 15.07.2022
- [15] <https://proselyte.net/tutorials/spring-tutorial-full-version/spring-mvc-framework/>. Dostęp 15.07.2022
- [16] <https://javastudy.ru/spring-mvc/spring-mvc-basic/>. Dostęp 15.07.2022
- [17] <http://spring-projects.ru/projects/spring-data/jpa/>. Dostęp 16.07.2022
- [18] <https://www.javappa.com/kurs-spring/spring-data-jpa-zapytania-wbudowane>. Dostęp 16.07.2022
- [19] <https://www.javappa.com/kurs-spring/spring-data-jpa-zapytania-wlasne>. Dostęp 16.07.2022
- [20] <https://spring-projects.ru/projects/spring-security/>. Dostęp 18.07.2022
- [21] <https://habr.com/ru/post/203318/>. Dostęp 18.07.2022
- [22] Felipe Gutierrez, Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices, str. 520.
- [23] Hellerstein J.M. Architecture of a database system. Foundations and Trends in Databases [Tekst]: J. M. Hellerstein, Michael Stonebraker, James Hamilton, 2007. – 141–259 str.
- [24] <https://www.oracle.com/pl/database/what-is-a-relational-database/>. Dostęp 18.07.2022

[25] <https://excelbi.pl/czym-jest-relacyjna-baza-danych-jak-tworzyc-relacje-w-powerpivot/>.

Dostęp 20.07.2022

[26] <http://www.deskbook.info/hibernate/entity/4-entity-about.html>. Dostęp 20.07.2022

STRESZCZENIE PRACY DYPLOMOWEJ INŻYNIERSKIEJ

SYSTEM POŚREDNICTWA USŁUG MECHANIKÓW SAMOCHODOWYCH

Autor: Mykola Nykolaichuk, nr albumu: EF-DI-150174

Opiekun: dr inż. Grzegorz Dec

Słowa kluczowe: Java, Spring, Thymeleaf, MVC, ORM, relacyjna baza danych.

Projekt inżynierski polegał na wytworzeniu systemu pośrednictwa między właścicielem samochodu a warsztatem samochodowym. Serwis internetowy został zaprojektowany w oparciu o architekturę MVC. Został zaimplementowany przy użyciu frameworku Spring, biblioteki Hibernate do współpracy z relacyjną bazą danych w technologii ORM, uwierzytelnienia użytkowników w technologii Spring. Relacyjna baza danych implementowana za pośrednictwem systemu zarządzania relacyjnymi bazami danych MySQL. Część klienta zaimplementowana przez bibliotekę Thymeleaf umożliwiającą przechowywanie szablonów widoków po stronie serwera.

RZESZOW UNIVERSITY OF TECHNOLOGY

Rzeszow, 2023

Faculty of Electrical and Computer Engineering

DIPLOMA THESIS (BS) ABSTRACT

WEBSITE FOR INTERMEDIATION BETWEEN CLIENT AND CAR REPAIR SERVICES

Author: Mykola Nykolaichuk, code: EF-DI -150174

Supervisor: EngD. Grzegorz Dec

Key words: Java, Spring, Thymeleaf, MVC, ORM, relational database.

The engineering project consisted in creating an intermediary system between the car owner and the car repair shop. The website was designed based on the MVC architecture. It was implemented using the Spring framework, Hibernate library for cooperation with a relational database in ORM technology, user authentication in Spring technology. A relational database implemented via the MySQL relational database management system. The client part implemented by the Thymeleaf library to store server-side view templates.