Spis treści

Rozdział 1

Model konceptualny bazy danych

1.1 Opis świata rzeczywistego

Warsztat samochodowy „Grzemar Repair" znajduje się we Wrocławiu, przy ulicy Kościuszki 43. Przeprowadzane są tam przeglądy i naprawy pojazdów związane z uszkodzeniami mechaniki, hydrauliki oraz elektryki. Dodatkowo istnieje możliwość wymiany oraz naprawy opon. W warsztacie pracuje 7 osób: właściciel, który jest jednocześnie mechanikiem, oraz 6 mechaników. Z racji wykwalifikowanej kadry oraz wysokiej jakości usług, warsztat ma duże perspektywy rozwoju oraz poszerzenia zatrudnienia. Jedną z form rozbudowy może być otwarcie drugiego warsztatu w innej części miasta, co zdecydowanie zwiększyłoby zatrudnienie oraz ilość przetwarzanych danych.

Dotychczasowo cała działalność dokumentowana była w formie papierowej, ale z powodu ciągłego rozwoju warsztatu, jak i szerokiej dostępności technologii, zdecydowano zmianę tradycyjnej formy dokumentacji, na przechowywanie dokumentacji w formie bazy danych. Zmiana ta przynosi korzyści zarówno dla warsztatu, ponieważ umożliwia prowadzenie dokumentacji w formie elektronicznej, usprawnia zarządzanie zasobami jak też umożliwia integrację z innymi systemami, np. bazą danych części zamiennych oraz hurtownią oferującą te części. Do dokumentacji należą m.in. dane pojedynczej naprawy, faktury za części, raporty tygodniowe/miesięczne, dane pracowników, klientów i samochodów.

1.1.1 Klient

Ze względu na podejście do klienta oraz usprawnienie pracy warsztatu, jedną z funkcji, które powinny być dostępne ze strony klienta, jest dostęp do *panelu klienta* z poziomu przeglądarki internetowej. W obrębie tego panelu, klient powinien mieć dostęp do:

- umówienia się na przegląd/naprawę samochodu,
- sprawdzenia stanu naprawy,
- wyświetlenia historii napraw wszystkich samochodów (wyświetlanie względem klienta) lub wybranego pojazdu (wyświetlanie względem samochodu),
- sprawdzenia terminu przeglądu okresowego.

Aby klient mógł skorzystać z powyższych funkcji, muszą być spełnione pewne wymagania. Po pierwsze, klient musi mieć założone konto oraz przypisany samochód. Powyższy warunek zostaje spełniony w momencie pierwszej wizyty w warsztacie. Po drugie, w przypadku rezerwacji terminu wizyty, dany termin musi być wolny. Należy również przewidzieć obsługę sytuacji, w których pracownik lub pracownicy są na zwolnieniu (nieobecność nieplanowana) lub urlopie (nieobecność planowana). W pierwszym przypadku należy zaznaczyć taki przypadek w systemie oraz wysłać do klienta powiadomienie o anulowaniu wizyty z powodu choroby pracownika i propozycji nowego terminu. Podobny proces postępowania powinien być użyty w przypadku awarii sprzętu niezbędnego do przeprowadzenia naprawy, np. awarii podnośnika. W drugiej sytuacji system powinien zablokować możliwość rezerwacji większej ilości wizyt w danym terminie, niż pozwalają na to zasoby ludzkie.

1.1.2 Mechanik

Po rezerwacji oraz potwierdzeniu terminu następuje etap naprawy. Przed rozpoczęciem naprawy mechanik powinien zweryfikować czy klient umówił się na wizytę poprzez panel klienta, gdzie jego dane oraz dane samochodu automatycznie są przypisane do naprawy, czy też klient zarejestrował wizytę w inny sposób (np. telefonicznie). Mechanik musi podjąć działanie jedynie w drugim przypadku i musi ręcznie utworzyć wpis wizyty oraz przypisać do niej klienta oraz samochód. Zmiana etapów naprawy również jest w gestii mechanika zajmującego się daną naprawą, a jej składowymi są:

1. Naprawa,
    (a) Opcjonalne: oczekiwanie na części,
2. do odbioru,
3. zakończone.

Opisując przebieg naprawy, należy wziąć pod uwagę obsługę reklamacji. Proces naprawy reklamacyjnej jest osobną naprawą, która jednak jest oznaczona jako naprawa reklamacyjna oraz posiada odnośnik do głównej naprawy. Aby można było zidentyfikować mechaników, którzy zajmowali się samochodem przy danej naprawie, musi znaleźć się odpowiednie pole, gdzie można przypisać jednego lub kilku mechaników do naprawy. Do każdej ewidencjonowanej wizyty należy dodać krótki opis elementów, które zostały poddane naprawie bądź wymianie, np. *„wymiana filtru oleju + wymiana klocków hamulcowych + regulacja świateł”*. Istnieje też forma wizyt nieewidencjonowanych, których nie ma potrzeby dodawać do bazy danych. Są to np. wizyty bez rejestracji od których nie pobiera się opłat. Przykładem takiej wizyty jest dopompowanie opon w samochodzie lub sprawdzenie stanu oleju.

1.1.3 Właściciele

W celu rozgraniczenia funkcji dostępnych dla mechaników oraz kierownictwa, wprowadzony jest wyższy poziom uprawnień: *Właściciel*. Jako, że właściciel może być jednocześnie mechanikiem, posiada on uprawnienia i funkcje należące do grupy uprawnień oraz funkcji mechanika. Do

uprawnień właściciela należą funkcje odpowiadające m.in. za wprowadzanie do systemu danych o pracownikach, generowanie raportów i statystyk, nadawanie uprawnień bądź funkcjonalności mechanikom.

1.1.4 Administratorzy

Ostatnią grupą użytkowników bazy danych jest grupa administratorów bazy danych. Podobnie jak w przypadku grupy właścicieli, którzy są rozszerzeniem grupy mechaników, administratorzy również są rozszerzeniem uprawnień właścicieli. Mimo tego faktu, ich głównym zadaniem jest spełnianie zupełnie odmiennej funkcji, a dokładniej są oni odpowiedzialni za zarządzanie bazą danych od strony technicznej. Do ich możliwości należy m.in. robienie kopii zapasowych bazy, dodawanie kont właścicieli, zmiana parametrów bazy danych oraz naprawianie błędów systemu.

1.1.5 Współpraca z systemami zewnętrznymi

Przy stworzeniu odpowiednich warunków, tj. jednolitego systemu wymiany danych, system bazodanowy może w znaczący sposób ułatwić kontakty z systemami zewnętrznymi. Przykładowo, gdyby ZUS umożliwiał komunikację pomiędzy systemami, to znacznie ułatwiłoby lub nawet całkowicie zautomatyzowało to cały proces rozliczania z tą instytucją. Podobnie w przypadku hurtowni części samochodowych - wzajemne wymienianie się informacjami przez poszczególne systemy mogłoby zniwelować możliwość wyczerpania którejś z części w warsztacie samochodowym poprzez automatyczne składanie zamówień.

1.2 Wymagania funkcjonalne i niefunkcjonalne

1.2.1 Wymagania funkcjonalne

Do wymagań funkcjonalnych aplikacji bazodanowej zaliczają się:

1. rejestracja użytkownika systemu,
2. sprawdzenie historii napraw,
3. sprawdzenie i zmiana stanu naprawy,
4. generowanie statystyk,
5. przypomnienia o najbliższych wizytach,
6. przechowywanie danych o pracownikach,
7. użytkownik może przeglądać, dodawać, edytować oraz usuwać dane jedynie w ramach swoich uprawnień,

1.2.2 Wymagania niefunkcjonalne

Do wymagań niefunkcjonalnych aplikacji bazodanowej należą:

1. odporność na utratę danych - poprzez odporność na utratę danych rozumiemy utworzenie mechanizmów odpowiedzialnych za cykliczną archiwizację stanu bazy danych oraz możliwość wczytania stanu bazy z pliku. Administrator będzie posiadał uprawnienia do ustalania harmonogramu tworzenia kopii zapasowych,

2. bezpieczeństwo poufności danych – system powinien dbać o zapewnienie ograniczonego dostępu do przechowywanych informacji. Realizacja tej funkcjonalności polegać będzie na utworzeniu osobnych kont dla każdego z użytkowników systemu. Do każdego konta w sposób indywidualny będą przypisywane uprawnienia do wykonywania poszczególnych operacji oraz dostępu do poszczególnych informacji. Ponadto system powinien być odporny na próby wyłudzenia danych przez nieautoryzowane osoby (np. w formie ataku *SQL Injection*),

3. wydajność – projektując system zakładamy, że maksymalnym czasem odpowiedzi programu na zapytanie użytkownika jest okres 5 sekund. Wyjątkiem są jedynie operacje wykonywane sporadycznie, jak np. generowanie rocznych raportów. Założenia projektowe obejmują wyżej wymienione czasy odpowiedzi na zapytania dla liczby pracowników nieprzekraczającej 15 mechaników oraz liczby klientów do 2000,

4. skalowalność - możliwość późniejszej rozbudowy aplikacji i bazy danych o kolejne funkcje bazująca na warstwowym modelu systemu. Baza danych oraz aplikacja powinny być zbudowane tak, aby możliwe było dołączenie modułów odpowiedzialnych za nowe funkcjonalności takie jak np. współpraca z innymi systemami zewnętrznymi, generowanie innych typów raportów, dostosowywanie się do zmieniających przepisów skarbowych,

5. komunikacja z innymi systemami bazodanowymi – system powinien wspierać automatyczną wymianę danych z systemami zewnętrznymi takimi jak bazy danych hurtowni części, bazy danych firm kurierskich i innych partnerów biznesowych itp.

1.2.3 Słownik

W celu zachowania jednoznaczności określeń, należy kierować się poniższym słownikiem.

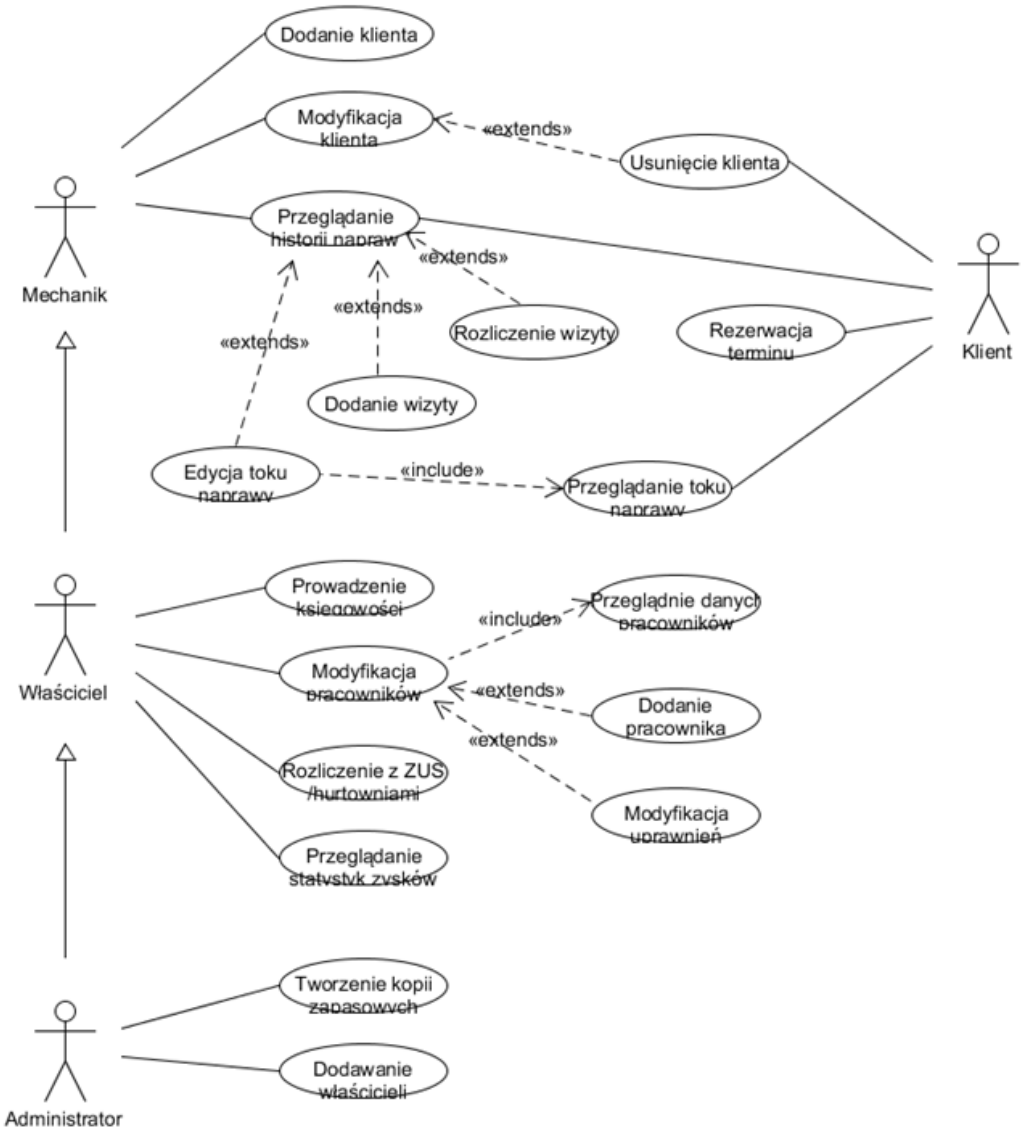
- **UŻYTKOWNIK** - osoba korzystająca z systemu, czyli KLIENT, MECHANIK, WŁAŚCICIEL lub ADMINISTRATOR,
- **KLIENT** - UŻYTKOWNIK, który ma dostęp jedynie do PANELU KLIENTA, a nie do aplikacji, z której korzystają pracownicy warsztatu, KLIENT musi posiadać co najmniej jeden samochód,
- **MECHANIK** - UŻYTKOWNIK, który ma możliwość tworzenia WIZYT, dodawania KLIENTÓW oraz SAMOCHODÓW, edycji TOKU NAPRAWY, przeglądania HISTORII NAPRAW każdego KLIENTA oraz rozliczania wizyty,
- **WŁAŚCICIEL** - UŻYTKOWNIK, który może być jednocześnie MECHANIKIEM, a więc posiada uprawnienia MECHANIKA oraz dodatkowo ma możliwość generacji oraz przegląd-

dania statystyk, dodawania kont MECHANIKÓW, rozliczania ubezpieczenia pracowników z ZUS oraz rozliczania płatności z dystrybutorami części, materiałów eksploatacyjnych itp.

- **ADMINISTRATOR** - UŻYTKOWNIK, który posiada uprawnienia WŁAŚCICIELA, a ponadto może dodawać konta WŁAŚCICIELI oraz ma uprawnienia do zarządzania bazą danych od strony technicznej, czyli przykładowo może utworzyć backup lub harmonogram backupów bazy danych,
- **PANEL KLIENTA** - strona internetowa, która po zalogowaniu się przez klienta umożliwia mu sprawdzenie stanu aktualnej naprawy oraz udostępnia mu następujące opcje: rezerwacja terminu, wyświetlenie historii napraw, ustawienia użytkownika, gdzie m.in. jest możliwość zarządzania przypisanymi do użytkownika samochodami, określenie chęci otrzymywania przyponień o zbliżających się przeglądach okresowych, zmiana hasła,
- **TOK NAPRAWY** - sekwencja następujących po sobie stanów naprawy w kolejności: *rezerwacja → potwierdzenie → w naprawie ↔ opcjonalne: oczekiwanie na części → zakończona*,
- **SAMOCHÓD** - pojazd znajdujący się w bazie danych, który jest przypisany do co najmniej jednego KLIENTA oraz posiada dane, takie jak marka, model, nr rejestracyjny, nr VIN,
- **WIZYTA** - wpis w bazie danych, zawierający odwołanie do klienta i samochodu oraz posiadający krótki opis słowny naprawy,
- **CZĘŚĆ** – element używany przy naprawach, system przewiduję możliwość pełnej ewidencji obrotu części. Pojawienie się części w firmie jest ewidencjonowane dzięki współpracy z systemem hurtowni, natomiast opuszczanie firmy przez części jest rejestrowane przez mechaników i zapisywane podczas rozliczania wizyty,
- **SPRZĘT** – fizyczne urządzenia znajdujące się w firmie i służące do ułatwienia/umożliwienia wykonania naprawy samochodów. System zakłada monitorowanie stanu zajętości sprzętu w celu ułatwienia planowania i umawiania terminów wizyt klientów,
- **UPRAWNIENIA** – informacja o możliwości wykonywania przez użytkownika systemu danych czynności, służą do ochrony poufności danych w systemie, są przydzielane indywidualnie wszystkim użytkownikom systemu.

1.3 Specyfikacja wymagań funkcjonalnych za pomocą diagramu przypadków użycia

1.3.1 Diagram przypadków użycia



Rysunek 1.1: Diagram przypadków użycia.

1.3.2 Opis słowny przypadków użycia

1. Dodanie klienta

   1.1. Cel:
   Utworzenie w bazie danych firmy rekordu odpowiadającego za przetrzymywanie historii transakcji klienta oraz przetrzymywanie jego danych osobowych (są one niezbędne np. w przypadku kontaktu oraz wystawiania rachunków)

   1.2. Warunek początkowy:
   Warunkiem koniecznym jest pojawienie się w warsztacie nowego klienta.

   1.3. Warunek końcowy:
   Kompletne wypełnienie formularza rejestracyjnego.

   1.4. Realizacja:

   a. Mechanik wybiera opcję DODAJ KLIENTA.
   b. Następuje wyświetlenie formularza, który zbiera niezbędne dane osobowe. Wypełnienie formularza skutkuje skokiem do punktu c.
   c. Umieszczenie w bazie danych rekordu z danymi z formularza.

2. Usunięcie klienta:

   2.1. Cel:
   Usunięcie z bazy danych wszystkich danych związanych z danym klientem oprócz danych niezbędnych do prowadzenia księgowości.

   2.2. Warunek początkowy:
   Posiadanie konta w firmie konta przez klienta.

   2.3. Warunek końcowy:
   W bazie nie ma już danych klienta (z wyjątkiem niezbędnych do prowadzenia księgowości).

   2.4. Realizacja:
   Możliwe są dwa scenariusze:

   a. Klient korzysta z funkcji Usuń konto w Panelu klienta. Potwierdza usunięcie danych przyciskiem na stronie.
   b. Na wniosek klienta Mechanik usuwa konto w placówce warsztatu. Jest to jeden z przypadków powiązanych z modyfikacją danych klienta.

3. Modyfikacja klienta:

   3.1. Cel:
   Zmiana danych klienta. Może być związana z modyfikacją/dodaniem/usunięciem danych klienta/samochodu.

   3.2. Warunek początkowy:
   Istnieje w bazie danych konto klienta.

3.3. Warunek końcowy:
Dane w bazie po aktualizacji odpowiadają stanowi rzeczywistemu.

3.4. Realizacja:

a. Mechanik korzysta z zakładki Modyfikacja danych klienta w aplikacji w warsztacie.
b. Wybiera pomiędzy: modyfikacja danych osobowych/modyfikacja posiadanych pojazdów.
c. Wprowadza dane do formularza.
d. Zatwierdza wprowadzenie zmian.
e. Baza tworzy rekord zawierający poprzednie i nowe dane oraz datę modyfikacji w tabeli przetrzymującej logi o zmianach, aby umożliwić ew. korektę pomyłki Mechanika.

4. Przeglądnie historii napraw:

4.1. Cel:
Klient lub mechanik może uzyskać informacje na temat napraw przeprowadzonych w przeszłości.

4.2. Warunek początkowy:
Klient miał już przeprowadzone wcześniej naprawy.

4.3. Warunek końcowy:
Uzyskanie przez Klienta/Mechanika pożądanych informacji.

4.4. Realizacja:
Istnieją dwa scenariusze:

a. Klient wybiera opcję *Przeglądaj historię* w Panelu klienta na stronie internetowej.
b. Mechanik przegląda historię za pomocą opcji Historia klienta w aplikacji w warsztacie. Mechanik ma także możliwość dokonania następujących operacji: dodanie wizyty/rozliczenie wizyty/edycja toku naprawy

5. Edycja toku naprawy:

5.1. Cel:
Umieszczenie w bazie informacji o aktualnym stanie naprawy, tak aby mechanicy oraz klienci mogli uzyskać tę informację.

5.2. Warunek początkowy:
Klient posiada założoną wizytę w firmie.

5.3. Warunek końcowy:
Zaakceptowanie wprowadzonej zmiany przez Mechanika.

5.4. Realizacja:

a. Mechanik przeglądając historię napraw klienta wybiera opcję zmień stan wizyty i wybiera jedną z opcji z wysuwanej listy.
b. Po zaakceptowaniu zmiany przyciskiem nowy stan jest widoczny w aplikacji w warsztacie oraz na stronie www.

6. Dodanie wizyty:

   6.1. Cel:
   Utworzenie w bazie danych rekordu odpowiadającego nowemu zdarzeniu – wizycie klienta.

   6.2. Warunek początkowy:
   Nowe zlecenie – najczęściej pojawienie się klienta w warsztacie, ale może to też być telefoniczne zgłoszenie potrzeby naprawy w terenie.

   6.3. Warunek końcowy:
   Utworzenie nowego rekordu w bazie odpowiadającego nowej wizycie.

   6.4. Realizacja:
   Mechanik wybiera opcję dodaj wizytę przeglądając historię napraw klienta. Od tego momentu jest możliwe m. in. Sprawdzenie stanu wizyty.

7. Rozliczenie wizyty:

   7.1. Cel:
   Wystawienie klientowi rachunku za wykonane usługi/sprzedane części. Zmiana statusu wizyty na zakończona. Dodanie wizyty do modułu rozliczeń księgowych.

   7.2. Warunek początkowy:

   - Istnienie rozpoczętej i niezakończonej naprawy.
   - Zakończenie prac związanych z daną wizytą.

   7.3. Warunek końcowy:
   Zakończenie wszystkich czynności z punktu 7.1.

   7.4. Realizacja:

   a. Mechanik przeglądając historię napraw klienta wybiera opcję Rozlicz wizytę przy aktualnie trwającej wizycie.
   b. Mechanik ma możliwość wydrukowania rachunków.
   c. Status zostaje zmieniony na Zakończona.
   d. Następuje dodanie wizyty do modułu rozliczeń księgowych.

8. Rezerwacja terminu:

   8.1. Cel:
   Umówienie się Klienta na wizytę w określonym terminie.

   8.2. Warunek początkowy:
   Istnienie wolnych terminów.

   8.3. Warunek końcowy:
   Potwierdzenie wizyty przez Mechanika.

   8.4. Realizacja:

   a. Klient poprzez panel klienta na stronie www wybiera termin wizyty.

b. Mechanik otrzymuje powiadomienie o czynnościach klienta, może potwierdzić wizytę lub zaproponować inny termin.

c. Klient dostaje powiadomienie o decyzji Mechanika. Jeżeli jest to potwierdzenie następuje koniec procedury. Jeśli natomiast otrzymuje propozycję innego terminu może ją przyjąć lub odrzucić.

9. Przeglądnie toku naprawy:

   9.1. Cel:
   Sprawdzenie w jakim stanie znajduje się dana naprawa (*rezerwacja* → *potwierdzenie* → *w naprawie* ↔ *opcjonalne: oczekiwanie na części* → *zakończona*)

   9.2. Warunek początkowy:
   Istnienie naprawy.

   9.3. Warunek końcowy:
   Sprawdzenie stanu.

   9.4. Realizacja:
   Umożliwia Klientowi oraz Mechanikowi sprawdzenie na jakim etapie znajdują się poszczególne wizyty. Ponadto zmiana toku naprawy jest obligatoryjnie związana z przeglądaniem toku – aby zmienić trzeba najpierw przeglądnąć.

10. Prowadzenie księgowości:

    10.1. Cel:
    Zautomatyzowanie procesu prowadzenia księgowości w warsztacie, w celu zaoszczędzenia czasu pracowników/właściciela.

    10.2. Warunek początkowy:
    Warunkiem jest prowadzenie przez warsztat działalności.

    10.3. Warunek końcowy:
    Rozliczenie wszystkich niezbędnych działań firmy.

    10.4. Realizacja:
    Właściciel z poziomu aplikacji ma dostęp do funkcji takich jak tworzenie oświadczeń podatkowych, rozliczanie podatku VAT do zapłacenia, automatyczne drukowanie faktur, ewidencjonowanie stanu magazynu w firmie.

11. Modyfikacja pracowników:

    11.1. Cel:
    Zmiana danych pracowników w bazie.

    11.2. Warunek początkowy:
    Aby dokonać zmiany danych trzeba najpierw te dane przeglądnąć.

    11.3. Warunek końcowy:
    Dodanie/usunięcie/modyfikacja wszystkich niezbędnych danych pracowników w bazie.

11.4. Realizacja:
Właściciel wypełniając formularze ma możliwość edycji wszystkich niezbędnych informacji. Ponadto może dodawać nowych pracowników. Modyfikacja pracowników może także wiązać się ze zmianą uprawnień pracownika do korzystania z aplikacji (np. przekazanie wszystkich uprawnień właściciela na czas nieobecności)

12. Przeglądanie danych pracowników:

12.1. Cel:
Uzyskanie informacji o pracownikach zawartych w bazie danych.

12.2. Warunek początkowy:
- Baza danych zawiera pracowników
- Baza danych zawiera właścicieli

12.3. Warunek końcowy:
Zakończenie przeglądania poprzez naciśnięcie przycisku wstecz.

12.4. Realizacja:
Właściciel poprzez aplikację na stacji roboczej w warsztacie ma możliwość przeglądania danych pracowników, które zostały wprowadzone do bazy. Dokonuje tego poprzez kliknięcie w zakładkę Dane pracowników.

13. Dodanie pracownika:

13.1. Cel:
Dodanie danych pracownika do bazy.

13.2. Warunek początkowy:
- Zatrudnienie nowego pracownika
- Wdrażanie systemu do warsztatu (wprowadzenie wszystkich aktualnych pracowników)

13.3. Warunek końcowy:
Wprowadzenie danych.

13.4. Realizacja:
Właściciel poprzez aplikację na stacji roboczej w warsztacie ma możliwość dodania nowych pracowników. Dokonuje tego poprzez kliknięcie w zakładkę Dane pracowników a następnie Dodaj pracownika. Po wypełnieniu formularza i zaakceptowaniu zmian baza zostaje powiększona o nowy rekord w tabeli pracowników oraz tabeli logów przetrzymującej wszystkie zmiany.

14. Rozliczenie z ZUS/hurtowniami:

14.1. Cel:
Automatyzacja procesu rozliczeń z zewnętrznymi systemami – ZUS oraz hurtowniami części samochodowych.

14.2. Warunek początkowy:
Posiadanie pełniej historii działalności warsztatu w elektronicznej bazie danych.

14.3. Warunek końcowy:
Wygenerowanie dokumentów przez właściciela.

14.4. Realizacja:
Właściciel poprzez aplikację na stacji roboczej w warsztacie ma możliwość generowani dokumentów niezbędnych do rozliczania się z partnerami handlowymi oraz Zakładem Ubezpieczeń Społecznych w zakładce Systemy zewnętrzne.

15. Przeglądanie statystyk zysków:

15.1. Cel:
Uzyskanie informacji o tym które zlecenia są najbardziej/najmniej zyskowne.

15.2. Warunek początkowy:
Baza danych zawiera dane o wizytach

15.3. Warunek końcowy:
Zakończenie przeglądania poprzez naciśnięcie przycisku wstecz.

15.4. Realizacja:
Właściciel poprzez aplikację na stacji roboczej w warsztacie ma możliwość przeglądania statystyk generowanych na podstawie zleceń, które zostały wprowadzone do bazy. Dokonuje tego poprzez kliknięcie w zakładkę Statystyki.

16. Tworzenie kopii zapasowych:

16.1. Cel:
Archiwizacja informacji zawartych w bazie danych na wypadek nieprzewidzianych zdarzeń prowadzących do ich utraty.

16.2. Warunek początkowy:
Baza danych zawiera jakiekolwiek dane

16.3. Warunek końcowy:
Zakończenie archiwizacji poprzez utworzenie pliku z którego możliwe jest przywrócenie stanu z chwili obecnej.

16.4. Realizacja:
Administrator w zakładce Kopia zapasowa, wybiera opcję utwórz kopię/harmonogram tworzenia kopii. Następnie po ustawieniu opcji zatwierdza wykonanie operacji. Możliwe jest też wczytanie stanu bazy z pliku.

17. Dodawanie właścicieli:

17.1. Cel:
Dodanie do bazy danych pracowników z domyślnymi uprawnieniami Właściciela.

17.2. Warunek początkowy:
Jest utworzone konto administratora.

17.3. Warunek końcowy:
Wypełnienie formularza i zaakceptowanie operacji.

17.4. Realizacja:
Administrator poprzez aplikację na stacji roboczej w warsztacie ma możliwość dodawania do bazy kont Właścicieli. Dokonuje tego poprzez kliknięcie w zakładkę Dane Właścicieli.

18. Modyfikacja uprawnień:

18.1. Cel:
Przekazanie Mechanikowi części lub wszystkich uprawnień Właściciela.

18.2. Warunek początkowy:
Baza posiada przynajmniej jednego Mechanika i Właściciela.

18.3. Warunek końcowy:
Wypełnienie formularza i zaakceptowanie operacji.

18.4. Realizacja:
Właściciel poprzez aplikację na stacji roboczej w warsztacie ma możliwość zmiany uprawnień poszczególnych kont Mechaników. Dokonuje tego poprzez kliknięcie w zakładkę Dane Mechaników/Uprawnienia.

1.4 Identyfikacja diagramu związków encji na podstawie analizy scenariuszy poszczególnych przypadków użycia

1.4.1 Określenie tabel

Użytkownik (id_uzytkownika, imię, nazwisko, PESEL, NIP, login, hasło, data_zatrudnienia, wynagrodzenie, data_urodzenia, adres, uprawnienia, nr_telefonu, stanowisko, dział, superuser, ostatnie_logowanie, teraz_aktywny)

Na podstawie analizy liczby instancji tabel przechowujących dane mechaników, właścicieli, administratora oraz klientów podjęliśmy decyzję o sklejeniu tych tabel pomimo powstania pustych komórek w niektórych rekordach (np. komórka stanowisko dla klienta albo komórka NIP dla mechanika). Liczba mechaników i właścicieli nie powinna przekroczyć 15, natomiast konto administratora jest tylko jedno. Zabieg scalenia powinien znacznie ułatwić projektowanie bazy oraz przyspieszyć jej działanie.

Ze względu na uproszczenie struktury oraz zwiększenie wydajności bazy zdecydowaliśmy się także na denormalizację – np. adres składający się z wielu elementów(miasto, kod pocztowy, ulica, nr domu itd.) został sklejony do jednej komórki.

Tabela **Użytkownik** przechowuje niezbędne dane osobowe użytkowników, a także datę ostatniego logowania, flagę aktualnej aktywności w systemie, informację czy użytkowik ma nieograniczone uprawnienia.

Samochód (marka, model, nr rejestracyjny, nr VIN, przebieg, id_uzytkownika)

Tabela **Samochód** przechowuje dane samochodu oraz informację o aktualnym przebiegu oraz właścicielu.

Wizyta (id_wizyty, data, status, klient, samochód, przebieg_w_momencie_wizyty, opis, części, cena, czas_pracowników, id_uzytkownika)

Tabela **Wizyta** przechowuje informacje na temat daty wizyty, jej statusu, kliencie, samochodzie, przebiegu auta w momencie wizyty – w celu przewidzenia kolejnych przeglądów i wysłania powiadomienia kierowcy, ewidencjonuje wymienione części, należności za usługę oraz informację o mechnikach wykonujących pracę.

Sprzęt (nazwa_sprzętu, status, opis, id_wizyty, data_przeglądu)

Tabela **Sprzęt** pełni funkcję informacji o zajętości urządzeń na warsztacie. Może także pomóc przy generowaniu przypomnień o cyklicznych przeglądach i pracach konserwacyjnych dotyczących maszyn.

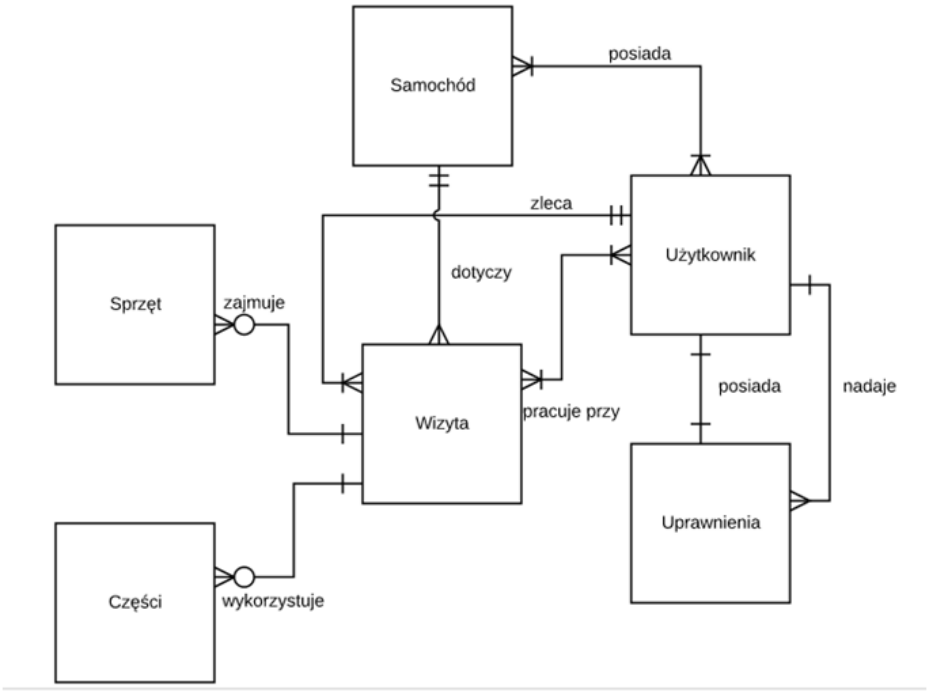
Części (id, nazwa, cena_w_hurtowni)

Tabela **Części** jest niezbędna do generowania statystyk zysków oraz ewidencjonowania przepływu części przez firmę.

Uprawnienia (id_uzytkownika, uprawniwnie_1, uprawniwnie_2, uprawniwnie_3, ... uprawniwnie_n)

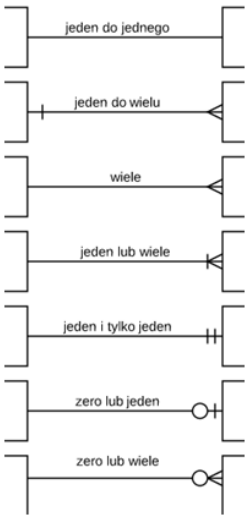
Tabela **Uprawnienia** przechowuje informacje o tym jakie operacje są dozwolone dla poszczególnych użytkowników systemu.

1.4.2 Diagram związków encji



Rysunek 1.2: Diagram związków encji.

Legenda:



Rysunek 1.3: Legenda do diagramu związków encji

Rozdział 2

Projekt bazy danych

2.1 Analiza liczby instancji każdej encji

W ramach analizy liczby instancji każdej encji należy oszacować maksymalną liczbę krotek (danych) czyli maksymalny rozmiar tabeli, co pozwala oszacować m.in. rozmiar pamięci dyskowej. Analiza ta zakłada, że dane zbierane są co najwyżej przez rok, a następnie część z nich, która ma znaczenie głównie archiwalne (jak w przypadku napraw, których jest w bazie danych najwięcej) jest archiwizowana. Archiwizacja ta nie dotyczy m. in. pracowników, uprawnień, sprzętu. Wyniki analizy liczby instancji przedstawione są w tabeli 2.1.

Tablica 2.1: Analiza liczby instancji każdej encji.

Tabela	**Maksymalna liczba wierszy**	**Średnia liczba wierszy**	**Rozmiar kolumn**	**Maksymalny rozmiar tabeli**	**Średni rozmiar tabeli**
Użytkownicy	2 000	1 000	18	36 000	18 000
Samochód	2 000	1 000	6	12 000	6 000
Wizyta	15 000	11 200	11	165 000	123 200
Sprzęt	10	4	5	50	20
Części	2 000	1 500	3	6 000	4 500
Uprawnienia	15	7	20	300	140
Terminy	3 400	2 700	2	6 800	5 400

2.2 Analiza użycia identyfikująca podstawowe rodzaje transakcji

Analiza użycia ma na celu identyfikację podstawowych rodzajów transakcji, czyli dodawania, usuwania, modyfikacji i wyszukiwania oraz określenie na tej podstawie zmienności zawartości poszczególnych tabel. Oznacza to, że dla każdej tabeli należy określić jaki rodzaj transakcji będzie przeprowadzany najczęściej. Oczywiście nie wyklucza to pozostałych operacji na tabelach. Przykładowo, najczęstszą operacją na tabeli Użytkownicy będzie operacja wyszukiwania, natomiast oczywistym jest, że aby wyszukiwać użytkowników, najpierw trzeba ich dodać. Zatem na

wszystkich tabelach można przeprowadzać wszystkie podstawowe transakcje, natomiast transakcje wymienione w tabeli 2.2 są operacjami wykonywanymi najczęściej, swego rodzaju operacjami podstawowymi.

Tablica 2.2: Analiza użycia identyfikująca podstawowe rodzaje transakcji.

Tabela	**Rodzaj transakcji**
Użytkownicy	wyszukiwanie, wstawianie, modyfikacja, wyszukiwanie
Samochód	wstawianie, modyfikacja, wyszukiwanie
Wizyta	wstawianie, modyfikacja, wyszukiwanie
Sprzęt	wyszukiwanie
Części	modyfikacja, wyszukiwanie
Uprawnienia	wyszukiwanie
Terminy	dodawanie, usuwanie, modyfikacja, wyszukiwanie

2.3 Sformułowanie wymagań dotyczących dostępu – określenie częstości wykonania operacji na danych

Opisy przebiegów operacji wymienionych w tabeli 2.3:
Ad 1. Wygenerowanie rocznego raportu przychodów i rozchodów wymaga dostępu do następujących tabel: *Użytkownik* (uzyskanie danych klientów oraz pracowników warsztatu), *Wizyta* (uzyskanie danych o wszystkich wizytach w roku), *Części* (uzyskanie danych o cenach części użytych w poszczególnych naprawach).
Ad 2. Operacja generująca roczny raport podatku dochodowego wymaga dostępu do następujących tabel: *Użytkownik* (Uzyskanie danych pracowników warsztatu oraz właściciela), *Wizyta* (uzyskanie danych o wszystkich wizytach w roku za które została pobrana opłata), *Części* (uzyskanie danych o cenach części użytych w poszczególnych naprawach).
Ad 3. Wygenerowanie kwartalnego raportu rozliczenia z podatku od wartości dodanej (VAT) wymaga dostępu do tabel: *Wizyta* (uzyskanie danych o wszystkich wizytach w roku za które została pobrana opłata), *Części* (uzyskanie danych o cenach części użytych w poszczególnych naprawach), *Sprzęt* (informacje na temat wydatków na wyposażenie firmy).
Ad 4. Miesięczny raport płac dla pracowników uwzględniający premie za brak reklamacji oraz nadgodziny to operacja korzystająca z tabel: *Użytkownik* (uzyskanie danych pracowników warsztatu oraz właściciela), *Wizyta* (uzyskanie danych o reklamacjach).
Ad 5. Operacja generowania miesięcznego raportu przychodów i rozchodów wymaga operacji na tabelach: *Użytkownik* (uzyskanie danych klientów oraz pracowników warsztatu), *Wizyta* (uzyskanie danych o wszystkich wizytach w roku), *Części* (uzyskanie danych o cenach części użytych w poszczególnych naprawach).
Ad 6. Operacja generowania dziennego raportu przychodów i rozchodów wymaga operacji na tabelach: *Użytkownik* (uzyskanie danych klientów oraz pracowników warsztatu), *Wizyta* (uzyskanie danych o wszystkich wizytach w roku), *Części* (uzyskanie danych o cenach części użytych w poszczególnych naprawach).

Tablica 2.3: Określenie częstości wykonywania poszczególnych operacji na danych.

Lp.	**Operacja**	**Częstotliwość wykonywania**	**Maksymalny czas wykonania zapytania**
1	Roczny raport przychodów i rozchodów	Raz na rok	30 min
2	Roczny raport podatku dochodowego	Raz na rok	30 min
3	Kwartalny raport rozliczenia z podatku od wartości dodanej (VAT)	Raz na kwartał	10 min
4	Miesięczny raport płac dla pracowników uwzględniający premie za brak reklamacji oraz nadgodziny	Raz na miesiąc	2 min
5	Miesięczny raport przychodów i rozchodów	Raz na miesiąc	2 min
6	Dzienny raport przychodów i rozchodów	Raz na dzień	1 min
7	Raport historii klienta – na żądanie	Do 500 razy dziennie	3 s
8	Sprawdzenie statusu naprawy	Do 500 razy dziennie	3 s
9	Sprawdzenie wolnych terminów	Do 500 razy dziennie	3 s
10	Dodanie klienta	Do 200 razy dziennie	3 s
11	Zmiana statusu naprawy	Do 500 razy dziennie	3 s
12	Generowanie przypomnień sms	Raz na dzień	5 min
13	Tworzenie kopii zapasowej bazy danych	Raz na tydzień	8 h
14	Wczytywanie kopii bazy z pliku	Raz w roku	8 h

Ad 7. Raport historii klienta to operacja wywoływana na żądanie klienta lub mechanika, polega ona na uzyskaniu wszystkich zapamiętanych wizyt danego użytkownika – potrzebne są tabele: *Użytkownik*, *Wizyta*, *Samochód* oraz *Uprawnienia*.
Ad 8. Sprawdzenie statusu naprawy jest operacją wykorzystującą dane zawarte w tabeli: *Wizyta* (przeglądnie wszystkich wizyt w celu odszukania bieżących wizyt danego klienta), *Użytkownik* (dostęp do wizyt konkretnego klienta).
Ad 9. Sprawdzenie wolnych terminów wymaga dostępu do tabeli *Wizyta* – w niej znajdują się informacje o terminach umówionych wizyt.
Ad 10. Dodanie klienta to operacja dodająca rekord do tabeli *Użytkownik*, najczęściej jednak wiąże się ona z dodaniem nowego samochodu do tabeli Samochody oraz zdefiniowaniem uprawnień nowego użytkownika znajdujących się w tabeli *Uprawnienia*.
Ad 11. Zmiana statusu naprawy przebiega identycznie jak sprawdzenie statusu naprawy - jest operacją wykorzystującą dane zawarte w tabeli: *Wizyta* (przeglądnie wszystkich wizyt w celu odszukania bieżących wizyt danego klienta), *Użytkownik* (dostęp do wizyt konkretnego klienta). Wymaga jednak dodatkowo dostępu do tabeli *Uprawnienia* w celu zweryfikowania, czy dany użytkownik może zmienić zawartość bazy danych.
Ad 12. Generowanie przypomnień sms to operacja wykonywana codziennie. Polega ona na przeszukaniu tabeli *Samochód* w celu odnalezienia pojazdów wymagających dokonania przeglądu lub

wymiany opon (sprawdzany jest przebieg samochodu oraz data ostatniej wizyty), aby wysłać wiadomość sms konieczne jest także uzyskanie danych oraz numeru telefonu klienta z tabeli *Użytkownik*.
Ad 13. Tworzenie kopii zapasowej bazy danych jest wykonywanie cyklicznie w odstępie siedmiu dni. Jest to zabieg mający na celu zapewnienie bezpieczeństwa danych. Tworzenie kopii zapasowej bazy danych jest za każdym razem wykonywane w nocy – jest to czas, w którym nigdy nie dochodzi do modyfikacji zawartości bazy. Operacja wymaga dostępu do wszystkich tabel w bazie.
Ad 14. Wczytywanie kopii bazy z pliku podobnie jak tworzenie kopii wymaga dostępu do każdej tabeli w bazie. Na wykonanie operacji jest cała noc, ponieważ firma jest wtedy nieczynna.

2.4 Analiza integralności

System bazodanowy powinien zawierać odpowiednie mechanizmy zabezpieczające przed skutkami przypadkowych błędów logicznych, konfliktów we współbieżnym dostępie do danych oraz skutkami awarii oprogramowania i sprzętu komputerowego.Ponadto system powinien zawsze dostarczać wiarygodne dane i być zabezpieczony przed nieautoryzowaną modyfikacją informacji. System baz danych powinien też zapewniać możliwość sprawdzania i ewentualnej korekty wprowadzanych danych oraz powinny zawierać odpowiednie mechanizmy zapewniające prawidłowe przetwarzanie danych. Konieczne jest także zapewnienie kompletności, poprawności i wiarygodności danych zgromadzonych w bazie.

Proces ochrony integralności obejmuje:

- kontrolę danych wejściowych oraz synchronizację dostępu do danych,
- poprawianie czyli korektę danych, cofanie i odtwarzanie stanu bazy,
- archiwizacje poprzez tworzenie kopii bazy oraz zapisów działania systemu,
- testowanie czyli sprawdzanie poprawności zawartości bazy.

Integralność dotyczy poprawnie zaprojektowanego schematu bazy danych oraz spełnienia ograniczeń nałożonych na wartości atrybutów opisujących obiekty w bazie.

Zakładamy trzy typy więzów integralności wewnętrznej:

- klucza głównego (np. Id_uzytkownika w tabeli Uzytkownik nie może być równy null),
- klucza obcego (np. Id_uzytkownika w tabeli Samochod nie może być równy null),
- dziedziny wartości np. atrybut stan w tabeli Wizyta może przybierać tylko następujące wartości: *rezerwacja* $\rightarrow$ *potwierdzenie* $\rightarrow$ *w naprawie* $\leftrightarrow$ *opcjonalne: oczekiwanie na części* $\rightarrow$ *zakończona.*

Zakładamy także następujące typy więzów integralności dodatkowej:

- więzy przejścia: np. jeśli klient nie korzystał z usług warsztatu przez ponad 3 lata jest usuwany z bazy,

- więzy statyczne: np. liczba rezerwacji danego terminu nie może przekroczyć ilości stanowisk roboczych w warsztacie,
- więzy atomowości transakcji: np. przy rezerwacji terminu wizyty nie można dopuścić do spowodowania anomalii wynikających z opóźnienia w komunikacji aplikacja-baza danych oraz anomalii wynikających z próby rejestracji w tym samym terminie przez więcej niż jedną osobę,
- więzy ochrony danych: np. z poziomu użytkownika aplikacji niemożliwe jest usunięcie danych o zakończonych wizytach.

Implementacja więzów integralności jest szerzej omówiona w punkcie 2.5.5.

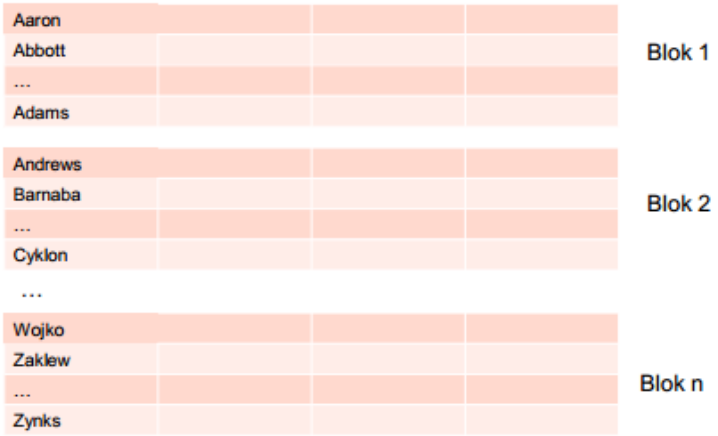
2.5 Dostrajanie bazy danych pod względem wydajności

2.5.1 Tworzenie mechanizmów dostępu związanych z przechowywaniem

Dzięki wynikom analizy liczby instancji oraz użycia, można zastosować mechanizmy poprawiające wydajność bazy danych. Zalicza się do nich sekwencyjność, haszowanie oraz klastry.

Sekwencyjność

Sekwencyjność (*ISAM, ang. Indexed Sequential Access Method*) można zastosować jedynie w przypadku plików uporządkowanych rekordów. Oznacza to, że rekordy pliku są porządkowane według wartości jednego pola (pole uporządkowania), co pokazuje rysunek 2.1.



Rysunek 2.1: Przykład pliku uporządkowanych rekordów.

Charakterystyka operacji na plikach sekwencyjnych:

- odczyt rekordów: wydajny, bez sortowania,
- wyszukiwanie: może być użyty mechanizm wyszukiwania binarnego, efektywny, koszt $log_2 b$, gdzie b – liczba bloków wymaganych dla pliku zawierającego r rekordów,
- wstawianie: kosztowne, gdyż wymaga fizycznego uporządkowania rekordów, należy wyszukać pozycję i przesuwać średnio połowę rekordów (odczytanie i zapisanie),

- usuwanie: kosztowny, tak jak wstawianie, można nadawać znaczniki usunięcia i dokonywać reorganizacji,
- wyszukiwanie wg innego klucza niż klucz uporządkowania: wyszukiwanie liniowe,
- modyfikacja: koszt zależy od warunku wyszukiwania (gdy uwzględnia pole porządkujące to binarnie) i modyfikowanego pola (porządkujące czy nie), modyfikacja pola porządkującego jest tak samo kosztowna jak wstawienie rekordu.

Na podstawie analizy charakterystyki stwierdzono, że sekwencyjność powinna zostać wykorzystana w przypadku tabel *Użytkownicy* oraz *Sprzęt*.

Klastry

Zapis kilku tabel w klaster sprawia, że ich złączenie staje się szybsze, ale za to operacje na pojedynczych tabelach stają się nieco wolniejsze niż bez klastra. Częstym złączeniem będzie wyszukanie klienta oraz samochodu w celu przypisania go do naprawy. Z tego powodu tabele *Użytkownicy* oraz *Samochód* powinny znaleźć się w jednym klastrze. Dodatkowo, w jednym klastrze powinny się znajdować tabele *Sprzęt* oraz *Terminy*, ponieważ częstym złączeniem będzie wyszukanie sprzętu, który jest wolny w danym terminie.

2.5.2 Dodawanie indeksów

Zastanawiając się nad dodawaniem do bazy indeksów koniecznie trzeba mieć na uwadze kompromis między czasem wykonywania zapytań oraz czasem modyfikowania, wstawiania i usuwania danych. Próbując określić miejsca, w których warto zastosować indeksy oraz typując miejsca, w których zastosowanie indeksów wpłynęłoby negatywnie na wydajność aplikacji bazodanowej opieraliśmy się na tabelach 2.2 oraz 2.3. Ponieważ indeks jest strukturą danych zwiększającą szybkość wykonywania operacji wyszukiwania na tabeli należało przede wszystkim zlokalizować pola po których najczęściej dokonywane jest wyszukiwanie:

- Użytkownik - id_uzytkownika, imie, nazwisko,
- Samochód - nr rejestracyjny, przebieg, id_uzytkownika,
- Wizyta - id_wizyty, data, status, id_uzytkownika,
- Sprzęt - nazwa_sprzetu, id_wizyty,
- Części – id_czesci,
- Uprawnienia - id_uzytkownika.

Należy jednak mieć na uwadze fakt, że użycie indeksów wpływa negatywnie na czas modyfikowania, wstawiania i usuwania danych. Ponieważ wszystkie tabele, z wyjątkiem tabeli Wizyta, są modyfikowane rzadko nie musieliśmy ograniczać się w nich przy dodawaniu indeksów. W przypadku tabeli Wizyta ograniczyliśmy indeksy do pól: id_wizyty, data, id_uzytkownika.

2.5.3 Denormalizacja

Denormalizacja bazy polega na wprowadzenie kontrolowanej nadmierności do bazy danych w celu przyśpieszenia wykonywania na niej operacji (np. obsługiwania zapytań). Dzięki denormalizacji bazy unika się kosztownych operacji połączeń tabel.

Stosowanie denormalizacji bazy danych jest stosowane, gdy inne metody dostrajania bazy danych pod względem wydajności zawiodą. Dzieje się tak, ponieważ denormalizacja zawsze wiąże się z redundancją danych.

Wstępna denormalizacja bazy została przeprowadzona już na etapie tworzenia modelu konceptualnego bazy danych. Dokonując identyfikacji diagramu związków encji na podstawie analizy scenariuszy poszczególnych przypadków użycia określaliśmy tabele, z których będzie zbudowana baza danych. Wstępnie mieliśmy bardzo wiele koncepcji podziału bazy danych na tabele.

1. Pierwotnie chcąc utrzymać bazę danych w 1PN Adres miał być oddzielną tabelą zawierającą pola takie jak: ulica, nr domu, nr lokalu, miasto, kod pocztowy, kraj. Zdecydowaliśmy się jednak na sklejenie tabel Adres oraz Użytkownik.

2. Na początku planowaliśmy przetrzymywać stawki pracowników w oddzielnej tabeli i uzależnić je od działu mechanika:

Tablica 2.4: Tabela „Pracownicy" po normalizacji.

Imię	**Nazwisko**	**Dział**
Jan	Kowalski	Wulkanizacja
Piotr	Nowak	Wulkanizacja
Paweł	Kowalski	Elektronika samochodowa

Tablica 2.5: Tabela „Wynagrodzenie" po normalizacji.

Dział	**Wynagrodzenie**
Wulkanizacja	2000 zł
Elektronika samochodowa	3000 zł

Jednak finalnie tabele te zostały sklejone w następujący sposób:

Tablica 2.6: Tabela „Pracownicy" po denormalizacji.

Imię	**Nazwisko**	**Dział**	**Wynagrodzenie**
Jan	Kowalski	Wulkanizacja	2000zł
Piotr	Nowak	Wulkanizacja	2000zł
Paweł	Kowalski	Elektronika samochodowa	3000zł

Analogiczne podejście w innych przypadkach pozwoliło nam znacząco uprościć strukturę bazy danych, przez co projektowanie jej stało się łatwiejsze.

3. Początkowo planowaliśmy utworzenie oddzielnej tabeli dla pracowników warsztatu oraz klientów. Jednak ze względu na niewielką liczbę pracowników zdecydowaliśmy się na utworzenie jednej tabeli o nazwie Użytkownicy, decyzja ta będzie wiązała się z uzupełnieniem niektórych pól w tej tabeli wartościami *null*.

4. Ponieważ bardzo często (wiele operacji wykonywanych kilkaset razy dziennie) sprawdzane są uprawnienia użytkownika zastanawiamy się nad sklejeniem tabeli uprawnień z tabelą użytkowników. Decyzję tę pozostawiliśmy sobie na później ze względu na zweryfikowanie skali problemu przy przeprowadzaniu wstępnych testów na gotowej bazie przy pomocy *SZBD*.

2.5.4 Wykorzystanie możliwości wybranego systemu zarządzania bazą danych

W kwestii wydajności bazy danych, znaczący jest wybór Systemu Zarządzania Bazą Danych (SZBD). SZBD różnią się m.in. typem optymalizacji, stopniem wielowątkowości, rodzajem zapytań (interpretowane bądź skompilowane). Zapytania interpretowane podczas wykonywania zapytania są wolniejsze - jednak poprawia się ich wydajność, gdy baza jest zmienna lub dostęp do danych ma charakter losowy. Wybrany przez nas SZBD, czyli MySQL, posiada liczne narzędzia służące zwiększeniu wydajności bazy danych. Zdecydowaliśmy się skorzystać z mechanizmu *InnoDB*, który jest nowszą wersją mechanizmu *MyISAM* i zastąpił go jako domyślny mechanizm w *MySQL v5.5*. Zastosowanie *InnoDB* umożliwia korzystanie z takich funkcji bazodanowych jak transakcje i klucze obce. Jest też zgodny ze standardem *ACID*. W silniku tym dostępne są dwa sposoby magazynowania danych: plik lub grupa plików wspólne dla wszystkich baz i tabel, lub też po jednym pliku z danymi dla każdej tabeli z osobna. Inne ważne cechy InnoDB to: blokady na poziomie wierszy, możliwość kompresji danych, oraz *MVCC*.

2.5.5 Obsługa więzów integralności

Więzy integralności to system reguł gwarantujących, że relacje między wierszami w tabelach pokrewnych są prawidłowe, a użytkownik nie może przypadkowo usunąć lub zmienić danych pokrewnych.

Zakładamy obsługę następujących więzów integralności:

1. więzy integralności encji (ze względu na klucz główny),

2. więzy integralności referencyjnej - klucze obce nie mogą być równe *null*,

3. więzy integralności dziedziny:

Tablica 2.7: Tabela Użytkownik.

Pole	**Ograniczenie**
id_uzytkownika	Różne od null
imie	Różne od null
nazwisko	Różne od null
PESEL	11 cyfrowa liczba dodatnia
NIP	10 cyfrowa liczba dodatnia lub null
login	Od 5 do 10 znaków (cyfr i liczb)
hasło	Musi zawierać liczby i cyfry, minimum 5 znaków
data_zatrudnienia	-
wynagrodzenie	Liczba dodatnia lub null (dla klienta)
data_urodzenia	-
adres	-
uprawnienia	Różne od null
nr_telefonu	Dodatnia liczba 9 cyfrowa lub null
stanowisko	{mechanik, właściciel, klient, admin}
dział	{wulkanizacja, elektronika, blacharstwo, mechanika, null}
superuser	{true, false}
ostatnie_logowanie	Format daty lub null
teraz_aktywny	{true, false}

Tablica 2.8: Tabela: Samochód.

Pole	**Ograniczenie**
marka	Różne od null
model	Różne od null
nr_rejestracyjny	Różne od null
nr_VIN	Różne od null
przebieg	Liczba dodatnia
id_uzytkownika	Różne od null

Tablica 2.9: Tabela: Wizyta.

Pole	**Ograniczenie**
id_wizyty	Różne od null
data	Format daty
status	{rezerwacja, potwierdzenie, w naprawie, oczekiwanie na części, zakonczona}
id_klienta	Różne od null
nr_rejestracyjny	Różne od null
przebieg_w_momencie_wizyty	Liczba dodatnia
opis	-
id_czesci	Różne od null
cena	Liczba dodatnia lub zero
czas_pracowników	Wartość z przedziału $< 0, 1000 >$
id_uzytkownika	Różne od null

Tablica 2.10: Tabela: Sprzet.

Pole	**Ograniczenie**
nazwa_sprzetu	Różne od null
status	{zajęty, wolny}
opis	-
id_wizyty	-
data_przegladu	Różne od null

Tablica 2.11: Tabela: Części.

Pole	**Ograniczenie**
id_czesci	Różne od null
nazwa	Różne od null
producent	Różne od null
cena_w_hurtowni	-

Tablica 2.12: Tabela: Uprawnienia.

Pole	**Ograniczenie**
id_uzytkownika	Różne od null
uprawnienie_n	true, false

4. więzy atomowości transakcji:
   Przy rezerwacji terminu wizyty nie można dopuścić do spowodowania anomalii wynikających z opóźnienia w komunikacji aplikacja-baza danych oraz z próby rejestracji w tym samym terminie przez więcej niż jedną osobę. W tym celu po wysłaniu przez aplikację klienta zapyta-

nia o zarezerwowanie terminu, system powinien zablokować innym użytkownikom możliwość wysłania podobnego zapytania. Ma to na celu zapobiegnięcie sytuacji, kiedy dochodzi do niespójności wewnątrz bazy spowodowanej tym, że transakcja została zainicjowana przez jednego użytkownika, natomiast dokończona przez innego. Możliwość wysyłania zapytań o rezerwację powinna zostać odblokowana dopiero po zakończeniu rozpoczętej operacji.

5. więzy ochrony danych:
   Z poziomu użytkownika aplikacji niemożliwe jest usunięcie i modyfikowanie danych:

   - usuwanie i modyfikacja zakończonych wizyt,
   - usuwanie rekordów w tabeli *Użytkownik*,
   - usuwanie rozpoczętych wizyt,
   - usuwanie części, które zostały użyte w wizycie,
   - usuwanie rekordów z tabeli *Samochody*,
   - modyfikacja pól takich jak np. *VIN* w tabeli *Samochody*.

Rozdział 3

Implementacja i testy bazy danych w wybranym systemie zarządzania bazą danych

3.1 Fizyczny projekt bazy danych

Zgodnie z wcześniejszymi założeniami, baza danych została zaimplementowana w środowisku *MySQL* (wersja 5.6) przy pomocy frameworku *Django* (wersja 1.7). Podczas implementacji bazy danych napotkaliśmy na sytuacje wymuszające na nas modyfikację struktury bazy względem wcześniejszych założeń. Głównym powodem owych zmian jest użycie frameworku *Django*, który automatycznie tworzy tabelę użytkowników, uprawnień, grup itp. Ponadto, już na etapie tworzenia modeli, określane są pola mające być indeksowane, co powoduje automatyczne dodanie indeksów do wskazanych pól przy generacji bazy na podstawie takich modeli.
Gotowy skrypt w języku **SQL**, który został wygenerowany przez *Django* na podstawie określenia pól modeli, przedstawiony jest na listingu 3.1.

Listing 3.1: Skrypt tworzący bazę danych.

```
BEGIN;
CREATE TABLE `warsztat_app_uzytkownik` (
    `id` integer AUTO_INCREMENT NOT NULL PRIMARY KEY,
    `user_id` integer NOT NULL UNIQUE,
    `PESEL` varchar(11) NOT NULL,
    `NIP` varchar(10),
    `data_zatrudnienia` date NOT NULL,
    `wynagrodzenie` integer UNSIGNED,
    `data_urodzenia` date NOT NULL,
    `adres` varchar(50) NOT NULL,
    `nr_telefonu` varchar(9),
    `stanowisko` varchar(4) NOT NULL,
    `dzial` varchar(4) NOT NULL
);
```

```
ALTER TABLE `warsztat_app_uzytkownik`
ADD CONSTRAINT `user_id_refs_id_aba8e5a3`
FOREIGN KEY (`user_id`) REFERENCES `auth_user` (`id`);
CREATE TABLE `warsztat_app_samochod` (
    `marka` varchar(30) NOT NULL,
    `model` varchar(30) NOT NULL,
    `nr_rejestracyjny` varchar(10) NOT NULL PRIMARY KEY,
    `nr_VIN` varchar(17) NOT NULL UNIQUE,
    `przebieg` integer UNSIGNED NOT NULL,
    `id_uzytkownika` integer NOT NULL
);
ALTER TABLE `warsztat_app_samochod`
ADD CONSTRAINT `id_uzytkownika_refs_id_6f6613aa`
FOREIGN KEY (`id_uzytkownika`) REFERENCES `auth_user` (`id`);
CREATE TABLE `warsztat_app_czesci` (
    `id_czesci` integer NOT NULL PRIMARY KEY,
    `nazwa` varchar(50) NOT NULL,
    `producent` varchar(50) NOT NULL,
    `cena_w_hurtowni` numeric(7, 2) NOT NULL
);
CREATE TABLE `warsztat_app_sprzet` (
    `id_sprzetu` integer NOT NULL PRIMARY KEY,
    `nazwa_sprzetu` varchar(20) NOT NULL,
    `status` varchar(3) NOT NULL,
    `opis` varchar(200) NOT NULL,
    `data_przegladu` date NOT NULL
);
CREATE TABLE `warsztat_app_wizyta` (
    `id_wizyty` integer NOT NULL PRIMARY KEY,
    `data` date NOT NULL,
    `status` varchar(6) NOT NULL,
    `id_uzytkownika` integer NOT NULL,
    `nr_rejestracyjny` varchar(10) NOT NULL,
    `przebieg_w_momencie_wizyty` integer UNSIGNED NOT NULL,
    `opis` varchar(200) NOT NULL,
    `id_czesci` integer NOT NULL,
    `cena` integer UNSIGNED NOT NULL,
    `czas_pracownikow` integer UNSIGNED NOT NULL,
    `id_sprzetu` integer NOT NULL
);
```

```
ALTER TABLE `warsztat_app_wizyta`
ADD CONSTRAINT `id_czesci_refs_id_czesci_d79b9e97`
FOREIGN KEY (`id_czesci`) REFERENCES `warsztat_app_czesci` (`id_czesci`);

ALTER TABLE `warsztat_app_wizyta`
ADD CONSTRAINT `id_sprzetu_refs_id_sprzetu_40478433`
FOREIGN KEY (`id_sprzetu`) REFERENCES `warsztat_app_sprzet` (`id_sprzetu`);

ALTER TABLE `warsztat_app_wizyta`
ADD CONSTRAINT `nr_rejestracyjny_refs_nr_rejestracyjny_a0df891d`
FOREIGN KEY (`nr_rejestracyjny`)
REFERENCES `warsztat_app_samochod` (`nr_rejestracyjny`);

ALTER TABLE `warsztat_app_wizyta`
ADD CONSTRAINT `id_uzytkownika_refs_id_a6f6d7d1`
FOREIGN KEY (`id_uzytkownika`) REFERENCES `auth_user` (`id`);

CREATE INDEX `warsztat_app_samochod_6b8a2a14`
ON `warsztat_app_samochod` (`przebieg`);

CREATE INDEX `warsztat_app_samochod_3b17592e`
ON `warsztat_app_samochod` (`id_uzytkownika`);

CREATE INDEX `warsztat_app_sprzet_7d943bc0`
ON `warsztat_app_sprzet` (`nazwa_sprzetu`);

CREATE INDEX `warsztat_app_wizyta_3d40d808`
ON `warsztat_app_wizyta` (`data`);

CREATE INDEX `warsztat_app_wizyta_48fb58bb`
ON `warsztat_app_wizyta` (`status`);

CREATE INDEX `warsztat_app_wizyta_3b17592e`
ON `warsztat_app_wizyta` (`id_uzytkownika`);

CREATE INDEX `warsztat_app_wizyta_0e1dc267`
ON `warsztat_app_wizyta` (`id_czesci`);

CREATE INDEX `warsztat_app_wizyta_c243bc7b`
ON `warsztat_app_wizyta` (`id_sprzetu`);

COMMIT;
```

Aby wykonać wiarygodne testy bazy danych, należy ją uzupełnić losowymi danymi w ilości odpowiadającej zakładanym w etapie 2 średnim rozmiarom bazy. W tym celu, korzystając ze stron internetowych `http://www.generatedata.com/` oraz `http://www.mockaroo.com/` wygenerowano losowe dane w formie zapytań SQL, a następnie zapisano te zapytania w formie skryptu SQL.

Kont użytkowników, w tabeli generowanej przez *Django*, nie można było wygenerować automatycznie, ponieważ hasło nie jest przechowywane jako zwykły tekst, lecz w formie zaszyfrowanej. Automatycznie zostały jednak wygenerowane dane dla tabeli uzupełniającej dane użytkownika. Poprzez panel administratora utworzonych zostało ręcznie 5 kont użytkowników, a skrypt przedstawiony na listingu 3.2 uzupełnił dane w drugiej tabeli.

Listing 3.2: Pełny skrypt wypełniający tabelę użytkowników.

```
INSERT INTO `warsztat_app_uzytkownik`
(`id`,`user_id`,`PESEL`,`data_zatrudnienia`,`wynagrodzenie`,
`data_urodzenia`,`adres`,`nr_telefonu`,`stanowisko`,`dzial`)
VALUES (2,2,78681966532,"2009-03-07",9817,
"1962-11-25","702-6619 Taciti Road","720707598","adm","wulk");

INSERT INTO `warsztat_app_uzytkownik` (`id`,`user_id`,`PESEL`,`data_zatrudnienia`,
`data_urodzenia`,`adres`,`nr_telefonu`,`stanowisko`,`dzial`)
VALUES (3,3,20395893706,"2011-12-15",16977,
"1967-08-24","9478 Cum Street","411704151","kli","mech");

INSERT INTO `warsztat_app_uzytkownik` (`id`,`user_id`,`PESEL`,`data_zatrudnienia`,
`data_urodzenia`,`adres`,`nr_telefonu`,`stanowisko`,`dzial`)
VALUES (4,4,27146147023,"2008-01-20",4457,
"1987-09-29","703 Turpis Road","210155944","wlas","mech");

INSERT INTO `warsztat_app_uzytkownik` (`id`,`user_id`,`PESEL`,`data_zatrudnienia`,
`data_urodzenia`,`adres`,`nr_telefonu`,`stanowisko`,`dzial`)
VALUES (5,5,80255860672,"2012-08-15",7921,
"1971-09-04","174-5773 Amet Road","924951444","wlas","elek");
```

Na poniższych listingach znajduje się treść skryptów SQL wypełniających bazę testowymi danymi. Aby zachować czytelność, zamieszczone zostaną jedynie pierwsze 3 rekordy dla danej tabeli.

Listing 3.3: Pierwsze trzy przykładowe rekordy w tabeli Samochody.

```
INSERT INTO `warsztat_app_samochod` (`marka`,`model`,`nr_rejestracyjny`,
`nr_VIN`,`przebieg`,`id_uzytkownika`)
VALUES ("faucibus","risus.","JE56278",43574008028954268,267997,4);
```

```
INSERT INTO `warsztat_app_samochod` (`marka`,`model`,`nr_rejestracyjny`,
`nr_VIN`,`przebieg`,`id_uzytkownika`)
VALUES ("et","per","SV33505",18930402677506208,408647,5);

INSERT INTO `warsztat_app_samochod` (`marka`,`model`,`nr_rejestracyjny`,
`nr_VIN`,`przebieg`,`id_uzytkownika`)
VALUES ("luctus","Integer","QO02975",77293810043483976,330102,5);
```

Listing 3.4: Pierwsze trzy przykładowe rekordy w tabeli Sprzęt.

```
INSERT INTO `warsztat_app_sprzet` (`id_sprzetu`,`nazwa_sprzetu`,`status`,
`opis`,`data_przegladu`) VALUES (1,"sodales","zaj","Fusce feugiat.","2015-10-05");

INSERT INTO `warsztat_app_sprzet` (`id_sprzetu`,`nazwa_sprzetu`,`status`,
`opis`,`data_przegladu`)
VALUES (2,"tempus","zaj","orci, in consequat enim diam vel arcu.","2015-11-03");

INSERT INTO `warsztat_app_sprzet` (`id_sprzetu`,`nazwa_sprzetu`,`status`,
`opis`,`data_przegladu`) VALUES (3,"nulla","wol","nascetur","2015-10-09");
```

Listing 3.5: Pierwsze trzy przykładowe rekordy w tabeli Wizyta.

```
INSERT INTO `warsztat_app_wizyta` (`id_wizyty`,`data`,`status`,
`id_uzytkownika`,`nr_rejestracyjny`, `przebieg_w_momencie_wizyty`,
`opis`,`id_czesci`,`cena`,`czas_pracownikow`,`id_sprzetu`)
VALUES (1,"2015-08-10","onc",3,"QS81046",984536,"lorem",80,233,728,4);

INSERT INTO `warsztat_app_wizyta` (`id_wizyty`,`data`,`status`,
`id_uzytkownika`,`nr_rejestracyjny`,`przebieg_w_momencie_wizyty`,
`opis`,`id_czesci`,`cena`,`czas_pracownikow`,`id_sprzetu`)
VALUES (2,"2015-02-22","w_napr",5,"LO84747",947859,
"Phasellus libero mauris, aliquam",35,809,678,2);

INSERT INTO `warsztat_app_wizyta` (`id_wizyty`,`data`,`status`,
`id_uzytkownika`,`nr_rejestracyjny`,`przebieg_w_momencie_wizyty`,
`opis`,`id_czesci`,`cena`,`czas_pracownikow`,`id_sprzetu`)
VALUES (3,"2015-11-20","onc",1,"LT42175",310390,
"Aliquam ultrices iaculis odio. Nam",68,655,34,1);
```

Listing 3.6: Pierwsze trzy przykładowe rekordy w tabeli Części.

```
INSERT INTO `warsztat_app_czesci` (`id_czesci`,`nazwa`,
`producent`,`cena_w_hurtowni`)
VALUES (3,"ultrices iaculis","Duis Risus Company","63.43");
```

```
INSERT INTO `warsztat_app_czesci` (`id_czesci`,`nazwa`,
`producent`,`cena_w_hurtowni`)
VALUES (13,"Donec","Egestas Ligula Nullam Corporation","22.18");

INSERT INTO `warsztat_app_czesci` (`id_czesci`,`nazwa`,
`producent`,`cena_w_hurtowni`)
VALUES (23,"libero. Proin","Lobortis PC","53.02");
```

3.2 Zbiór zapytań zoptymalizowanych

3.2.1 Testy przykładowych zapytań

1. Przykładowe pobranie danych niezbędnych do wysłania powiadomienia SMS:
   Treść zapytania:

Listing 3.7: Pobranie danych do powiadomienia.

```
SELECT marka, model, nr_rejestracyjny, nr_telefonu
FROM warsztatdb.warsztat_app_samochod samochod
INNER JOIN warsztatdb.warsztat_app_uzytkownik uzytkownik
ON samochod.id_uzytkownika = uzytkownik.id;
```

marka	model	nr_rejestracyjny	nr_telefonu
nibh	nec,	OE11229	667715649
leo.	eu	QR12018	667715649
non	nascetur	RE00052	667715649
magna.	interd...	RO41405	667715649
mollis	eget	RT96968	667715649
est,	tristique	UB12035	667715649
ante.	libero	VO66456	667715649
quis	comm...	XF05357	667715649
faucibus.	vel	ZO06363	667715649
scelerisque	iaculis	ZP13097	667715649
Sed	molestie	BN38786	720707598
Mauris	et	BU71731	720707598
nec,	pede	CD81953	720707598
elit.	tempus	CP29361	720707598
Nullam	dolor	DD03425	720707598
gravida	nunc	EN02084	720707598
risus.	Sed	FV61365	720707598
arcu.	ac	GY18077	720707598
vitae	vel	HH17073	720707598
dolor	netus	JA83356	720707598
elit,	pede,	KL19070	720707598

Rysunek 3.1: Fragment wyników zwróconych przez zapytanie z listingu 3.7.

2. Przykładowe zapytanie wyliczające zyski z poszczególnych wizyt:

Listing 3.8: Wyliczenie zysku z poszególnych wizyt.

```
SELECT
id_wizyty,
cena AS 'cena_wizyty',
czas_pracownikow,
nazwa AS 'nazwa_czesci',
producent,
cena_w_hurtowni,
wizyta.cena - czesci.cena_w_hurtowni - czas_pracownikow*0.16 AS 'zysk'
FROM warsztatdb.warsztat_app_wizyta wizyta
INNER JOIN warsztatdb.warsztat_app_czesci czesci
ON wizyta.id_czesci = czesci.id_czesci;
```

id_wizyty	cena wizyty	czas_pracownikow	nazwa	producent	cena_w_hurtowni	zysk
10799	346	902	Nunc ac	Sit Amet Ultricies Limited	58	143.68
10801	567	263	Nunc ac	Sit Amet Ultricies Limited	58	466.92
10924	944	545	Nunc ac	Sit Amet Ultricies Limited	58	798.80
11017	282	725	Nunc ac	Sit Amet Ultricies Limited	58	108.00
11205	625	762	Nunc ac	Sit Amet Ultricies Limited	58	445.08
11210	696	951	Nunc ac	Sit Amet Ultricies Limited	58	485.84
11261	364	773	Nunc ac	Sit Amet Ultricies Limited	58	182.32
11333	125	220	Nunc ac	Sit Amet Ultricies Limited	58	31.80
11405	364	332	Nunc ac	Sit Amet Ultricies Limited	58	252.88
11478	336	450	Nunc ac	Sit Amet Ultricies Limited	58	206.00
75	85	192	erat.	Posuere LLC	22	32.28
164	953	229	erat.	Posuere LLC	22	894.36
311	371	354	erat.	Posuere LLC	22	292.36
493	906	847	erat.	Posuere LLC	22	748.48
506	531	460	erat.	Posuere LLC	22	435.40
624	544	208	erat.	Posuere LLC	22	488.72
654	453	259	erat.	Posuere LLC	22	389.56

Rysunek 3.2: Fragment wyników zwróconych przez zapytanie z listingu 3.8.

3. Przykładowe zapytanie wyliczające zyski z danego okresu:

Listing 3.9: Wyliczenie zysku z danego okresu.

```
SELECT sum(wizyta.cena - czesci.cena_w_hurtowni - czas_pracownikow*0.16) as 'z
FROM warsztatdb.warsztat_app_wizyta wizyta
INNER JOIN warsztatdb.warsztat_app_czesci czesci
ON wizyta.id_czesci = czesci.id_czesci
WHERE data_wizyty BETWEEN '2012-03-01' AND '2015-03-30';
```

zysk
225455.28

Rysunek 3.3: Fragment wyników zwróconych przez zapytanie z listingu 3.11.

4. Przykładowe zapytanie wyliczające zyski z danego okresu:

Listing 3.10: Wyliczenie zysku z danego okresu.

```
SELECT id_wizyty, wizyta.id_sprzetu, status_sprzetu
FROM warsztatdb.warsztat_app_wizyta wizyta
JOIN warsztatdb.warsztat_app_sprzet sprzet
ON wizyta.id_sprzetu = sprzet.id_sprzetu
WHERE wizyta.stan = 'w_napr';
```

id_wizyty	id_sprzetu	status_sprzetu
11442	2	zaj
11467	2	zaj
11472	2	zaj
11474	2	zaj
11493	2	zaj
11498	2	zaj
26	3	wol
56	3	wol
62	3	wol
97	3	wol
98	3	wol
116	3	wol
129	3	wol
149	3	wol
150	3	wol
164	3	wol
184	3	wol

Rysunek 3.4

3.2.2 Pomiary czasu wykonania zapytań dla zapytań krytycznych

Opis metody pomiarowej:
Ponieważ nie udało nam się znaleźć gotowego rozwiązania umożliwiającego zmierzenie czasów wykonywania badanych zapytań z dostatecznie dobrą dokładnością pozwalającą na porównanie otrzymanych wyników zdecydowaliśmy się na napisanie własnej procedury wykonującej pomiary.

Napisana przez nas procedura umożliwia wykonanie testowanego zapytania zadaną ilość razy oraz zmierzenia czasu z dokładnością do jednej mikrosekundy.

Listing 3.11: Kod procedury do wykonywania pomiarów.

```
delimiter //
CREATE PROCEDURE pomiar_czasu1 (in v_max int)
```

```
BEGIN
    DECLARE v_counter int default 0;
    DECLARE start_time bigint;
    DECLARE finish_time bigint;
    DECLARE duration bigint;

    SET start_time = MICROSECOND(sysdate(6)) + 1000000*SECOND(sysdate(6));

        WHILE v_counter < v_max DO
                /* TUTAJ NALEŻY UMIEŚCIĆ TESTOWANE ZAPYTANIE*/
                SET v_counter=v_counter+1;
        END WHILE;



    SET finish_time = MICROSECOND(sysdate(6)) + 1000000*SECOND(sysdate(6));

    SET duration = finish_time - start_time;
    /*SELECT duration;*/

        INSERT INTO `czasy`  (`repeats`, `duration` )
        VALUES(v_max, duration);

END
//
```

Listing 3.12: Przykład wywołania procedury.

```
drop table czasy;
create table czasy
(
repeats int not null,
duration bigint not null
);

call pomiar_czasu1 (2);
call pomiar_czasu1 (3);
call pomiar_czasu1 (4);
call pomiar_czasu1 (5);

select * from czasy;
```

Badane zapytania:

Wszystkie pomiary zostały wykonane stukrotnie, a wyniki uśrednione.

Tablica 3.1: Czasy cząstkowe.

Operacja	**Raport przychodów i rozchodów za dany okres**	**Raport historii klienta**	**Sprawdzenie statusu naprawy**	**Dodanie wizyty**	**Zmiana statusu wizyty**	**Generowanie przypomnień SMS**
$\bar{t}$	51035,667	44531,5	4170,5	69547,67	58427,67	1000,833
t_1	60042	60042	3002	108365	53796	1001
t_2	52036	52036	4002	62391	47441	1000
t_3	60044	60044	5004	53371	53428	1000
t_4	42029	42029	4004	85003	89702	1001
t_5	47033	47033	5007	54209	54043	1002
t_6	45030	6005	4004	53947	52156	1001

Tablica 3.2: Zestawienie długości trwania poszczególnych operacji.

Operacja	$\bar{t}[\mu s]$
Raport przychodów i rozchodów za dany okres	51035
Raport historii klienta	44531
Sprawdzenie statusu naprawy	4170
Dodanie wizyty	68804
Zmiana statusu wizyty	58427
Generowanie przypomnień SMS	1001

Treści testowanych zapytań:

Listing 3.13: Raport przychodów i rozchodów za dany okres.

```
SELECT
id_wizyty,
cena AS 'cena_wizyty',
czas_pracownikow,
nazwa AS 'nazwa_czesci',
producent,
cena_w_hurtowni,
wizyta.cena*0.77 AS 'przychod',
czesci.cena_w_hurtowni * czas_pracownikow*0.16 AS 'koszty',
wizyta.cena*0.77 - czesci.cena_w_hurtowni - czas_pracownikow*0.16 AS 'zysk'
FROM warsztatdb.warsztat_app_wizyta wizyta
INNER JOIN warsztatdb.warsztat_app_czesci czesci
```

```
ON wizyta.id_czesci = czesci.id_czesci
WHERE data_wizyty BETWEEN '1950-03-01' AND '2016-03-30';
```

Listing 3.14: Raport historii klienta.

```
select
id_wizyty,
data_wizyty,
stan,
nr_rejestracyjny,
przebieg_w_momencie_wizyty,
opis,
cena
from warsztatdb.warsztat_app_wizyta
where id_uzytkownika = '2';
```

Listing 3.15: Sprawdzenie statusu naprawy.

```
select data,status
from warsztatdb.warsztat_app_wizyta
where id_uzytkownika = 'id zalogowanego uzytkownika' and status != 'zak';
```

Listing 3.16: Dodanie wizyty.

```
INSERT INTO warsztatdb.warsztat_app_wizyta
(id_wizyty,data_wizyty,stan,id_uzytkownika,nr_rejestracyjny,
przebieg_w_momencie_wizyty,opis,id_czesci,cena,czas_pracownikow,id_sprzetu)
VALUES ('11504', '2015-09-10', 'onc', '3', 'QS81046',
  '984536', 'lorem', '80', '233', '728', '4');
```

Listing 3.17: Zmiana statusu naprawy.

```
UPDATE warsztatdb.warsztat_app_wizyta SET stan='zak' WHERE id_wizyty='11504';
```

Listing 3.18: Generowanie przypomnień sms.

```
SELECT marka, model, nr_rejestracyjny, nr_telefonu
FROM warsztatdb.warsztat_app_samochod samochod
INNER JOIN warsztatdb.warsztat_app_uzytkownik uzytkownik
ON samochod.id_uzytkownika = uzytkownik.id;
```

3.2.3 Rozważenie alternatywnych rozwiązań(bez indeksów) - pomiary i porównanie

Tablica 3.3: Czasy cząstkowe.

Operacja	**Raport przychodów i rozchodów za dany okres**	**Raport historii klienta**	**Sprawdzenie statusu naprawy**	**Dodanie wizyty**	**Zmiana statusu wizyty**	**Generowanie przypomnień SMS**
$\bar{t}$	72889,33	15611,33	15596,17	52084	59899,17	15531,5
t_1	62498	15606	15578	46903	62529	15599
t_2	78140	15578	15596	62461	62525	15572
t_3	62481	15627	15572	46901	62494	15545
t_4	78073	15600	15613	62473	46890	15518
t_5	78072	15632	15647	46875	62508	15491
t_6	78072	15625	15571	46891	62449	15464

Tablica 3.4: Zestawienie rozwiązania zawierającego indeksy z rozwiązaniem bez indeksów.

Operacja	**Czas przed zmianą** $[\mu s]$	**Czas po zmianie** $[\mu s]$	**Różnica** $[\mu s]$	**Różnica [%]**	**Opis dokonanej zmiany**
Raport przychodów i rozchodów za dany okres	51035	72889,33	21854,33	43%	Usunięcie indeksu z: status naprawy, data wizyty
Raport historii klienta	44531	15611,33	-28919,7	-65%	Usunięcie indeksu z: status naprawy, data wizyty
Sprawdzenie statusu naprawy	4170	15596	11426	274%	Usunięcie indeksu ze statusu naprawy
Dodanie wizyty	68804	52084	-16720	-24%	Usunięcie indeksu z: status naprawy, data wizyty
Zmiana statusu naprawy	58427	59899	1472	3%	Usunięcie indeksu ze statusu naprawy
Generowanie przypomnień SMS	1001	15531,5	14530,5	1452%	Usunięcie indeksu z: status naprawy, data wizyty

3.2.4 Wnioski

Zgodnie z oczekiwaniami usunięcie indeksów z kolumn po których dokonywane jest wyszukiwanie wydłużyło czas generowania Raportu przychodów i rozchodów za dany okres. Dziwi jednak fakt, że generowanie Raportu historii klienta przebiegło szybciej w tablicy pozbawionej indeksów. Ciężko jest nam znaleźć uzasadnienie tej sytuacji. Sprawdzenie statusu naprawy zajęło znacznie więcej czasu po usunięciu indeksu z kolumny statusu naprawy, co było oczekiwanym rezultatem. Usunięcie indeksu ze statusu naprawy i daty wizyty przyspieszyło dodanie wizyty, wynika to z faktu, że baza danych nie musiała przebudowywać struktury indeksu po dodaniu elementu do tablicy. Zmiana statusu naprawy zachowała się niemal obojętnie wobec usunięcia indeksu ze statusu naprawy. Czas generowania przypomnień wydłużył się gigantycznie, ciężko stwierdzić, czy jest to spowodowane jedynie usunięciem indeksów, czy też coś innego mogło mieć wpływ na takie wyniki (np. uruchomienie na komputerze innego procesu absorbującego procesor).

Ostatecznie ze względu na znacznie większą częstotliwość wyszukań w bazie, niż modyfikacji zawartości bazy lepszym rozwiązaniem w naszym przypadku okazuje się pozostawienie indeksów na status naprawy oraz datę wizyty.

3.2.5 Sprawdzenie jakie powinny być parametry serwera na którym działa baza danych

Obliczenia zostały wykonane na komputerze wyposażonym w procesor Intel Core i5 U4200. W wynikach benchmarków procesor ten otrzymał uśredniony wynik 44171 MIPS. [Źródło: `http://www.notebookcheck.net/Intel-Core-i5-4200U-Notebook-Processor.93563.0.html`, stan na dzień 22.01.2015]

Na podstawie testów wydajności bazy danych najbardziej krytycznym zapytaniem okazało się utworzenie nowej wizyty – 0.068 s. Przy założeniu responsywności aplikacji na poziomie 3s oraz narzutowi warstwy sieciowej na poziomie 0.5s pozostaje 2.5s na wykonanie operacji. Zakładamy, że system powinien być w stanie obsłużyć dwa takie zapytania jednocześnie.

Oszacowanie mocy procesora serwera:

$$2 \cdot 0.068s = 0.136s$$
$$2.5s \div 0.136s \approx 18$$
$$44171MIPS \div 18 \approx 2500MIPS$$

Oszacowanie ilości pamięci operacyjnej serwera oraz łącza internetowego jest bardzo trudnym zadaniem. Baza przechowuje jedynie dane tekstowe, nie występują w niej żadne pliki graficzne lub dokumenty. Baza danych ma niezwykle mały rozmiar – plik .sql z którego można odtworzyć stan bazy posiada zaledwie 963KB pamięci. Empirycznie sprawdziliśmy, że baza danych prawidłowo funkcjonyje na serwerze wyposażonym w 1GB pamięci ram. Sprawa łącza internetowego również jest sprawą drugorzędną ze względu na znikome ilości danych przesyłanych z bazy oraz fakt, że wszystkie obliczenia wykonywane są po stronie serwera, a dopiero gotowe wyniki przesyłane do aplikacji klienckiej.

3.3 Polityka bezpieczeństwa

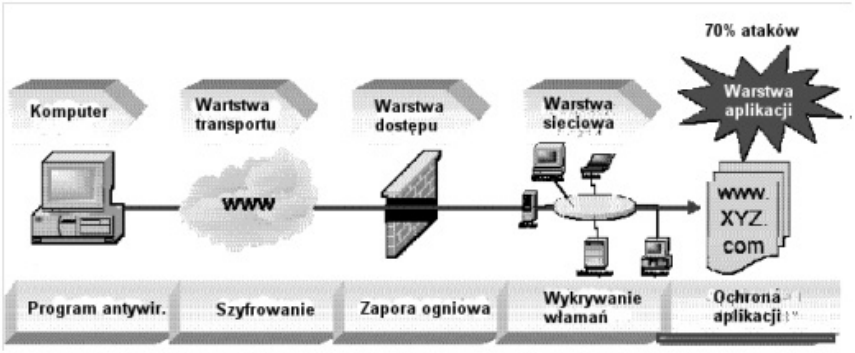
W procesie projektowania aplikacji bazodanowej niezwykle istotny jest aspekt bezpieczeństwa. Należy przeprowadzić analizę mechanizmów ochrony zarówno na poziomie bazy danych, jak i na poziomie aplikacji bazodanowej.

Konieczne jest rozważenie zastosowania m. in.:

1. metod weryfikacji dostępu do danych oraz modyfikacji danych na podstawie uprawnień zalogowanego użytkownika,
2. metod ochrony procesu logowania takich jak:
   (a) czas oczekiwania między nieudanymi próbami logowania,
   (b) odzyskiwania hasła,
   (c) okresowe wymuszanie zmiany hasła,
   (d) logowanie jako administrator jedynie z określonej domeny,
   (e) logowanie jako administrator jedynie w określonych przedziałach czasowych.

Atak przebiega zazwyczaj według następującego schematu:

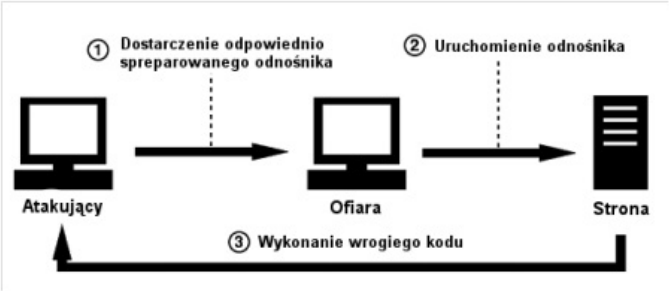
1. Skanowanie – atakujący skanuje porty, aby wykryć otwarte porty HTTP oraz HTTPS i wyszukać domyślne strony dla każdego otwartego portu,
2. Gromadzenie informacji – atakujący identyfikuje typ serwera dla każdego portu, a następnie analizuje stronę w celu poznania jej budowy i zachowania,
3. Testowanie – atakujący sprawdza kolejne elementy i podstrony witryny w celu odnalezienia błędów programistycznych, które pozwolą mu na uzyskanie większego dostępu,
4. Planowanie ataku – w oparciu o informacje zgromadzone w poprzednich krokach, atakujący określa słabe punkty aplikacji i planuje atak,
5. Dokonywanie ataku – atakujący dokonuje właściwego ataku wykorzystując wcześniej wykryte podatności; dodatkowo na koniec może on zatrzeć wszelkie ślady dokonanego właśnie włamania.



Rysunek 3.5: Większość ataków realizowanych jest poprzez warstwę aplikacji.

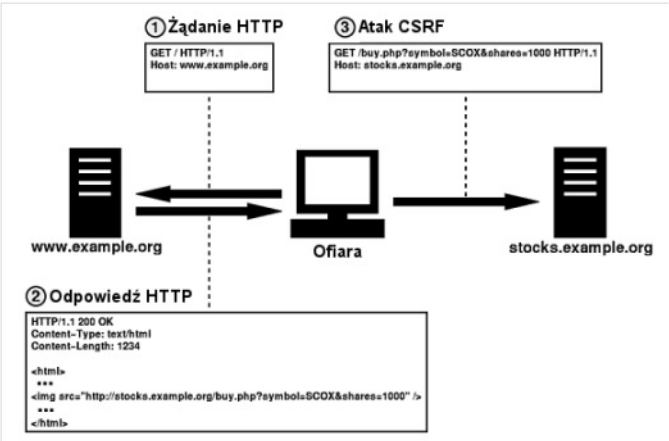
Ze względu na to, że większość ataków odbywa się poprzez warstwę aplikacji, należy podczas jej projektowania zwrócić szczególną uwagę na bezpieczeństwo. Do ataków takich można zaliczyć:

1. Iniekcja kodu SQL - ataki polegające na wstrzyknięciu i wykonaniu wrogiego kodu,
2. Cross Site Scripting (XSS) – skryptowanie skośne - polega na iniekcji złośliwego kodu w oryginalną treść strony.



Rysunek 3.6: Przebieg ataku bezpośredniego Cross Site Scripting.

3. Cross Site Request Forgeries (CSRF) - polega na wykorzystaniu przeglądarki internetowej użytkownika (ofiary ataku) do wysyłania żądań HTTP bez jego wiedzy,



Rysunek 3.7: Przebieg przykładowego ataku typu Cross Site Request Forgeries.

4. Iniekcja poleceń systemowych - dostęp do zasobów systemowych, takich jak polecenia systemu operacyjnego, skrypty powłoki oraz zewnętrzne programy wywoływane z poziomu systemu operacyjnego,

5. Iniekcja znaku końca wiersza.

Działanie każdej aplikacji internetowej oparte jest na parametrach przesyłanych pomiędzy przeglądarką użytkownika a stroną WWW. Jeśli parametry te nie są odpowiednio weryfikowane lub przechowywane są w nich dane kluczowe dla działania aplikacji, to atakujący zyskuje możliwość przeprowadzenia skutecznego ataku poprzez modyfikację tych parametrów. Manipulowanie parametrami może być przeprowadzone przy użyciu adresu URL, ciasteczek (cookies), pól formularza, a także nagłówków HTTP.

1. Manipulowanie łańcuchem żądania - zagrożenie polega na tym, że atakujący może samodzielnie zmodyfikować składniki adresu URL i w ten sposób uzyskać dostęp do poufnych danych albo je nieprawidłowo zmodyfikować,

2. Manipulowanie polami formularza - atakujący może bowiem pobrać kod HTML formularza, a następnie zmodyfikować go i wysłać żądanie z własnego komputera albo wstrzyknąć odpowiednio przygotowany kod HTML formularza bezpośrednio na stronę,

3. Manipulowanie nagłówkami żądania http - polega na tym, że analogicznie do wszystkich innych informacji pochodzących od klienta, nagłówki HTTP mogą być zmodyfikowane przez napastnika,

4. Manipulowanie wartościami ciasteczek,

5. Ataki na proces uwierzytelniania – atak siłowy, atak słownikowy, odwrócony atak siłowy,

6. Ataki na sesje użytkownika – domniemanie sesji, podsłuchiwanie ruchu sieciowego (użycie przez atakującego tego samego identyfikatora co prawowity użytkownik prowadzi bowiem do nieuprawnionego dostępu do aplikacji i wykonania operacji w jego imieniu).

Bibliografia

[1] dr inż. A. Bołtuć, *BAZY DANYCH: Wykład 8*, Uniwersytet w Białymstoku, 2010 dostęp pod adresem: `http://ii.uwb.edu.pl/~aboltuc/images/stories/bazy_danych_1/wykad_8.pdf`, aktualne na dzień 08.12.2014r

[2] dr hab. Lech Banachowski, *Wykład 11: Planowanie indeksów*, dostępne pod adresem: `http://edu.pjwstk.edu.pl/wyklady/sbd/scb/w11.htm`, aktualne na dzień 08.12.2014r

[3] `http://pl.wikipedia.org/wiki/InnoDB`, aktualne na dzień 08.12.2014r

[4] dr inż. Zofia Kruczkiewicz, *Fizyczne projektowanie bazy danych: wykład 5*, Politechnika Wrocławska