**Politechnika Wrocławska**

**Wydział Elektroniki**

PROJEKT Z BAZ DANYCH

System bazodanowy wspomagający zarządzanie sprzętem muzycznym   
w zespole

|  |  |
| --- | --- |
| Autor:  Paweł Biel 225949  Mateusz Wójtowicz 226069 | Prowadzący zajęcia:  Dr inż. Robert Wójcik, W4/K-9 |
|  | Ocena pracy: |

### 

Wrocław 2018

Spis treści

[**Spis rysunków** 3](#_Toc504496687)

[1. Wstęp 4](#_Toc504496688)

[1.1. Cel projektu 4](#_Toc504496689)

[1.2. Zakres Projektu 4](#_Toc504496690)

[2. Analiza Wymagań 4](#_Toc504496691)

[2.1. Opis działania i schemat logiczny 4](#_Toc504496692)

[2.2.1. Diagram przypadków użycia 5](#_Toc504496696)

[2.2.2. Scenariusze wybranych przypadków użycia 6](#_Toc504496697)

[2.2. Wymagania niefunkcjonalne 7](#_Toc504496698)

[2.3.1. Wykorzystywane technologie i narzędzia 7](#_Toc504496700)

[2.3.2. Wymagania dotyczące rozmiaru bazy danych 7](#_Toc504496701)

[2.3.3. Wymagania dotyczące bezpieczeństwa systemu 7](#_Toc504496702)

[2.3. Przyjęte założenia projektowe 8](#_Toc504496703)

[3. Projekt systemu 9](#_Toc504496704)

[3.1. Projekt bazy danych 9](#_Toc504496705)

[3.1.1. Analiza rzeczywistości i uproszczony model konceptualny 9](#_Toc504496708)

[3.1.2. Model logiczny i normalizacja 9](#_Toc504496709)

[3.1.3. Model fizyczny i ograniczenia integralności danych 10](#_Toc504496710)

[3.1.4. Inne elementy Schematu - mechanizmy przetwarzania danych 11](#_Toc504496711)

[3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych 11](#_Toc504496712)

[3.2. Projekt aplikacji użytkownika 11](#_Toc504496713)

[3.2.1. Architektura aplikacji i diagramy projektowe 11](#_Toc504496715)

[3.2.2. Interfejs graficzny i struktura menu 12](#_Toc504496716)

[3.2.3. Projekt wybranych funkcji systemu 14](#_Toc504496717)

[3.2.4. Metoda podłączenia do bazy danych - integracja z bazą danych 17](#_Toc504496719)

[3.2.5. Projekt zabezpieczeń na poziomie aplikacji 17](#_Toc504496720)

[4. Implementacja Systemu 18](#_Toc504496721)

[4.1. Realizacja bazy danych 18](#_Toc504496722)

[4.1.1. Tworzenie i definiowanie ograniczeń 18](#_Toc504496725)

[4.1.2. Implementacja przetwarzania danych 19](#_Toc504496726)

[4.1.2.1. Widok sprzętów sprawnych 19](#_Toc504496727)

[4.1.2.2. Widok sprzętów niesprawnych 20](#_Toc504496728)

[4.1.3. Implementacja uprawnień i innych zabezpieczeń 20](#_Toc504496729)

[4.2. Realizacja elementów aplikacji 21](#_Toc504496730)

[4.2.1. Obsługa menu 21](#_Toc504496731)

[4.2.2. Walidacja i filtracja 22](#_Toc504496732)

[4.2.2.1. Dla okna osoby 22](#_Toc504496733)

[4.2.2.2. Dla tabeli sprzęt 22](#_Toc504496734)

[4.2.2.3. Dla tabeli lokalizacja 23](#_Toc504496735)

[4.2.3. Implementacja interfejsu dostępu do bazy danych 23](#_Toc504496736)

[4.2.4. Implementacja wybranych funkcjonalności systemu 24](#_Toc504496737)

[4.2.4.1. Dodawanie danych 24](#_Toc504496738)

[4.2.4.2. Przeglądanie widoków 24](#_Toc504496739)

[4.2.5. Implementacja mechanizmów bezpieczeństwa 25](#_Toc504496740)

[5. Testowanie systemu 26](#_Toc504496741)

[5.1. Instalacja i konfigurowanie systemu 26](#_Toc504496742)

[5.2. Testowanie opracowanych funkcji systemu 26](#_Toc504496743)

[5.2.1. Testowanie funkcji addData(String table, String[] dataToAdd) 26](#_Toc504496747)

[5.2.2. Testowanie funkcji getViewsNames() 29](#_Toc504496748)

[5.2.1. Test funkcji removeDate() 31](#_Toc504496749)

[5.2.2. Testy funkcji updateDate() 32](#_Toc504496750)

[5.2.3. Testy funkcji getDate() 33](#_Toc504496751)

[5.2.4. Testowanie widoku automatycznego zliczania sprzętów sprawnych i niesprawnych 34](#_Toc504496752)

[5.3. Testowanie mechanizmów bezpieczeństwa 35](#_Toc504496753)

[5.4. Inne testy 35](#_Toc504496754)

[5.5. Wnioski z testów 37](#_Toc504496755)

[6. Podsumowanie 37](#_Toc504496756)

[7. Literatura 37](#_Toc504496757)

# **Spis rysunków**

[Rysunek 1: Schemat logiczny aplikacji 5](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496654)

[Rysunek 2: Diagram przypadków użycia 5](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496655)

[Rysunek 3: Model logiczny bazy danych 9](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496656)

[Rysunek 4:Model fizyczny bazy danych 10](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496657)

[Rysunek 5: Okno logowania aplikacji 12](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496658)

[Rysunek 7: Widok tabeli "Lokalizacja" 13](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496659)

[Rysunek 6: Widok tabeli "Sprzęt" 13](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496660)

[Rysunek 8: Widok tabeli "Osoby" w aplikacji 14](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496661)

[Rysunek 9: Widok widoków w aplikacji 14](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496662)

[Rysunek 10: Diagram Funkcji logowania do systemu 15](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496663)

[Rysunek 11: Diagram wyszukiwania danych w bazie 16](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496664)

[Rysunek 12: Testowanie funkcji addData()(1) 26](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496665)

[Rysunek 14: Testowanie funkcji addData()(3) 27](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496666)

[Rysunek 13: Testowanie funkcji addData()(4) 27](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496667)

[Rysunek 15: Testowanie funkcji addData()(2) 27](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496668)

[Rysunek 16: Testowanie funkcji addData()(5) 28](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496669)

[Rysunek 17: Testowanie funkcji addData()(6) 28](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496670)

[Rysunek 18: Testowanie funkcji getViewsNames()(1) 29](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496671)

[Rysunek 19: Testowanie funkcji getViewsNames()(2) 29](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496672)

[Rysunek 20: Testowanie funkcji getViewsNames()(3) 30](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496673)

[Rysunek 21: Testowanie funkcji getViewsNames()(4) 30](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496674)

[Rysunek 22: Testowanie funkcji removeDate()(1) 31](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496675)

[Rysunek 23: Testowanie funkcji removeDate()(2) 31](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496676)

[Rysunek 24: Testowanie funkcji updateDate()(1) 32](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496677)

[Rysunek 25: Testowanie funkcji updateDate()(2) 32](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496678)

[Rysunek 26: Testowanie funkcji getDate()(1) 33](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496679)

[Rysunek 27: Testowanie funkcji getDate()(2) 33](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496680)

[Rysunek 28: Testowanie funkcji zliczania sprzętu(1) 34](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496681)

[Rysunek 29: Testowanie funkcji zliczania sprzętu(2) 34](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496682)

[Rysunek 30: Testowanie mechanizmów bezpieczeństwa 35](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496683)

[Rysunek 31: Inne testy(1) 35](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496684)

[Rysunek 32: Inne testy(2) 36](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496685)

[Rysunek 33: Inne testy(3) 36](file:///E:\Obrazy\Studia\!Semestr%205\Projekt_Bazy_Danych_2\Sprawozdanie-skończone.docx#_Toc504496686)

1. Wstęp

### Cel projektu

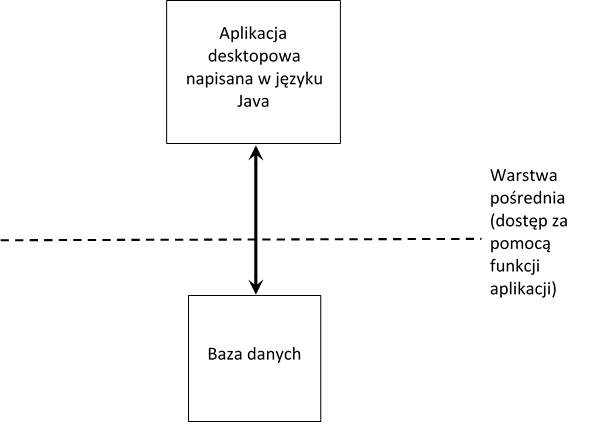
Celem jest zaprojektowanie oraz implementacja aplikacji umożliwiającej dostęp do bazy danych, przeznaczonej do monitorowania oraz zarządzania sprzętem muzycznym w zespole.

* 1. Zakres Projektu

Stworzenie aplikacji desktopowej łączącej się z bazą danych i umożliwiającej przeprowadzanie na niej podstawowych operacji. Aplikacja powinna mieć prosty oraz responsywny interfejs graficzny, umożliwiający łatwe wykorzystywanie jej funkcjonalności.

1. Analiza Wymagań
   1. Opis działania i schemat logiczny

System będzie umożliwiać monitorowanie/zarządzanie sprzętem muzycznym w zespole w oparciu o relacyjną bazę danych (tabele opisujące dane o sprzęcie np. marka, parametry, czy dany sprzęt jest zepsuty, w jakiej lokalizacji przebywa, kto go wypożyczył i inne). Dostęp do danych będzie możliwy za pomocą aplikacji (klienta) łączącego się z bazą danych umieszczoną na serwerze. By uzyskać dostęp do funkcjonalności oprogramowania należy przejść przez procedurę logowania. Wszyscy członkowie będą równorzędnymi użytkownikami systemu. Aplikacja udostępniać będzie podstawowe operacje na bazie danych takie jak: wyszukiwanie, dodawanie, edytowanie oraz usuwanie danych. Dodatkowo można będzie przeglądać stworzone w bazie widoki.

Wymagania funkcjonalne

Rysunek 1: Schemat logiczny aplikacji

● Możliwość wyświetlenia danych wyszukanych w bazie

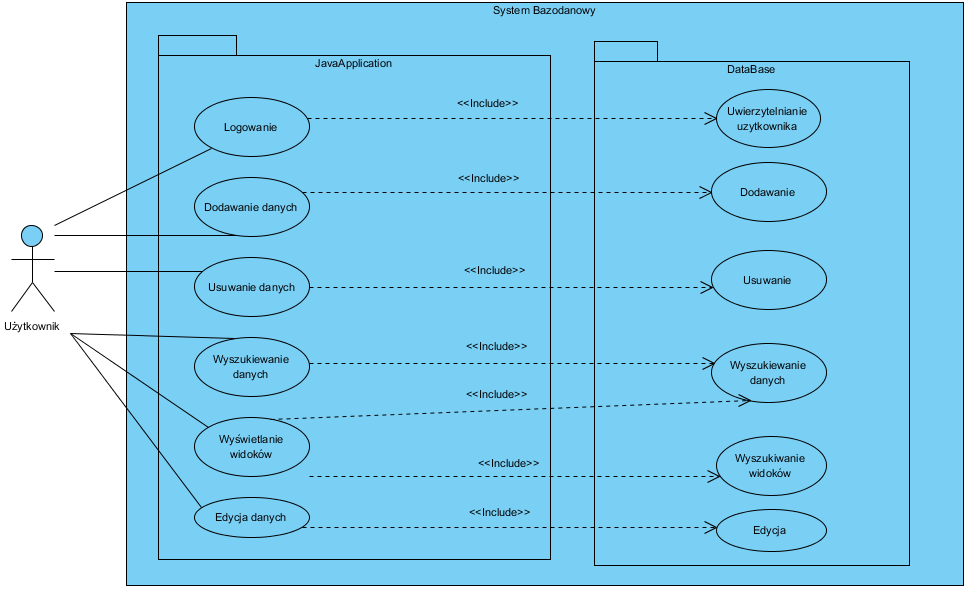
● Możliwość dodawania oraz usuwania danych

● Rozróżnianie użytkowników korzystających z systemu



Rysunek 2: Diagram przypadków użycia

* 2. 1. Diagram przypadków użycia



* + 1. Scenariusze wybranych przypadków użycia

PU - przypadek użycia

WS - warunki wstępne

WK - warunki końcowe

**PU Logowanie**

**OPIS:**

**CEL:** Przyznanie użytkownikowi praw do korzystania z funkcji systemu.

**WS:** Inicjalizacja po uruchomieniu programu.

**WK:** Pobranie loginu i hasła użytkownika celem uwierzytelnienia go z danymi w bazie.

**PRZEBIEG:**

1. Należy pobrać podany przez użytkownika login oraz hasło.

2. Wywoływane jest PU Uwierzytelnianie użytkownika w celu sprawdzenia czy podany login jest zarejestrowany w bazie i jeżeli tak to czy hasło do niego przypisane się zgadza z tym, które zostało podane.

3. Jeżeli zwrócony wynik oznacza brak dostępu należy wyświetlić stosowną informację, w przeciwnym wypadku należy udzielić dostępu do funkcji systemu.

**PU Uwierzytelnianie użytkownika**

**OPIS:**

**CEL:** Sprawdzanie czy podane hasło zgadza się z hasłem danego użytkownika.

**WS:** Może być wywołany z PU Logowanie.

**WK:** Potwierdzenie tożsamości osoby logującej się do systemu.

**PRZEBIEG:**

1. Sprawdzane jest czy hasło przypisane do logującego się użytkownika jest takie samo jak otrzymane hasło.

2. Jeżeli hasła się zgadzają zwracana jest informacja o uwierzytelnieniu. W przeciwnym wypadku zwracana jest informacja o braku dostępu.

**PU Dodawanie danych**

**OPIS:**

**CEL:** Dodanie nowej pozycji do bazy.

**WS:** Inicjalizacja po uruchomieniu programu i zalogowaniu się.

**WK:** Pobranie atrybutów opisujących dodawaną pozycję celem dodania go do bazy.

**PRZEBIEG:**

1. Należy pobrać podane przez użytkownika atrybuty opisujące dodawaną pozycję.

2. Wywoływane jest PU Dodawanie.

**PU Dodawanie**

**OPIS:**

**CEL:** Dodanie do bazy nowej pozycji.

**WS:** Może być wywołany z PU Dodawanie danych.

**WK:** Dodanie nowej krotki do odpowiedniej tabeli w bazie.

**PRZEBIEG:**

1. Stworzenie odpowiedniej transakcji dodającej określoną przez atrybuty pozycje do bazy.

2. Przeprowadzenie transakcji na bazie danych.

* 1. Wymagania niefunkcjonalne

● Dostęp do systemu poprzez oprogramowanie z interfejsem graficznym, z poziomu komputerów PC

● Zapewnienie stabilności i możliwie maksymalnej bezawaryjności opisywanego systemu

● Przechowywanie danych w relacyjnej bazie danych opartej na technologii MySQL

● Uwzględnienie możliwości rozbudowy i rozwoju systemu w przyszłości

● Procedura logowania oraz walidacji danych.

* 1. 1. Wykorzystywane technologie i narzędzia

Baza danych będzie obsługiwana za pośrednictwem serwera bazy danych MySQL. Interfejs użytkownika zostanie zrealizowany w postaci aplikacji obiektowej w języku Java, uruchamianej lokalnie na komputerze, będącym równocześnie serwerem aplikacji. Do specyfikacji funkcji systemu wykorzystany zostanie zunifikowany język modelowania UML.

* + 1. Wymagania dotyczące rozmiaru bazy danych

Rozmiar bazy danych będzie się ograniczał do stałej ilości tabel tzn.

* Tabela zawierająca informacje o sprzęcie muzycznym
* Tabela zawierająca informacje o osobach
* Tabela zawierająca informacje o lokalizacjach

oraz do dwóch, widoku zawierającego informacje o sprzęcie, który jest sprawny i gotowy do użycia i widoku, który zawiera informacje o sprzęcie niesprawnym.

* + 1. Wymagania dotyczące bezpieczeństwa systemu

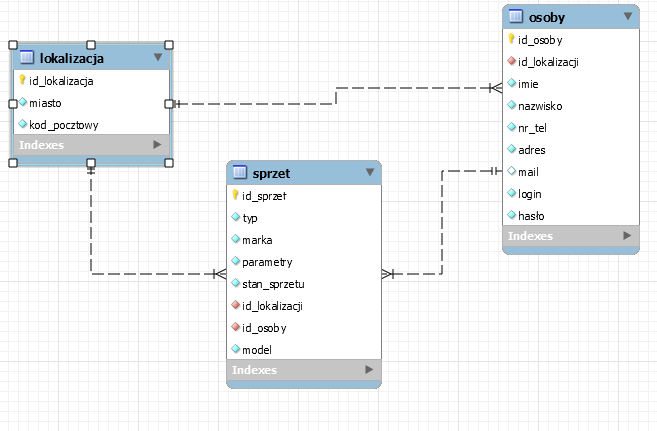
Dostęp do funkcjonalności aplikacji, a więc i możliwość wprowadzania zmian w bazie, należy umożliwić wszystkim członkom zespołu. Każdy z nich powinien otrzymać własny login oraz hasło, umożliwiające im zalogowanie się do aplikacji. Wszyscy użytkownicy będą posiadali takie same uprawnienia w zakresie możliwości korzystania z funkcji udostępnianych przez oprogramowanie.

* 1. Przyjęte założenia projektowe

System będzie stworzony w formie desktopowej aplikacji okienkowej, będącej interfejsem umożliwiającym użytkownikowi wykonywanie różnych operacji na relacyjnej bazie danych. Aplikacja napisana zostanie w języku Java co zapewni łatwą możliwość rozbudowy jej funkcjonalności w przyszłości (obiektowe podejście). Jako system służący do zarządzania stworzoną relacyjną bazą danych wybrany został MySQL firmy Oracle. Użytkownik będzie mógł wyszukiwać, dodawać, edytować oraz usuwać dane z bazy za pomocą aplikacji, w każdej z istniejących w bazie tabeli. Nie będzie mógł on jednak usuwać istniejących ani dodawać nowych tabeli do bazy, gdyż te są zaprojektowane w ten sposób, że nie należy ich zmieniać. Możliwe będzie, również przeglądanie widoków stworzonych w bazie. Widoków można będzie tworzyć dowolnie dużo, jednak będzie trzeba to robić z poziomu systemu zarządzania bazą danych. Sama aplikacja będzie jednak w stanie umożliwić przeglądanie wszystkich widoków niezależnie od ich liczby, wielkości i rodzaju. Walidacja danych będzie odbywać się na poziomie oprogramowania jak i samej bazy danych.

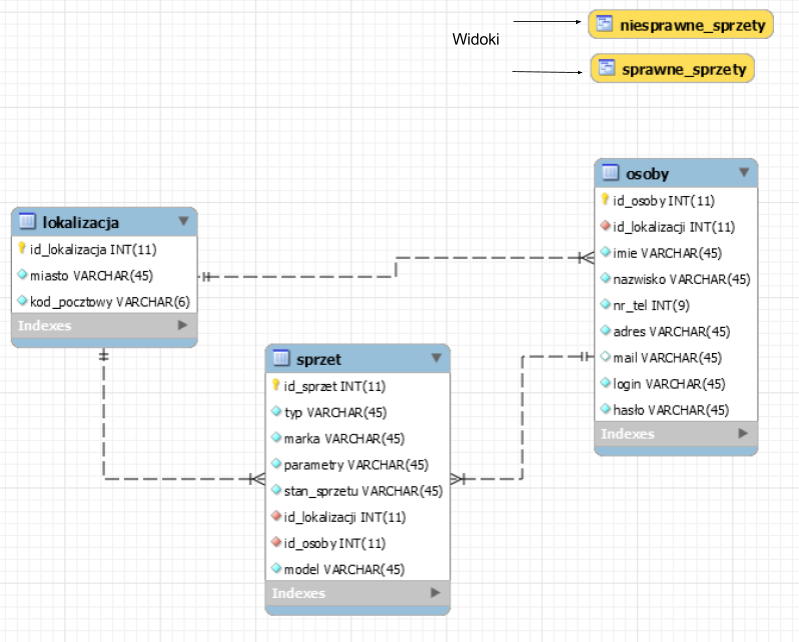
1. Projekt systemu
   1. Projekt bazy danych
2. 1. 1. Analiza rzeczywistości i uproszczony model konceptualny

W bazie powinny się znajdować trzy podstawowe tabele: tabela użytkowników, tabela lokalizacji oraz tabela sprzętów. Pierwsza z nich zawierać będzie podstawowe informacje o użytkownikach, czyli członkach zespołu, a także o przypisanych do nich loginach i hasłach. Sprzęt muzyczny, który posiada zespół nie jest składowany w żadnym wspólnym magazynie, a w domach wszystkich członków zespołu, którzy mieszkają w różnych miastach. Dlatego potrzebna jest druga tabela przechowująca informacje o różnych lokalizacjach, w których może znajdować się sprzęt, czyli po prostu miejscach zamieszkania członków zespołu. Ostatnia z tabel będzie przechowywać informacje o sprzęcie opisujące jaki to sprzęt, jaki jest jego stan oraz gdzie się znajduje.

* + 1. Model logiczny i normalizacja

Rysunek 3: Model logiczny bazy danych

* + 1. Model fizyczny i ograniczenia integralności danych



Rysunek 4:Model fizyczny bazy danych

* + 1. Inne elementy Schematu - mechanizmy przetwarzania danych

W bazie znajdują się dwa najbardziej potrzebne z punktu widzenia użytkownika widoki. Pierwszy z nich sprawne\_sprzety wyciąga z tabeli sprzętów tylko te które są w danym momencie sprawne. Analogicznie drugi z widoków niesprawne\_sprzęty wyciąga z bazy, których aktualny stan to niesprawny.

CREATE VIEW sprawne\_sprzety AS

SELECT s.id\_sprzet, s.typ, s.marka , s.parametry, l.miasto, o.imie, o.nazwisko

FROM sprzet s

JOIN osoby o

ON o.id\_osoby = s.id\_osoby

JOIN lokalizacja l

ON l.id\_lokalizacja = s.id\_lokalizacji

WHERE s.stan\_sprzetu LIKE 'SPRAWNY';

CREATE VIEW niesprawne\_sprzety AS

SELECT s.id\_sprzet, s.typ, s.marka , s.parametry, l.miasto, o.imie, o.nazwisko

FROM sprzet s

JOIN osoby o

ON o.id\_osoby = s.id\_osoby

JOIN lokalizacja l

ON l.id\_lokalizacja = s.id\_lokalizacji

WHERE s.stan\_sprzetu LIKE 'NIESPRAWNY';

* + 1. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

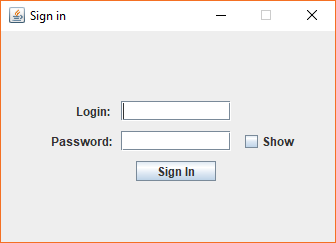
W naszej bazie danych nie mamy rozróżnienia na użytkowników i ich uprawnienia stąd też nie występują tego typu zabezpieczenie. Jedyne zabezpieczenia bazy danych to hasło i login nadawany przy tworzeniu bazy danych dla użytkownika “root”.

* 1. Projekt aplikacji użytkownika
  2. 1. Architektura aplikacji i diagramy projektowe

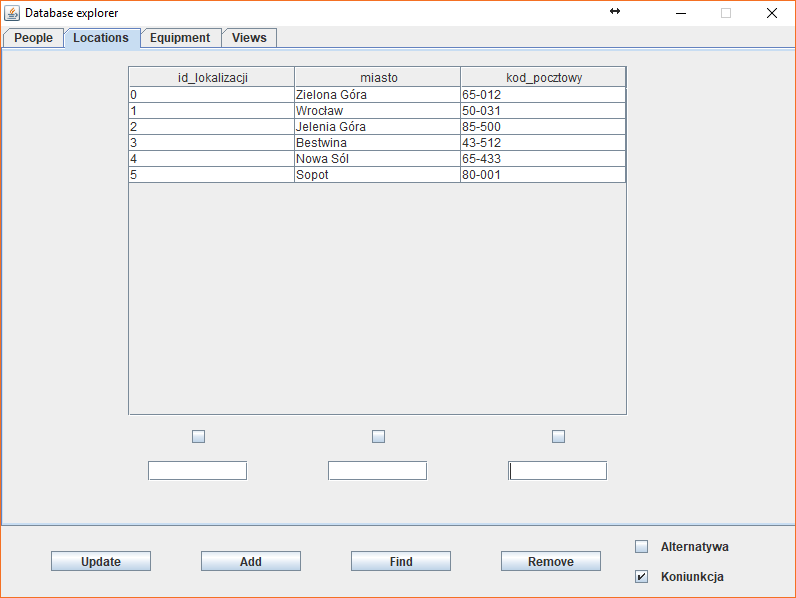
Oprogramowanie napisane jest w języku Java co pozwoli na uruchomienie go na wielu platformach. Taki wybór języka narzuca konieczność programowania zorientowanego obiektowo, co sprawia że aplikacja jest łatwa w rozbudowie. Każde z okien reprezentuje oddzielna klasa. Stworzona została również oddzielna klasa służąca do przeprowadzania wszelkich procedur bezpośrednio związanych z bazą danych. Językiem aplikacji jest język angielski.

* + 1. Interfejs graficzny i struktura menu

Po uruchomieniu programu pojawia się okienko logowania, którego rozmiaru nie można zmieniać. Widoczne są w nim dwa pola tekstowe: górne i dolne, w które użytkownik wpisuje kolejno swój login i hasło. Znajdujący się obok drugiego pola tekstowego checkbox pozwala odsłonić wpisywane hasło, które domyślnie jest wypisywane niejawnie (wpisywane znaki zastąpione są znakiem \* ). Do zalogowania się służy jedyny znajdujący się w oknie przycisk (Sign In). Jeżeli hasło lub login jest niepoprawne to poniżej tego przycisku, kolorem czerwonym wyświetlana jest odpowiednia informacja. Jeżeli jednak login i hasło są poprawne, to okno to jest zamykane, a otwarte zostanie główne okno programu.

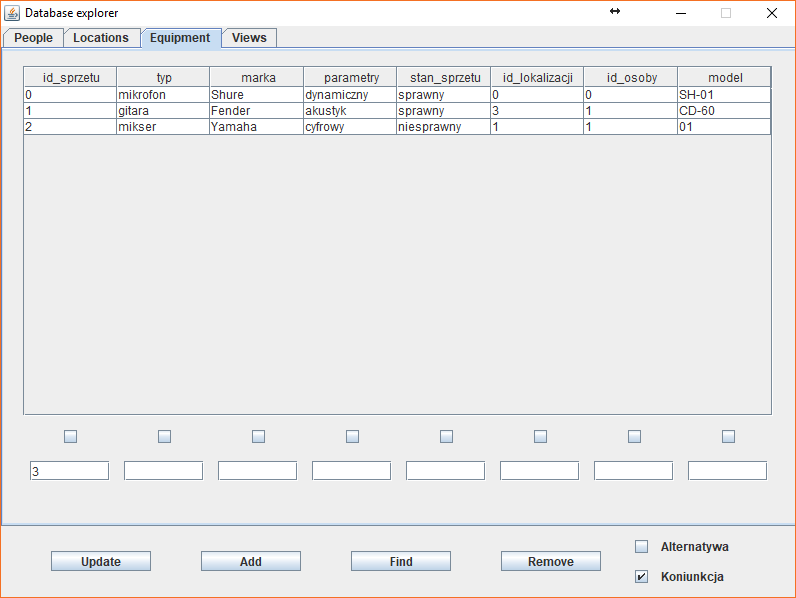


Rysunek 5: Okno logowania aplikacji

Główne okno programu jest podzielone na dwie części. Część z zakładkami, dzięki której możemy przełączać się między tabelami oraz otworzyć sekcje przeglądania widoków oraz część z przyciskami służącą do przeprowadzania operacji na bazie. W pierwszej części mamy cztery zakładki. Pierwsze trzy odpowiadają tabelą istniejącym w bazie. Każda z tych zakładek jest zbudowana na tej samej zasadzie. W centralnym punkcie znajduje się tabela wyświetlająca dane pobrane z bazy. Panel, w którym znajduje się taka tabela ma możliwość przewijania w razie większej ilości danych do wyświetlenia. Poniżej tabeli znajdują się rząd checkboxów i rząd pól tekstowych. Każdej kolumnie w tabeli odpowiada po jednym elemencie z obu tych rzędów. W pola tekstowe wpisuje się wartości dla danych kolumn, według których chcemy przeprowadzać operacje na bazie. Jednak by wartość z pola tekstowego była brana pod uwagę, musi być zaznaczony znajdujący się nad danym polem checkbox. Ostatnią z zakładek jest zakładka służąca do przeglądania stworzonych w bazie widoków. Po lewej stronie mamy listę checkboxów będącą listą istniejących widoków. Natomiast po prawej mamy tabelę wyświetlającą dane z widoku, umieszczoną w panelu, który można przewijać w razie większej ilości danych. W tabeli wyświetlane są dane z widoku, który jest aktualnie wybrany na liście z lewej strony (checkbox przy jego nazwie jest zaznaczony). Lista widoków jest aktualizowana przy zalogowaniu do programu i również jest umieszczona w panelu, który można przewijać gdy w bazie znajdzie się większa ilość widoków. W przypadku gdy w bazie nie istnieją żadne widoki, w panelu listy w kolorze czerwonym wyświetlana jest odpowiednia informacja. Drugą częścią głównego okna jest panel z przyciskami Edit, Add, Find, Remove oraz checkboxami Alternatywa i Koniunkcja. Przyciski służą do przeprowadzania przypisanych do nich procedur na bazie danych kolejno: dodawania danych, wyszukiwania danych i usuwania danych. Checkboxy są modyfikatorami dla procedur wyszukiwania i usuwania. Określają jak traktowane potraktowane zostaną wartości wpisane w pola tekstowe pod tabelą. W danym momencie zaznaczony może być tylko jeden checkbox. Główne okno programu ma zablokowaną możliwość zmiany rozmiaru.

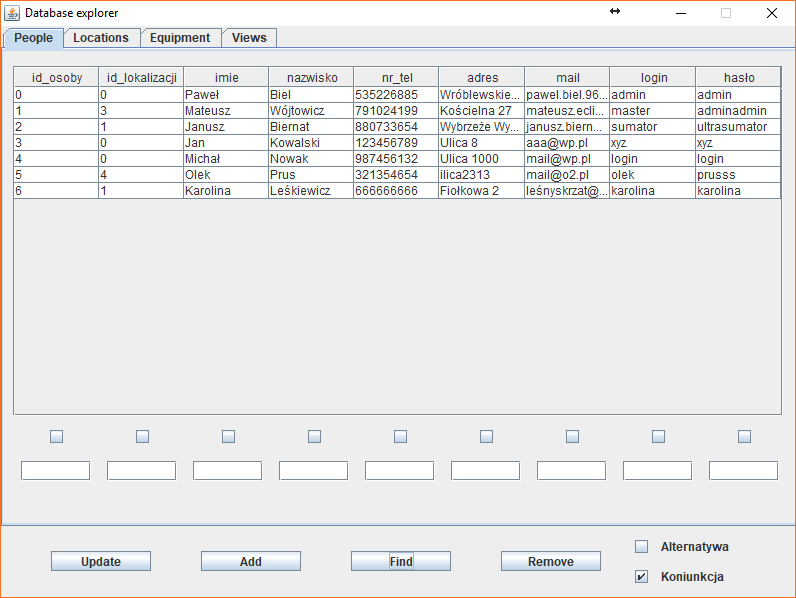
Rysunek 7: Widok tabeli "Lokalizacja"

w aplikacji

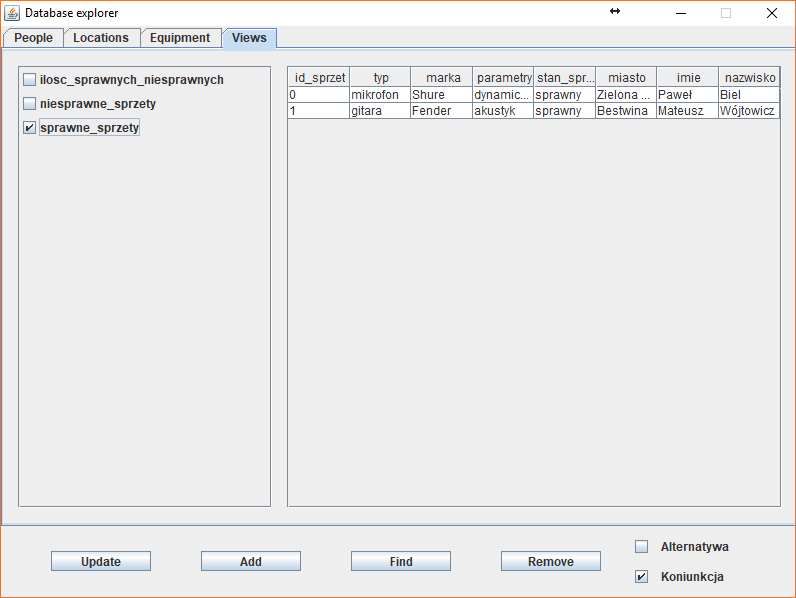


Rysunek 6: Widok tabeli "Sprzęt"

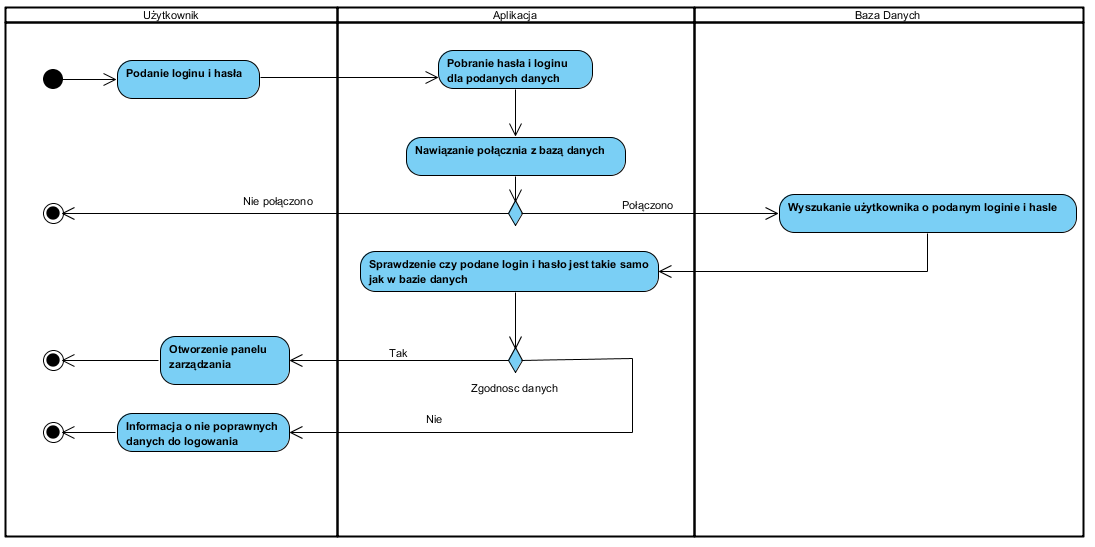
w aplikacji

* + 1. Projekt wybranych funkcji systemu

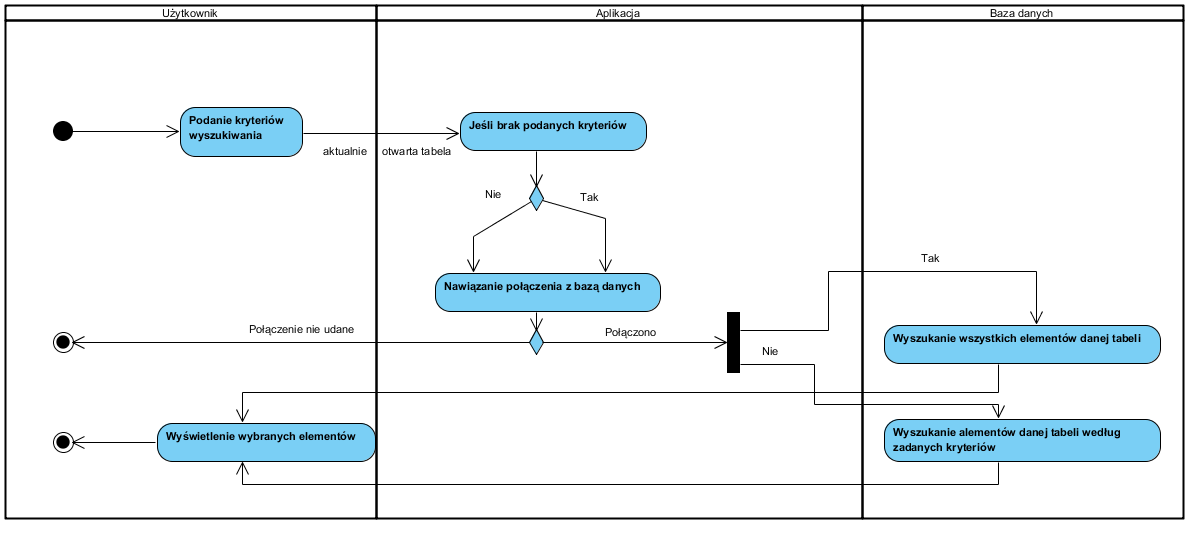
Rysunek 8: Widok tabeli "Osoby" w aplikacji



Rysunek 9: Widok widoków w aplikacji



Rysunek 10: Diagram Funkcji logowania do systemu



Rysunek 11: Diagram wyszukiwania danych w bazie

* + 1. Metoda podłączenia do bazy danych - integracja z bazą danych

Baza danych, z którą łączy się aplikacja znajduje się na serwerze MySQL Server postawionym na hoście lokalnym. Do porozumiewania się z bazą w języku SQL aplikacja korzysta z interfejsu JDBC. Wymagane biblioteki znajdują się w pakietach java.sql oraz com.mysql.jdbc.

* + 1. Projekt zabezpieczeń na poziomie aplikacji

Dostęp do funkcjonalności oprogramowania jest osiągalny po pomyślnym zalogowaniu w oknie logowania, widocznym po uruchomieniu aplikacji.

1. Implementacja Systemu
   1. Realizacja bazy danych
2. 1. 1. Tworzenie i definiowanie ograniczeń

Tworzenie tabel (tabele zostały wygenerowane za pomocą MySql Workbanch)

* + - 1. Utworzenie tabeli sprzęt

CREATE TABLE `sprzet` (

`id\_sprzet` int(11) NOT NULL,

`typ` varchar(45) NOT NULL,

`marka` varchar(45) NOT NULL,

`parametry` varchar(45) NOT NULL,

`stan\_sprzetu` varchar(45) NOT NULL DEFAULT 'SPRAWNY',

`id\_lokalizacji` int(11) NOT NULL,

`id\_osoby` int(11) NOT NULL,

`model` varchar(45) NOT NULL,

PRIMARY KEY (`id\_sprzet`),

KEY `id\_uzytkownika\_idx` (`id\_osoby`),

KEY `lokal\_idx` (`id\_lokalizacji`),

CONSTRAINT `lokal` FOREIGN KEY (`id\_lokalizacji`) REFERENCES `lokalizacja` (`id\_lokalizacja`),

CONSTRAINT `sprzet\_ibfk\_1` FOREIGN KEY (`id\_osoby`) REFERENCES `osoby` (`id\_osoby`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Tabela zawierajaca inforamcje o sprzecie'

* + - 1. Tworzenie tabeli lokalizacja

CREATE TABLE `lokalizacja` (

`id\_lokalizacja` int(11) NOT NULL,

`miasto` varchar(45) NOT NULL,

`kod\_pocztowy` varchar(6) NOT NULL,

PRIMARY KEY (`id\_lokalizacja`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8

* + - 1. Tworzenie tabeli osoby

CREATE TABLE `osoby` (

`id\_osoby` int(11) NOT NULL,

`id\_lokalizacji` int(11) NOT NULL,

`imie` varchar(45) NOT NULL,

`nazwisko` varchar(45) NOT NULL,

`nr\_tel` int(9) NOT NULL,

`adres` varchar(45) NOT NULL,

`mail` varchar(45) DEFAULT NULL,

`login` varchar(45) NOT NULL,

`hasło` varchar(45) NOT NULL,

PRIMARY KEY (`id\_osoby`),

KEY `for\_idx` (`id\_lokalizacji`),

CONSTRAINT `lokal\_os` FOREIGN KEY (`id\_lokalizacji`) REFERENCES `lokalizacja` (`id\_lokalizacja`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8

* + 1. Implementacja przetwarzania danych
       1. Widok sprzętów sprawnych

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`

SQL SECURITY DEFINER VIEW `sprawne\_sprzety` AS select`s`.`id\_sprzet`AS `id\_sprzet`,

`s`.`typ`AS `typ`,

`s`.`marka`AS `marka`,

`s`.`parametry`AS`parametry`,

`s`.`stan\_sprzetu`AS`stan\_sprzetu`,

`l`.`miasto`AS `miasto`,

`o`.`imie`AS `imie`,

`o`.`nazwisko`AS `nazwisko`

from ((`sprzet` `s` join `osoby` `o`on

((`o`.`id\_osoby` = `s`.`id\_osoby`)))

join `lokalizacja` `l`on((`l`.`id\_lokalizacja` = `s`.`id\_lokalizacji`)))

where (`s`.`stan\_sprzetu` like 'SPRAWNY')

* + - 1. Widok sprzętów niesprawnych

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost`

SQL SECURITY DEFINER

VIEW `niesprawne\_sprzety` AS

select`s`.`id\_sprzet` AS `id\_sprzet`,

`s`.`typ` AS `typ`,

`s`.`marka` AS `marka`,

`s`.`parametry` AS `parametry`,

`s`.`stan\_sprzetu` AS `stan\_sprzetu`,

`l`.`miasto` AS `miasto`,

`o`.`imie` AS `imie`,

`o`.`nazwisko` AS `nazwisko`

from((`sprzet` `s` join `osoby` `o` on

((`o`.`id\_osoby` = `s`.`id\_osoby`)))

join `lokalizacja`

`l`on((`l`.`id\_lokalizacja`= `s`.`id\_lokalizacji`)))

where(`s`.`stan\_sprzetu` like 'NIESPRAWNY')

* + 1. Implementacja uprawnień i innych zabezpieczeń

W naszej bazie danych nie ma rozróżnienia na użytkowników, co też wiąże się z tym,że nie ma rozróżnienia uprawnień i zabezpieczeń.

* 1. Realizacja elementów aplikacji
  2. 1. Obsługa menu

W głównym oknie możemy przełączać się pomiędzy zakładkami widocznymi w górnym lewym rogu zwyczajnie na nie klikając. Przy przełączaniu się między zakładkami zmieniać będzie się centralna część głównego okna (zawartość zakładki), jednak dolna część (zawierająca przyciski Edit, Add, Find, Remove i dwa checkboxy) zawsze jest taka sama. Dolna część głównego okna jest funkcjonalna gdy znajdujemy się w jednej z trzech pierwszych zakładek (odpowiadających tabelom). Również tylko w tych trzech zakładkach pod tabelami znajdują się rząd checkboxów i rząd pól tekstowych. W pola tekstowe możemy dowolnie wpisywać wartości, według których chcemy podejmować dalej różne procedury (np. wyszukiwanie z danej wartości). Żeby jednak wartość wpisana w dane pole była brana pod uwagę podczas tych procedur (za wyjątkiem dodawania danych), należy zaznaczyć znajdujący się nad nią checkbox.

Jeżeli jednak chcemy wykonać operację wyszukiwania, tak by wyciągnąć wszystkie dane z tabeli (SELECT \* FROM table), należy odznaczyć wszystkie checkboxy. W dolnej części oprócz przycisków służących do przeprowadzania zaimplementowanych procedur na bazie, znajdują się również dwa checkboxy, z których w danym momencie zaznaczony musi być jeden i tylko jeden. Decydują one o tym jak traktowane będą wartości w polach tekstowych. W przypadku zaznaczenia niewybranego checkboxa drugi odznacza się automatycznie, ponieważ w danym momencie tylko jeden może być zaznaczony. Podobnie jeżeli odznaczymy zaznaczonego checkboxa drugi zaznaczy się automatycznie, gdyż zawsze któryś z nich musi być wybrany. Identycznie działa to w przypadku listy widoków w zakładce *views,* gdzie również jedynie jeden widok może być jednocześnie wyświetlany, a więc i wybrany na liście. Jeżeli więc na liście widoków zaznaczymy checkboxa odpowiadającego innemu widokowi to aktualnie zaznaczony, odznaczy się automatycznie. Natomiast gdy sami odznaczymy zaznaczony checkbox, to automatycznie wybrany oraz zaznaczony zostanie checkbox pierwszego widoku z listy.

* + 1. Walidacja i filtracja
       1. Dla okna osoby

**id\_osoby** - jest kluczem głównym w tabeli osoby, a więc musi być unikalny, aplikacja ignoruje wpisany klucz i nadaje numer według kolejności w bazie danych.

**id\_lokalizacji** - aplikacja pobiera z bazy danych czy podana lokalizacja istnieje oraz zostaje sprawdzone czy podana wartość jest liczbą i czy nie jest puste.

**imie** - sprawdzenie czy polę nie jest puste.

**nazwisko** - sprawdzenie czy polę nie jest puste.

**nr\_tel** - sprawdzenie czy pole nie jest puste oraz czy wartość jest liczbą.

**mail** - sprawdzenie czy “@” występuje w podane frazie oraz czy nie występuje na pierwszej lub ostatniej pozycji danej frazy.

**login** - sprawdzenie czy podany login już istnieje, czy pole nie jest puste oraz czy nie występują kropki, przecinki,ukośniki itp.

**hasło** - sprawdzenie czy pole nie jest puste oraz czy hasło jest dłuższe od 5 znaków.

* + - 1. Dla tabeli sprzęt

**id\_sprzet** - jest kluczem głównym w tabeli sprzet, a więc musi być unikalny, aplikacja ignoruje wpisany klucz i nadaje   
numer według kolejności w bazie danych.

**stan\_sprzetu** - zostaje sprawdzone czy pole nie jest puste oraz czy wpisaną wartością jest “sprawny” lub “niesprawny”, jeśli w polu zostało wpisane coś innego aplikacja domyślnie wstawi słowo “sprawny”.

**id\_osoby** i **id\_lokalizacji** - sprawdzenie czy podane klucze obce istnieją w bazie danych oraz czy podane pola nie są puste.

* + - 1. Dla tabeli lokalizacja

**id\_lokalizacja**- jest kluczem głównym w tabeli lokalizacja, a więc musi być unikalny, aplikacja ignoruje wpisany klucz i nadaje   
numer według kolejności w bazie danych.

**miasto** - zostaje sprawdzone czy podane miasto nie zawiera cyfr oraz czy polę nie jest puste.

**kod\_pocztowy** - zostaje sprawdzone czy kod pocztowy ma długość 6 znaków, czy dwa pierwsze znaki to cyfry, następnie czy trzeci znak jest myślnikiem/dash’em oraz czy pozostałe trzy znaki to cyfry.

* + 1. Implementacja interfejsu dostępu do bazy danych

Do komunikacji z bazą oraz przeprowadzania na niej zaimplementowanych procedur służy klasa DataBaseFactory. Łączenie z bazą ma więc miejsce tylko w obiektach tej klasy, w momencie gdy dany obiekt jest tworzony. Połączenie odbywa się za pomocą biblioteki java.jdbc w aplikacji został zdefiniowany URL, login i hasło do bazy danych, w naszym wypadku baza danych znajduje się lokalnie więc URL odnosi się do localhost:  
 **MYSQL\_URL = "jdbc:mysql://localhost:3306/mydb"**;

a login i hasło zostaje zdefiniowane w funkcji nawiązującej połączenie  
z bazą:

**conn = (Connection)DriverManager.getConnection(MYSQL\_URL,**

**"root","admin");**

dla każdej operacji (dodawanie, wyszukiwanie czy walidacja danych) jednorazowo zostaje nawiązywane połączenie oraz wykonywane odpowiednie zapytanie np.:

**statement = conn.createStatement();**

**resultSet = statement.executeQuery(query)**

lub

**statement = conn.createStatement();**

**statement.executeUpdate(query);**

Jeżeli z jakiegoś powodu nie uda się nawiązać połączenia z bazą aplikacja będzie działać dalej, ponieważ wyjątki tego typu są obsłużone.

* + 1. Implementacja wybranych funkcjonalności systemu
       1. Dodawanie danych

W momencie wciśnięcia przycisku Add, z pól tekstowych pod tabelą w aktualnie wybranej zakładce, wczytywane są dane do dodania. Następnie dane te oraz informacja o wybranej tabeli (zakładce) jest przekazywana do funkcji odpowiedzialnej za dodawanie danych do bazy. W pierwszej kolejności funkcja łączy się z bazą w celu automatycznego nadania dodawanej przez nas krotce danych id, niezależnie od tego jakie id wpisaliśmy. Następnie dla wszystkich wczytanych wartości następuje walidacja danych. Jeżeli któraś z danych jest wprowadzona niepoprawnie wyświetlony zostanie odpowiedni komunikat, który informuje o typie błędu. Jeżeli wszystkie podane wartości są poprawne, tworzona jest transakcja: INSERT INTO table\_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...); i jest ona przeprowadzana na bazie danych.

* + - 1. Przeglądanie widoków

Po zalogowaniu, gdy otwiera się główne okno programu nawiązywane jest połączenie z bazą w celu zaktualizowania listy widoków. Lista nazw widoków jest pobierana z bazy po wysłaniu do niej zapytania: SELECT TABLE\_NAME FROM information\_schema.tables WHERE TABLE\_TYPE LIKE 'VIEW' AND TABLE\_SCHEMA LIKE 'mydb';. Lista ta pozwala stworzyć listę widoków dodając kolejne checkboxy z nazwami widoków. Żeby jednak wyświetlić wybrany widok w tabeli musimy znać nazwy kolumn w danym widoku. Informacje o nazwach kolumn widoku wyciągamy z bazy za pomocą zapytania: SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'viewName';, gdzie viewName jest nazwą widoku, którego nazwy kolumn chcemy poznać. Na koniec należy jeszcze tylko wyciągnąć dane z widoku i wstawić do tabeli. Dane zawarte w widoku wyciągamy z bazy za pomocą zapytania: SELECT \* FROM view;. Przy każdej zmianie widoku powtarzane są dwa ostatnie kroki czyli pobieranie nazw kolumn nowo wybranego widoku i danych w nim zawartych.

* + 1. Implementacja mechanizmów bezpieczeństwa

Podczas logowania, w momencie wciśnięcia przycisku Sign In pobierane są podane przez użytkownika login i hasło. Następnie nawiązywane jest połączenie z bazą. W bazie wyszukiwane jest hasło przypisane dla wprowadzonego loginu. Jeżeli hasło zapisane w bazie różni się od podanego przez użytkownika lub podany login nie znajduje się w bazie, dostęp do aplikacje nie jest przydzielany co okraszone jest odpowiednim komunikatem, widocznym w oknie logowania.

1. Testowanie systemu
   1. Instalacja i konfigurowanie systemu

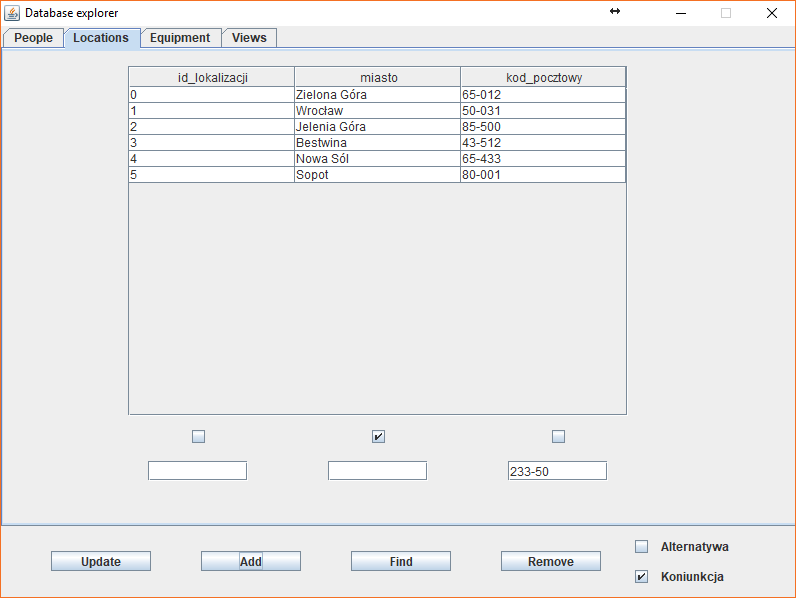
Należy zainstalować/posiadać wirtualną maszynę javy   
do uruchomienia pliku .jar oraz dostęp do serwera bazy danych.

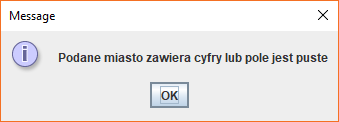
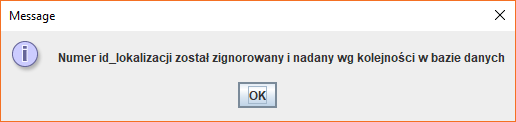
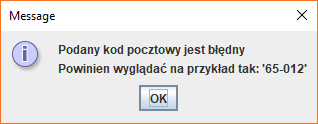
* 1. Testowanie opracowanych funkcji systemu

1. 2. 1. Testowanie funkcji addData(String table, String[] dataToAdd)

Kiedy pola są puste lub nieprawidłowo wpisane:

Rysunek 12: Testowanie funkcji addData()(1)



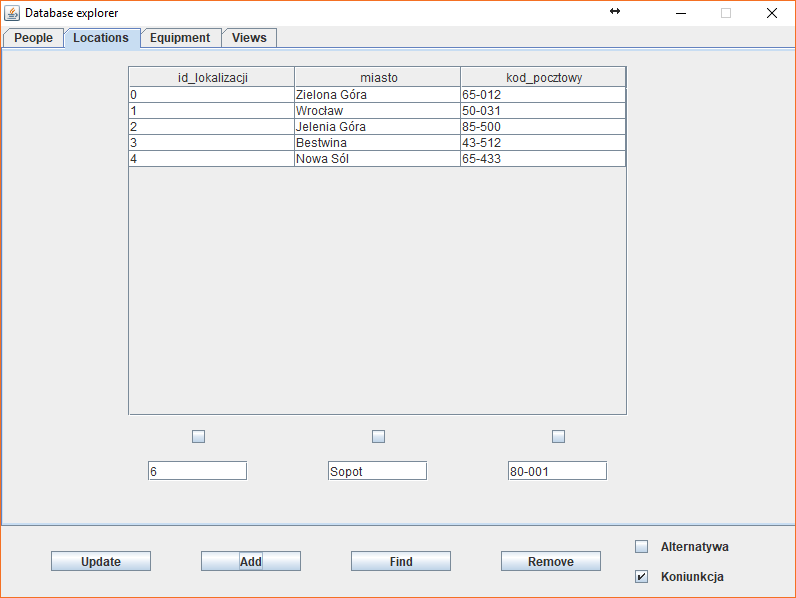
Zostają wywołane powiadomienia :

Rysunek 14: Testowanie funkcji addData()(3)

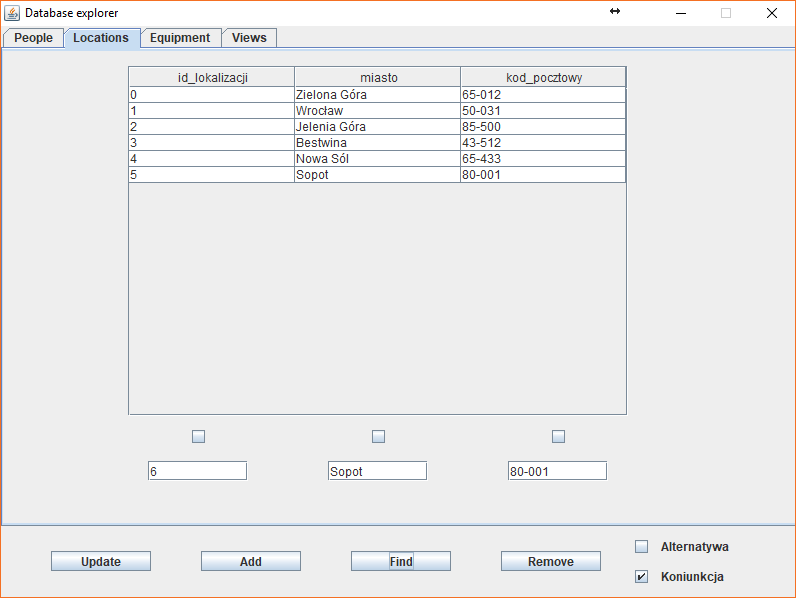
Rysunek 13: Testowanie funkcji addData()(4)

Rysunek 15: Testowanie funkcji addData()(2)

Co spowoduje, że podane dane nie zostaną dodane do bazy danych, a powiadomienia na kierują użytkownika co powinien poprawić podczas podawania danych.

Jeśli dane zostały podane prawidłowo:

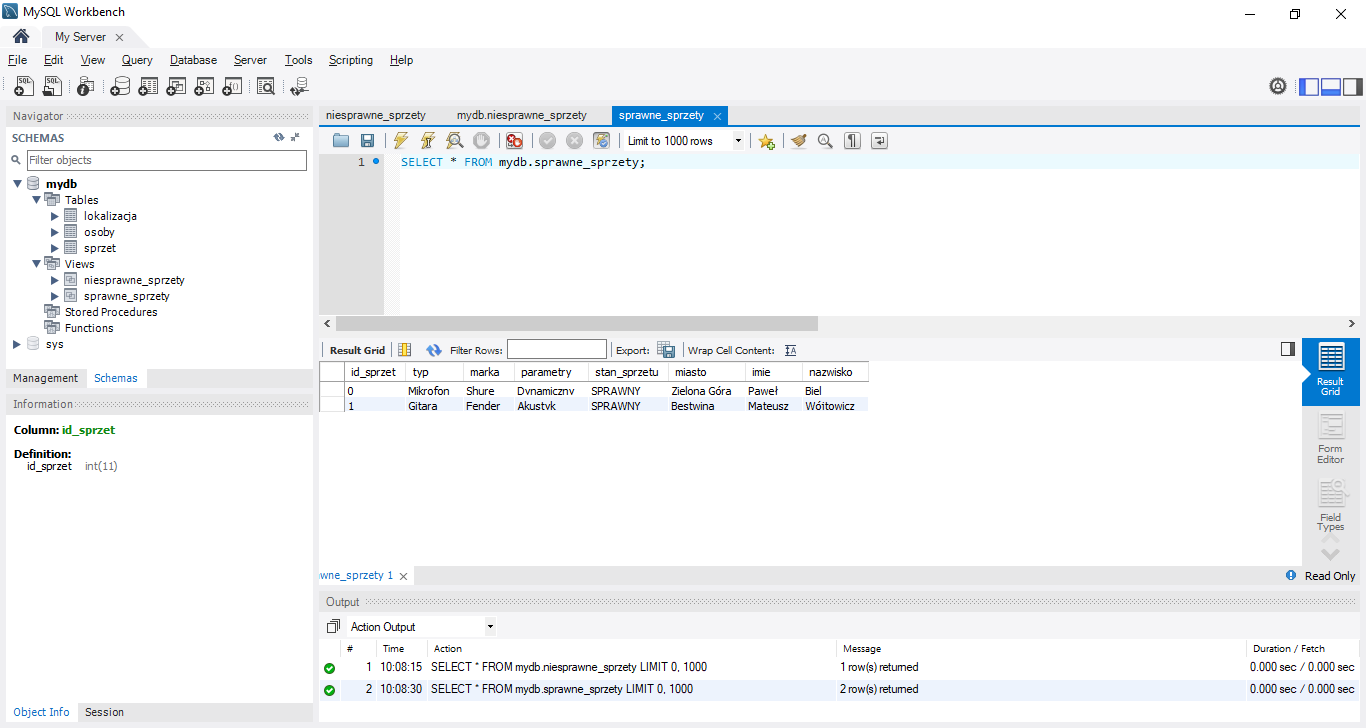
Rysunek 16: Testowanie funkcji addData()(5)

Można zauważyć wynik:

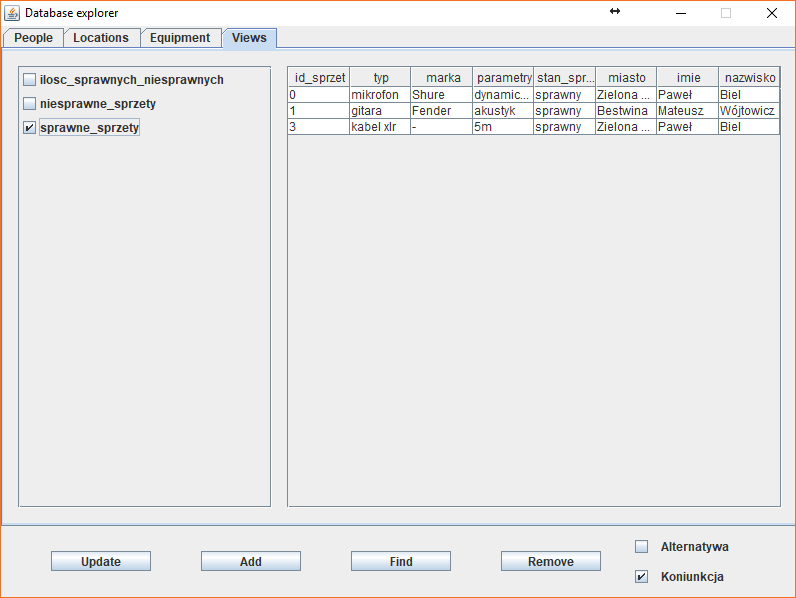
Rysunek 17: Testowanie funkcji addData()(6)

* + 1. Testowanie funkcji getViewsNames()

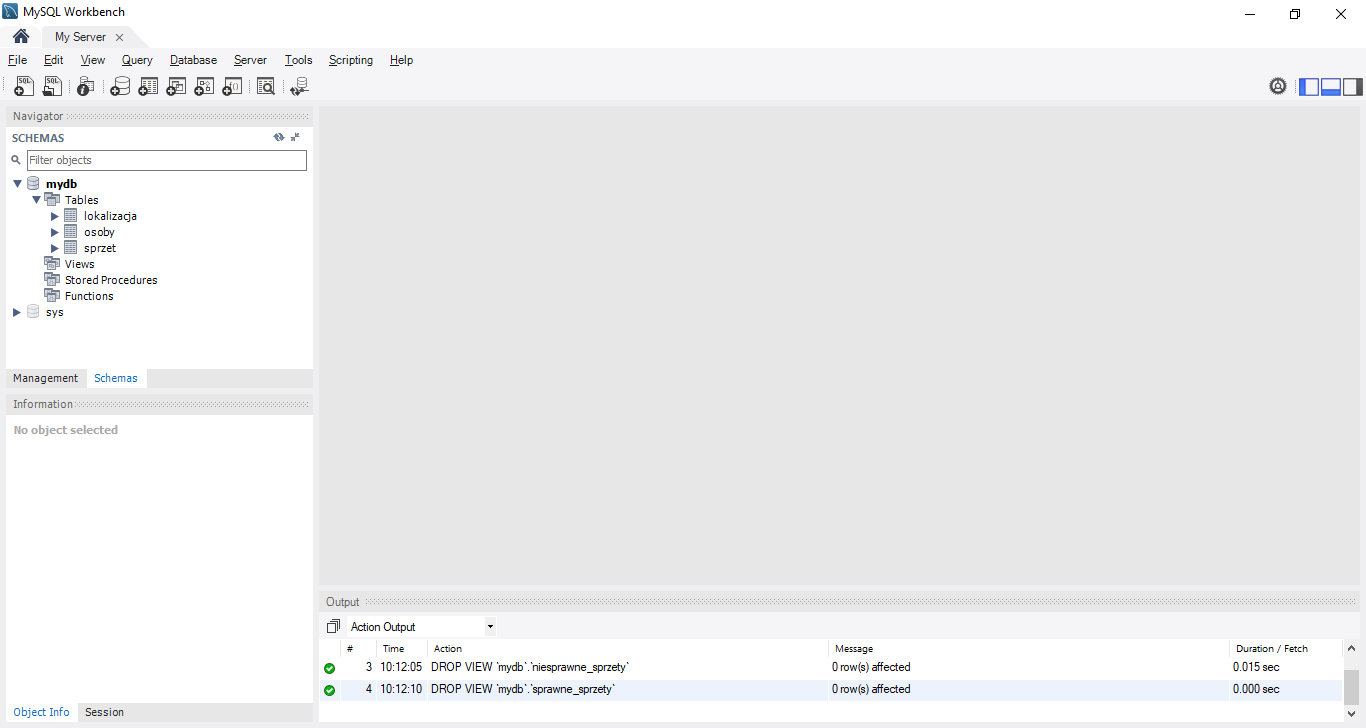
Przypadek kiedy w bazie danych są stworzone widoki:

Aplikacja pobierze i wyświetli zaznaczone widoki:

Rysunek 18: Testowanie funkcji getViewsNames()(1)

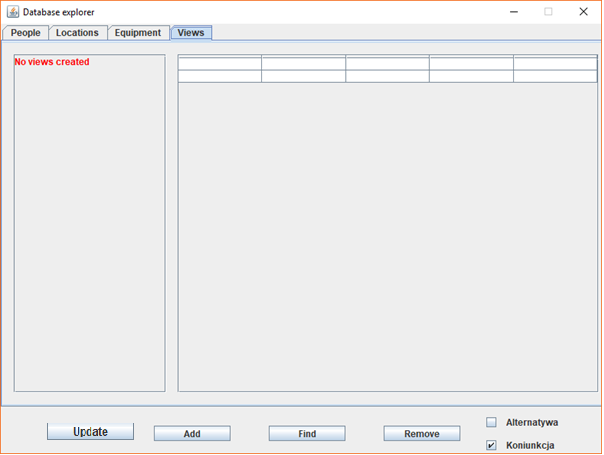


Rysunek 19: Testowanie funkcji getViewsNames()(2)

Jeśli w bazie danych nie będzie żadnych widoków:

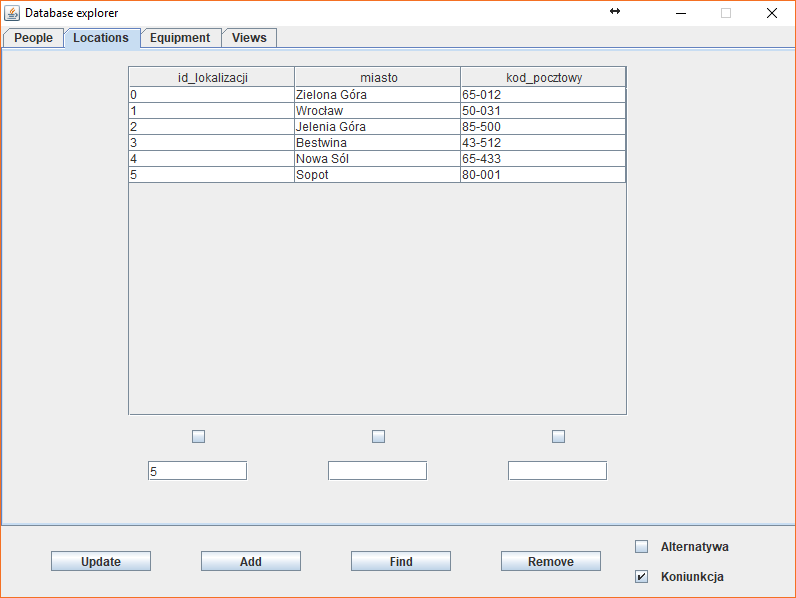
Rysunek 20: Testowanie funkcji getViewsNames()(3)

to aplikacja obsłuży wyjątek i nie wyświetli widoków:



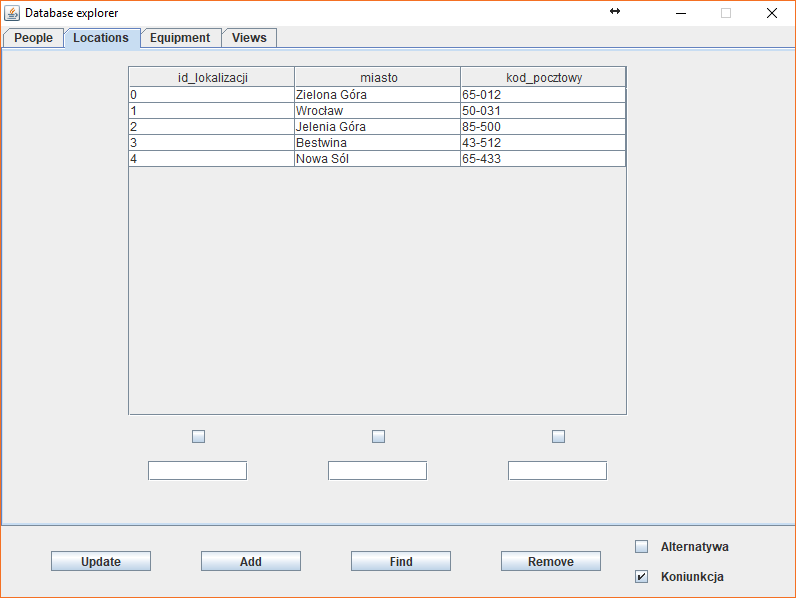
Rysunek 21: Testowanie funkcji getViewsNames()(4)

* + 1. Test funkcji removeDate()

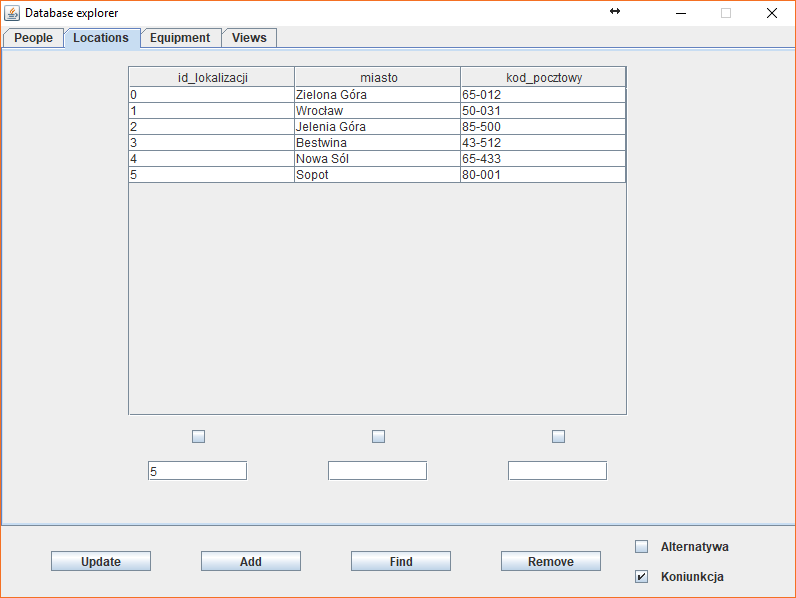


Rysunek 22: Testowanie funkcji removeDate()(1)

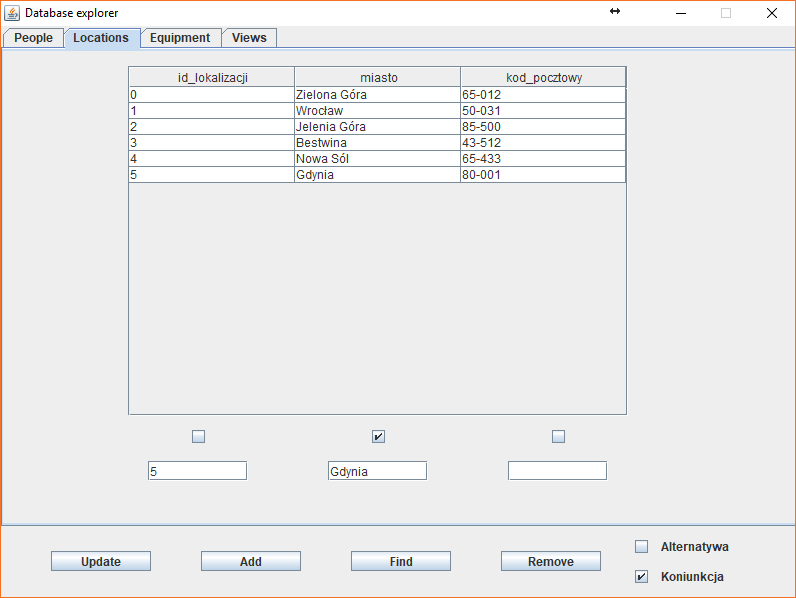




Rysunek 23: Testowanie funkcji removeDate()(2)

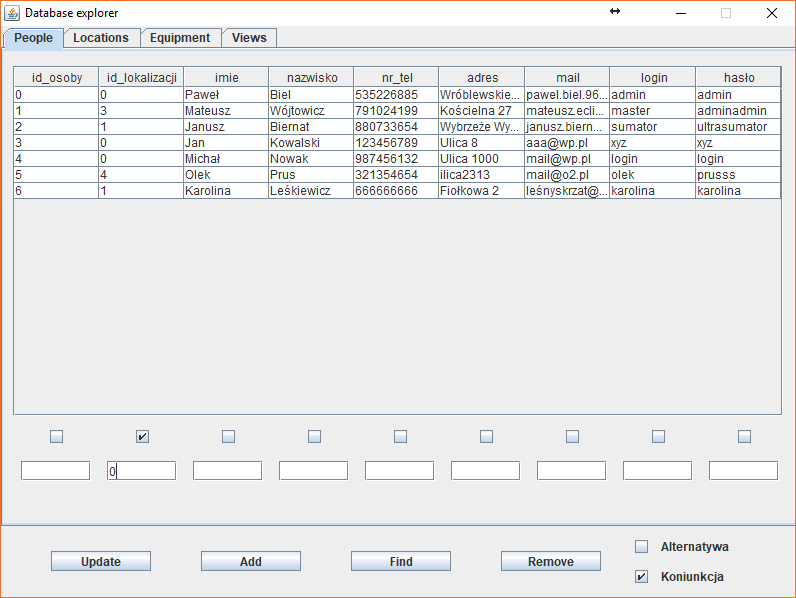
* + 1. Testy funkcji updateDate()

Rysunek 24: Testowanie funkcji updateDate()(1)

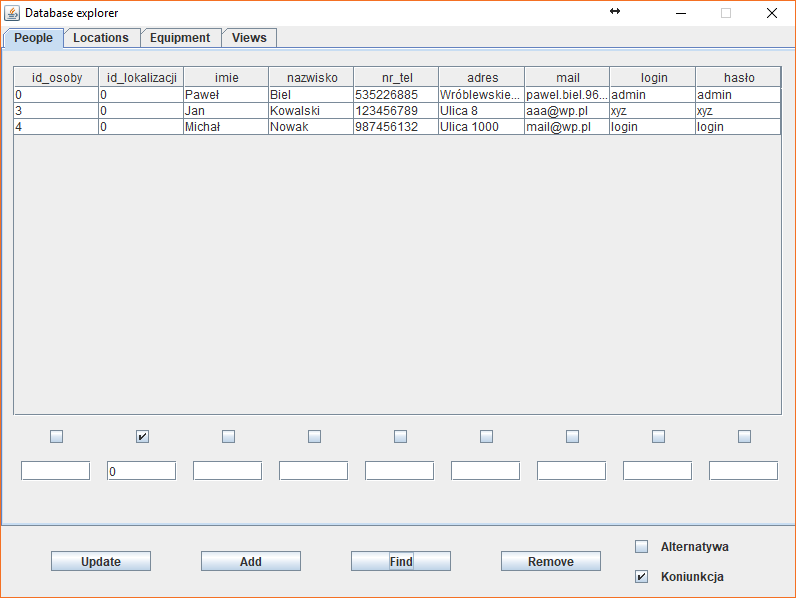


Rysunek 25: Testowanie funkcji updateDate()(2)

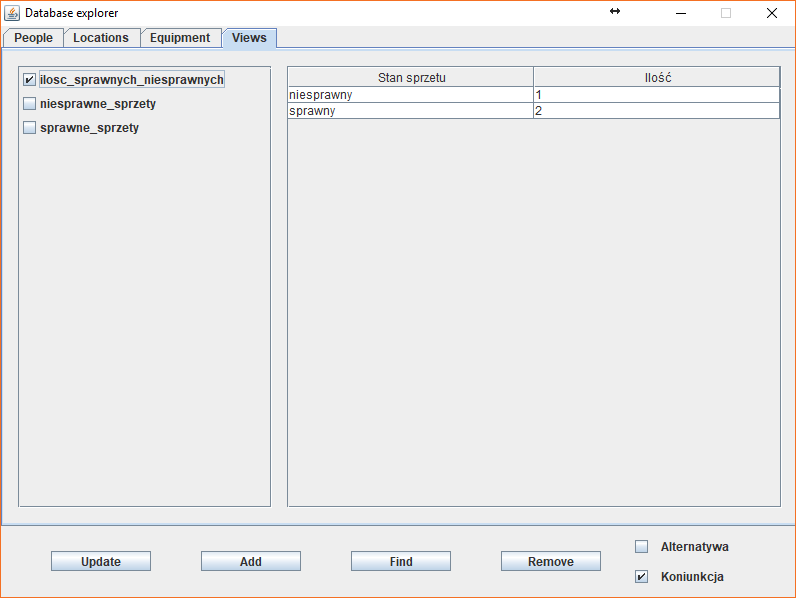
* + 1. Testy funkcji getDate()



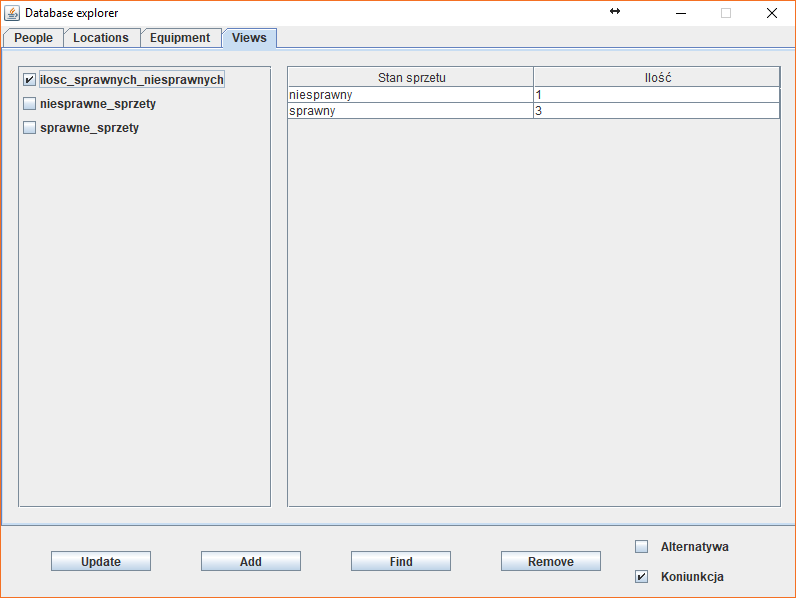
Rysunek 26: Testowanie funkcji getDate()(1)



Rysunek 27: Testowanie funkcji getDate()(2)

* + 1. Testowanie widoku automatycznego zliczania sprzętów sprawnych i niesprawnych

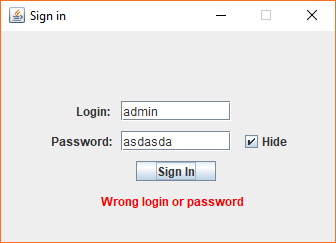
Rysunek 28: Testowanie funkcji zliczania sprzętu(1)



Rysunek 29: Testowanie funkcji zliczania sprzętu(2)

* 1. Testowanie mechanizmów bezpieczeństwa

Testowanie mechanizmu logowania do aplikacji:

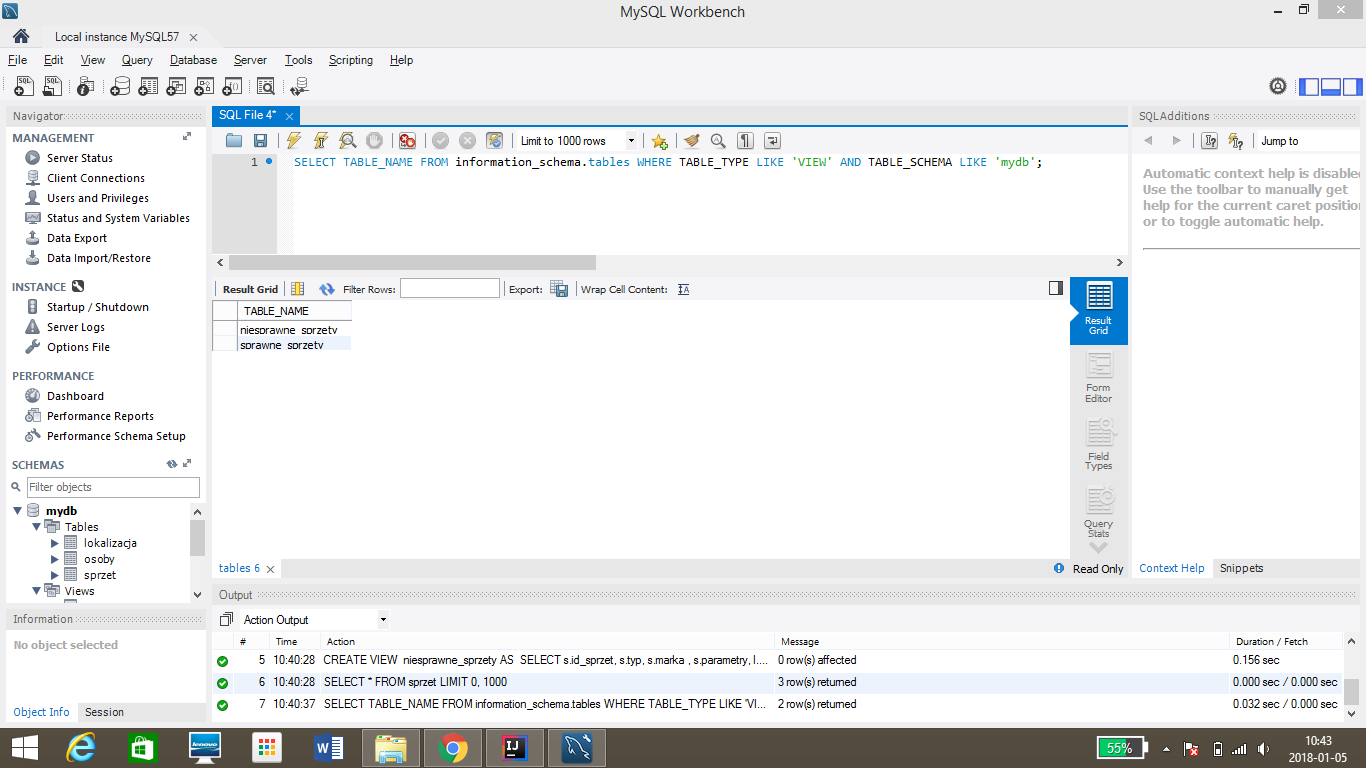
Przypadek, gdy podamy złe hasło lub login lub login i hasło:

Rysunek 30: Testowanie mechanizmów bezpieczeństwa

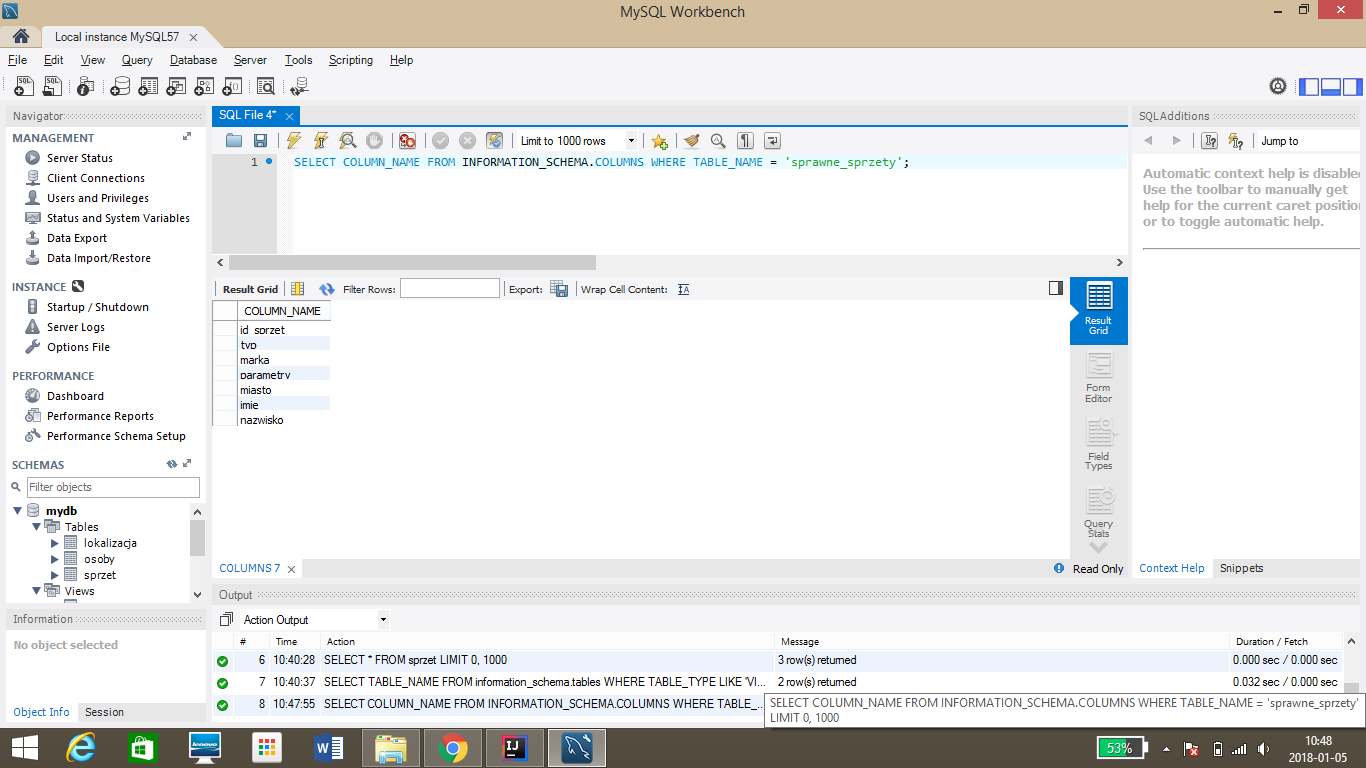
W przypadku prawidłowo podanego loginu i hasła aplikacja wpuści nas do panelu głównego aplikacji.

* 1. Inne testy

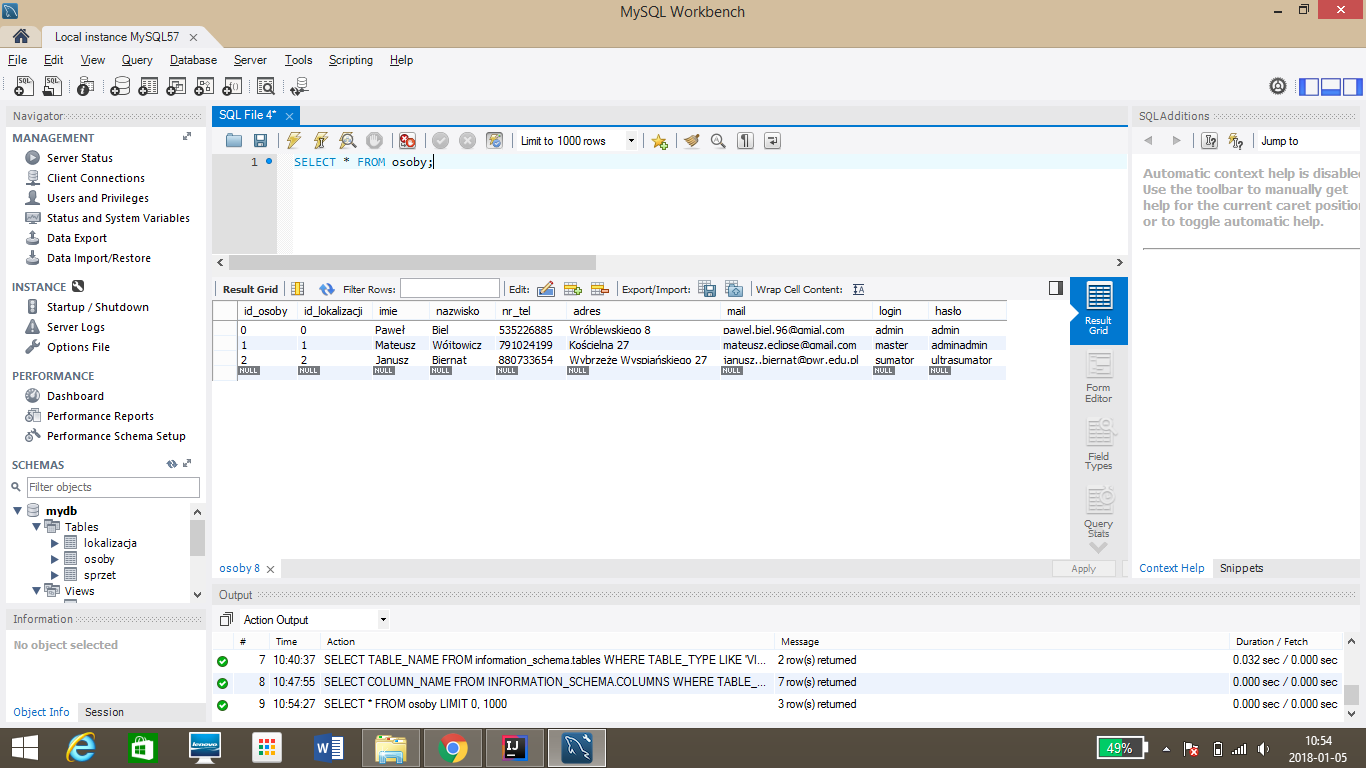
Poniżej przedstawione są testy zapytań wykorzystywanych przez funkcje aplikacji.



Rysunek 31: Inne testy(1)



Rysunek 32: Inne testy(2)



Rysunek 33: Inne testy(3)

* 1. Wnioski z testów

Testy aplikacji wykazały, że:

* aplikacja jest stabilna
* działająca prawidłowo walidacja danych chroni przed wprowadzeniem niepożądanych danych
* wszystkie funkcjonalności, które zostały zaplanowane działają prawidłowo
* aplikacja jest zabezpieczona przed niepożądanymi użytkownikami

1. Podsumowanie

Udało się stworzyć oprogramowanie realizujące założenia wstępne projektu, pomyślnie przechodzące testy funkcjonalności. Program działa stabilnie, jest dobrze zabezpieczony, a mechanizmy walidacji danych chronią logikę systemu. Efektem prac jest aplikacja wraz z bazą danych oraz niniejsza dokumentacja.

1. Literatura

[1] Górski J.,Inżynieria oprogramowania w projekcie informatycznym, Mikom,   
Warszawa, 2000

[2] Kathy S., Bert B., Head First Java, O’REILLY 2003

[3] MySQL, <https://dev.mysql.com/doc/>