

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:

Spis treści

Spis rysunków	3
1. Wstęp	4
1.1. Cel projektu	4
1.2. Zakres Projektu	4
2. Analiza Wymagań	5
2.1. Opis działania i schemat logiczny	5
2.2.1. Diagram przypadków użycia	6
2.2.2. Scenariusze wybranych przypadków użycia	6
2.2. Wymagania niefunkcjonalne	8
2.3.1. Wykorzystywane technologie i narzędzia	8
2.3.2. Wymagania dotyczące rozmiaru bazy danych	8
2.3.3. Wymagania dotyczące bezpieczeństwa systemu	8
2.3. Przyjęte założenia projektowe	8
3. Projekt systemu	10
3.1. Projekt bazy danych	10
3.1.1. Analiza rzeczywistości i uproszczony model konceptualny	10
3.1.2. Model logiczny i normalizacja	10
3.1.3. Model fizyczny i ograniczenia integralności danych	11
3.1.4. Inne elementy Schematu - mechanizmy przetwarzania danych	12
3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych	12
3.2. Projekt aplikacji użytkownika	12
3.2.1. Architektura aplikacji i diagramy projektowe	12
3.2.2. Interfejs graficzny i struktura menu	13
3.2.3. Projekt wybranych funkcji systemu	15
3.2.4. Metoda podłączenia do bazy danych - integracja z bazą danych	18
3.2.5. Projekt zabezpieczeń na poziomie aplikacji	18
4. Implementacja Systemu	19
4.1. Realizacja bazy danych	19
4.1.1. Tworzenie i definiowanie ograniczeń	19
4.1.2. Implementacja przetwarzania danych	20
4.1.2.1. Widok sprzętów sprawnych	20
4.1.2.2. Widok sprzętów niesprawnych	21
4.1.3. Implementacja uprawnień i innych zabezpieczeń	21

4.2.	Realizacja elementów aplikacji	22
4.2.1.	Obsługa menu	22
4.2.2.	Walidacja i filtracja	23
4.2.2.1.	Dla okna osoby	23
4.2.2.2.	Dla tabeli sprzęt	23
4.2.2.3.	Dla tabeli lokalizacja	24
4.2.3.	Implementacja interfejsu dostępu do bazy danych	24
4.2.4.	Implementacja wybranych funkcjonalności systemu	25
4.2.4.1.	Dodawanie danych	25
4.2.4.2.	Przeglądanie widoków	25
4.2.5.	Implementacja mechanizmów bezpieczeństwa	26
5.	Testowanie systemu	27
5.1.	Instalacja i konfigurowanie systemu	27
5.2.	Testowanie opracowanych funkcji systemu	27
5.2.1.	Testowanie funkcji addData(String table, String[] dataToAdd)	27
5.2.2.	Testowanie funkcji getViewsNames()	30
5.2.1.	Test funkcji removeData()	32
5.2.2.	Testy funkcji upDataData()	33
5.2.3.	Testy funkcji getData()	34
5.2.4.	Testowanie widoku automatycznego zliczania sprzętów sprawnych i niesprawnych	35
5.3.	Testowanie mechanizmów bezpieczeństwa	36
5.4.	Inne testy	36
5.5.	Wnioski z testów	38
6.	Podsumowanie	39
	Literatura	40

Spis rysunków

Rysunek 1: Schemat logiczny aplikacji.....	5
Rysunek 2: Diagram przypadków użycia.....	6
Rysunek 3: Model logiczny bazy danych.....	10
Rysunek 4: Model fizyczny bazy danych.....	11
Rysunek 5: Okno logowania aplikacji.....	13
Rysunek 7: Widok tabeli "Lokalizacja".....	14
Rysunek 6: Widok tabeli "Sprzęt".....	14
Rysunek 8: Widok tabeli "Osoby" w aplikacji.....	15
Rysunek 9: Widok widoków w aplikacji.....	15
Rysunek 10: Diagram Funkcji logowania do systemu.....	16
Rysunek 11: Diagram wyszukiwania danych w bazie.....	17
Rysunek 12: Testowanie funkcji addData()(1).....	27
Rysunek 14: Testowanie funkcji addData()(3).....	28
Rysunek 13: Testowanie funkcji addData()(4).....	28
Rysunek 15: Testowanie funkcji addData()(2).....	28
Rysunek 16: Testowanie funkcji addData()(5).....	29
Rysunek 17: Testowanie funkcji addData()(6).....	29
Rysunek 18: Testowanie funkcji getViewsNames()(1).....	30
Rysunek 19: Testowanie funkcji getViewsNames()(2).....	30
Rysunek 20: Testowanie funkcji getViewsNames()(3).....	31
Rysunek 21: Testowanie funkcji getViewsNames()(4).....	31
Rysunek 23: Testowanie funkcji removeData()(2).....	32
Rysunek 24: Testowanie funkcji upDataData()(1).....	33
Rysunek 25: Testowanie funkcji upDataData()(2).....	33
Rysunek 26: Testowanie funkcji getData()(1).....	34
Rysunek 27: Testowanie funkcji getData()(2).....	34
Rysunek 28: Testowanie funkcji zliczania sprzętu(1).....	35
Rysunek 29: Testowanie funkcji zliczania sprzętu(2).....	35
Rysunek 30: Testowanie mechanizmów bezpieczeństwa.....	36
Rysunek 31: Inne testy(1).....	36
Rysunek 32: Inne testy(2).....	37
Rysunek 33: Inne testy(3).....	37

1. Wstęp

1.1. Cel projektu

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

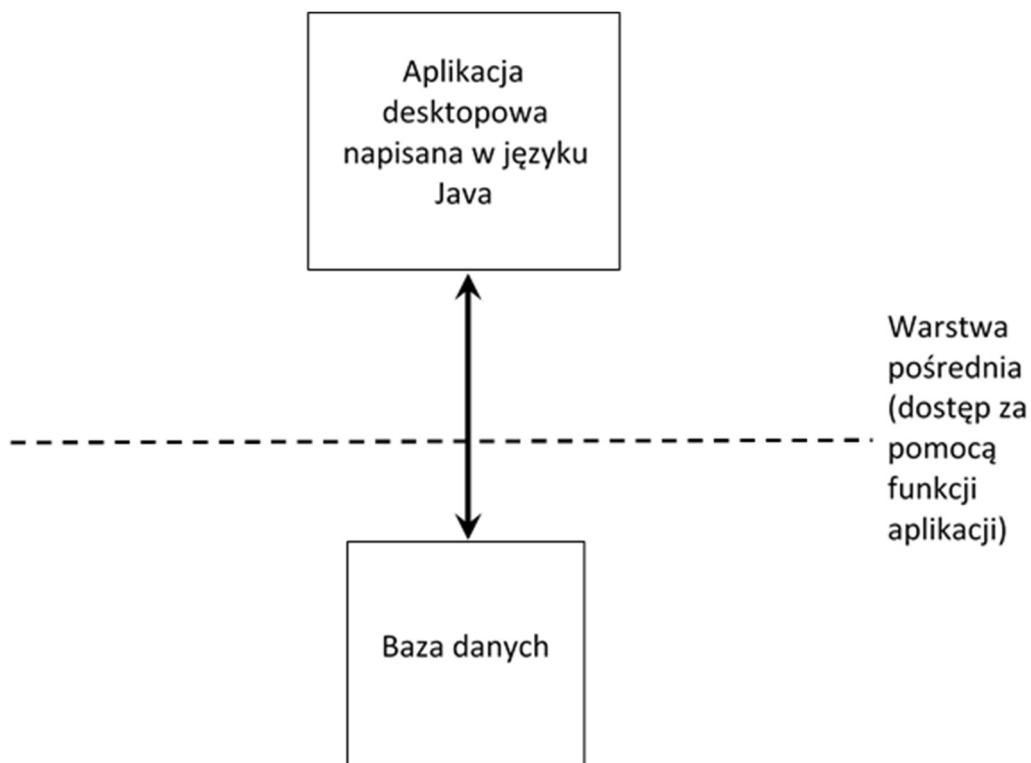
1.2. Zakres Projektu

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.

2. Analiza Wymagań

2.1. Opis działania i schemat logiczny

System będzie umożliwiał 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



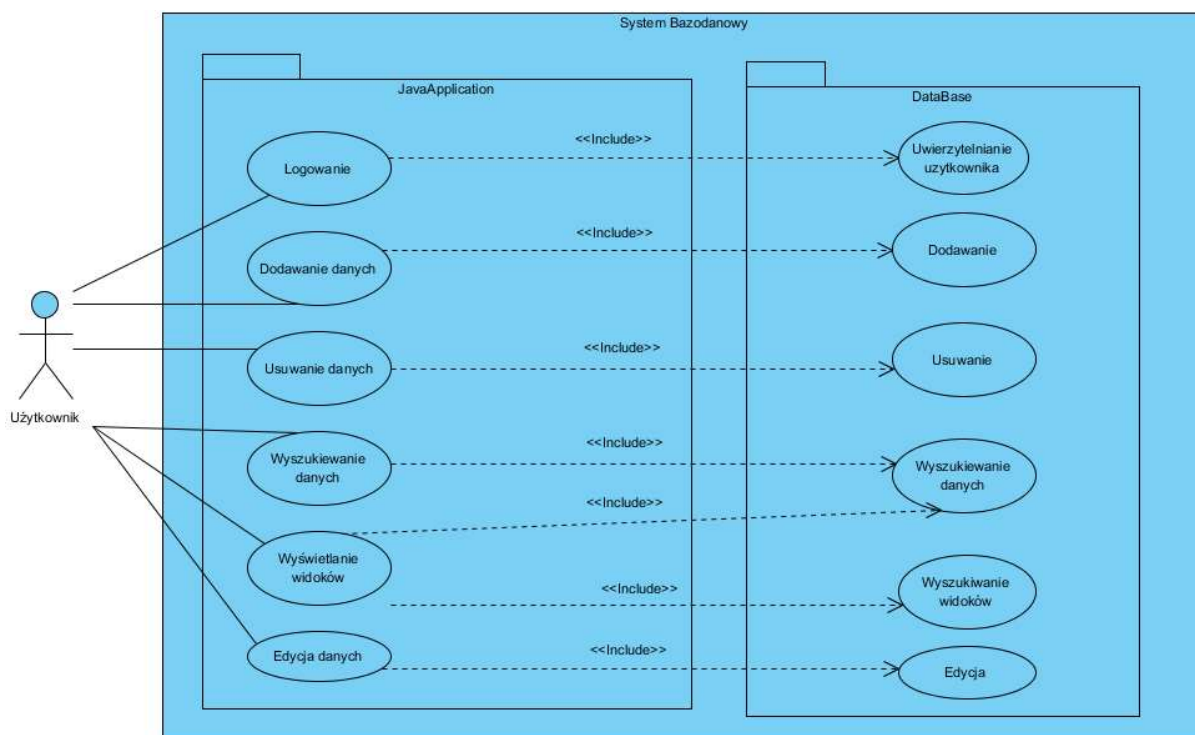
w bazie widoki.

Wymagania funkcjonalne

- Możliwość wyświetlenia danych wyszukanych w bazie
- Możliwość dodawania oraz usuwania danych
- Rozróżnianie użytkowników korzystających z systemu

Rysunek 1: Schemat logiczny aplikacji

2.2.1. Diagram przypadków użycia



2.2.2. Scenariusze wybranych przypadków użycia

PU - przypadek użycia

Rysunek 2: Diagram przypadków 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 pozycję do bazy.
2. Przeprowadzenie transakcji na bazie danych.

2.2. 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.

2.3.1. 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.

2.3.2. 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.

2.3.3. 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.

2.3. 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 (obiektywne 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.

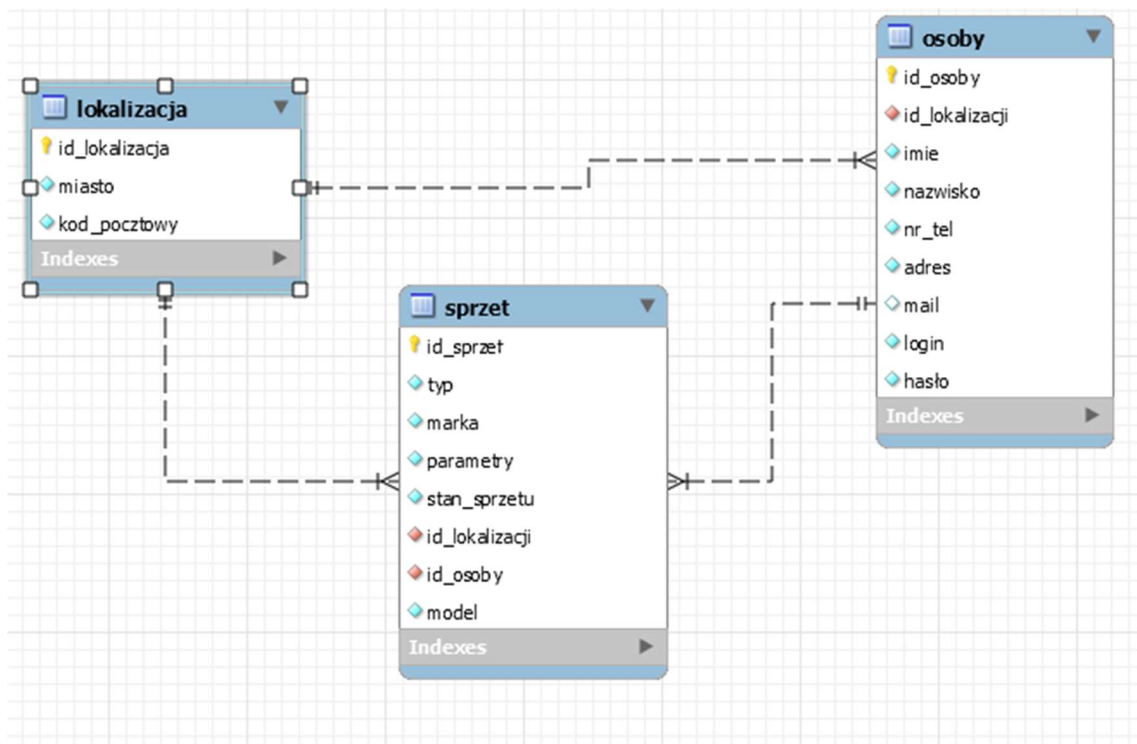
3. Projekt systemu

3.1. Projekt bazy danych

3.1.1. Analiza rzeczywistości i uproszczony model konceptualny

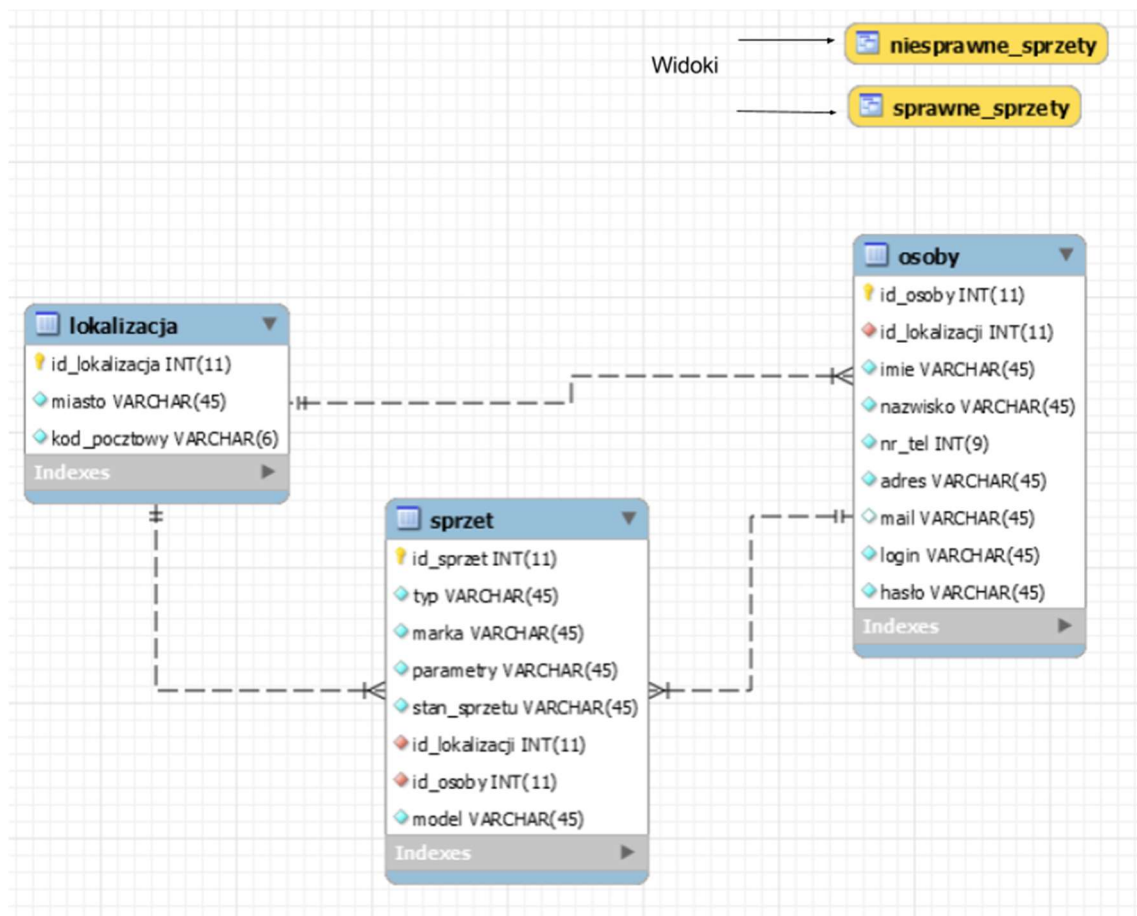
W bazie powinny się znajdować trzy podstawowe tabele: tabela użytkowników, tabela lokalizacji oraz tabela sprzętów. Pierwsza z nich zawierać będzie podstawowe informacje o użytkownikach, czyli członkach zespołu, a także o przypisanych do nich loginach i hasłach. Sprzęt muzyczny, który posiada zespół 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.

3.1.2. Model logiczny i normalizacja



Rysunek 3: Model logiczny bazy danych

3.1.3. Model fizyczny i ograniczenia integralności danych



Rysunek 4: Model fizyczny bazy danych

3.1.4. 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_sprzety` 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';
```

3.1.5. 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".

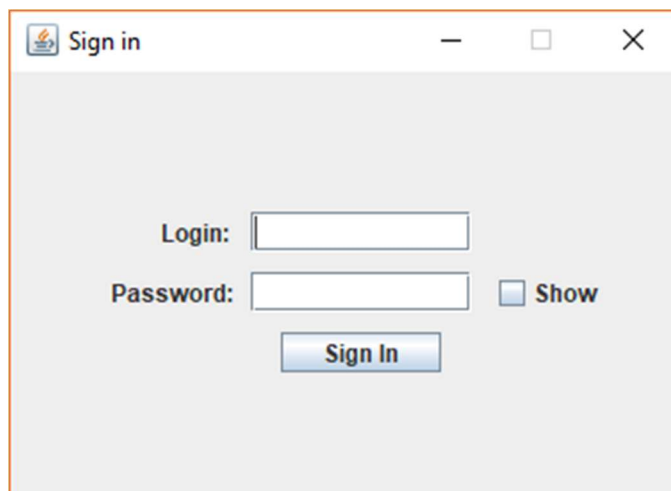
3.2. Projekt aplikacji użytkownika

3.2.1. 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 oddzielną klasę. Stworzona została również oddzielna klasa służąca do przeprowadzania wszelkich procedur bezpośrednio związanych z bazą danych. Językiem aplikacji jest język angielski.

3.2.2. Interfejs graficzny i struktura menu

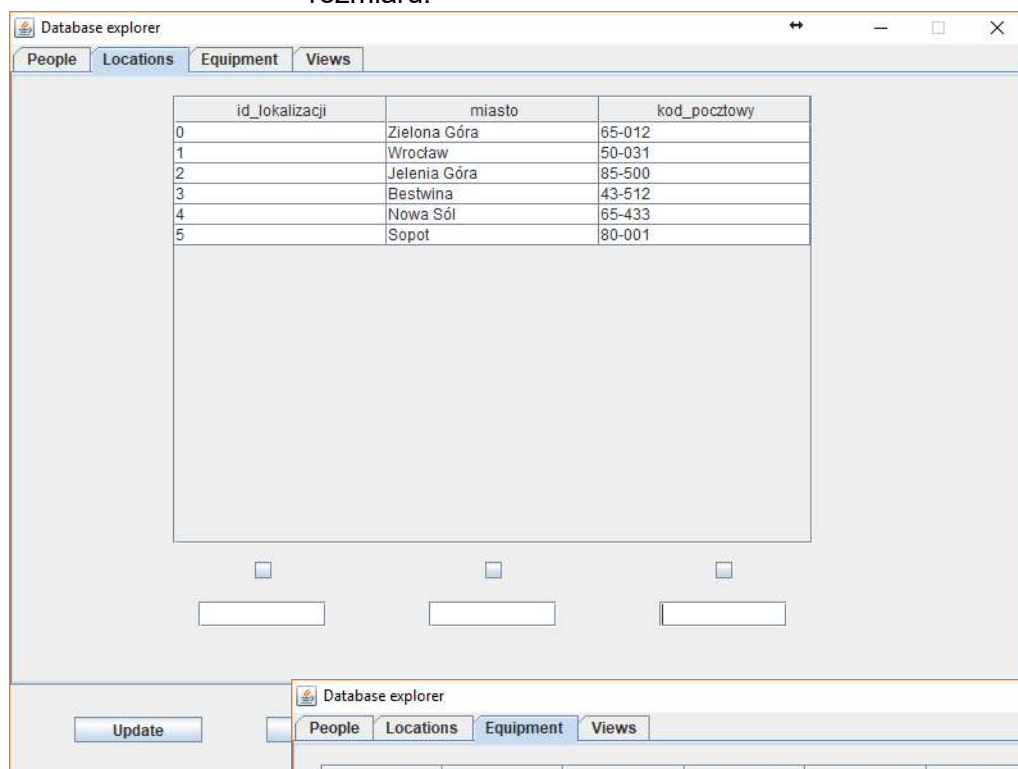
Po uruchomieniu programu pojawia się okienko logowania, którego rozmiaru nie można zmieniać. Widoczne są w nim dwa pola tekstowe: górne i dolne, w które użytkownik wpisuje kolejno swój login i hasło. Znajdujący się obok drugiego pola tekstowego checkbox pozwala odsłonić wpisywane hasło, które domyślnie 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

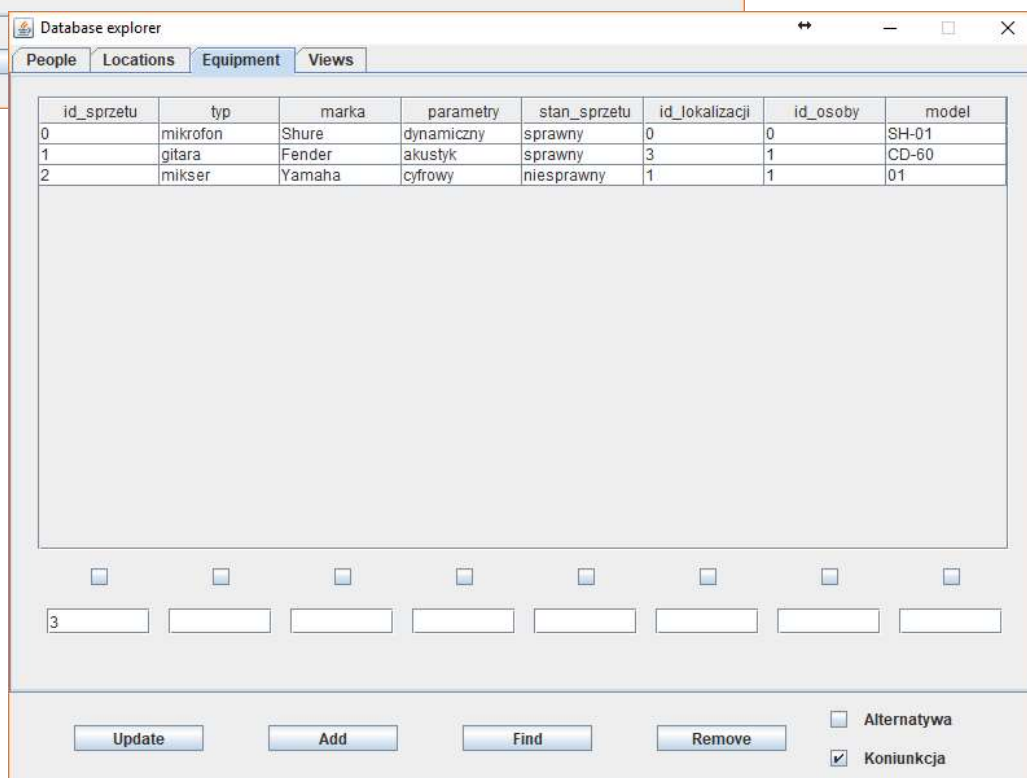
Główne okno programu jest podzielone na dwie części. Część z zakładkami, dzięki której możemy przełączać się między tabelami oraz otworzyć sekcje przeglądania widoków oraz część z przyciskami służącą do przeprowadzania operacji na bazie. W pierwszej części mamy cztery zakładki. Pierwsze trzy odpowiadają 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 6: Widok tabeli "Lokalizacja" w aplikacji

Rysunek 7: Widok tabeli "Sprzęt" w aplikacji



3.2.3. Projekt wybranych funkcji systemu

Database explorer

People Locations Equipment Views

id_osoby	id_lokalizacji	imie	nazwisko	nr_tel	adres	mail	login	haslo
0	0	Paweł	Biel	535226885	Wróblewskie...	pawel.biel.96...	admin	admin
1	3	Mateusz	Wójtowicz	791024199	Kościelna 27	mateusz.eci...	master	adminadmin
2	1	Janusz	Biernat	880733654	Wybrzeże Wy...	janusz.biern...	sumator	ultrasumator
3	0	Jan	Kowalski	123456789	Ulica 8	aaa@wp.pl	xyz	xyz
4	0	Michał	Nowak	987456132	Ulica 1000	mail@wp.pl	login	login
5	4	Olek	Prus	321354654	ilica2313	mail@o2.pl	olek	prusss
6	1	Karolina	Leśkiewicz	666666666	Fiołkowa 2	leśnyskrzat@...	karolina	karolina

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Update Add Find Remove ☐ Alternatywa ☒ Koniunkcja

Rysunek 8: Widok tabeli "Osoby" w aplikacji

Database explorer

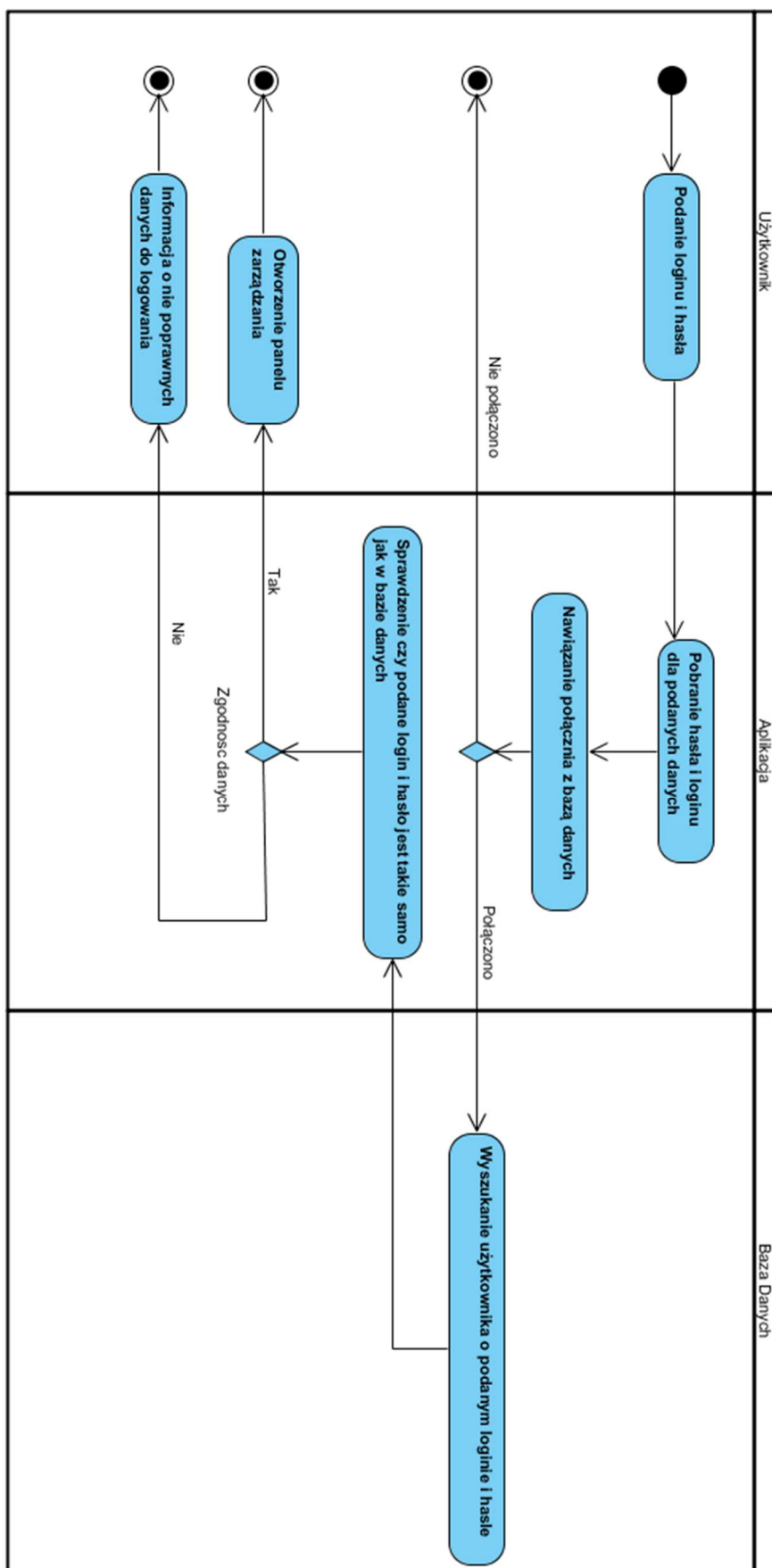
People Locations Equipment Views

☐ ilosc_sprawnych_niesprawnych
☐ niesprawne_sprzety
☒ sprawne_sprzety

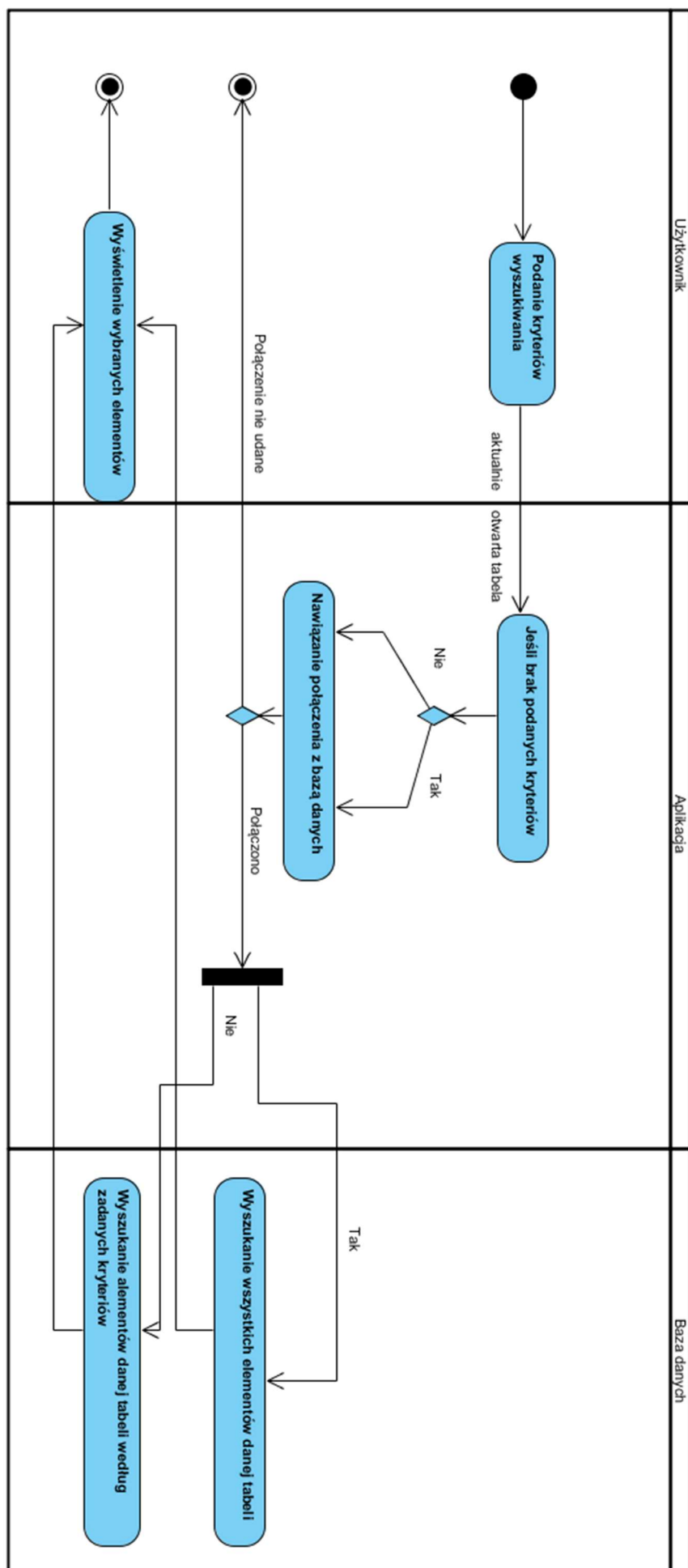
id_sprzet	typ	marka	parametry	stan_spr...	miasto	imie	nazwisko
0	mikrofon	Shure	dynamic...	sprawny	Zielona ...	Paweł	Biel
1	gitara	Fender	akustyk	sprawny	Bestwina	Mateusz	Wójtowicz

Update Add Find Remove ☐ Alternatywa ☒ Koniunkcja

Rysunek 9: Widok widoków w aplikacji



Rysunek 10: Diagram Funkcji logowania do systemu



Rysunek 11: Diagram wyszukiwania danych w bazie

3.2.4. 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`.

3.2.5. Projekt zabezpieczeń na poziomie aplikacji

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

4. Implementacja Systemu

4.1. Realizacja bazy danych

4.1.1. Tworzenie i definiowanie ograniczeń

Tworzenie tabel (tabele zostały wygenerowane za pomocą MySQL Workbench)

4.1.1.1. 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'
```

4.1.1.2. 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
```

4.1.1.3. 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,  
  `haslo` 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
```

4.1.2. Implementacja przetwarzania danych

4.1.2.1. 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')
```

4.1.2.2. 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')
```

4.1.3. 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.

4.2. Realizacja elementów aplikacji

4.2.1. 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.

4.2.2. Walidacja i filtracja

4.2.2.1. 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 pole nie jest puste.

nazwisko - sprawdzenie czy pole nie jest puste.

nr_tel - sprawdzenie czy pole nie jest puste oraz czy wartość jest liczbą.

mail - sprawdzenie czy "@" występuje w podanej 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.

4.2.2.2. Dla tabeli sprzęt

id_sprzet - jest kluczem głównym w tabeli sprzęt, 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.

4.2.2.3. 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 pole 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.

4.2.3. 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 obsługiwane.

4.2.4. Implementacja wybranych funkcjonalności systemu

4.2.4.1. 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.

4.2.4.2. 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.

4.2.5. 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 aplikacji nie jest przydzielany co okraszone jest odpowiednim komunikatem, widocznym w oknie logowania.

5. Testowanie systemu

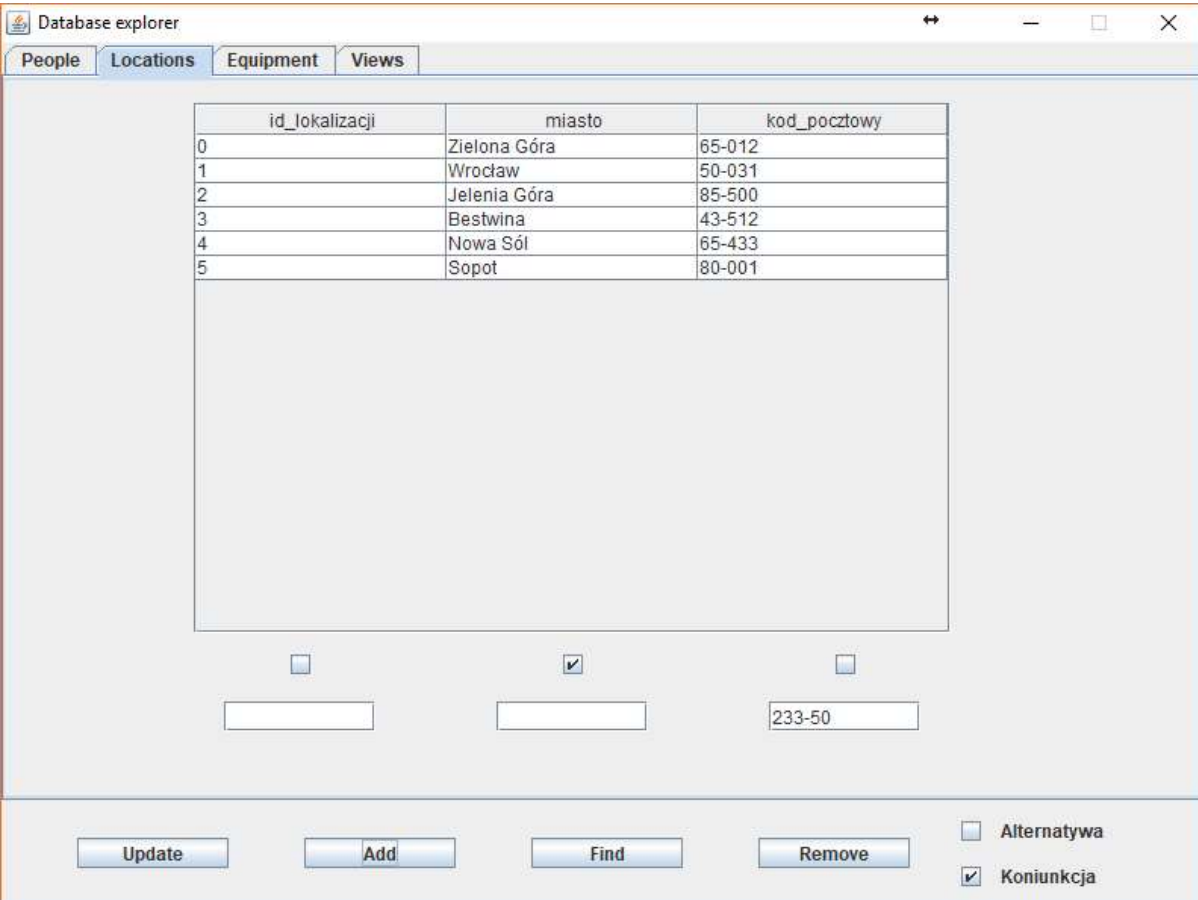
5.1. Instalacja i konfigurowanie systemu

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

5.2. Testowanie opracowanych funkcji systemu

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

Kiedy pola są puste lub nieprawidłowo wpisane:



id_lokalizacji	miasto	kod_pocztowy
0	Zielona Góra	65-012
1	Wrocław	50-031
2	Jelenia Góra	85-500
3	Bestwina	43-512
4	Nowa Sól	65-433
5	Sopot	80-001

☐

☒

☐

233-50

Update

Add

Find

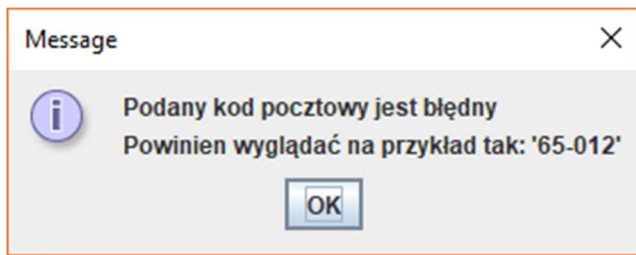
Remove

☐ Alternatywa

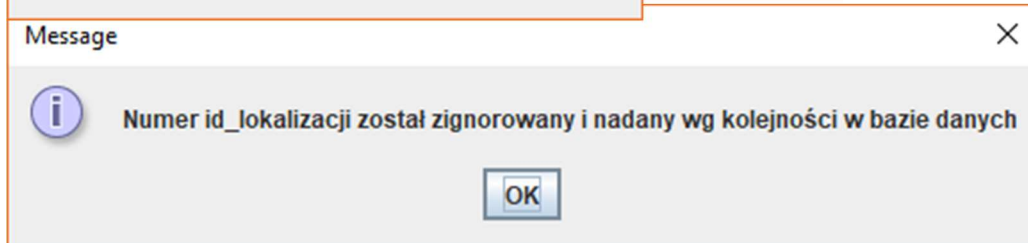
☒ Koniunkcja

Rysunek 12: Testowanie funkcji addData()(1)

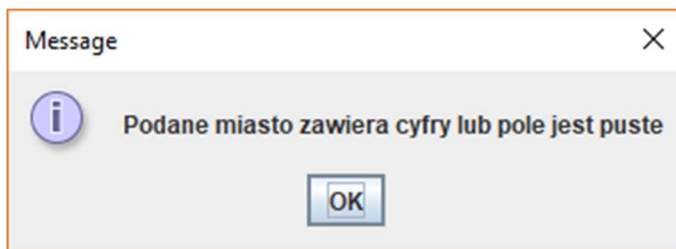
Zostają wywołane powiadomienia :



Rysunek 15: Testowanie funkcji addData()(2)



Rysunek 13: Testowanie funkcji addData()(3)



Rysunek 14: Testowanie funkcji addData()(4)

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:

id_lokalizacji	miasto	kod_pocztowy
0	Zielona Góra	65-012
1	Wrocław	50-031
2	Jelenia Góra	85-500
3	Bestwina	43-512
4	Nowa Sól	65-433

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	Sopot	80-001

☐ Alternatywa
☒ Koniunkcja

Rysunek 16: Testowanie funkcji addData(5)

Można zauważyć wynik:

id_lokalizacji	miasto	kod_pocztowy
0	Zielona Góra	65-012
1	Wrocław	50-031
2	Jelenia Góra	85-500
3	Bestwina	43-512
4	Nowa Sól	65-433
5	Sopot	80-001

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	Sopot	80-001

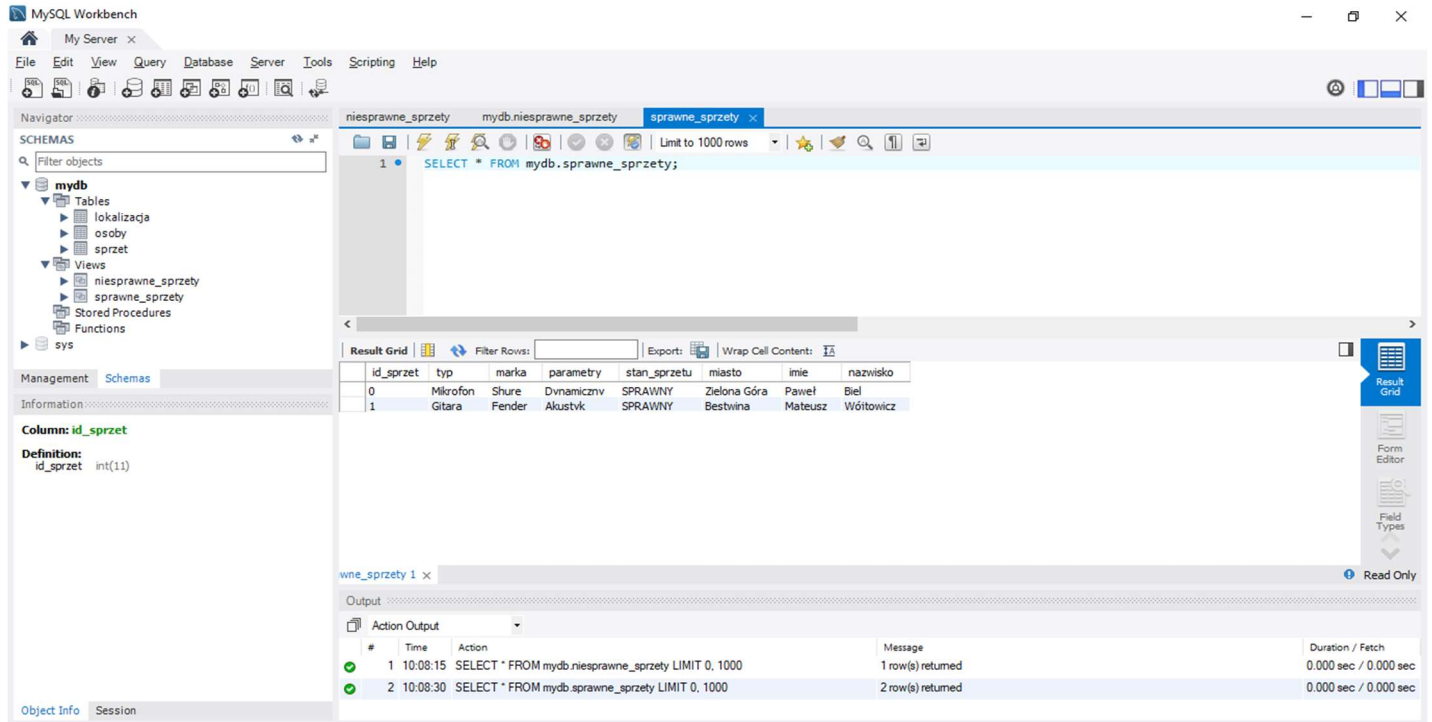
☐ Alternatywa
☒ Koniunkcja

Rysunek 17: Testowanie funkcji addData(6)

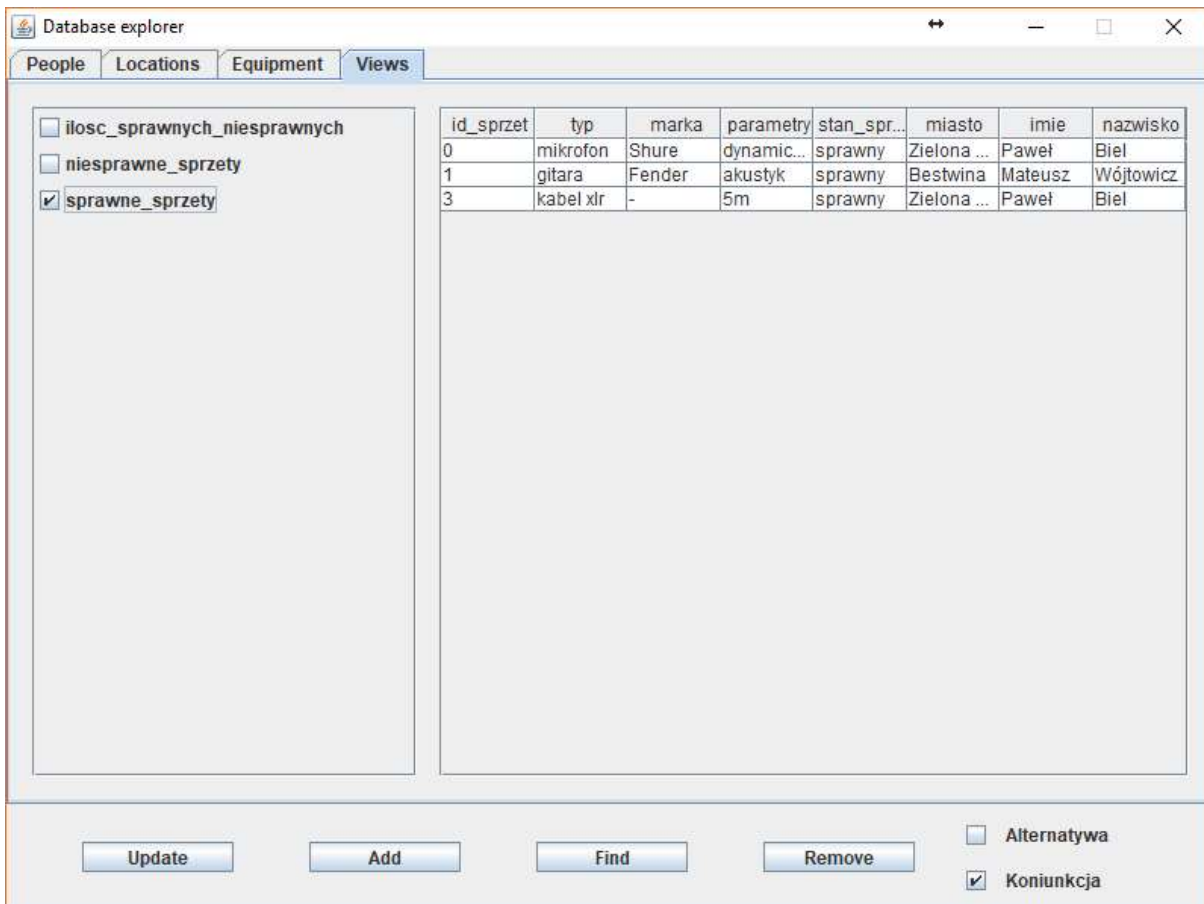
5.2.2. Testowanie funkcji getViewNames()

Przypadek kiedy w bazie danych są stworzone widoki:

Aplikacja pobierze i wyświetli zaznaczone widoki:

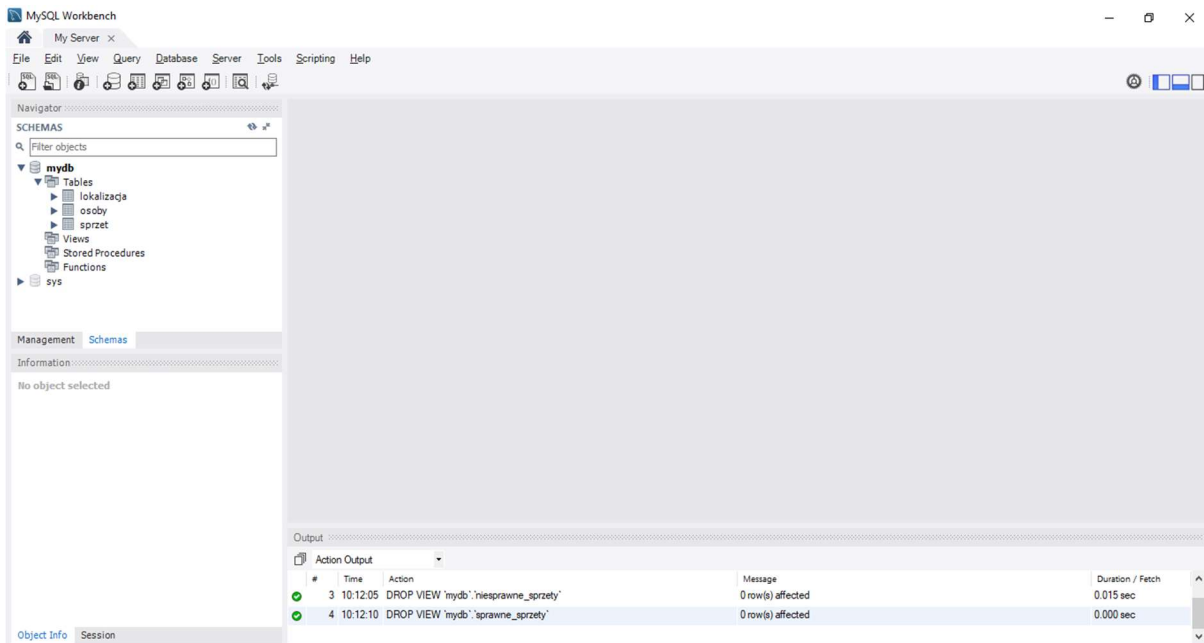


Rysunek 18: Testowanie funkcji getViewNames()(1)



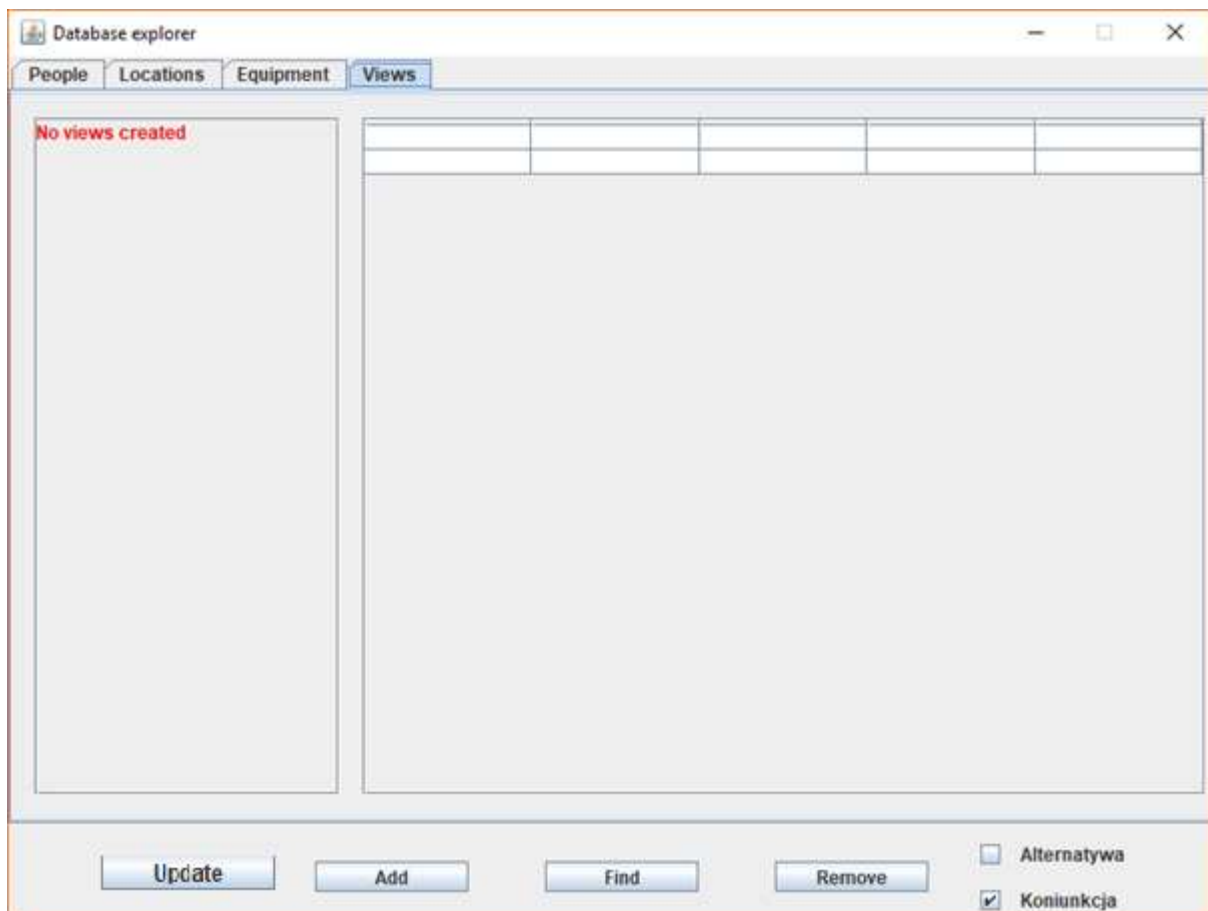
Rysunek 19: Testowanie funkcji getViewNames()(2)

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)

5.2.1. Test funkcji removeData()

Database explorer

People Locations **Equipment** Views

id_lokalizacji	miasto	kod_pocztowy
0	Zielona Góra	65-012
1	Wrocław	50-031
2	Jelenia Góra	85-500
3	Bestwina	43-512
4	Nowa Sól	65-433
5	Sopot	80-001

☒ ☐ ☐

5

Update Add Find Remove

☐ Alternatywa
☒ Koniunkcja

Rysunek 22: Testowanie funkcji removeData()(1)

Database explorer

People Locations **Equipment** Views

id_lokalizacji	miasto	kod_pocztowy
0	Zielona Góra	65-012
1	Wrocław	50-031
2	Jelenia Góra	85-500
3	Bestwina	43-512
4	Nowa Sól	65-433

☐ ☐ ☐

Update Add Find Remove

☐ Alternatywa
☒ Koniunkcja

Rysunek 23: Testowanie funkcji removeData()(2)

5.2.2. Testy funkcji upDataData()

Database explorer

People Locations Equipment Views

id_lokalizacji	miasto	kod_pocztowy
0	Zielona Góra	65-012
1	Wrocław	50-031
2	Jelenia Góra	85-500
3	Bestwina	43-512
4	Nowa Sól	65-433
5	Sopot	80-001

☐ ☐ ☐

5

Update Add Find Remove

☐ Alternatywa
☒ Koniunkcja

Rysunek 24: Testowanie funkcji upDataData(1)

Database explorer

People Locations Equipment Views

id_lokalizacji	miasto	kod_pocztowy
0	Zielona Góra	65-012
1	Wrocław	50-031
2	Jelenia Góra	85-500
3	Bestwina	43-512
4	Nowa Sól	65-433
5	Gdynia	80-001

☐ ☒ ☐

5 Gdynia

Update Add Find Remove

☐ Alternatywa
☒ Koniunkcja

Rysunek 25: Testowanie funkcji upDataData(2)

5.2.3. Testy funkcji getData()

Database explorer

People Locations Equipment Views

id_osoby	id_lokalizacji	imie	nazwisko	nr_tel	adres	mail	login	haslo
0	0	Paweł	Biel	535226885	Wróblewskie...	pawel.biel.96...	admin	admin
1	3	Mateusz	Wójtowicz	791024199	Kościelna 27	mateusz.ecli...	master	adminadmin
2	1	Janusz	Biernat	880733654	Wybrzeże Wy...	janusz.biern...	sumator	ultrasumator
3	0	Jan	Kowalski	123456789	Ulica 8	aaa@wp.pl	xyz	xyz
4	0	Michał	Nowak	987456132	Ulica 1000	mail@wp.pl	login	login
5	4	Olek	Prus	321354654	ilica2313	mail@o2.pl	olek	prusss
6	1	Karolina	Leśkiewicz	666666666	Fiołkowa 2	leśnyskrzat@...	karolina	karolina

☐ ☒ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Update Add Find Remove

☐ Alternatywa
☒ Koniunkcja

Rysunek 26: Testowanie funkcji getData()(1)

Database explorer

People Locations Equipment Views

id_osoby	id_lokalizacji	imie	nazwisko	nr_tel	adres	mail	login	haslo
0	0	Paweł	Biel	535226885	Wróblewskie...	pawel.biel.96...	admin	admin
3	0	Jan	Kowalski	123456789	Ulica 8	aaa@wp.pl	xyz	xyz
4	0	Michał	Nowak	987456132	Ulica 1000	mail@wp.pl	login	login

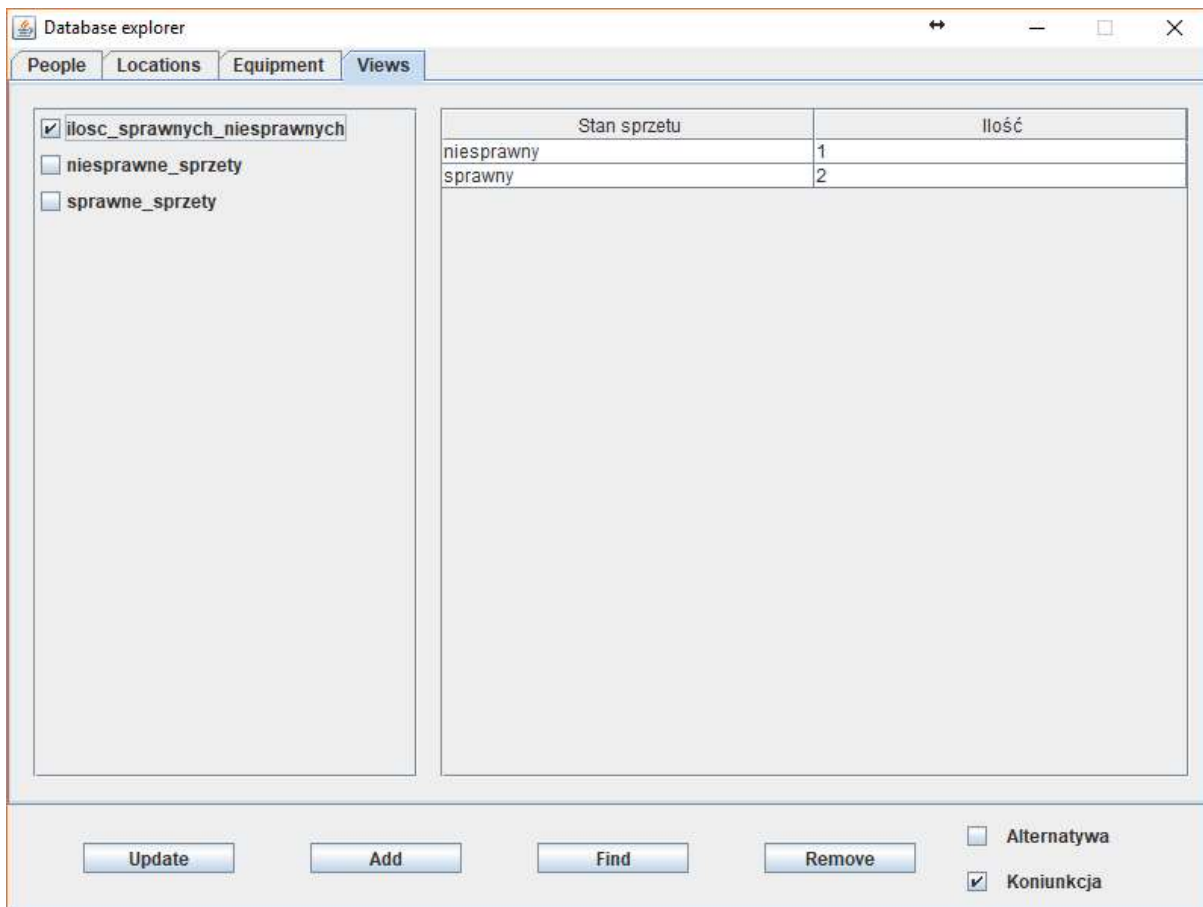
☐ ☒ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Update Add Find Remove

☐ Alternatywa
☒ Koniunkcja

Rysunek 27: Testowanie funkcji getData()(2)

5.2.4. Testowanie widoku automatycznego zliczania sprzętów sprawnych i niesprawnych



Database explorer

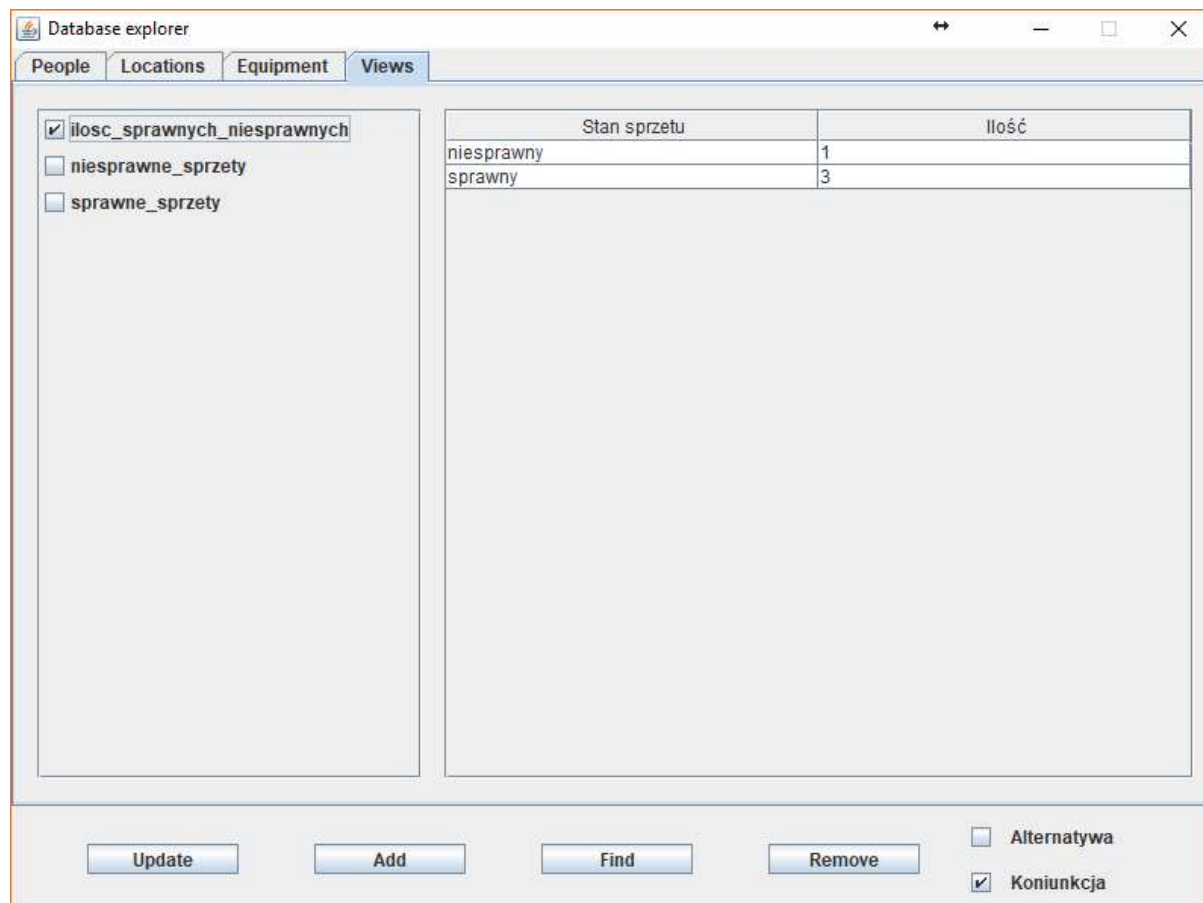
People Locations Equipment Views

☒ ilosc_sprawnych_niesprawnych
☐ niesprawne_sprzety
☐ sprawne_sprzety

Stan sprzetu	Ilość
niesprawny	1
sprawny	2

Update Add Find Remove ☐ Alternatywa ☒ Koniunkcja

Rysunek 28: Testowanie funkcji zliczania sprzętu(1)



Database explorer

People Locations Equipment Views

☒ ilosc_sprawnych_niesprawnych
☐ niesprawne_sprzety
☐ sprawne_sprzety

Stan sprzetu	Ilość
niesprawny	1
sprawny	3

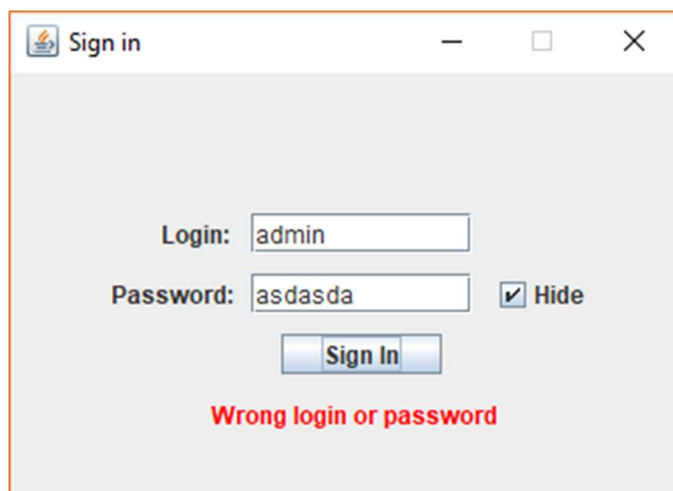
Update Add Find Remove ☐ Alternatywa ☒ Koniunkcja

Rysunek 29: Testowanie funkcji zliczania sprzętu(2)

5.3. Testowanie mechanizmów bezpieczeństwa

Testowanie mechanizmu logowania do aplikacji:

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

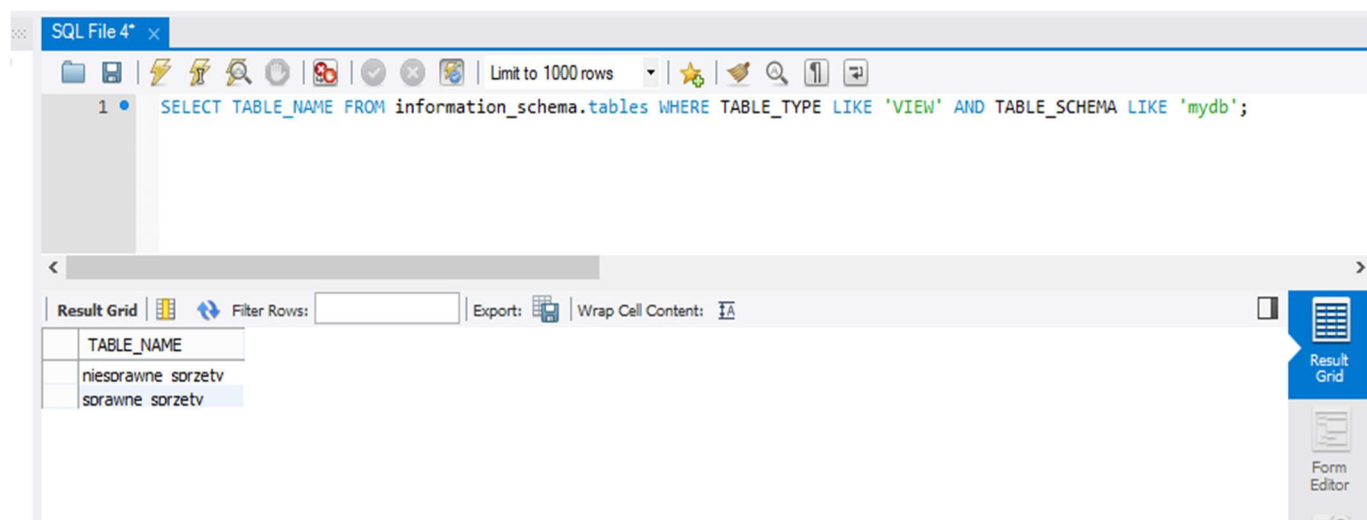


Rysunek 30: Testowanie mechanizmów bezpieczeństwa

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

5.4. Inne testy

Poniżej przedstawione są testy zapytań wykorzystywanych przez funkcje aplikacji.



Rysunek 31: Inne testy(1)

SQL File 4*

Limit to 1000 rows

1 • `SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'sprawne_sprzety';`

Result Grid

COLUMN_NAME
id sorzet
tvp
marka
parametr
miasto
imie
nazwisko

Result Grid

Form Editor

Field

Rysunek 32: Inne testy(2)

SQL File 4*

Limit to 1000 rows

1 • `SELECT * FROM osoby;`

Result Grid

id_osoby	id_lokalizacji	imie	nazwisko	nr_tel	adres	mail	login	haslo
0	0	Paweł	Biel	535226885	Wróblewskiego 8	pawel.biel.96@omail.com	admin	admin
1	1	Mateusz	Wójtowicz	791024199	Kościelna 27	mateusz.ediose@gmail.com	master	adminadmin
2	2	Janusz	Biernat	880733654	Wybrzeże Wvspiańskiego 27	ianusz..biernat@pwr.edu.pl	sumator	ultrasumator
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid

Form Editor

Rysunek 33: Inne testy(3)

5.5. 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

6. Podsumowanie

Udało się stworzyć oprogramowanie realizujące założenia wstępne projektu, pomyślnie przechodzące testy funkcjonalności. Za pomocą aplikacji można ma wiele różnych sposobów wyszukiwać dane zawarte w bazie danych systemu. Można również usuwać dane z bazy, przy czym dane do usunięcia można wyznaczać analogicznie jak dane do wyszukania, co znacznie ułatwia zbiorowe usuwanie danych. Kolejnymi rzeczami, które udało się zaimplementować jest możliwość dodawania nowych danych do bazy oraz zmieniania tych, które już istnieją w bazie. Poza podstawowymi funkcjami jakimi są dodawanie, usuwanie, zmienianie oraz wyszukiwanie danych, udało się również zaimplementować dodatkową funkcjonalność, polegającą na wyświetlaniu w formie tabeli, wybranego przez nas widoku, stworzonego w bazie. W celu autoryzacji dostępu do aplikacji udało się zaimplementować system logowania, wymagający podania poprawnych loginu i hasła, które przypisane są do użytkownika w bazie danych. Po stronie bazy danych zaimplementowany został również mechanizm zliczający na bieżąco ilość sprzętów sprawnych i niesprawnych, którego wyniki można przejrzeć w odpowiednim widoku, do którego dostęp mamy bezpośrednio z aplikacji. Program działa stabilnie, jest dobrze zabezpieczony, a mechanizmy walidacji danych chronią logikę systemu. Efektem prac jest aplikacja wraz z bazą danych oraz niniejsza dokumentacja.

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/>