Temat nr 11. Baza wypożyczalni rowerów i akcesoriów.

# Opis świata rzeczywistego

Na zamówienie wypożyczalni sprzętu sportowego należy stworzyć aplikację użytkową wraz z bazą danych do przechowywania istotnych danych dla właściwego funkcjonowania przedsiębiorstwa. Wypożyczalnia znajduje się w rejonach górskich, gdzie największa aktywność turystyczna, a co za tym idzie aktywność wypożyczeń przypada na okres letni. Firma posiada różne rodzaje rowerów jak również akcesoria takie jak rękawice, kaski okulary. Sprzęt zazwyczaj jest wypożyczany na okres od kilku godzin do kilku dni na podstawie dokumentu tożsamości. Przedsiębiorstwo zatrudnia kilku pracowników, których kompetencjami są m.in. wypożyczanie, przyjmowanie zwrotów wypożyczeń, wystawianie faktur na życzenie klientów.

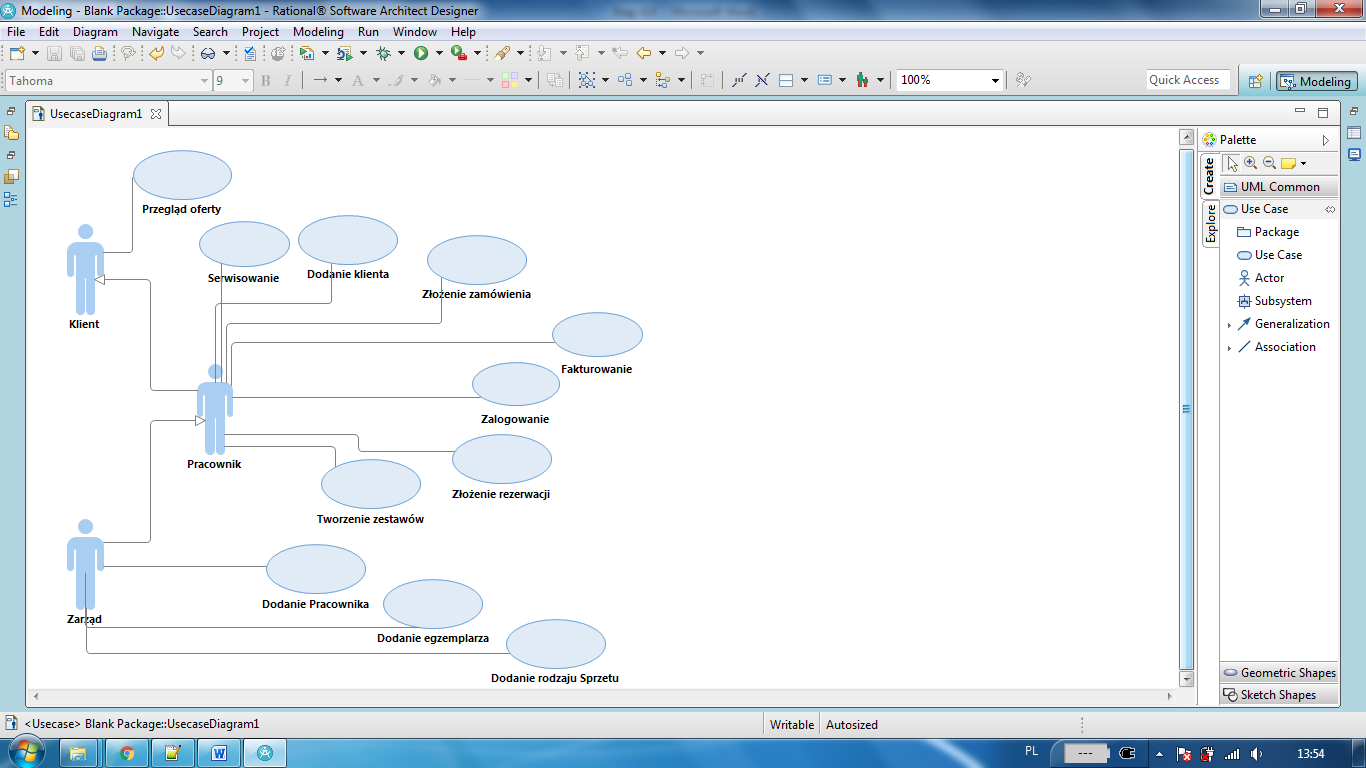
# Wymagania funkcjonalne

Zadaniem aplikacji jest wspomaganie pracowników w przyjmowaniu, wydawaniu oraz zarządzaniu wypożyczeniami. Tylko pracownicy będący w systemie mają możliwość dokonywania zmian. Wymagane jest, aby była możliwość rozszerzania bazy danych klientów, rozszerzania oferty wypożyczeń, dodawanie do systemu nowo zakupionych elementów takich jak np. rowery. Wsparciem w zakresie przestrzegania terminów zwrotów oraz naliczania opłat, a także wynagrodzeń dla pracowników. Należy również przechowywać informację o zużyciu sprzętu, wykonanych naprawach. Ponadto ważną funkcjonalnością jest możliwość naliczania opłat dodatkowych na nierzetelnych, nieterminowych klientów (przekraczających czas wypożyczenia lub niszczących sprzęt). Musi istnieć możliwość rezerwacji sprzętu, jeśli w obecnej chwili jest on niedostępny, wypożyczony lub w naprawie a także informowania o pojawieniu się możliwości wypożyczenia zarezerwowanego sprzętu.

# Wymagania niefunkcjonalne

Aplikacja musi spełniać wymogi bezpieczeństwa i ochrony danych osobowych zarówno pracowników jak i klientów. Powinna posiadać jasny i przejrzysty interfejs graficzny przyjazny dla użytkownika, powinna być dostępna tylko lokalnie z siedzib wypożyczalni (również z filii), powinna oferować efektywność działania, szczególnie w kluczowych aspektach działania przedsiębiorstwa. Musi umożliwiać skalowalność i rozszerzalność a także posiadać system uprawnień dostępu, aby kadra zarządzająca miała większe uprawnienia niż szeregowy pracownik.

# Specyfikacja wymagań funkcjonalnych za pomocą diagramu przypadków użycia

Diagram przypadków użycia został zrealizowany za pomocą narzędzia **IBM Software Architect Designer.**  Kompletny projekt został dołączony do sprawozdania, poniżej prezentujemy zdjęcia diagramu.

Rysunek . Diagram przypadków użycia

# Analiza liczby instancji dla każdej encji

Zakłada się, że maksymalny czas funkcjonowania projektowanej bazy danych wynosić będzie 5 lat. Po tym okresie, najpotrzebniejsze informacje zostaną zagregowane i zarchiwizowane, zaś stan bazy zostanie przywrócony do stanu początkowego. Działanie to zapobiegnie ciągłemu spadkowi wydajności wraz ze wzrostem ilości przechowywanych przez bazę informacji.

Poniższe szacunki przedstawiają zakładany stan bazy po roku użytkowania firmy:

* **pracownicy** – szacuje się, że wypożyczalnia będzie stale się rozwijać, natomiast w ciągu roku można oczekiwać, że szeregi firmy zasili około 30 pracowników. Liczba ta nie będzie malała, bowiem pracownik, który zostanie zwolniony, nie zostanie usunięty z systemu. Za możliwe maksimum przyjmuje się 60 pracowników pod koniec pierwszego roku funkcjonowania wypożyczalni.
* **klienci** – ich ilość w bazie, za sprawą stałych klientów, będzie rosła logarytmicznie. Wypożyczalnia zakłada obsługiwać dziennie około 100 klientów, w tej liczbie stałych klientów po roku funkcjonowania będzie około 80%. Przyniesie to około 8000 klientów w bazie. Biorąc pod uwagę różne okoliczności, za maksymalną liczbę klientów w bazie po roku funkcjonowania firmy przyjęto liczbę 12000.
* **wypożyczany sprzęt** – na początku wypożyczalnia będzie w posiadaniu stu rodzajów sprzętu, jaki wypożycza, oraz po 20 egzemplarzy z każdego rodzaju. Średni roczny przyrost mienia firmy szacuje się na około 10% stanu z początku roku. Dotyczy to zarówno rodzajów sprzętu jak i ilości egzemplarzy. Reasumując, po roku funkcjonowania systemu zakłada się 2200 egzemplarzy sprzętu do wypożyczenia, w porywach do 4000.
* **zestawy (tabela asocjacyjna)** – będzie to tabela asocjacyjna, która w danym okresie czasu będzie przechowywała proponowane przez wypożyczalnie promocyjne zestawy sprzętów. Zazwyczaj będzie miała ona od kilkudziesięciu do kilkuset wpisów, okresowo wymienianych z częstotliwością zmian promocji, ilości dostępnego na stanie sprzętu lub innych czynników.
* **rezerwacja sprzętu (tabela asocjacyjna)** – będzie to tabela asocjacyjna identyczna jak tabela wypożyczeń, jednak o nieco innym przeznaczeniu. Będzie w niej zazwyczaj kilkadziesiąt (w zależności od okresu roku) krotek.
* **dostawcy, hurtownie** – lista aktualnych, a także archiwalnych dostawców, z którymi wypożyczalnia współpracuje lub współpracowała w przeszłości. Szacuje się, że takich wpisów będzie od kilkudziesięciu do nawet stu.
* **wypożyczenia (aktualne)** – lista aktualnych wypożyczeń. Na jedno wypożyczenie będzie przypadało kilka krotek (w zależności od ilości wypożyczonego sprzętu przez klientów). Sezonowo będzie to duża tabela, do której opłacalne jest włączenie opcji indeksowania.
* **wypożyczenia (archiwalne)** – jest to idealna kopia tabeli opisanej powyżej, z tą różnicą, że będą tutaj przechowywane wszystkie zwrócone wypożyczenia przez okres czasu, jaki wynika z regulacji prawnych. Tabela ta będzie archiwizowała dziesiątki tysięcy wpisów w zależności od ilości wypożyczeń oraz od górnego limitu czasowego (szacowany jest przedział od miesiąca do połowy roku).
* **serwisy** – lista serwisów świadczących usługi naprawy sprzętu (egzemplarzy), zakłada się około pięciu.

# Analiza użycia identyfikująca podstawowe rodzaje transakcji

1. **Pracownicy** – zmienność tabeli na niskim poziomie.
   1. Wstawianie – z częstotliwością zatrudniania nowego pracownika (kilka/kilkanaście razy w ciągu roku).
   2. Modyfikacja – do kilkudziesięciu razy w ciągu roku.
   3. Usuwanie – brak.
   4. Wyszukiwanie – dość częste.
2. **Klienci** – zmienność tabeli na średnim poziomie.
   1. Wstawianie – z częstotliwością pozyskiwania klientów – około kilkaset razy w miesiącu.
   2. Modyfikacja – do kilkuset razy w ciągu roku.
   3. Usuwanie – brak.
   4. Wyszukiwanie – kilkadziesiąt razy dziennie.
3. **Wypożyczany sprzęt** – zmienność tabeli na poziomie dużym.
   1. Wstawianie – z częstotliwością poszerzanej oferty – kilkadziesiąt razy w ciągu roku.
   2. Modyfikacja – z każdym wypożyczeniem (true/false w kolumnie „wypożyczony”)
   3. Usuwanie – bardzo rzadko – przypadki jednostkowe.
   4. Wyszukiwanie – kilkadziesiąt – kilkaset razy dziennie.
4. **Zestawy** – zmienność tabeli na poziomie średnim.
   1. Wstawianie – do kilkudziesięciu razy w miesiącu.
   2. Modyfikacja – do kilkudziesięciu razy w miesiącu.
   3. Usuwanie – do kilkudziesięciu razy w miesiącu.
   4. Wyszukiwanie – kilkadziesiąt – kilkaset razy dziennie.
5. **Dostawcy, hurtownie** – zmienność tabeli na poziomie bardzo niskim.
   1. Wstawianie – bardzo rzadko.
   2. Modyfikacja – bardzo rzadko.
   3. Usuwanie – brak.
   4. Wyszukiwanie – rzadko.
6. **Rezerwacja sprzętu** – zmienność tabeli na poziomie dużym.
   1. Wstawianie – kilkanaście razy dziennie.
   2. Modyfikacja – kilka razy miesięcznie.
   3. Usuwanie – kilkanaście razy dziennie.
   4. Wyszukiwanie – kilkanaście razy dziennie.
7. **Wypożyczenia** – zmienność tabeli na poziomie bardzo dużym.
   1. Wstawianie – kilkadziesiąt razy dziennie.
   2. Modyfikacja – kilka razy dziennie.
   3. Usuwanie – kilkanaście razy dziennie.
   4. Wyszukiwanie – do kilkuset razy dziennie.
8. **Wypożyczenia (archiwum)** – zmienność tabeli na poziomie dużym.
   1. Wstawianie – Z każdym zwrotem wypożyczenia – kilkadziesiąt razy dziennie.
   2. Modyfikacja – brak.
   3. Usuwanie – kilkadziesiąt razy dziennie (z nadejściem limitu czasu przechowywania dla danych).
   4. Wyszukiwanie – kilkanaście razy rocznie.

# Sformułowanie wymagań dotyczących dostępu

1. Tworzenie raportów odbywać się będzie raz na miesiąc.
2. Generowanie nowych rekordów będzie mieć miejsce kilkadziesiąt razy dziennie.
3. Wystawianie faktur, a co za tym idzie – agregacja danych z kilku tabel, odbywać się będzie kilkadziesiąt razy dziennie.
4. Każdy użytkownik za pomocą systemu będzie mógł uzyskać informację do której otrzymania jest upoważniony, odpowiednio szybko.

# ****Analiza integralności****

1. Wszelkie zmiany notowane będą w tabeli Log za pomocą wyzwalaczy (ang. *trigger*).
2. Do ważnych operacji wymagających jednoczesnej edycji więcej niż jednej tabeli (np. wypożyczenie sprzętu) zastosowane zostaną mechanizmy transakcyjne. W przypadku awarii w trakcie realizacji transakcji – nie zostanie ona zrealizowana. Dane pozostaną spójne, zaś pracownik zostanie poinformowany o zaistniałym problemie.
3. Integralność danych zapewnia organizacja danych w bazie – przechowywane będą one w trzeciej postaci normalnej, co zapobiegnie anomaliom i niezgodności podczas ich aktualizacji oraz zredukuje stopień redundancji danych.
4. Wybrane pola w tabelach będą typu ENUM, co pozwoli na łatwiejszą kontrole ich zawartości i uniemożliwi wpisywanie wartości wg. Uznania użytkowników. Wszystkie wartości będą musiały być z wcześniej wybranego zakresu lub listy.

# ****Dostrajanie bazy danych pod względem wydajności****

1. W przypadku kiedy baza danych nie będzie osiągała zakładanej wydajności, możliwe do zastosowania będzie częściowe obniżenie stopnia jej normalizacji (do 2NF). Zwiększy to redundancję danych i objętość bazy, jednak ze względu na mniejszą ilość operacji złączeń i relacji – w znacznym stopniu poprawi wydajność.
2. Dużą rolę w poprawie wydajności bazy zapewnia podzielenie tabeli wypożyczeń na część aktualną i archiwalną. Do znalezienia konkretnego aktualnego wypożyczenia, operacja szukania zostanie wykonana na znacznie mniejszej ilości wpisów.
3. Istotnym wyborem, jeśli chodzi o szybkość działania, jest także serwer. Musi posiadać on odpowiednią ilość pamięci RAM, mieć procesor najlepiej dedykowany usługom serwerowym, zaś jego łącze powinno być szybsze niż zakłada górna granicą szacunkowych przepustowości potrzebnych do obsłużenia analizowanego ruchu sieciowego.
4. Tabele, które będą eksploatowane w największym stopniu jak np. wypożyczenia otrzymają dodatkowo indeksację na poziomie bazy danych. Indeksacja pozwoli na znaczne przyśpieszenie operacji wyszukiwania przy nieznacznym zwiększeniu zajmowanej przestrzeni dyskowej serwera.

# Wybór serwera i systemu  zarządzania  relacyjną bazą danych

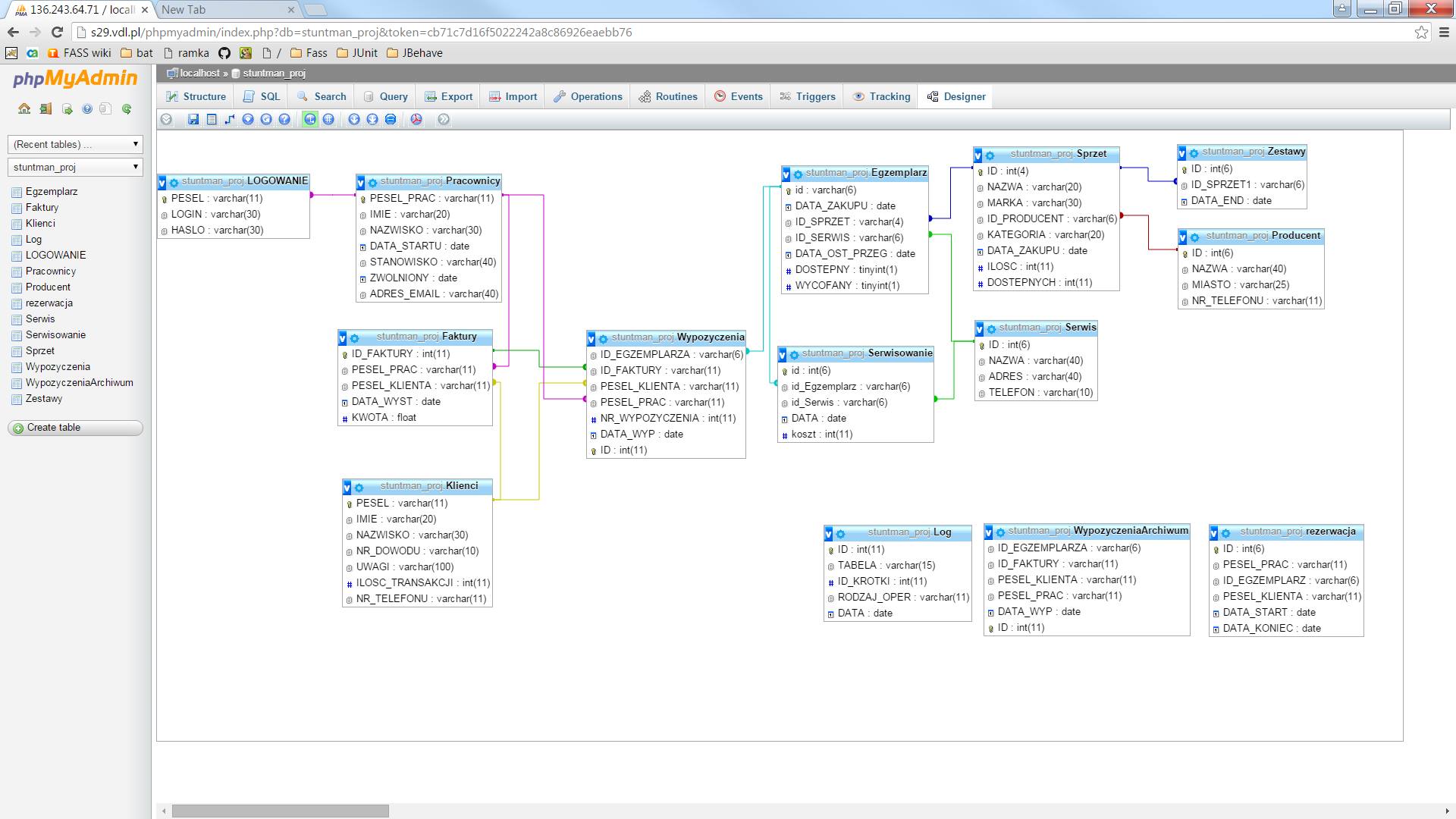
Wybór serwera jest rzeczą niewątpliwie istotną podczas realizacji projektu bazodanowego. Dobrze dopasowane środowisko potrafi w znaczący sposób ułatwić pracę, zaś jego zły wybór w najgorszym przypadku może doprowadzić nawet do fiaska projektu.

W oparciu o podane odnośniki na stronie projektu oraz własne doświadczenie w pracy z różnymi środowiskami zarządzania bazą danych, podjęto decyzję o wyborze darmowego serwera **MySQL** w wersji 5.5. Serwer ten znajduje się „w chmurze” na dedykowanym serwerze. Domyślnie dołączone jest środowisko **PHPMyAdmin** w wersji 3.5.8.2, które realizuje dostęp i graficzny interfejs użytkownika. Dodatkowo wykorzystywany jest **SQL Developer** w wersji 4.1.2.20 firmy Oracle, jako równorzędne środowisko obsługi bazy danych. Testy zostały przeprowadzone na komputerze firmy Lenovo, Thinkpad E540 z procesorem i3 4000M oraz8 GB pamięci RAM.

# Fizyczny projekt bazy danych

Baza danych została najpierw zaprojektowana i skonsultowana podczas zajęć projektowych, a następnie wykonano jej implementację na serwerze. Oczywiście podczas wykonywania implementacji zostały zauważone pewne braki i niejasności, które na bieżąco były naprawiane. Obecnie baza posiada 14 tabel zorganizowanych w paradygmacie relacyjnym. Występują wszystkie rodzaje relacji, takie jak: jeden do jeden, jeden do wielu, oraz wiele do wielu. Rysunek 1 prezentuje graficzną reprezentację tabel, szczegółowy opis znajduje się w pliku .sql dołączonym do dokumentu.

Dane do tabel zostały wprowadzone z wykorzystaniem strony internetowej https://www.mockaroo.com/



Rysunek . Graficzna interpretacja stworzonej bazy danych

Testy wydajnościowe zostały przeprowadzone dla pojedynczych jak i dla grupowanych tabel   
w operacjach SELECT, INSERT oraz UPDATE. Zostały wykorzystane do tego celu operacje INSERT takie same, jak podczas generowania danych dla bazy danych oraz polecenia z pkt.3 „zbiór zapytań zoptymalizowanych”. Tabela 1 przedstawia wyniki eksperymentów.

Tabela . Wyniki testów wydajnościowych utworzonej bazy danych

|  |  |  |  |
| --- | --- | --- | --- |
| **Zapytanie SQL** | **PHPMyAdmin [s]** | **Oracle SQLDeveloper  online [s]** | **Oracle SQLDeveloper lokalnie [s]** |
| SELECT \* FROM Klienci; | 0.0003 | 0,25 | 0,125 |
| SELECT imie, nazwisko FROM Klienci; | 0.0002 | 0,156 | 0 |
| SELECT \* FROM Wypozyczenia w INNER JOIN Klienci k ON w.pesel\_klienta = k.pesel; | 0,0002 | 0,078 | 0,094 |
| SELECT COUNT(w.id), imie, nazwisko FROM Wypozyczenia w INNER JOIN Klienci k ON w.pesel\_klienta = k.pesel | 0,0001 | 0,063 | 0 |
| SELECT COUNT(w.id), imie, nazwisko FROM Wypozyczenia w INNER JOIN Klienci k ON w.pesel\_klienta = k.pesel GROUP BY k.pesel; | 0,0002 | 0,047 | 0,027 |
| INSERT INTO `stuntman\_proj`.`Klienci` (`PESEL` ,`IMIE` ,`NAZWISKO` ,`NR\_DOWODU` ,`UWAGI` ,`ILOSC\_TRANSAKCJI` ,`NR\_TELEFONU`)  VALUES ('90231548987', 'Maciej', 'Dowbor', 'AFT123489', 'brak', '0', '500014897'); | 0,0001 | 0,078 | 0,063 |
| DELETE FROM `Klienci` WHERE pesel = '90231548987'; | 0,0014 | 0,062 | 0,109 |

Uzyskane czasy znacząco różnią się od siebie. Zostały one uzyskane dzięki wbudowanym w każde ze środowisk funkcjom do mierzenia czasu. Dodatkowo należy mieć na uwadze sposób działania. Baza danych znajduje się na serwerze internetowym. PHPMyAdmin jest domyślnym klientem dostarczonym wraz z systemem zarządzania bazą danych. Jest on przetwarzany na tym samym serwerze. Oracle SQLDeveloper jest zainstalowany lokalnie na komputerach. Przesyłanie zapytań do serwera oraz oczekiwanie na odpowiedzi z serwera powoduje dodatkowe opóźnienia. Niemniej, można łatwo zauważyć jak bardzo różnią się uzyskane czasy. Wykres 1 prezentuje omawiane zależności. W łatwy sposób możemy także porównać czasy dostępu do bazy danych z poziomu SQL Developera do bazy danych zdalnej oraz lokalnej. Można zauważyć znaczną różnicę, co oznacza, że przesyłanie danych poprzez sieć internetową wprowadza zauważalne opóźnienia.

Wykres . Zależności czasowe pomiędzy dwoma różnymi systemami zarządzania bazą danych

# Zbiór zapytań zoptymalizowanych

SELECT \* from Pracownicy p JOIN Wypozyczenia w ON p.pesel\_prac = w.pesel\_prac where w.PESEL\_PRAC = '90022101337';

SELECT p.imie, p.nazwisko from Pracownicy p JOIN Wypozyczenia w ON p.pesel\_prac = w.pesel\_prac where w.PESEL\_PRAC = '90022101337';

SELECT \* FROM Egzemplarz e WHERE e.id\_sprzet LIKE '%';

SELECT COUNT(w.id), imie, nazwisko FROM Wypozyczenia w INNER JOIN Klienci k ON w.pesel\_klienta = k.pesel GROUP BY k.pesel;

SELECT COUNT(id\_Egzemplarz) FROM Serwisowanie s GROUP BY data;

SELECT COUNT(e.id), s.marka, s.nazwa FROM Egzemplarz e INNER JOIN Sprzet s ON e.id\_sprzet = s.id GROUP BY s.nazwa;

SELECT \* FROM Faktury f WHERE data\_wyst + INTERVAL 5 DAY < NOW();

SELECT \* FROM Pracownicy p where p.zwolniony IS NULL;

# Polityka bezpieczeństwa

Bezpieczeństwo danych jest kluczowym aspektem w kontekście projektowania, wdrażania oraz utrzymywania baz danych. Podstawową, a zarazem prozaiczną rzeczą jaką każdy może wykonać, jest backup danych. Okresowy zrzut bazy danych jest operacją prostą, można ją zautomatyzować. Nleży wykonywać co najmniej raz dziennie przyrostową kopię bazy danych, całościowy zrzut zawartości powinien być wykonywany raz w miesiącu lub częściej. Potrafi to zaoszczędzić mnóstwo czasu, niejednokrotnie chroniąc wrażliwe dane, które bardzo trudno odtworzyć. Integralność danych jest drugim aspektem. Odwołania zawsze powinny znajdować odpowiednie rekordy, w przeciwnym razie może skończyć się to nieoczekiwanymi błędami, a także wyciekiem danych. Utrzymywanie porządku wspomaga aplikacja bazodanowa, która będzie następnym etapem projektu. Dla wysokiego poziomu bezpieczeństwa, wskazane jest korzystanie z trudnych, długich i złożonych haseł. Odpowiednie poziomy uprawnień przydzielane powinny być właściwym osobom. Silnym filarem bezpieczeństwa będzie również wprowadzenie restrykcji co do miejsc, z których można uzyskać dostęp do danych, bądź zarządzać bazą. Dobrą praktyką jest także stworzenie użytkowników bazodanowych ze zróżnicowanymi uprawnieniami. Pozwala to na budowanie hierarchicznej struktury uprawnień dostępu do zasobów. Kluczowe jest także pamiętanie o zabezpieczaniu stacji dostępowych, np. blokowaniem systemu, gdy użytkownik odchodzi od komputera. Wszystko to ma znaczący wpływ na ochronę danych przechowywanych w projektowanym systemie.

# Wykorzystane technologie

Zadaniem ostatniego etapu projektu bazy danych jest stworzenie aplikacji. Program ten jest bardzo ważnym elementem ułatwiającym dostęp oraz manipulację danymi. Dla zwykłego użytkownika polecenia języka SQL są dość niejasne, niezrozumiałe i trudne. Oczekuje on jasnego, prostego i intuicyjnego sposobu na dostęp do zgromadzonych informacji. Naprzeciw tym wymaganiom wychodzi stworzona aplikacja. Oprócz ułatwienia obsługi bazy, posiada ona jeszcze jedną jakże ważną zaletę. Ukrywając przed nieświadomym użytkownikiem samą implementację bazy danych, nie tylko znacząco podnosi poziom bezpieczeństwa, ale także w naturalny sposób dba o poprawność i spójność wprowadzanych, bądź usuwanych danych. Pozwala w bardzo łatwy sposób kontrolować, a także wykrywać wszelkie próby działania niezgodnie z zasadą funkcjonowania firmy, dla której została stworzona.

Z analizy potrzeb firmy stwierdzono, że najlepszą opcją będzie stworzenie aplikacji desktopowej działającej na każdej platformie sprzętowej. Jest ona przeznaczona dla pracowników oraz kierownictwa przedsiębiorstwa. Ze względu na charakter działalności, dostęp online dla klientów ogranicza się jedynie do wyszukania i ewentualnie złożenia rezerwacji na potrzebny sprzęt. Użytkownik nie ma uprawnień do wypożyczenia sprzętu on-line. Pozostałe operacje polegające na zarządzaniu bazą należy przeprowadzać lokalnie z komputerów należących do wypożyczalni (baza zezwala na dodatkowe uprawnienia jedynie dla uprzednio zdefiniowanych, stałych adresów IP). Ograniczona liczba osób, które mają dostęp pełnego zakresu funkcji aplikacji w znaczny sposób podnosi bezpieczeństwo projektowanego systemu.

Aplikacja napisana została w środowisku *Eclipse Mars.1 w wersji 4.5.1,* z wykorzystaniem języka *Java EE 1.8*. Ponadto, wykorzystano ogólnodostępny na rynku framework *Hibernate* z zestawem znormalizowanych funkcji *JPA*.

*Hibernate* jest framework-iem służącym do realizacji warstwy dostępu do danych (ang. *persistence layer*). Zapewnia przede wszystkim translację danych pomiędzy relacyjną bazą danych, a światem obiektowym (ang. *O/R mapping*). Opiera się on na wykorzystaniu opisu struktury danych za pomocą języka XML, dzięki czemu można "rzutować" obiekty, stosowane w obiektowych językach programowania takich jak *Java*, bezpośrednio na istniejące tabele bazy danych. Dodatkowo *Hibernate* zwiększa wydajność operacji na bazie danych dzięki buforowaniu i minimalizacji liczby przesyłanych zapytań. Jest to projekt rozwijany na licencji *open source*.

W skład projektu Hibernate wchodzą następujące podprojekty:

* Hibernate Core
* Hibernate Annotations
* Hibernate EntityManager
* Hibernate Shards
* Hibernate Validator
* Hibernate Search
* Hibernate Tools
* NHibernate

**Hibernate Core** stanowi centralną część wszystkich projektów. Pozostałe stanowią albo rozszerzenia (np. *Hibernate Annotations*), albo translację dla innych języków (*NHibernate*). Stąd podprojekt *Hibernate Core* jest utożsamiany z całym projektem *Hibernate*. *Hibernate Core* jest biblioteką umożliwiającą wykonywanie mapowania obiektowo-relacyjnego dla języka Java, w którym mapowania zdefiniowane są w dokumentach XML.

**Hibernate Annotations** jest rozszerzeniem projektu *Hibernate Core* pozwalającym na stosowanie adnotacji (zamiast dokumentów XML) do definiowania mapowań pomiędzy obiektami Javy oraz tabelami.

**Hibernate EntityManager** jest nakładką na projekt *Hibernate Core* wprowadzająca *EntityManager* jako centralną klasę, z poziomu której wykonywana jest komunikacja z bazą danych. Połączenie podprojektów *Hibernate EntityManager* oraz *Hibernate Annotations* pozwala posługiwać się techniką mapowania obiektowo-relacyjnego zgodnie ze standardem *Java Persistence API*.

**Hibernate Shards** ułatwia stosowanie *Hibernate Core* w przypadku stosowania wielu baz danych.

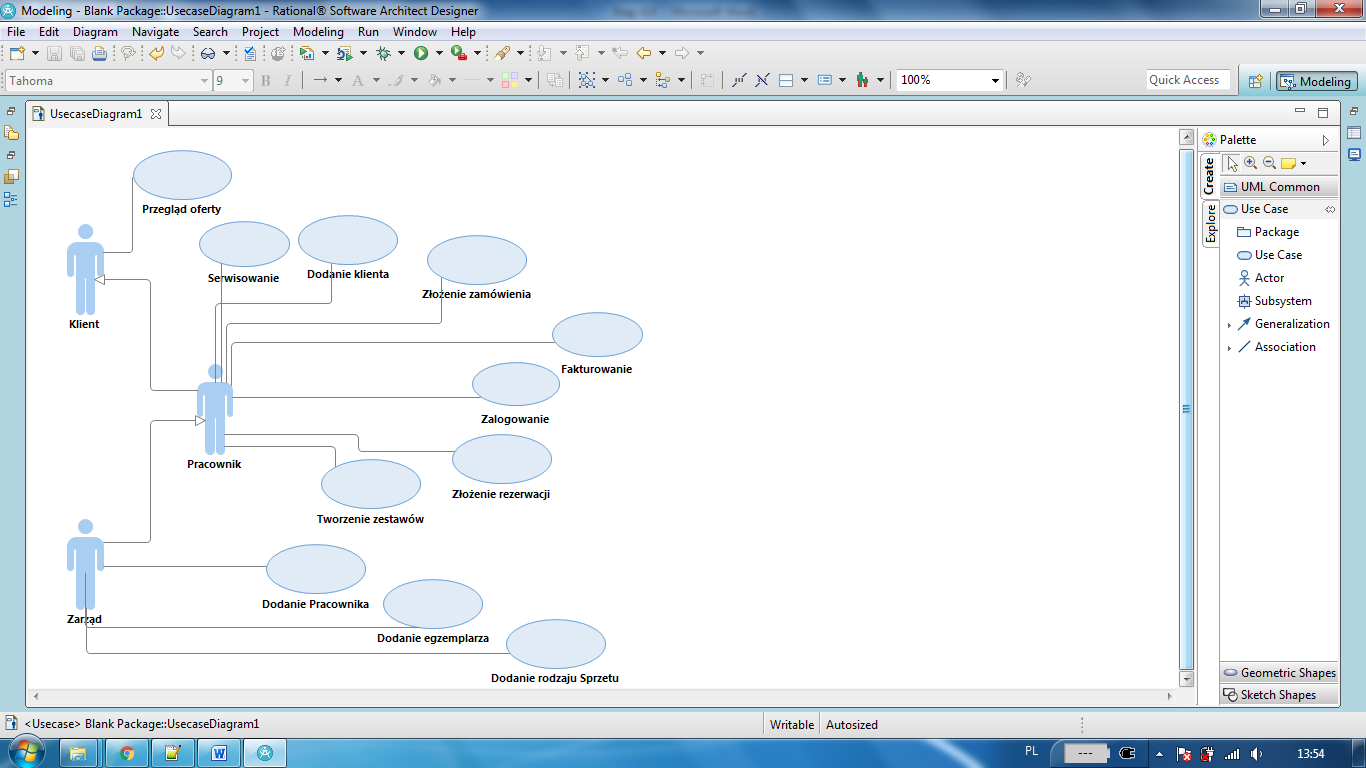
**Hibernate Validator** rozszerza zestaw adnotacji wprowadzonych przez *Hibernate Annotations* o takie, które umożliwiają dodanie ograniczeń na wartości pól podlegających mapowaniu. Ograniczenia te są podobne jak ograniczenia na pola w bazie danych (ang. *constraints*). Ograniczenia te są definiowane na poziomie obiektów podlegających zapisywaniu do bazy danych, jednakże dotyczą zarówno ograniczeń nakładanych na pola w bazie danych przy generowaniu schematu bazy danych jak i ograniczeń wprowadzanych na poziomie języka programowania.

**Hibernate Search** Rozszerza możliwości *Hibernate Core* o usługę pełnotekstowego wyszukiwania (ang. *full text search*) implementowanego w oparciu o bibliotekę *Lucene*. Podprojekt ten wykorzystuje również mechanizm adnotacji.

**Hibernate Tools** obejmuje zestaw narzędzi ułatwiających wykorzystanie *Hibernate Core* w tworzeniu projektów w języku *Java*. W szczególności zawiera on zadania dla narzędzia *Ant* umożliwiające generowanie schematu bazy danych na podstawie mapowań, czy też generowanie klas języka Java na podstawie mapowań oraz wtyczkę dla Eclipse IDE ułatwiającą pracę z *Hibernate*.

**NHibernate** jest implementacją usługi mapowania obiektowo-relacyjnego dla platformy *.NET*.

# Projektowanie i realizacja



Rys. . Diagram przypadków użycia dla projektu

Diagram przypadków użycia (Rys. 1) ilustruje funkcjonalność projektowanej aplikacji. Z uwagi na to, że aplikacja jest napisana w wersji desktopowej, nie istnieje możliwość udostępnienia oferty klientom online. W tym celu należałoby stworzyć dodatkowo niewielki serwis realizujący to zadanie.

Główny ciężar obsługi programu leży po stronie pracowników, dla których zostało przeznaczonych najwięcej dedykowanych możliwych do wykonania akcji. Wszystkie kluczowe czynności z obsługą klientów oraz bieżącym funkcjonowaniem przedsiębiorstwa. Rozszerzeniem tego użytkownika jest zarząd, który pełni rolę kierownictwa w firmie. Ten użytkownik ma uprawnienia do wszystkich oferowanych przez aplikację funkcjonalności.

W ramach prac projektowych, zgodnie z porozumieniem z prowadzącym zostało zrealizowana demonstracyjna wersja programu oferująca wybraną część kluczowych operacji na zbiorze danych, takie jak:

* Logowanie
* Dodawanie pracowników
* Dodanie rezerwacji
* Dodanie sprzętu
* Wyświetlanie pracowników
* Wyświetlanie rezerwacji
* Wyświetlanie klientów
* Wyświetlanie sprzętu
* Wyświetlanie zestawów
* Okno dialogowe dla wyboru odwołań do innych encji
* Dodawanie oraz zwracanie wypożyczeni
* Interfejs graficzny
* Menu kontekstowe

# Mechanizmy aplikacji

## Walidacja danych

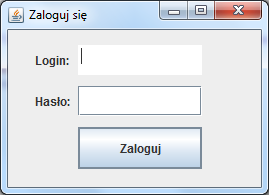
Sprawdzanie poprawności syntaktycznej wprowadzanych danych jest jednym z kluczowych elementów tworzenia aplikacji. Odbywa się to na dwa sposoby. Jedną linią są klasy mapujące w których odbywa się sprawdzanie typów danych, co umożliwia język Java. Drugim miejscem jest interfejs graficzny użytkownika, w którym w jawny sposób są stworzone mechanizmy walidacji wprowadzanych danych.

## Interakcja z bazą danych a SQL Injection

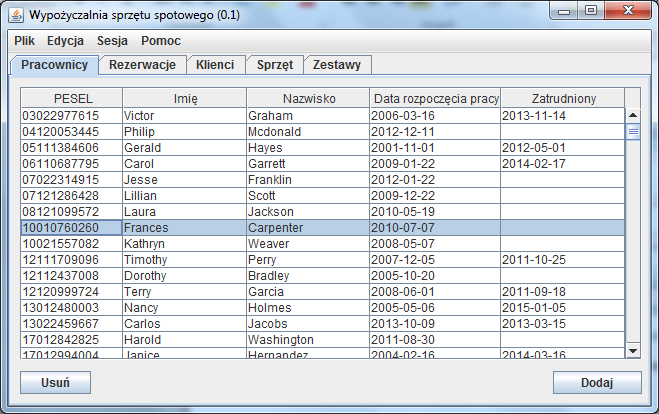
*Hibernate* realizuje odwzorowanie tabel bazy danych na rekordy. Po poprawnym skonfigurowaniu, framework sam generuje zapytania do bazy danych na podstawie pliku konfiguracyjnego oraz klas mapujących Entity. Wszystkie przeprowadzane operacje w programie odbywają się na obiektach, co znacznie przyspiesza oraz ułatwia pracę. Takie rozwiązanie eliminuje problem *SQL Injection*, bowiem nigdzie zapytania SQL nie są przekazywane bezpośrednio do bazy danych. Sytuacja komplikuje się nieco w momencie, kiedy zaistnieje potrzeba wyszukiwania parametrycznego. Hibernate oferuje możliwość wysłania zapytania SQL do bazy danych, ale w taki sposób zagrożenie ataku wzrasta. Podczas połączenia tylko z użyciem konektora JDBC zostałaby wykorzystana strategia „*Prepared Statement*”, która pomaga chronić użytkownika. W użytym w projekcie framework’u występuje podobny mechanizm. Są nim parametryzowane zapytania, które zachowują się bardzo podobnie do „*Prepared Statement*”.

# Interfejs aplikacji

Pierwszym oknem widocznym po uruchomieniu aplikacji jest aplet logowania. Pełni on ważną rolę w kwestii bezpieczeństwa. Celem zalogowania się do systemu, można użyć zarówno loginu, jak i numeru PESEL. Po pomyślnym zalogowaniu, aplikacja otrzymuje informacje o korzystającym z systemu użytkowniku. Następuje aktualizacja tabel danymi, do których dany użytkownik posiada uprawnienia. Dane te są przechowywane w pamięci i w każdej chwili program może się do nich natychmiastowo odwołać.



Rysunek 4. Panel logowania



Rysunek 5. Po pomyślnym zalogowaniu, uzyskiwany jest dostęp do panelu głównego

Interfejs aplikacji zaprojektowany został w oparciu o wygodne zakładki (panele), które to przełączając można uzyskiwać dostęp do zawartości poszczególnych działów. Dostępne są operacje takie jak dodawanie, usuwanie czy aktualizacja (modyfikacja) zgromadzonych danych. W przypadkach bardziej skomplikowanych, do dyspozycji użytkownika zaimplementowano przejrzyste okna dialogowe, które pomagają w wyborze np. właściwego producenta spośród dostępnych, co niezbędne jest do zachowania spójności danych znajdujących się w bazie.



Rysunek 6. Okno dialogowe wyboru Producenta w menu dodawania nowego sprzętu

# Wnioski i spostrzeżenia

Zastosowanie frameworka ORM *Hibernate* było bardzo dobrym posunięciem. Czas poświęcony na konfigurację wymaganą dla prawidłowego funkcjonowania zwrócił się z nawiązką podczas kolejnych etapów projektowych. Ponadto należy zaznaczyć, że środowisko programistyczne *Eclipse* znacząco wspiera *Hibernate*. Po zdefiniowaniu połączenia z bazą danych, *Eclipse* potrafi wygenerować klasy mapujące „*Entity”* wraz z relacjami. Jest to znaczące ułatwienie w procesie implementowania obsługi baz danych w oprogramowaniu. Należy pamiętać, że po wygenerowaniu klas, trzeba je sprawdzić klasy, bowiem czasem nie są one zrealizowane w sposób dający optymalną wydajność.