**Dokumentacja projektu wykonanego w ramach przedmiotu   
BAZY DANYCH I**

**Hotel**

**Wykonał Kamil Sudoł**

**I. Projekt koncepcji, założenia**

**1. Zdefiniowanie tematu projektu:**

Tematem projektu jest obsługa bazy danych dedykowanej sprawom hotelarskim. Dzięki napisanej aplikacji możliwe jest obsłużenie podstawowych działań związanych z branżą hotelarską, takich, jak możliwość dokonania rezerwacji danego pokoju na konkretny termin, możliwość opłaty bądź rezygnacji z rezerwacji.

**2. Analiza wymagań użytkownika:**

Użytkownik zalogowany jako klient powinien mieć możliwość:

- dokonania rezerwacji na pokój o określonej kategorii,

- dokonania rezerwacji na ten sam pokój w różnych terminach,

- wyboru dodatkowych usług do rezerwacji pokoju,

- podglądu informacji o pokojach podzielonych na kategorie,

- podglądu informacji o swoich rezerwacjach,

- opłacenia lub rezygnacji z rezerwacji.

Ponadto użytkownik zalogowany jako administrator hotelu poza możliwościami wymienionymi powyżej powinien mieć dodatkowo możliwość:

- podglądu wszystkich danych w bazie danych,

- zablokowania konkretnego użytkownika,

- dodania nowego pracownika do obsługi hotelu.

Dodatkowo:

- rezerwacja nie może zostać zaakceptowania, jeżeli termin zakwaterowania nastąpi szybciej,  
 niż 7 dni od daty złożenia rezerwacji,

- rezerwacje przydzielane są na konkretny pokój,

- każdy gość może dokonywać wiele rezerwacji na wiele kategorii pokojów,

- gościowi przydzielamy pokój nie kategorię,

- gość, który nie opłaci swojej rezerwacji do dnia zakwaterowania, zostaje automatycznie przeniesiony na czarną listę,

- gość, który dobrowolnie zrezygnuje z rezerwacji więcej, niż 5 razy, zostaje automatycznie przeniesiony na czarną listę.

**3. Zaprojektowanie funkcji:**

Użytkownik zalogowany jako klient może dokonać rezerwacji na jeden konkretny pokój z wielu dostępnych o określonej kategorii, w konkretnym terminie (o ile wybrany pokój będzie dostępny w tymże terminie) na konkretną liczbę osób uwzględniając podział na dzieci i dorosłych, mając przy tym możliwość wyboru dodatkowych usług, takich, jak dostęp do basenu, kręgielni, bilarda czy siłowni. Ponadto klient ma możliwość podglądu swoich rezerwacji, wyświetlenia szczegółowych informacji każdej z nich oraz możliwość zdecydowania, czy daną rezerwację klient chce opłacić bądź zrezygnować z niej.

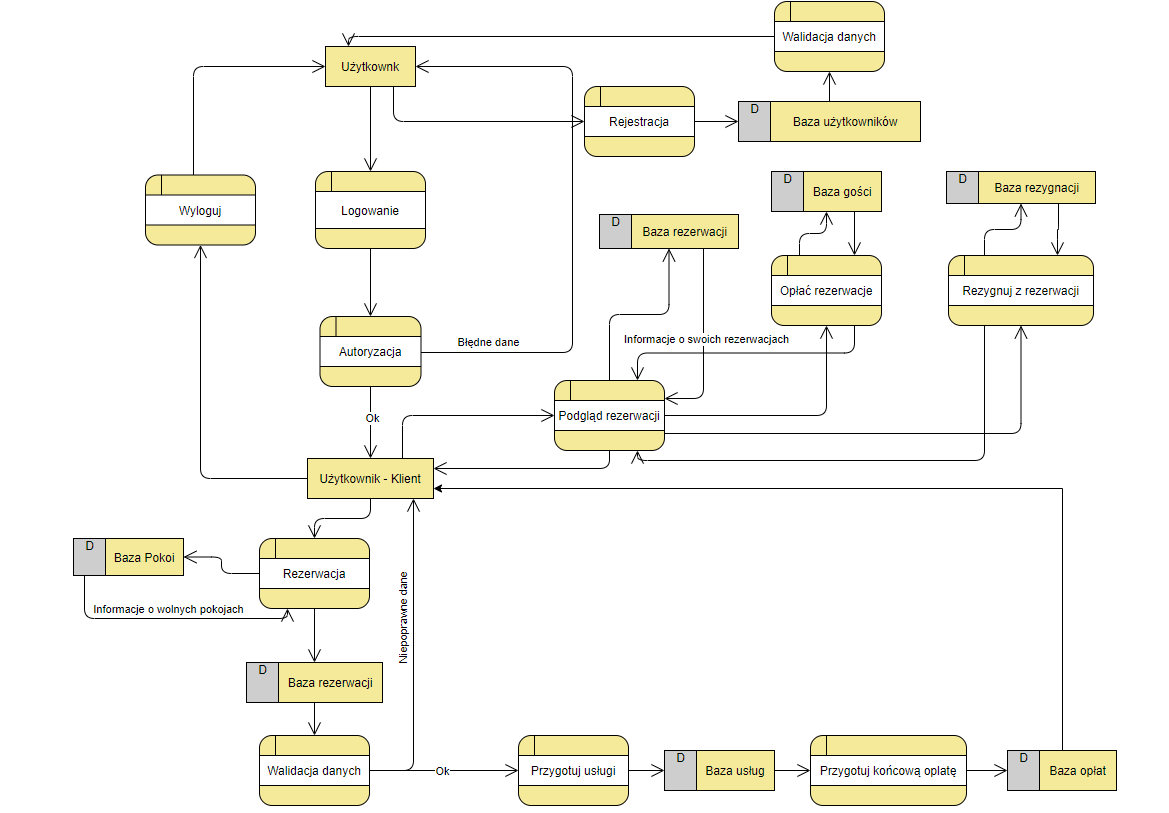
Użytkownik zalogowany jako headadmin lub pracownik poza możliwościami wymienionymi dla klientów ma udostępniony panel administratora. Na omawiany panel składa się możliwość dodania nowego pracownika do obsługi hotelu a także możliwość podglądu wszystkich raportów oraz tabel hotelowej bazy danych.

Dodatkowo baza danych ma funkcjonalność zapobiegania wprowadzania nieprawidłowych danych oraz możliwość zwalniania pokoi, jeżeli dany klient zrezygnował ze swojej rezerwacji.

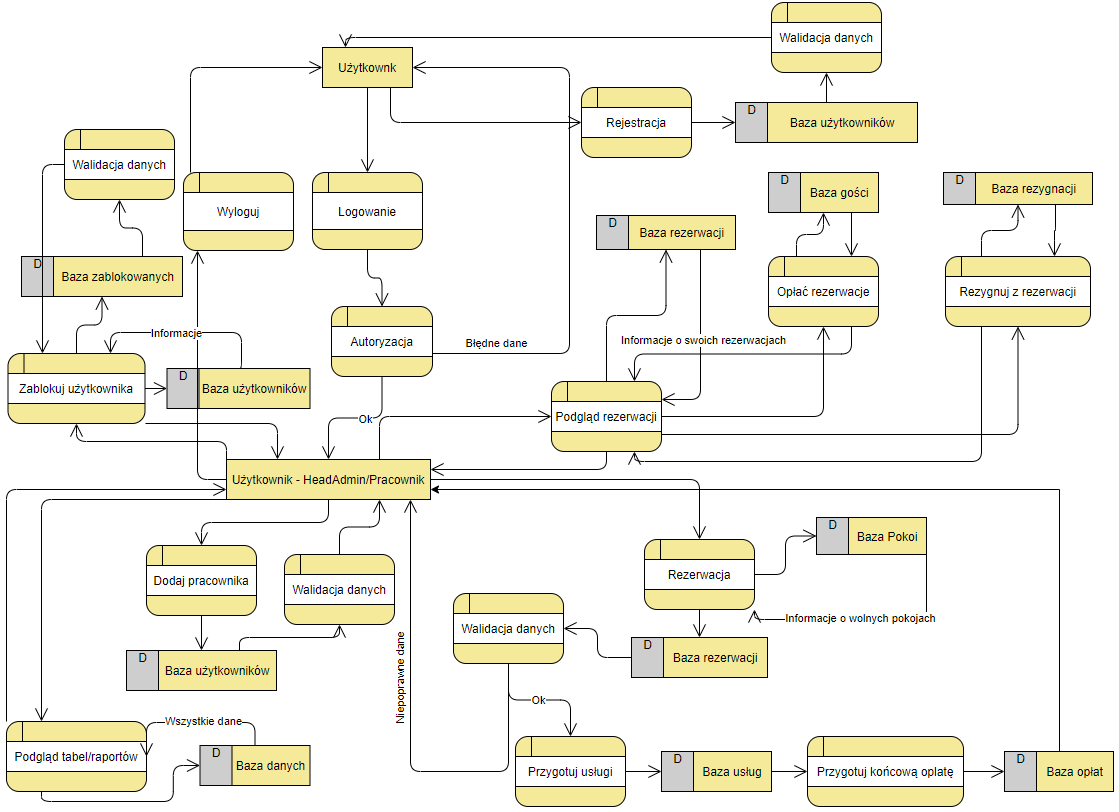
**II. Projekt diagramów (konceptualny)**

**4. Budowa i analiza diagramu przepływu danych (DFD):**

**Diagram DFD dla użytkownika zalogowanego jako Klient:**

****

**Diagram DFD dla użytkownika zalogowanego jako HeadAdmin lub Pracownik:**

****

**5. Zdefiniowanie encji (obiektów) oraz ich atrybutów:**

***Panel:*** *uzyt\_id*, *login*, *haslo*,

***Uzytkownik***: *uzytkownik\_id*, *imie*, *nazwisko*, *e\_mail*, *numer*, *typ*,

***Rezerwacje***: *rezerwacja\_id*, *uzytkownik\_id*, *pokoj\_id*, *data\_rezerwacji*, *od\_kiedy*, *do\_kiedy*, *liczba\_dzieci*, *liczba\_doroslych*,

***Dodatkowe\_uslugi***: *dodatkowe\_uslugi\_id*, *nazwa\_uslugi*, *cena\_od\_osoby*, *rezerwacja\_id*,

***Oplata***: *oplata\_id*, *rezerwacja\_id,* *status\_czy\_oplacone*, *kwota*,

***Pokoj***: *pokoj\_id*, *numer\_pokoju*, *pietro, liczba\_miejsc*, *kategoria\_id*,

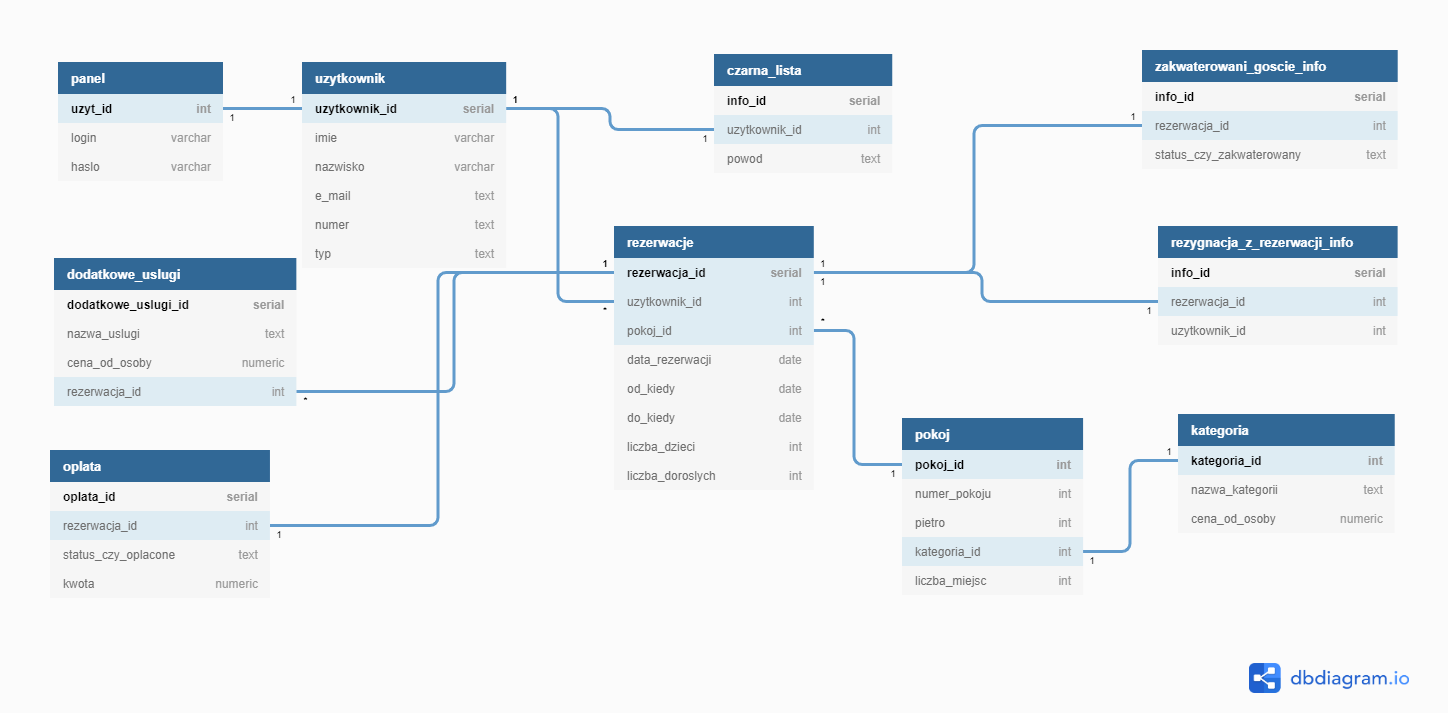
***Kategoria***: *kategoria\_id*, *nazwa\_kategorii*, *cena\_od\_osoby*,

***Rezygnacja\_z\_rezerwacji\_info:*** *info\_id*, *rezerwacja\_id*, *uzytkownik\_id*,

***Zakwaterowani\_goscie\_info:*** *info\_id*, *rezerwacja\_id*, *status\_czy\_zakwaterowany*,

**C*zarna\_lista***: *info\_id*, *uzytkownik\_id, powod.*

**6. Zaprojektowanie relacji pomiędzy encjami:**

****

**III. Projekt logiczny**

**7. Projektowanie tabel, kluczy, indeksów:**

Schemat ***projekt***:

- tabela ***panel:***

- *uzyt\_id* INT PRIMARY KEY – jednoznacznie identyfikuje id użytkownika, połączenie z tabelą *uzytkownik(uzytkownik\_id) -* relacja 1:1*,*

- tabela ***uzytkownik***:

- *uzytkownik\_id* SERIAL PRIMARY KEY – jednoznacznie identyfikuje id użytkownika,

- tabela ***rezerwacje***:

- *rezerwacja\_id* SERIAL PRIMARY KEY– jednoznacznie identyfikuje id rezerwacji,

- *uzytkownik\_id* INT klucz obcy do *uzytkownik(uzytkownik\_id) -* relacja N:1,

- *pokoj\_id* INT klucz obcy do *pokoj(pokoj\_id) -* relacja N:1,

- tabela ***dodatkowe\_uslugi***:

- *dodatkowe\_uslugi\_id* SERIAL PRIMARY KEY – jednoznacznie identyfikuje id dodatkowej usługi,

- *rezerwacja\_id* INT klucz obcy do *rezerwacje(rezerwacja\_id) -* relacja 1:1,

- tabela ***oplata***:

- *oplata\_id* SERIAL PRIMARY KEY – jednoznacznie identyfikuje id opłaty,

- *rezerwacja\_id* INT klucz obcy do *rezerwacje(rezerwacja\_id) -* relacja 1:1,

- tabela ***pokoj***:

- *pokoj\_id* INT PRIMARY KEY – jednoznacznie identyfikuje id pokoju,

- *kategoria\_id* INT klucz obcy do *kategoria(kategoria\_id) -* relacja 1:1

- tabela ***kategoria***:

- *kategoria\_id* INT PRIMARY KEY – jednoznacznie identyfikuje id kategorii,

- tabela ***rezygnacja\_z\_rezerwacji\_info:***

- *info\_id* SERIAL PRIMARY KEY – jednoznacznie identyfikuje id informacji,

- *uzytkownik\_id* INT klucz obcy do *uzytkownik(uzytkownik\_id) -* relacja 1:1,

- tabela ***zakwaterowani\_goscie\_info:***

- *info\_id* SERIAL PRIMARY KEY – jednoznacznie identyfikuje id informacji,

- *rezerwacja\_id* INT klucz obcy do *rezerwacje(rezerwacja\_id) -* relacja 1:1,

- tabela ***czarna\_lista***:

- *info\_id* SERIAL PRIMARY KEY – jednoznacznie identyfikuje id informacji,

- *uzytkownik\_id* INT klucz obcy do *uzytkownik(uzytkownik\_id) -* relacja 1:1,

**8. Słowniki danych:**

Schemat ***projekt***:

- tabela ***panel:***

- *uzyt\_id* INT > 0,

- *login* VARCHAR długość > 5,

- *haslo* VARCHAR długość > 5

- tabela ***uzytkownik***:

- *uzytkownik\_id* SERIAL > 0,

- *imie* VARCHAR,

- *nazwisko* VARCHAR,

- *e\_mail* TEXT,

- *numer* TEXT,

- *typ* TEXT

- tabela ***rezerwacje***:

- *rezerwacja\_id* SERIAL > 0,

- *uzytkownik\_id* INT > 0,

- *pokoj\_id* INT > 0,

- *data\_rezerwacji* DATE,

- *od\_kiedy* DATE,

- *do\_kiedy* DATE,

- *liczba\_dzieci* INT >= 0,

- *liczba\_doroslych* INT >= 0

- tabela ***dodatkowe\_uslugi***:

- *dodatkowe\_uslugi\_id* SERIAL > 0,

- *nazwa\_uslugi* TEXT,

- *cena\_od\_osoby* NUMERIC >= 0,

- *rezerwacja\_id* INT > 0,

- tabela ***oplata***:

- *oplata\_id* SERIAL > 0,

- *rezerwacja\_id* INT > 0,

- *status\_czy\_oplacone* TEXT,

- *kwota* NUMERIC >= 0

- tabela ***pokoj***:

- *pokoj\_id* INT > 0,

- *numer\_pokoju* INT > 0,

*- pietro INT* >= 0*,*

- *liczba\_miejsc* INT > 0,

- *kategoria\_id* INT > 0

- tabela ***kategoria***:

- *kategoria\_id* INT > 0,

- *nazwa\_kategorii* TEXT,

- *cena\_od\_osoby* NUMERIC >= 0

- tabela ***rezygnacja\_z\_rezerwacji\_info:***

- *info\_id* SERIAL > 0,

- *rezerwacja\_id* INT > 0,

- *uzytkownik\_id* INT > 0

- tabela ***zakwaterowani\_goscie\_info:***

- *info\_id* SERIAL > 0,

- *rezerwacja\_id* INT > 0,

- *status\_czy\_zakwaterowany* TEXT

- tabela ***czarna\_lista***:

- *info\_id* SERIAL > 0,

- *uzytkownik\_id* INT > 0,

- *powod* TEXT

**9. Analiza zależności funkcyjnych i normalizacja tabel:**

Pierwsza postać normalna (1NF):

W każdej tabeli jest jednoznacznie definiujący encję klucz główny. Wszystkie dane są atomowe.

Druga postać normalna (2NF):

Każda tabela przechowuje dane dotyczące tylko konkretnej klasy obiektów. Każdy niekluczowy atrybut jest nieredukowalnie zależny od klucza głównego.

Trzecia postać normalna (3NF):

Wszystkie niekluczowe atrybuty (niewchodzące w skład klucza głównego) są

**10. Zaprojektowanie operacji na danych:**

**Funkcje (dostępne w pliku SQL/funkcje.sql):**

*Numeric\_check* – pomocnicza funkcja wykorzystywana przy walidacji danych rejestracyjnych sprawdzająca czy podany numer telefonu składa się z samych cyfr.

*Get\_pokoje* – funkcja zwracająca z bazy listę dostępnych pokoi na podany termin i deklarowaną liczbę osób. Do tego celu został użyty kursor iterujący po tabeli *pokoj,* gdzie w pętli sprawdzany był warunek, czy wskazywany pokój na podany termin przypisany jest już do jakiejś rezerwacji – jeżeli nie, wówczas id pokoju było zapisywane do zwracanej tabeli.

*Get\_liczba\_dni* – pomocnicza funkcja wyznaczająca przedział czasowy pomiędzy podanymi datami.

*Last\_rezerwacja\_id* – funkcja zwracająca z bazy numer id ostatniej złożonej rezerwacji. W tym celu wykorzystano funkcję agregującą *MAX()*, która została wywołana na atrybucie *rezerwacja\_id* z tabeli *rezerwacje*.

*Last\_uzytkownik\_id* – funkcja zwracająca z bazy numer id ostatniego zarejestrowanego użytkownika. W tym celu wykorzystano funkcję agregującą *MAX()*, która została wywołana na atrybucie *uzytkownik\_id* z tabeli *uzytkownik*.

*OplataZaplac* – funkcja aktualizująca wartość atrybutu *status\_czy\_oplacone* na wartość *‘Oplacone’* krotki z tabeli *oplata* o podanym numerze id.

*OplataRezygnuj* – funkcja aktualizująca wartość atrybutu *status\_czy\_oplacone* na wartość *‘Nieoplacone - rezygnacja’* krotki z tabeli *oplata* o podanym numerze id.

*ZbanowanyCheck* – funkcja sprawdzająca, czy użytkownik o podanym id został z jakiegoś powodu zablokowany. W tym celu sprawdzane jest funkcja agregującą *COUNT()*, czy podany id sprawdzanego użytkownika znajduje się w tabeli *czarna\_lista*.

*Autoryzacja* – funkcja sprawdzająca, czy logujący się użytkownik jest klientem, pracownikiem czy headadminem. W tym celu sprawdzany jest atrybut *typ* z tabeli *uzytkownik*.

*StartUpdate* – funkcja wywoływana na bazie danych przy każdym starcie aplikacji. Funkcja ta wykorzystuje kursor iterujący po widoku *OplataRezerwacje* sprawdzając między innymi:  
- czy dzisiejszego dnia następuje jakieś zakwaterowanie,  
- czy dzisiejszego dnia następuje jakieś wykwaterowanie,  
- czy dzisiejszego dnia mija termin opłaty rezerwacji któregoś z klientów.

W przypadku zakwaterowania czy wykwaterowania modyfikowana jest krotka z tabeli *zakwaterowani\_goscie\_info* aktualizująca wartość atrybutu *status\_czy\_zakwaterowany*, zaś w przypadku braku uregulowania opłaty w terminie użytkownik ten automatycznie jest przenoszony na czarną listę, innymi słowy wykonywany jest *INSERT* do tabeli *czarna\_lista* zawierająca id tegoż użytkownika oraz powód o treści *‘Nieoplacenie rezerwacji w terminie.’*.

**Wyzwalacze (dostępne w pliku SQL/triggery.sql):**

*Register\_validator* – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *użytkownik*. Wyzwalacz ten sprawdza poprawność wprowadzonych danych, czyli sprawdza, czy wszystkie pola są wypełnione, czy adres email jest poprawny bądź unikatowy oraz czy podany numer telefonu jest składa się z cyfr. W przypadku niepowodzenia do aplikacji wysyłany jest odpowiedni komunikat z bazy.

Panel\_validator – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *panel*. Wyzwalacz ten sprawdza poprawność wprowadzonych danych, czyli sprawdza, czy wszystkie pola są wypełnione, czy login oraz hasło są dłuższe, niż 5 znaków oraz czy nazwa użytkownika jest unikatowa. W przypadku niepowodzenia do aplikacji wysyłany jest odpowiedni komunikat z bazy.

*Platnosc\_rezerwacja* – wyzwalacz uruchamiany po *INSER’cie* do tabeli *rezerwacje*. Wyzwalacz ten oblicza sumaryczny koszt pobytu w zależności od ilości dni uwzględniając dzieci oraz dorosłych, wykonując *INSERT* do tabeli *oplata*.

*Platnosc\_update* – wyzwalacz uruchamiany po *INSER’cie* do tabeli *dodatkowe\_uslugi*. Wyzwalacz ten aktualizuje sumaryczny koszt pobytu w hotelu z tabeli *oplata* o koszt za dodatkowe usługi uwzględniając dzieci oraz dorosłych.

*OplataStatus* – wyzwalacz uruchamiany przed *UPDATE’em* do tabeli *oplata*. Wyzwalacz ten odpowiedzialny jest za aktualizacje statusu opłaty. W celu uniknięcia problemów z aktualizacjami wynikającymi z dwóch wcześniej opisanych wyzwalaczy sprawdzane jest najpierw, czy nowa kwota rożni się od kwoty poprzedniej, wówczas wyzwalacze *Platnosc\_rezerwacje* oraz *Platnosc\_update* mogą spokojnie się wykonać. W przeciwnym wypadku sprawdzany jest warunek, czy stary status jest określony jako *‘Nieoplacone’*. Jeżeli tak, wówczas w przypadku nowego statusu *‘Oplacone’* wykonywany jest *INSERT* do tabeli *zakwaterowani\_goscie\_info* zawierający id rezerwacji wraz z informacja *‘Oczekiwanie na zakwaterowanie’*, zaś w przypadku nowego statusu *‘Nieoplacone - rezygnacja’* wykonywany jest *INSERT* do tabeli *rezygnacja\_z\_rezerwacji\_info* zawierający id rezerwacji oraz id użytkownika. Gdy jednak żaden z wyżej wymienionych warunków nie jest spełniony, do aplikacji wysyłany jest komunikat informujący o braku możliwości modyfikacji wcześniej zmodyfikowanego statusu opłaty rezerwacji.

*RezerwacjaNaTydzienPrzed* – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *rezerwacje*. Wyzwalacz ten sprawdza, czy wybrany termin zakwaterowania jest określony na więcej niż siedem dni od dnia złożenia rezerwacji. W przeciwnym wypadku do aplikacji wysyłany jest odpowiedni komunikat z bazy.

*RezerwacjaOsobyValidator* - wyzwalacz uruchamiany przed *INSERT’em* do tabeli *rezerwacje*. Wyzwalacz ten sprawdza, czy podana liczba osób w rezerwacji jest poprawna. Jeżeli sumaryczna liczba osób wynosi zero bądź przekracza pojemność wybranego pokoju, wówczas do aplikacji wysyłany jest odpowiedni komunikat z bazy.

*RezygnacjaChecker* – wyzwalacz uruchamiany po *INSER’cie* do tabeli *rezygnacja\_z\_rezerwacji\_info*. Wyzwalacz ten usuwa rekordy związane z podaną rezerwacją w celu zwolnienia pokoi oraz sprawdza liczbę rekordów w tejże tabeli dla danego użytkownika. Jeżeli liczba rezygnacji przekroczy 5 rekordów, wówczas użytkownik zostaje dodany na czarną listę, czyli wykonywany jest *INSERT* do tabeli *czarna\_lista* zawierający id użytkownika oraz powód o treści *‘Ciagle rezygnowanie ze skladanych rezerwacji.’*.

*CzarnaListaChecker* - wyzwalacz uruchamiany przed *INSERT’em* do tabeli *czarna\_lista*. Wyzwalacz ten jest głównie używany w czasie manualnego blokowania użytkownika przez administratora – sprawdzane jest, czy powód blokady jest uzupełniony oraz czy blokowany użytkownik nie jest headadminem lub pracownikiem – w przeciwnym wypadku do aplikacji zostanie wysłany odpowiedni komunikat z bazy.

*TransakcjaRezerwacjaChecker* – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *dodatkowe\_uslugi*. Wyzwalacz ten jest dosyć prostym i prymitywnym zabezpieczeniem przed wymieszaniem się danych w momencie, kiedy dwie osoby w tym samym czasie próbują dokonać rezerwacji na ten sam pokój w pokrywających się terminach. Jeżeli warunki te zostaną spełnione, do aplikacji zostanie wysłany odpowiedni komunikat z bazy. Dlaczego wyzwalacz ten jest podpięty pod tabelę *dodatkowe\_uslugi*? Otóż proces rezerwacji został zaimplementowany jako procedura transakcji, gdzie INSERT do tabeli *dodatkowe\_uslugi* wykonuje się zaraz po *INSER’cie* do tabeli *rezerwacje* i INSERT ten jest ściśle związany z poprzednim poprzez wartość atrybutu *rezerwacja\_id*. Jest to mało prawdopodobny scenariusz, jednakże, jeżeli w tym samym momencie dwie osoby przy wyżej wymienionych warunkach będą chciały dokonać rezerwacji, wówczas bez tego zabezpieczenia może nastąpić przypisanie informacji o dodatkowych usługach do zupełnie innej rezerwacji.

*TransakcjaRejestracjChecker* – wyzwalacz uruchamiany przed *INSERT’em* do tabeli *panel*. Wyzwalacz ten spełnia podobną rolę, co wyzwalacz opisany powyżej. Sprawdzane jest, czy nie następuje dopisanie informacji do istniejącego już użytkownika, a w przypadku spełnienia warunku do aplikacji zostanie wysłany odpowiedni komunikat z bazy.

**Widoki** **(dostępne w pliku SQL/widoki.sql):**

*Uzytkownicy* – widok będący złączeniem tabel *uzytkownik* oraz *panel* po atrybutach odpowiednio *uzytkownik\_id* oraz *uzyt\_id*.

*PokojeView* - widok będący złączeniem tabel *pokoj* oraz *kategoria* po atrybucie *kategoria\_id*.

*PokojeRezerwacjeView* – pomocniczy widok będący złączeniem tabel *pokoj* oraz *rezerwacje* po atrybucie *pokoj\_id*.

*UslugiPokojeRezerwacje* – pomocniczy widok będący złączeniem tabeli *dodatkowe\_uslugi* oraz widoku *PokojeRezerwacjeView* po atrybucie *rezerwacja\_id*.

*RezerwacjeInfoView* – pomocniczy widok będący złączeniem tabeli *oplata* oraz widoku *UslugiPokojeRezerwacje* po atrybucie *rezerwacja\_id*.

*RezerwacjeAllInfoView* – widok będący złączeniem tabeli *uzytkownik* oraz widoku *RezerwacjeInfoView* po atrybucie *uzytkownik\_id*.

*OplataRezerwacje* – pomocniczy widok będący złączeniem tabel *oplata* oraz *rezerwacje* po atrybucie *rezerwacja\_id*.

*RezerwPokojView* – pomocniczy widok będący złączeniem tabel *pokoj* oraz *rezerwacje* po atrybucie *rezerwacja\_id*.

*ZakwaterowaniView* –widok będący złączeniem tabel *uzytkownik* i *zakwaterowani\_goscie\_info* oraz widoku *RezerwPokojView*.

*ZablokowaniView* – widok będący złączeniem tabel *czarna\_lista* oraz *uzytkownik* po atrybucie *uzytkownik\_id*.

*UzytRezerwView* – pomocniczy widok będący złączeniem tabel *rezerwacje* oraz *uzytkownik* po atrybucie *uzytkownik\_id*.

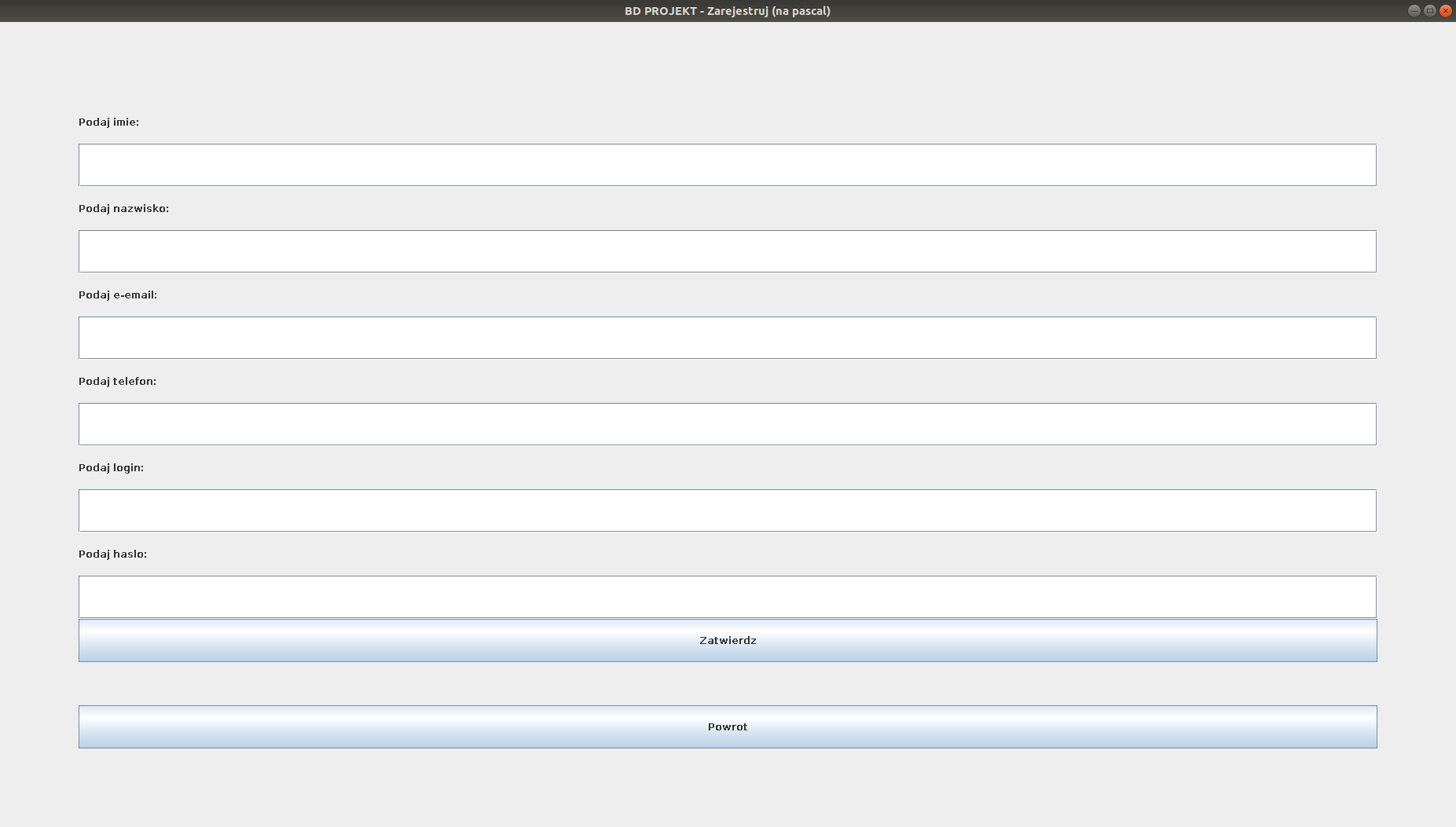
*RezygnacjaView* – widok będący złączeniem tabeli *rezygnacja\_z\_rezerwacji\_info* oraz widoku *UzytRezerwView* po atrybucie *rezerwacja\_id*.

**IV. Projekt funkcjonalny**

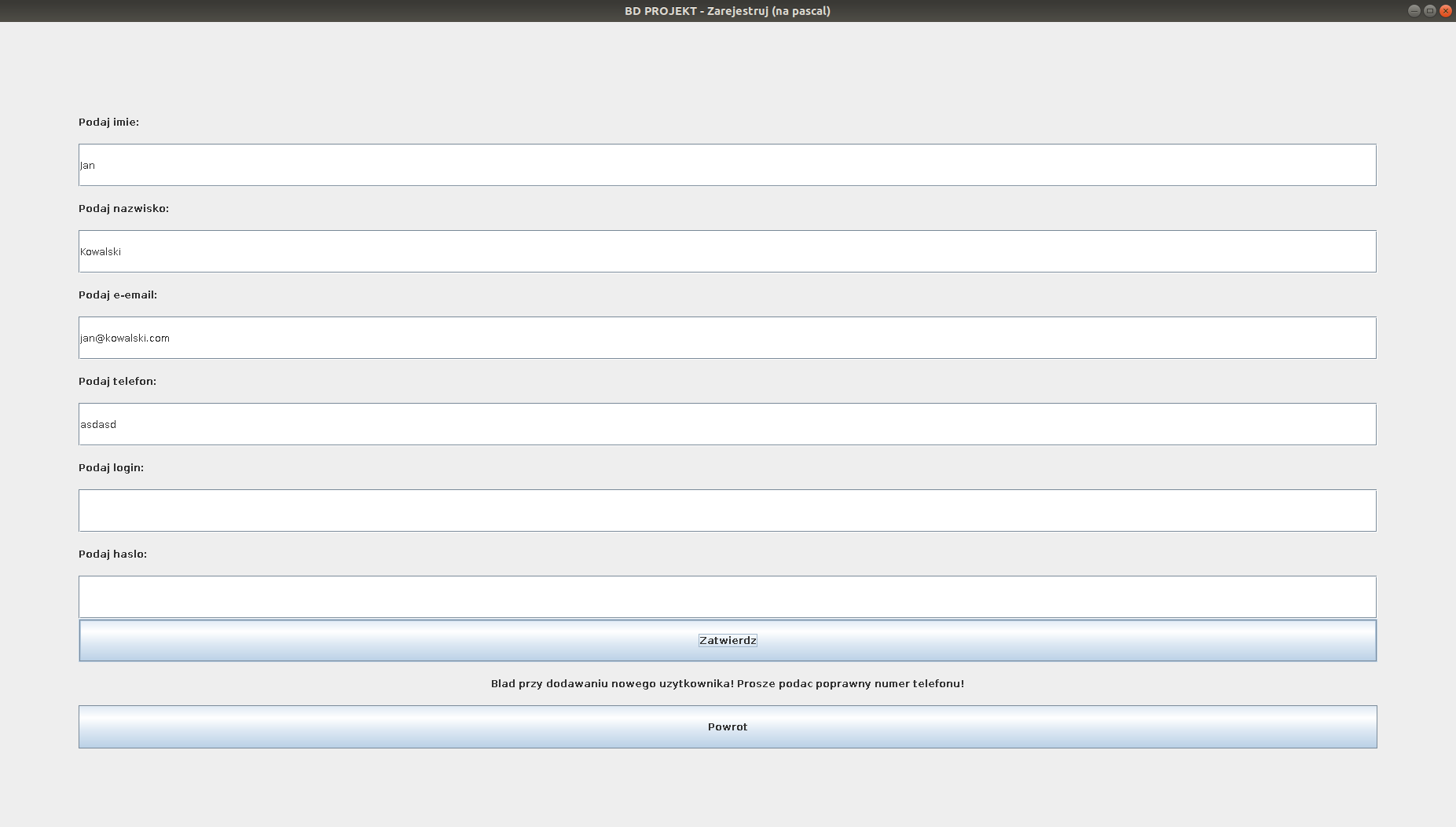
**11.Interfejsy do prezentacji, edycji i obsługi danych:**

Obsługa oraz edycja danych jest dostępna zarówno z poziomu administratora, jak i klienta. Poniżej przedstawiono dostępne formularze:

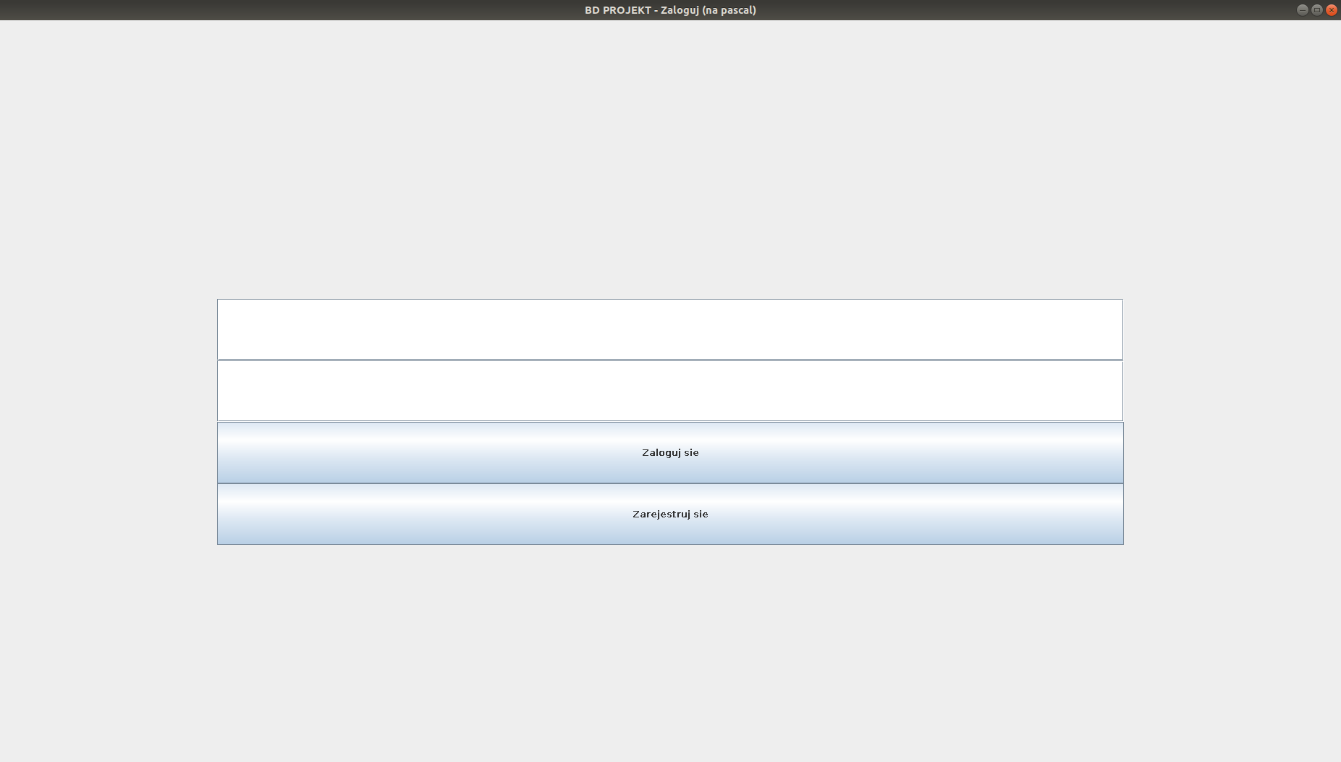
- formularz rejestracji nowego użytkownika – nowy użytkownik ma możliwość wprowadzenia swoich danych personalnych, po kliknięciu przycisku ‘Zatwierdz’ rozpoczyna się procedura transakcji, dane wysyłane są do bazy danych, a następnie odpowiednie wyzwalacze sprawdzają poprawność wprowadzonych danych (czy wszystkie pola są wypełnione, czy użytkownik o podanym e-mailu już istnieje, czy podany login jest unikatowy itp.),

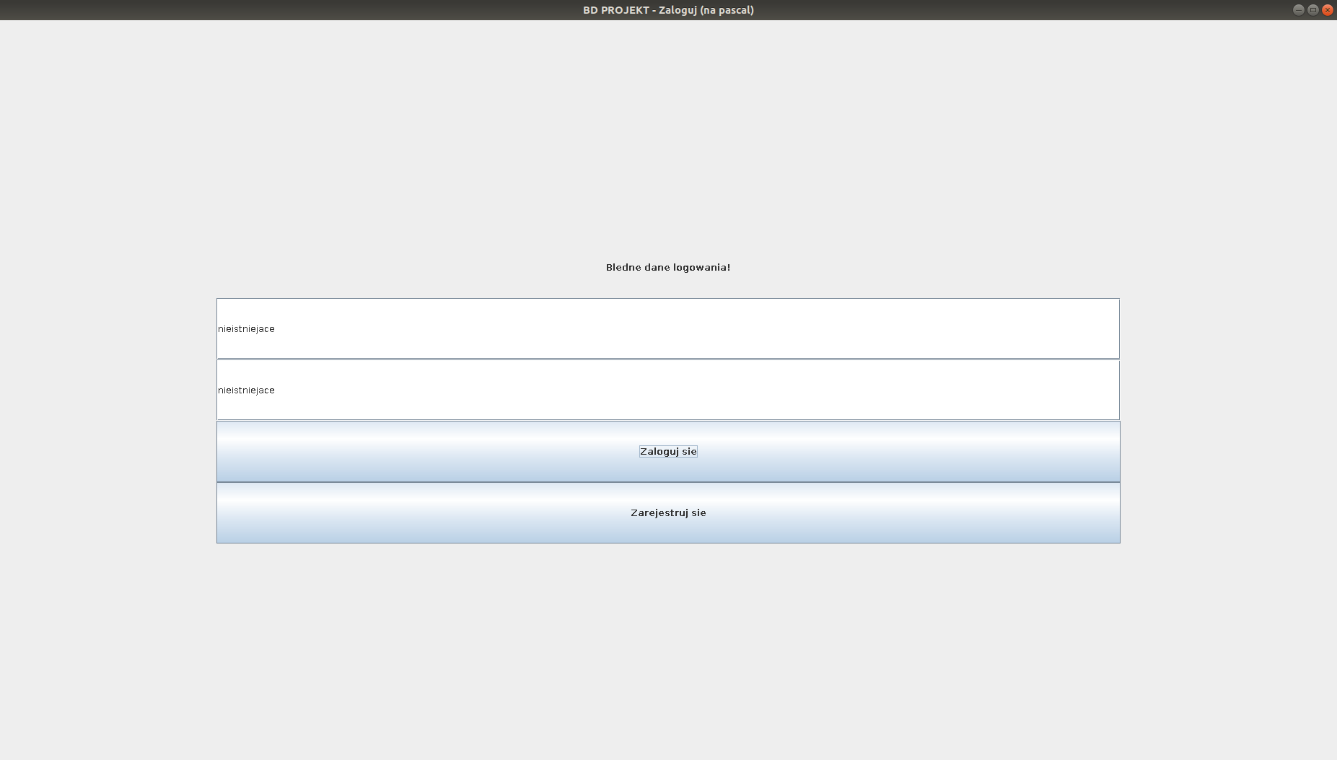


W przypadku powodzenia bądź też niepowodzenia wyświetlane są odpowiednie komunikaty wysyłane z bazy. Poniżej przedstawiono przykład takiego komunikatu:

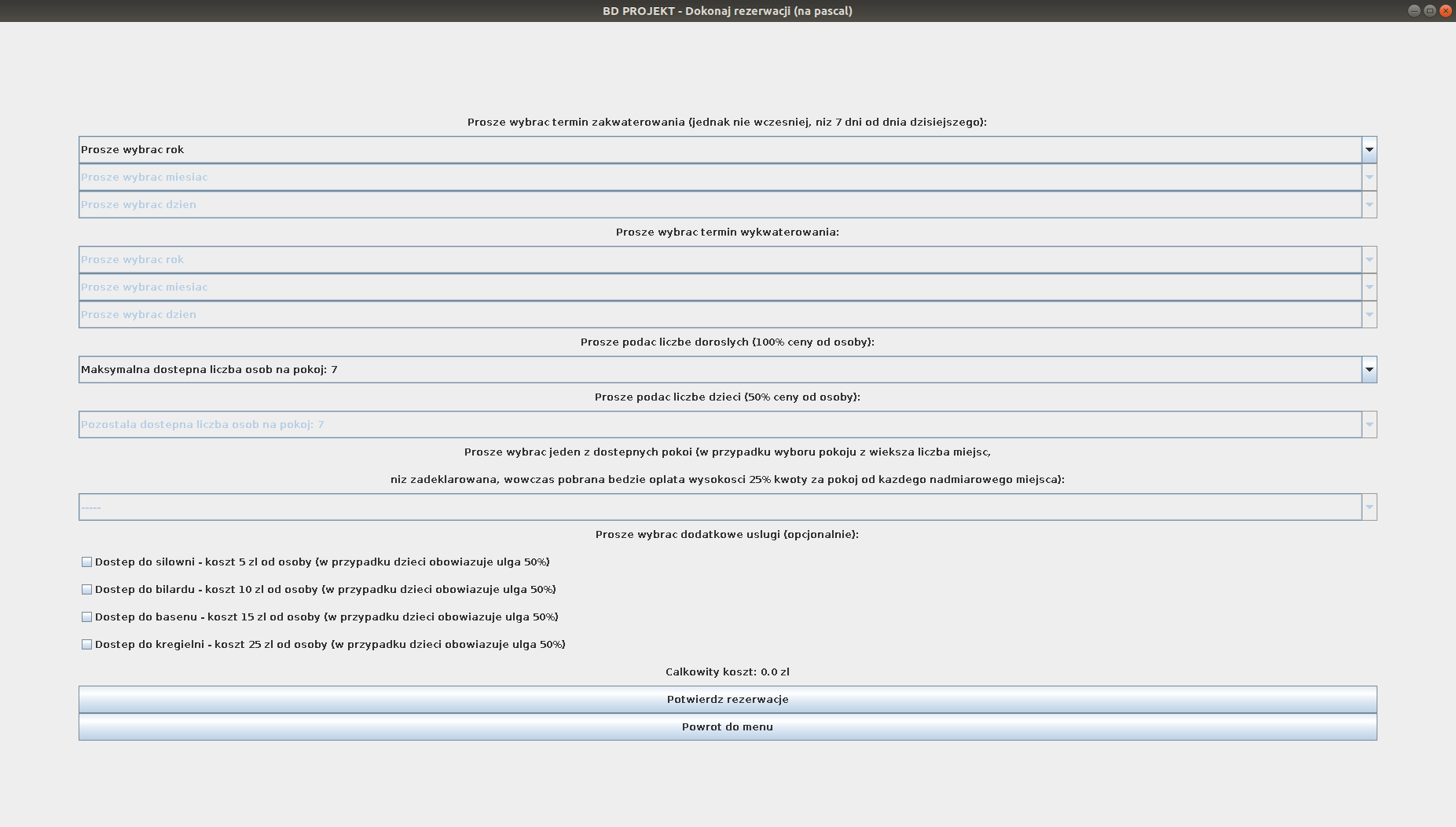


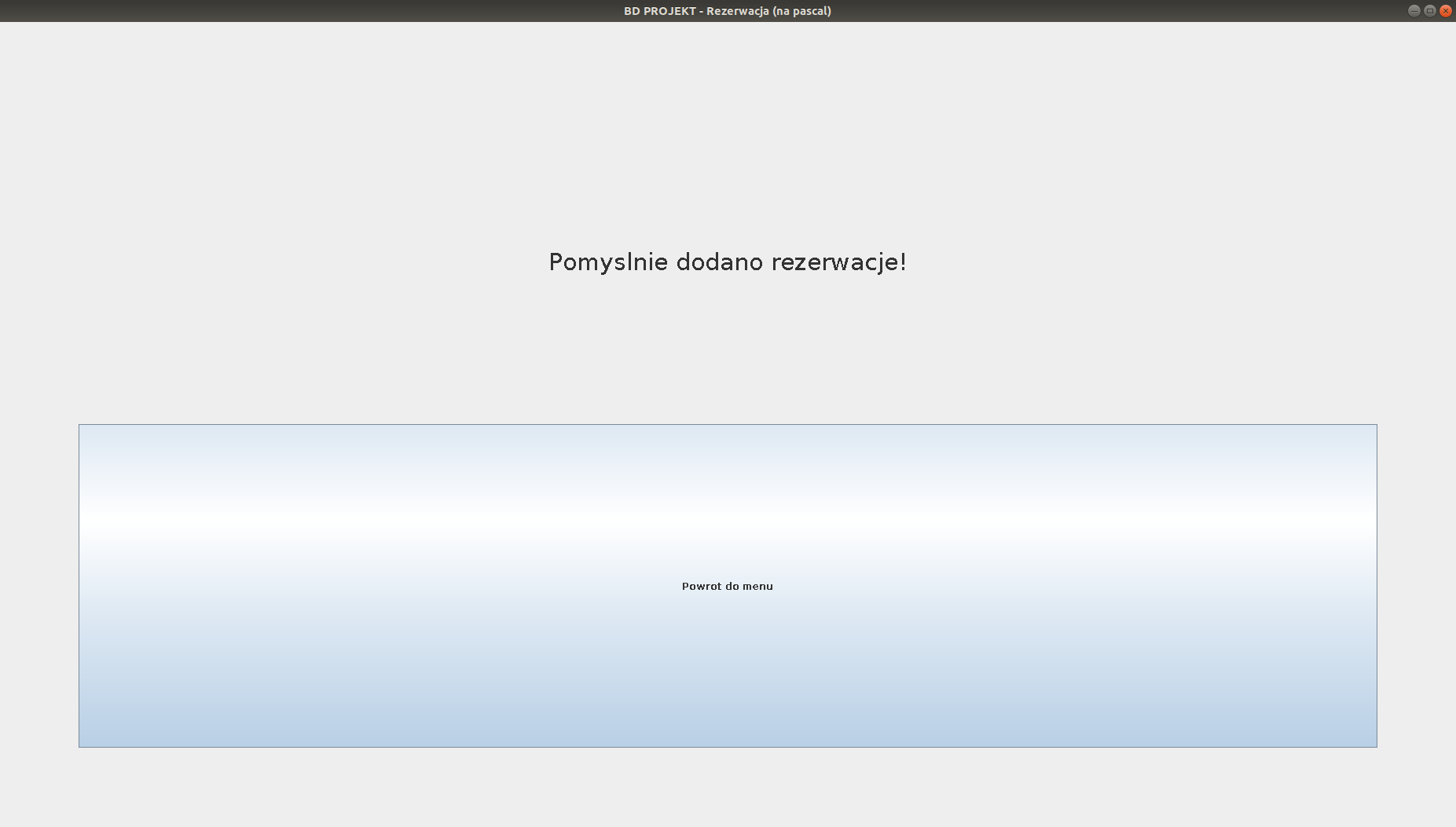
- formularz logowania – każdy z użytkowników ma możliwość zalogowania się na swoje konto. Po wciśnięciu guzika ‘Zaloguj’ do bazy wysyłane jest zapytanie, czy dane logowania są poprawne, a następnie w zależności od rezultatu wyświetlany jest odpowiedni komunikat:



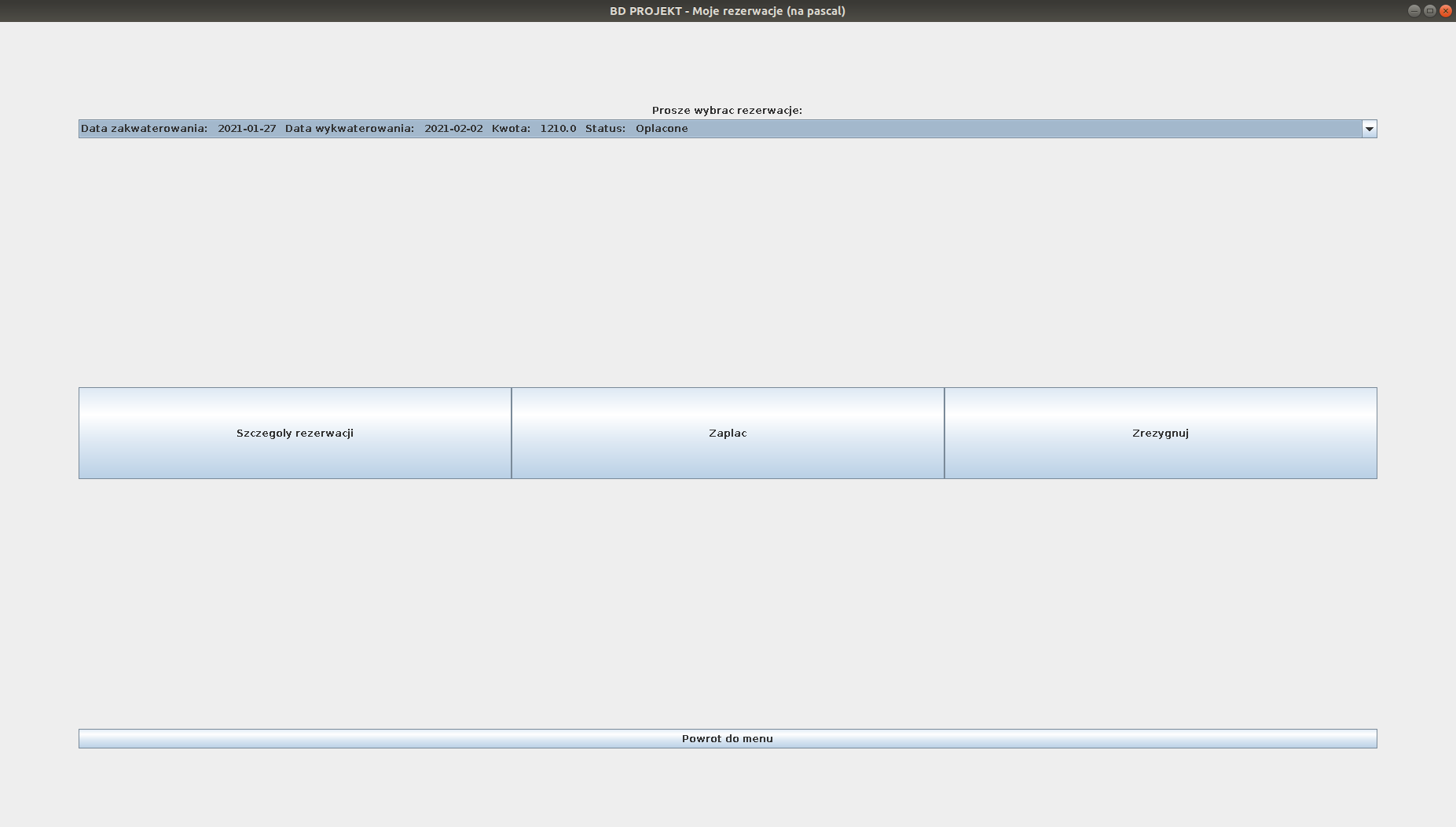


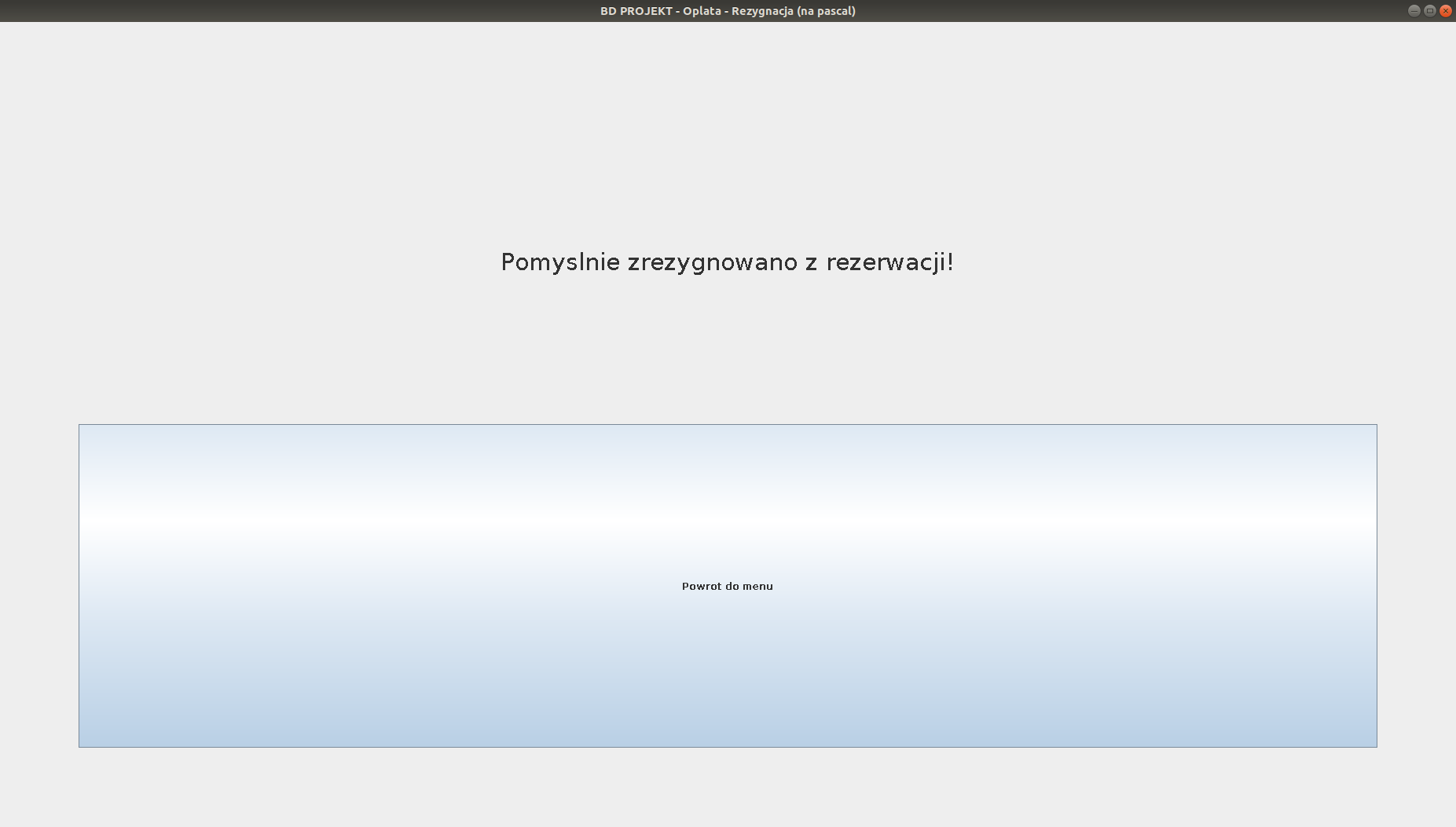
- formularz dokonywania rezerwacji – każdy użytkownik ma możliwość dokonania rezerwacji w hotelu. Użytkownik ma możliwość wyboru daty zakwaterowania oraz wykwaterowania, wyboru pokoju czy dodatkowych usług. Po wciśnięciu guzika ‘Potwierdz rezerwacje’ rozpoczynana jest procedura transakcji, podczas której uruchamiane są odpowiednie wyzwalacze sprawdzające poprawność wprowadzonych danych. W zależności od rezultatu operacji wyświetlany jest odpowiedni komunikat z bazy:



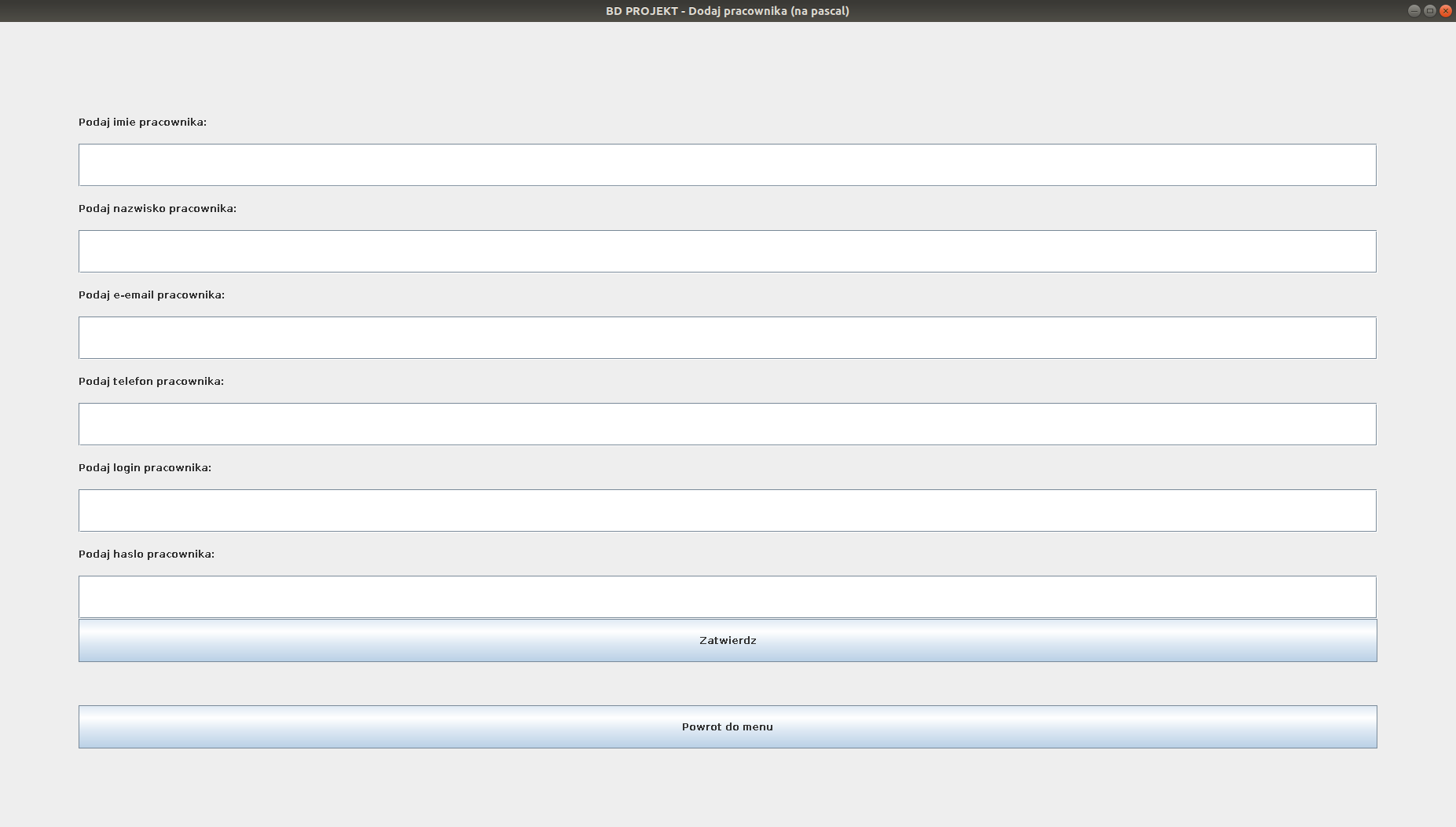


- formularz opłaty bądź rezygnacji z rezerwacji – każdy użytkownik ma możliwość zdecydowania, czy dana rezerwacja ma być opłacona czy też nie. Po wciśnięciu odpowiedniego guzika uruchamiany jest odpowiedni wyzwalacz walidujący, a na koniec z bazy wysyłany jest odpowiedni komunikat:





- formularz rejestracji nowego pracownika – opcja ta dostępna jest tylko dla użytkowników zalogowanych jako HeadAdmin lub Pracownik. Procedura ta jest niemalże taka sama jak w przypadku rejestracji nowego użytkownika – wpisy do bazy różnią się jedynie typem konta. Poniżej przedstawiono wygląd tegoż formularza:



**12.Wizualizacja danych:**

Wszelkie dane z hotelowej bazy danych zostały przedstawione w postaci interaktywnych raportów oraz bezpośredniego podglądu wszystkich tabel. Każda pojedyncza tabela z omawianego projektu ma swój własny podgląd, który zawiera nazwy wszystkich atrybutów oraz odpowiadające im wartości. Poniżej przedstawiono jeden z takich podglądów:

W przypadku raportów wykorzystywane są odpowiednie widoki z bazy danych. Dane w raportach wyświetlane są w ten sam sposób, jak w przypadku podglądu tabel, jednakże posiadają one dodatkowo możliwość operowania na danych – wyszukiwanie rekordów odpowiadających podanej wartości oraz wyświetlanie danych wykorzystując funkcje agregujące, klauzule GROUP BY, HAVING oraz ORDER BY. Stworzono dostęp do sześciu takich raportów:

- Raport o użytkownikach – wykorzystano tutaj widok *uzytkownicy*,

- Raport o rezerwacjach – wykorzystano tutaj widok *RezerwacjeAllInfoView*,

- Raport o pokojach – wykorzystano tutaj widok *pokojeView*,

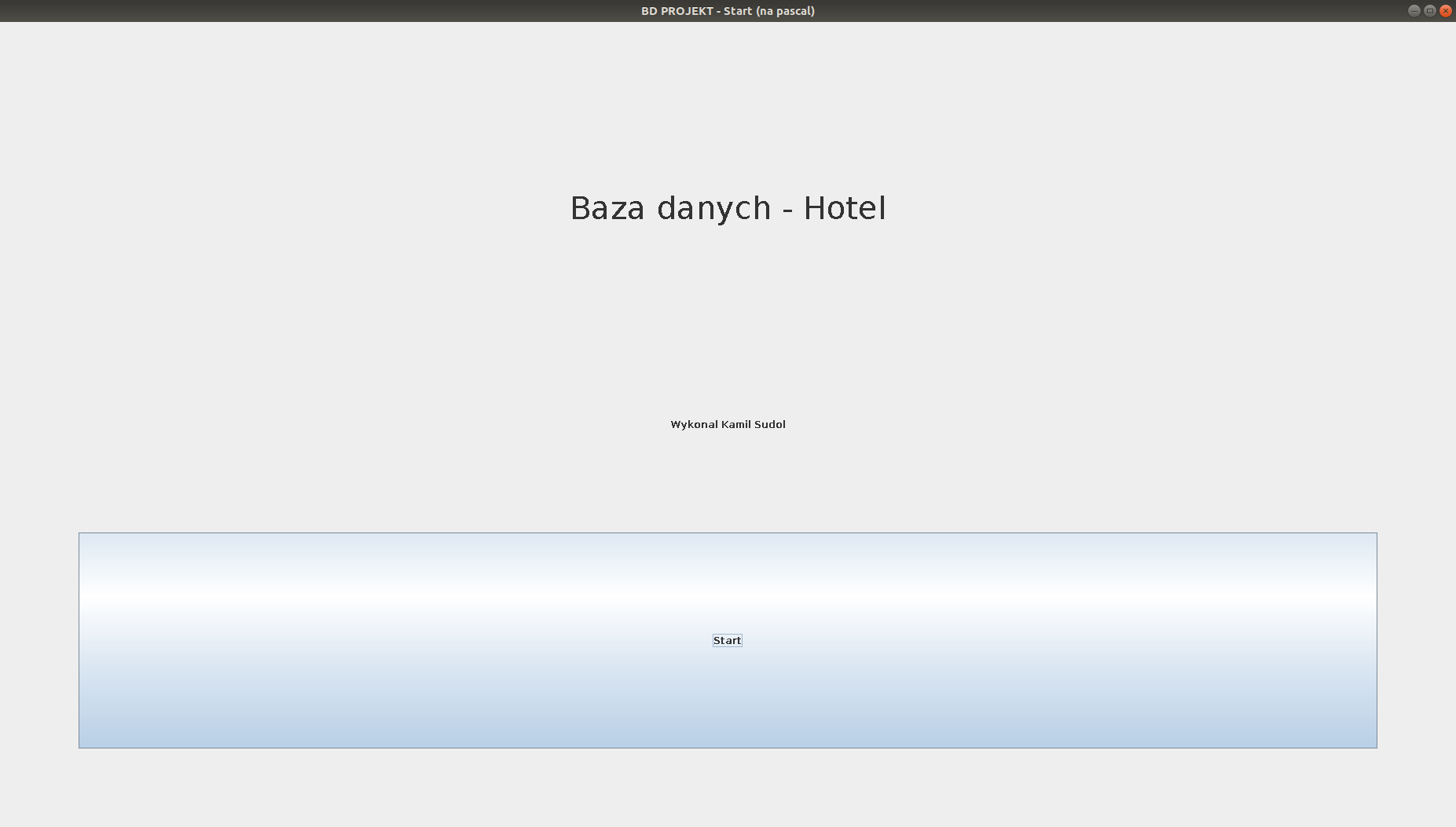
- Raport o zakwaterowanych gościach – wykorzystano tutaj widok *zakwaterowaniView*,

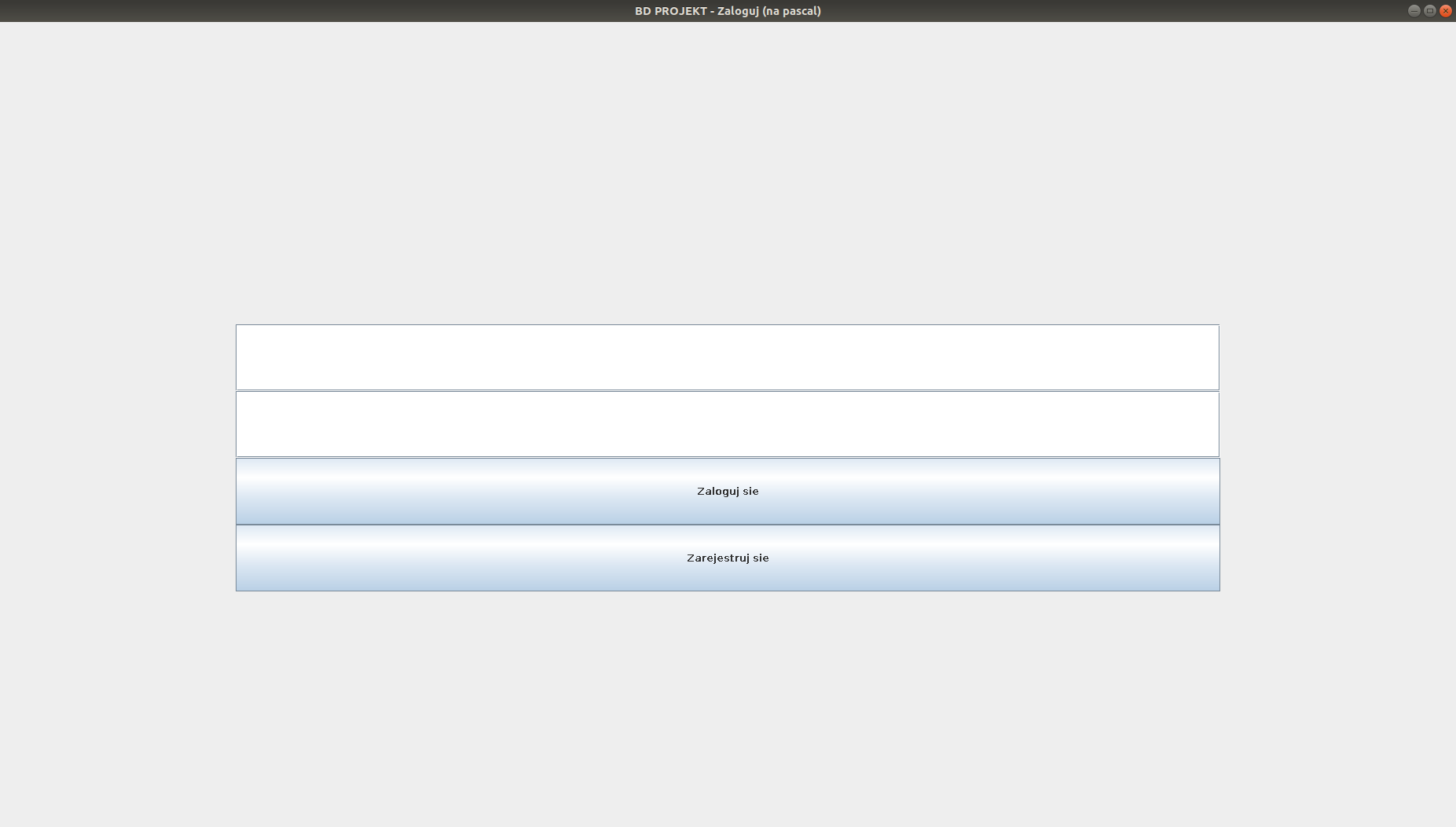
- Raport o rezygnacjach z rezerwacji – wykorzystano tutaj widok *rezygnacjaView*,

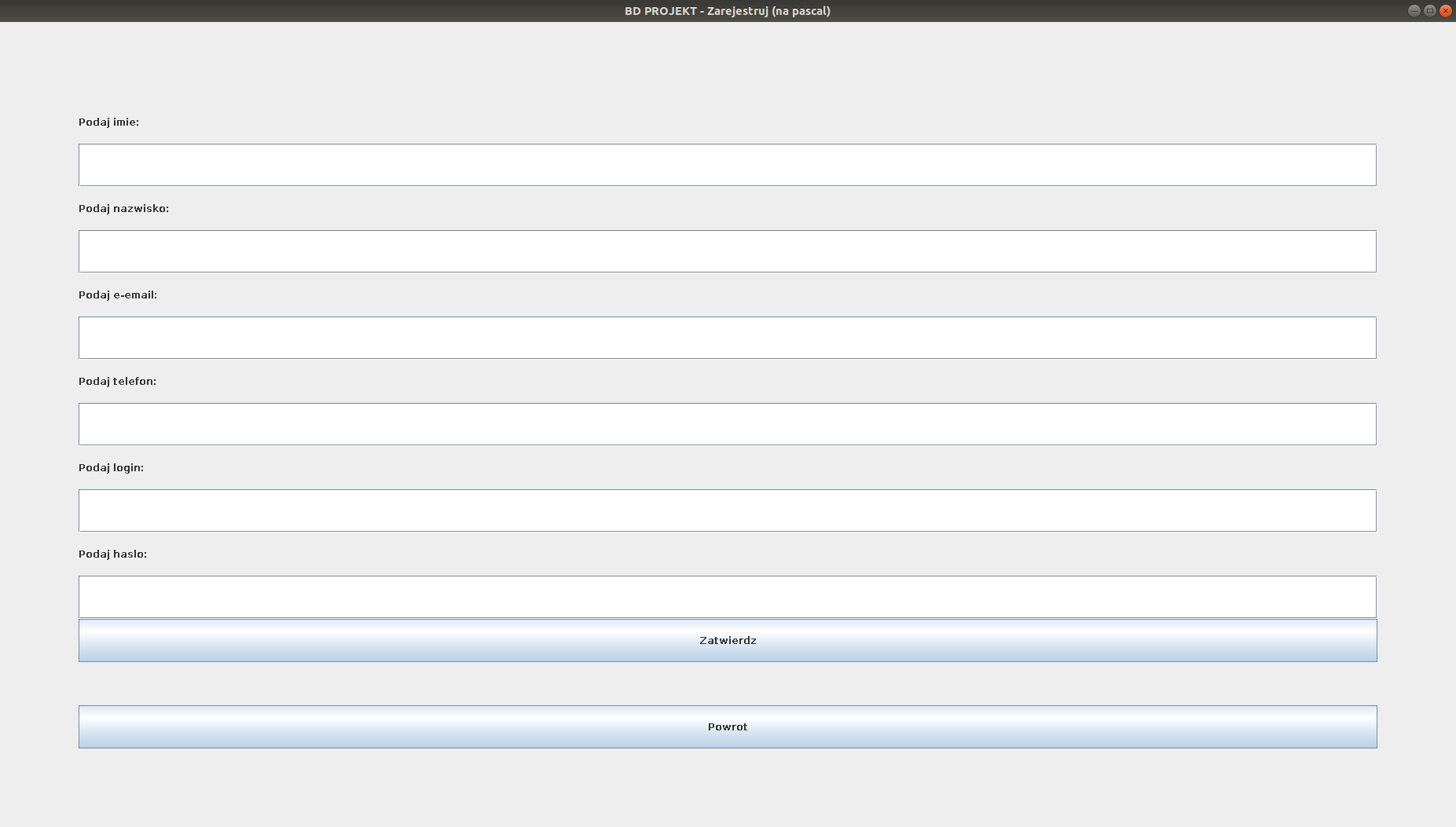
- Raport o zablokowanych użytkownikach – wykorzystano tutaj widok *zablokowaniView*.

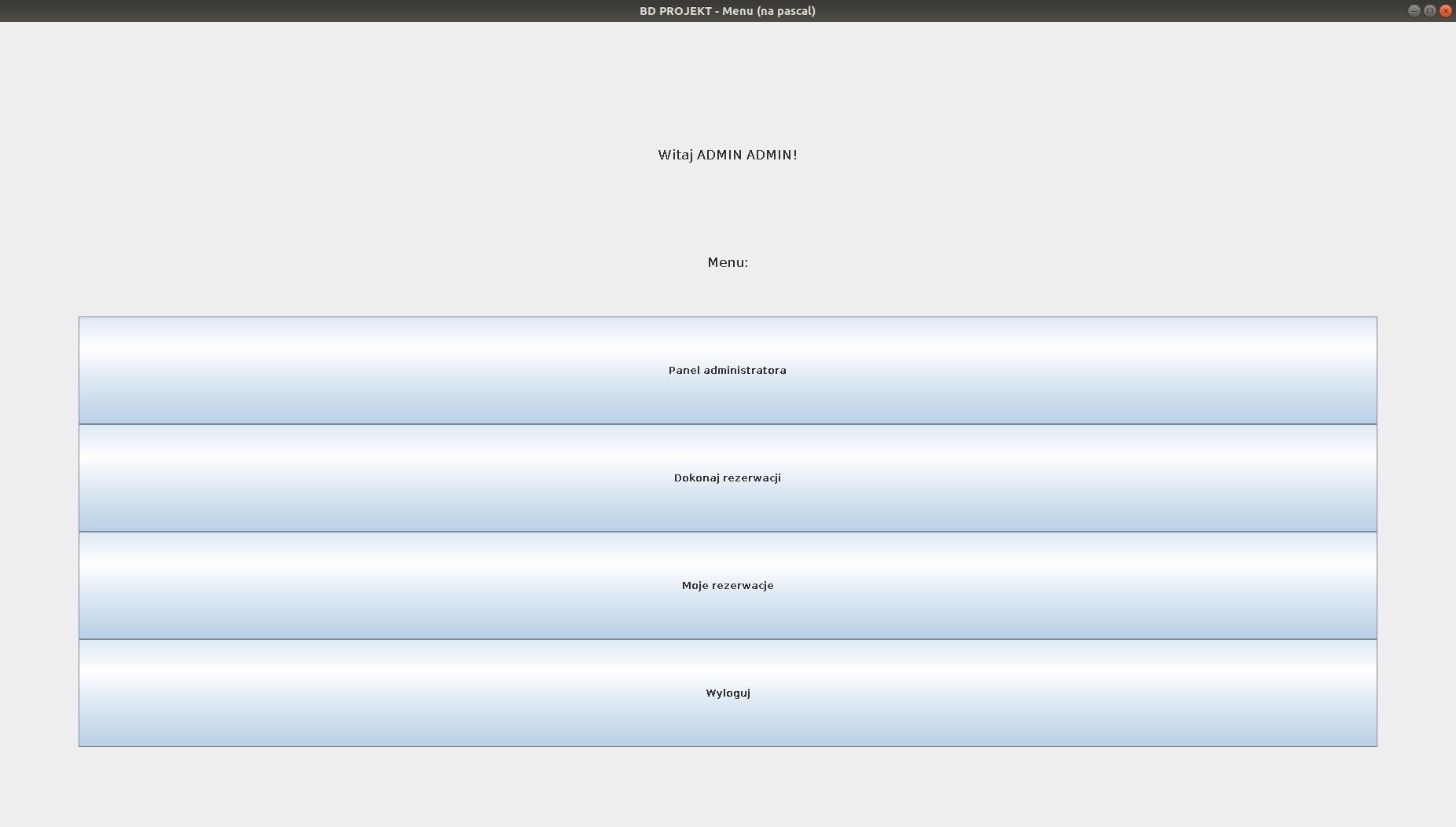
Interaktywność w każdym raporcie poza raportem o rezerwacjach jest taka sama, ta wcześniej wspomniana różni się tylko większym wyborem funkcji agregujących. Poniżej przedstawiono wygląd przykładowych raportów:

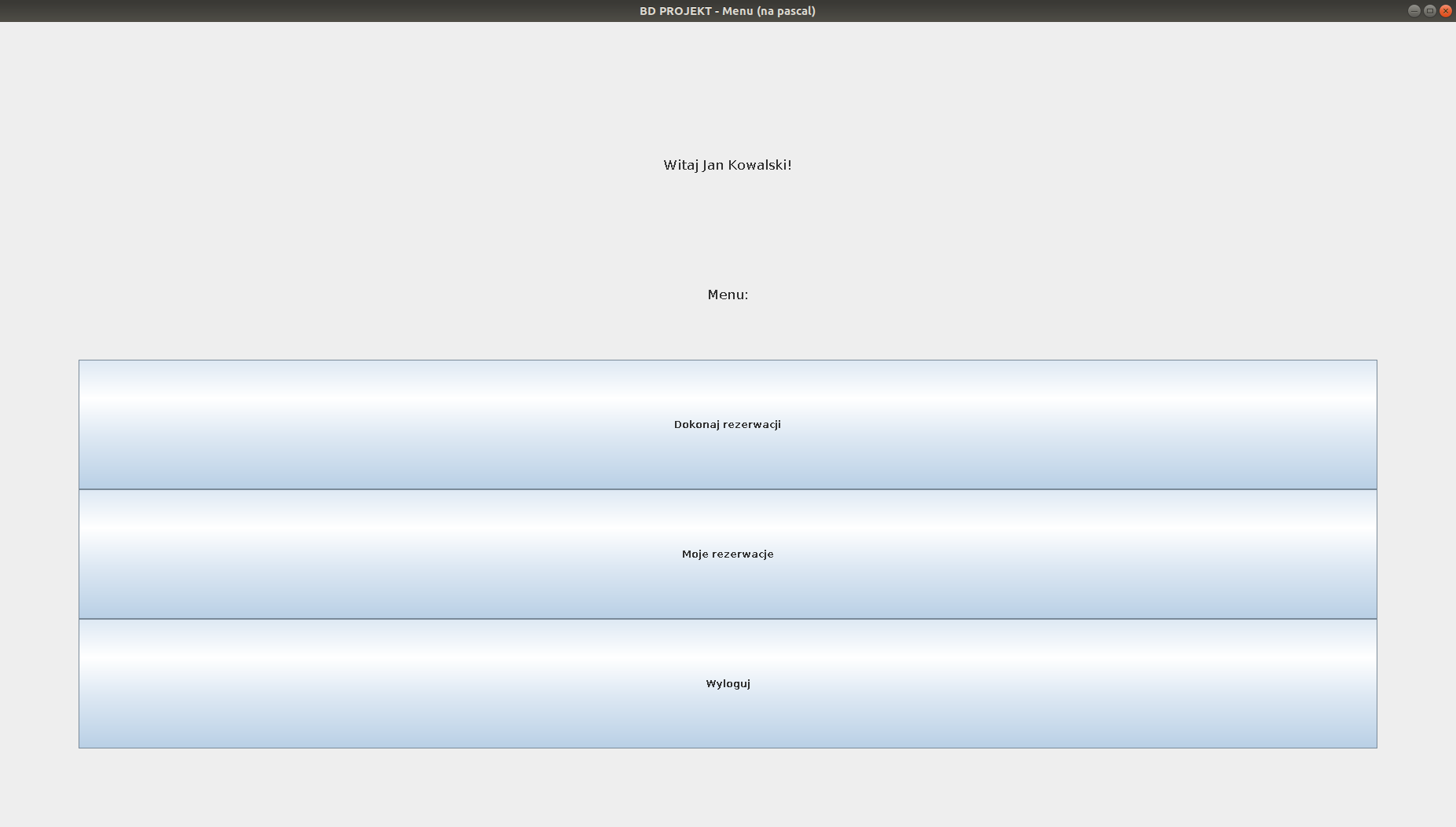
**13.Zdefiniowanie panelu sterowania aplikacji:**

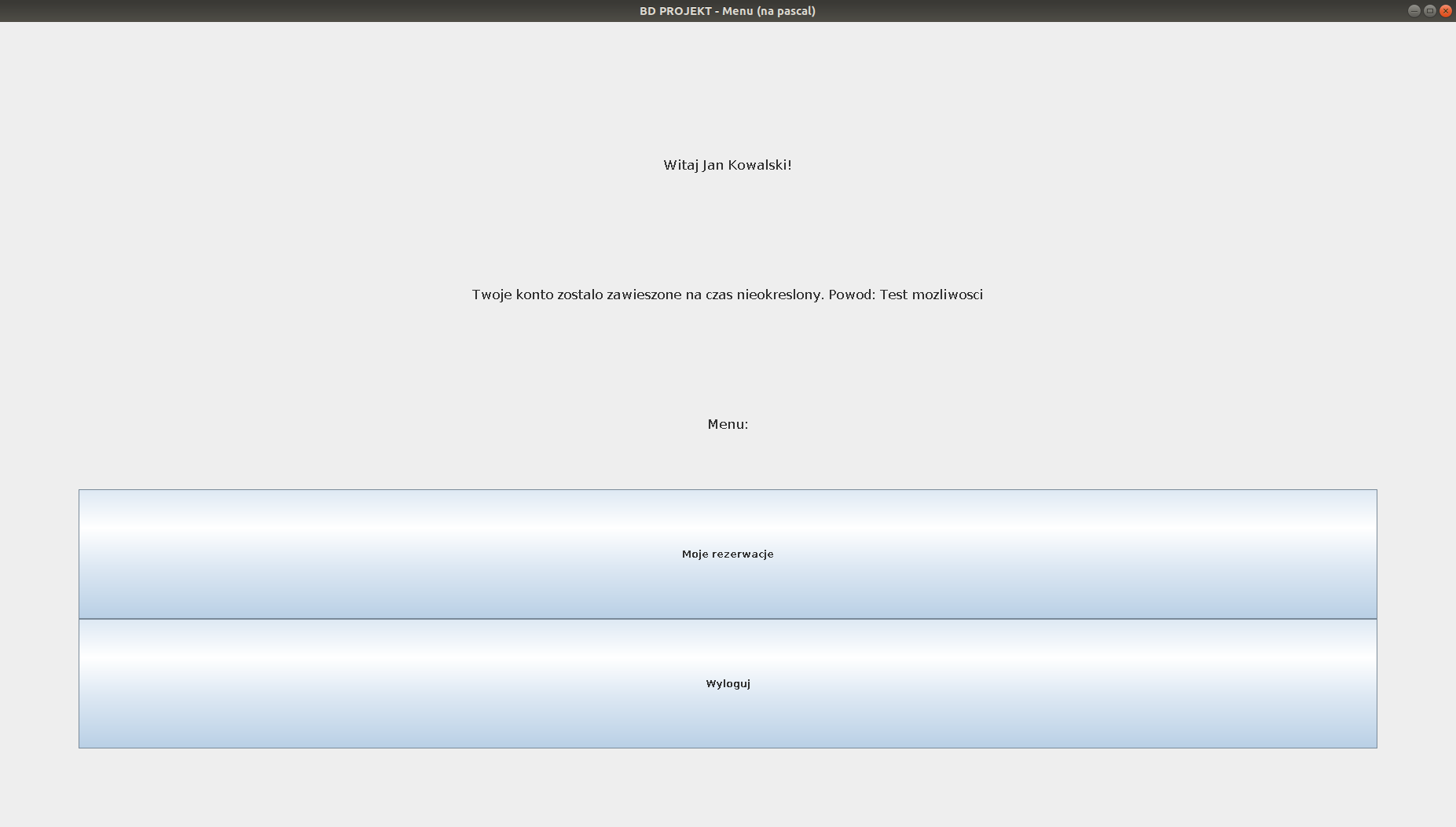
****

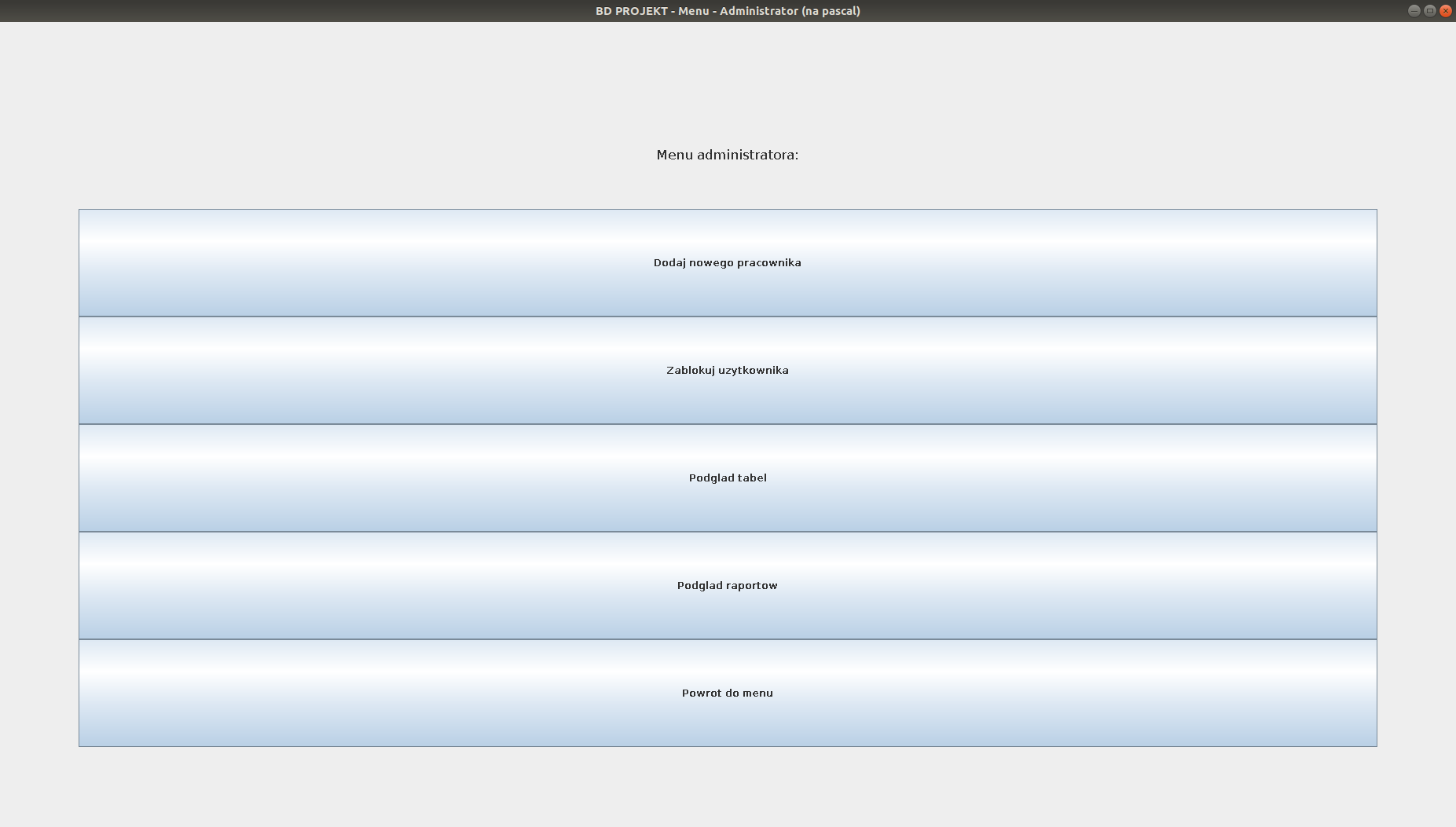
****

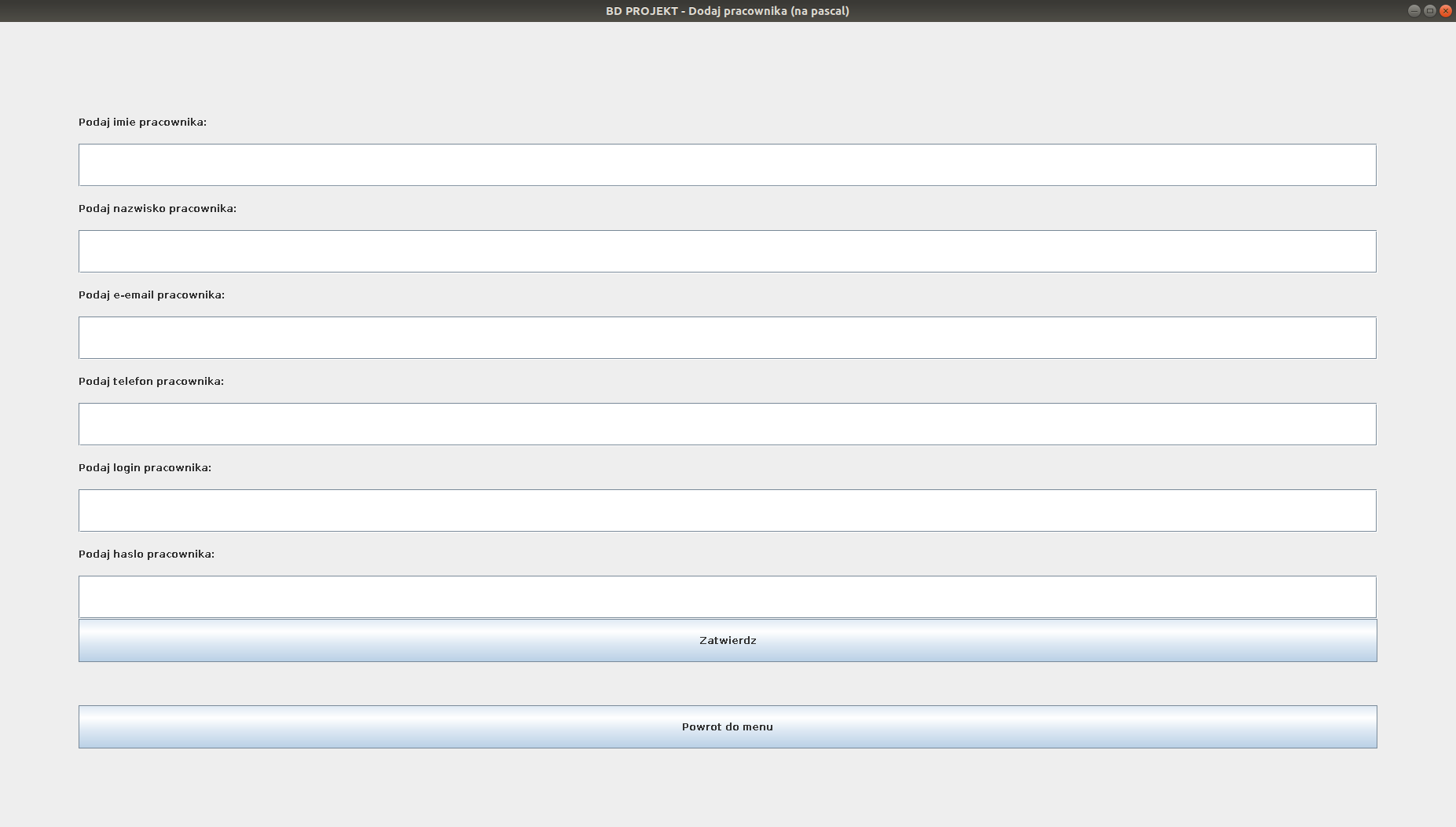
****

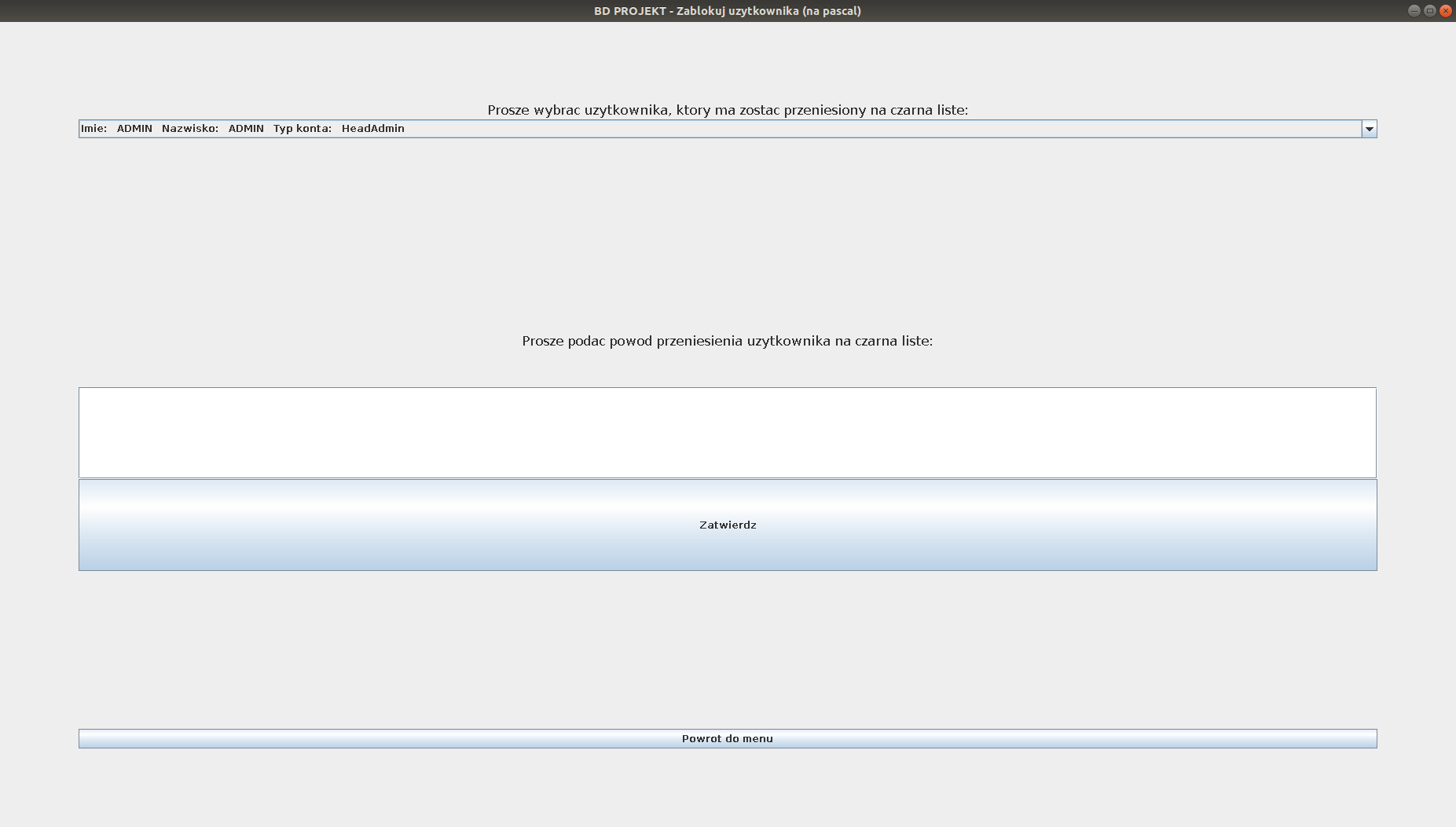
****

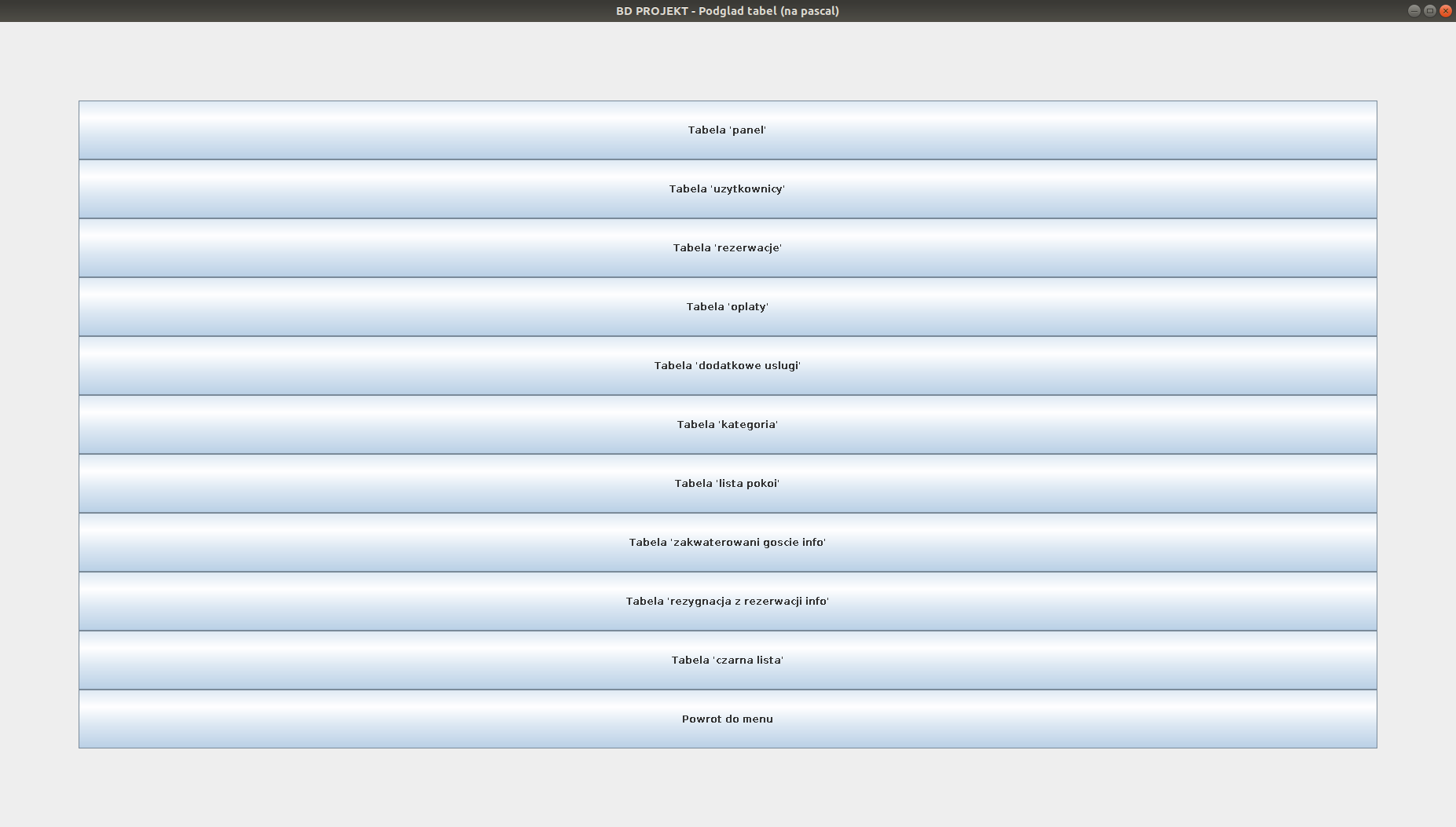
****

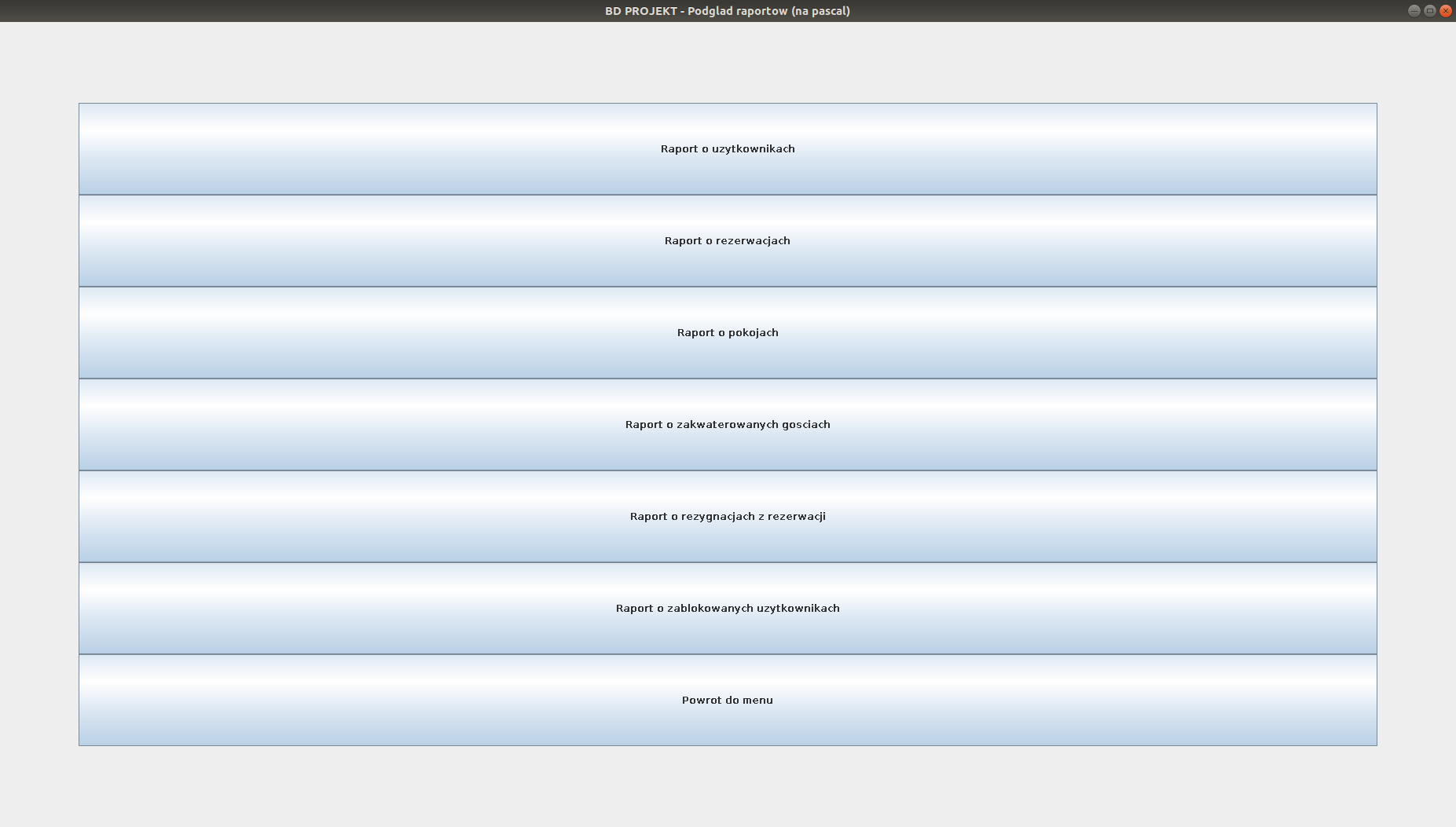
****

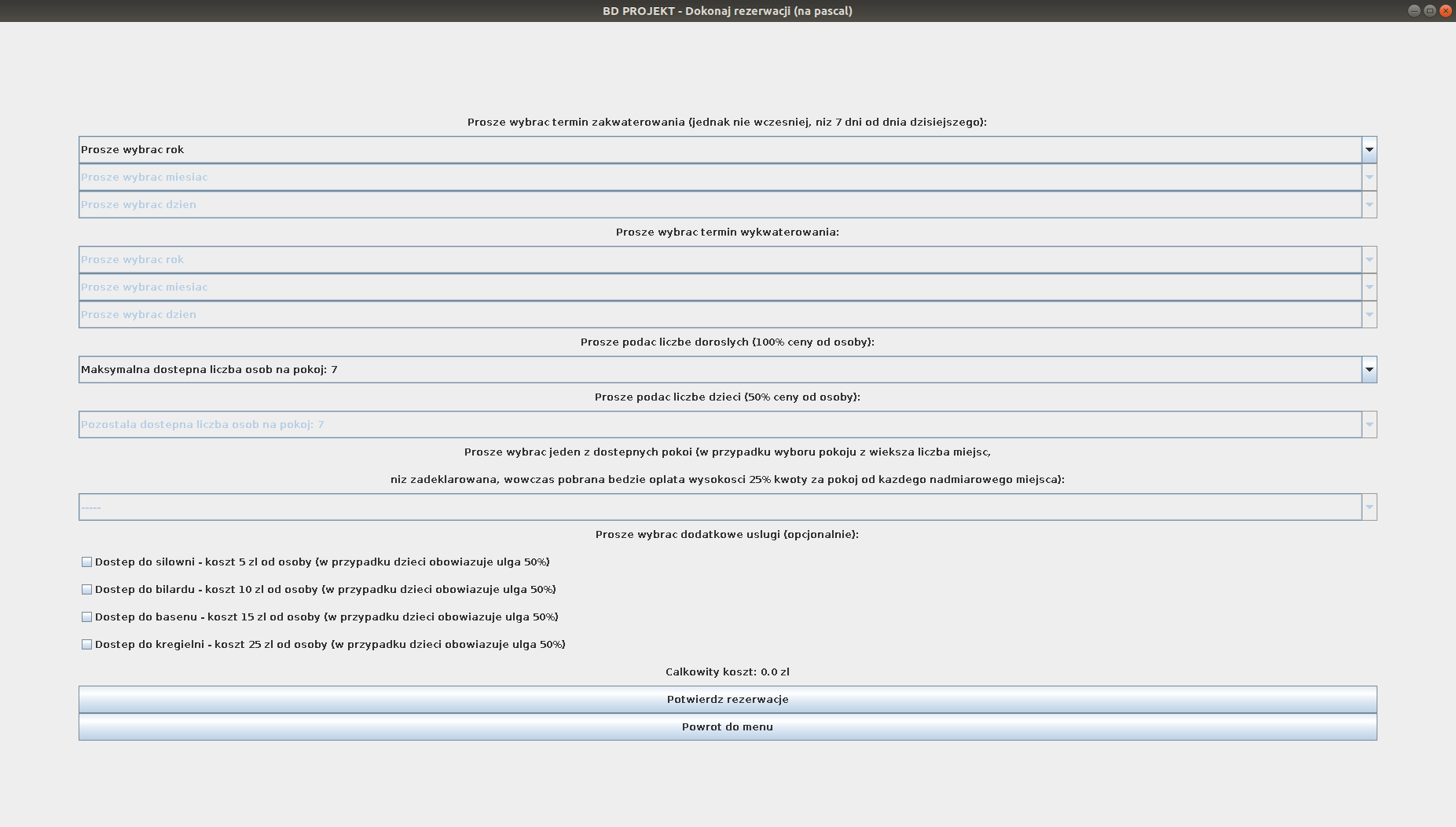
****

****

****

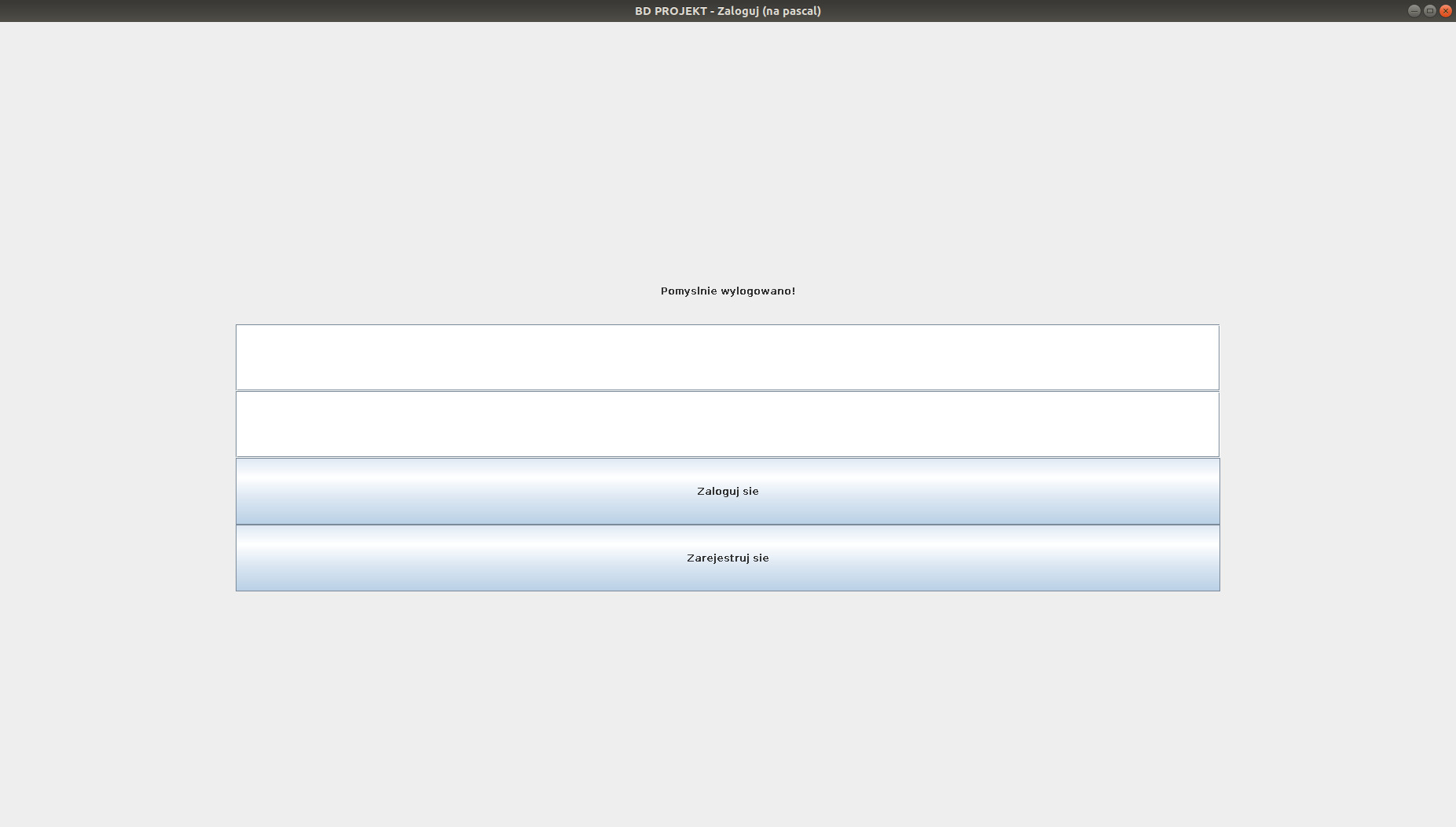
****

****

****

****

****

****

**14.Makropolecenia:**

**V. Dokumentacja**

**15. Wprowadzanie danych:**

**Dane do bazy mogą zostać wprowadzone jedynie poprzez interfejs uzytkownika**

**16. Dokumentacja użytkownika:**

**17.Opracowanie dokumentacji technicznej:**

Kod został udokumentowany w kodzie źródłowym.

**18.Wykaz literatury.**

- <https://dbdiagram.io/>,

- <https://newton.fis.agh.edu.pl/~antek/index.php?sub=db_wykl>,

- <https://www.postgresql.org/docs/>,

- <https://docs.oracle.com/javase/8/docs/>,

- <https://stackoverflow.com/>

. Graficzny interfejs użytkownika w całości został napisany w języku Java. Wykorzystując przy tym rozne poziomy autoryzacji