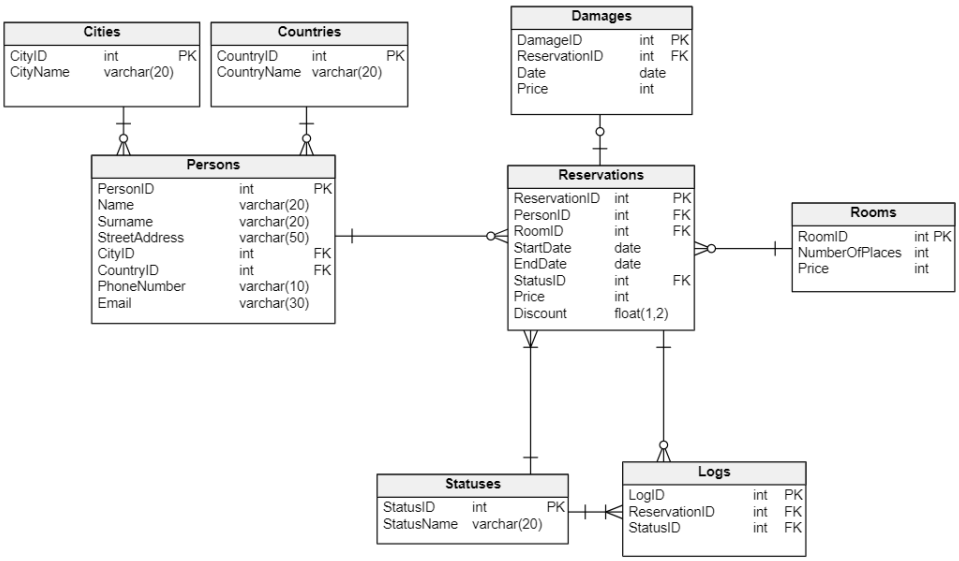
MiniProjekt BedBooker

BedBooker to narzędzie ułatwiające zarządzanie hotelem. Jego głównym zastosowaniem jest obsługa bazodanowa rezerwacji. Projekt tworzony jest przy pomocy MySQL oraz Javy.

Imiona i nazwiska autorów : Gabriela Dumańska, Katarzyna Lisiecka

Tabele



- `Persons` - osoby
  - `PersonID` - identyfikator, klucz główny
  - `Name` - imię
  - `Surname` - nazwisko
  - `StreetAddress` - adres
  - `CityID` - identyfikator miasta, klucz obcy
  - `CountryID` - identyfikator kraju, klucz obcy
  - `PhoneNumber` - numer telefonu
  - `Email` - adres e-mail
- `Cities` - słownik miast
  - `CityID` - identyfikator, klucz główny
  - `CityName` - nazwa miasta
- `Countries` - słownik państw
  - `CountryID` - identyfikator, klucz główny
  - `CountryName` - nazwa państwa

- `Rooms` - pokoje
    - `RoomID` - identyfikator, klucz główny
    - `NumberOfPlaces` - liczba miejsc
    - `Price` - cena za pokój za dobę hotelową
- `Reservations` - rezerwacje
    - `ReservationID` - identyfikator, klucz główny
    - `PersonID` - identyfikator osoby, klucz obcy
    - `RoomID` - identyfikator pokoju, klucz obcy
    - `StartDate` - początek pobytu
    - `EndDate` - koniec pobytu
    - `StatusID` - identyfikator statusu, klucz obcy
    - `Price` - cena rezerwacji
    - `Discount` - zniżka
- `Damages` - pokoje
    - `DamageID` - identyfikator, klucz główny
    - `ReservationID` - identyfikator rezerwacji, klucz obcy
    - `Date` - data zniszczenia
    - `Price` - wartość szkód
- `Logs` - dziennik zmian statusów rezerwacji
    - `LogID` - identyfikator, klucz główny
    - `ReservationID` - identyfikator rezerwacji, klucz obcy
    - `StatusID` - identyfikator statusu, klucz obcy
    - `Date` - data zmiany
- `Statuses` - słownik statusów
    - `StatusID` - identyfikator, klucz główny
    - `StatusName` - nazwa statusu- rezerwacja nowa, potwierdzona i zapłacona, anulowana

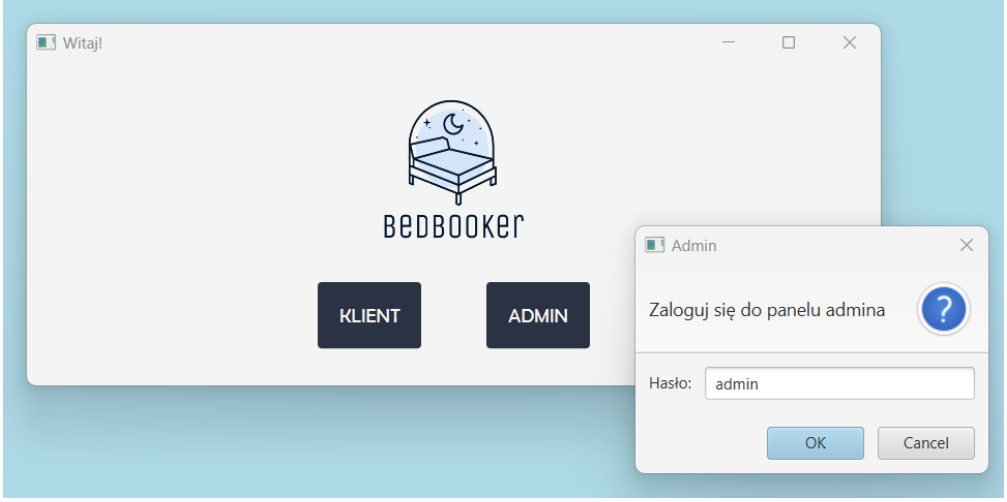
---

Widok administratora

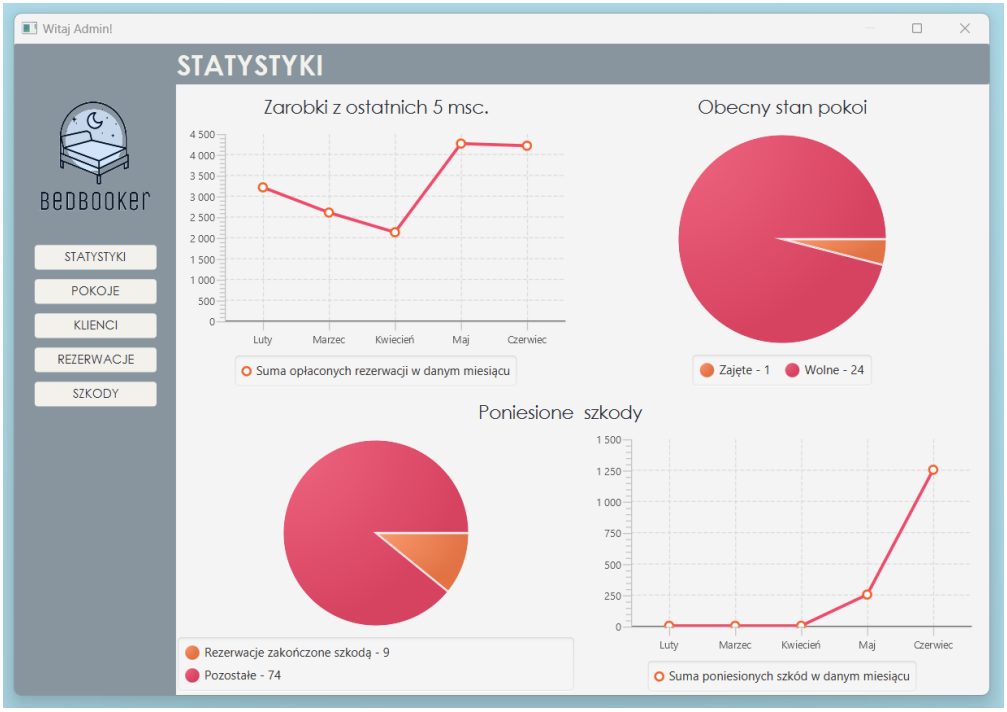
Dla widoku administratora zaprojektowano kluczowe funkcje dla kontrolowania pracy hotelu.

Logowanie do systemu

Widok administratora został zabezpieczony hasłem przed nieporządanymi działaniami.



Statystyki



Ten panel pomaga pracownikowi szybko zorietnować się w jakim stanie jest obecnie hotel.

Zarobki z ostatnich 5 miesięcy

Tę statystykę uzyskano korzystając z następującego widoku:

```
CREATE VIEW Earnings AS
  SELECT EXTRACT(YEAR FROM StartDate) AS Rok,
         EXTRACT(MONTH FROM StartDate) AS Miesiąc,
         SUM(Price * ((100 - Discount) / 100)) AS Zarobki
  FROM Reservations
    WHERE StatusID = 3 AND StartDate <= NOW()
    GROUP BY EXTRACT(YEAR FROM StartDate), EXTRACT(MONTH FROM StartDate)
    ORDER BY Rok DESC, Miesiąc DESC
    LIMIT 5;
```

Zarobki to suma płatności za rezerwacje w danym miesiącu. Upewniono się, że liczone są rezerwacje jedynie opłacone- nieoczekujące, ani nieodwołane, czyli te ze statusem 3. Nie liczono również rezerwacji opłaconych, które się jeszcze nie odbyły.

Obecny stan pokoi

Ten wykres uzyskano z dwóch widoków- liczby pokoi obecni zamieszkałych oraz wszystkich pokoi.

```
CREATE VIEW NumberOfOccupiedRooms AS
  SELECT COUNT(res.ReservationID) AS OccupiedRoomCount
  FROM Rooms r
      LEFT JOIN Reservations res ON r.RoomID = res.RoomID
    WHERE res.StartDate <= CURDATE() AND res.EndDate >= CURDATE();
```

```
CREATE VIEW NumberOfRooms AS
  SELECT COUNT(r.RoomID) AS RoomCount
  FROM Rooms r
```

W kolejnym punkcie w razie potrzeby wyciągnięcia dwóch liczb z bazy danych wykonano to w ramach jednego widoku, co uznano za lepsze rozwiązanie.

Poniesione szkody

Pierwszy wykres kołowy to prosta statystyka unikalnych rezerwacji z tabeli Damages do pozostałych rezerwacji.

```
  CREATE VIEW DamagesPerReservations AS
    SELECT
    (SELECT COUNT(DISTINCT ReservationID) FROM Damages) AS
ReservationsWithDamages,
    (SELECT COUNT(ReservationID) FROM Reservations
                                WHERE EndDate < NOW()) AS UniqueReservations;
```

Takie rozwiązanie wyciągania dwóch liczb uznano za bardziej czytelne- od razu wiadomo do czego są potrzebne. Redukuje to również liczbę widoków.

Drugi wykres liniowy informuje o poniesionych szkodach w ostatich 5 miesiącach. W tym przypadku należało zmienić podejście względem statystki zarobków z ostatnich 5 miesięcy. Możemy się spodziewać, że w każdym miesiącu wystąpi chociaż jedna rezerwacja, więc wystarczyło z tabeli Reservations pogrupowane dane z ostatnich 5 miesięcy. Szkody nie muszą występować co miesiąc. Dlatego zastosowano inne podejście z zastosowaniem tabeli pomocniczej Miesiące.

Tabela zawiera wszystkie miesiące od 2014 do 2034 w formacie np. '2024-06-01'

- `Miesiące`
    - `Miesiąc` - typu date

```
CREATE VIEW SumOfDamages AS
  SELECT EXTRACT(YEAR FROM m.Miesiąc)  AS Rok,
       EXTRACT(MONTH FROM m.Miesiąc)   AS Miesiąc,
       COALESCE(SUM(d.Price), 0)       AS Szkody
  FROM Miesiące m
      LEFT JOIN Damages d ON date_format(d.Date, '%Y-%m-01') = m.Miesiąc
    WHERE EXTRACT(YEAR FROM m.Miesiąc) <= YEAR(NOW())
      AND EXTRACT(MONTH FROM m.Miesiąc) <= MONTH(NOW())
    GROUP BY EXTRACT(YEAR FROM m.Miesiąc), m.Miesiąc
    ORDER BY EXTRACT(YEAR FROM m.Miesiąc) DESC,
             EXTRACT(MONTH FROM m.Miesiąc) DESC
    LIMIT 5;
```

Pokoje

To prosty ekran do przeglądania dostępnych pokoi oraz dodawania nowych. Przycisk plus ukazuje okno do dodawania nowych pokoi, a odświeżenie ładuje ponownie wyniki, by pokazywały się dodane na nowo pokoje.

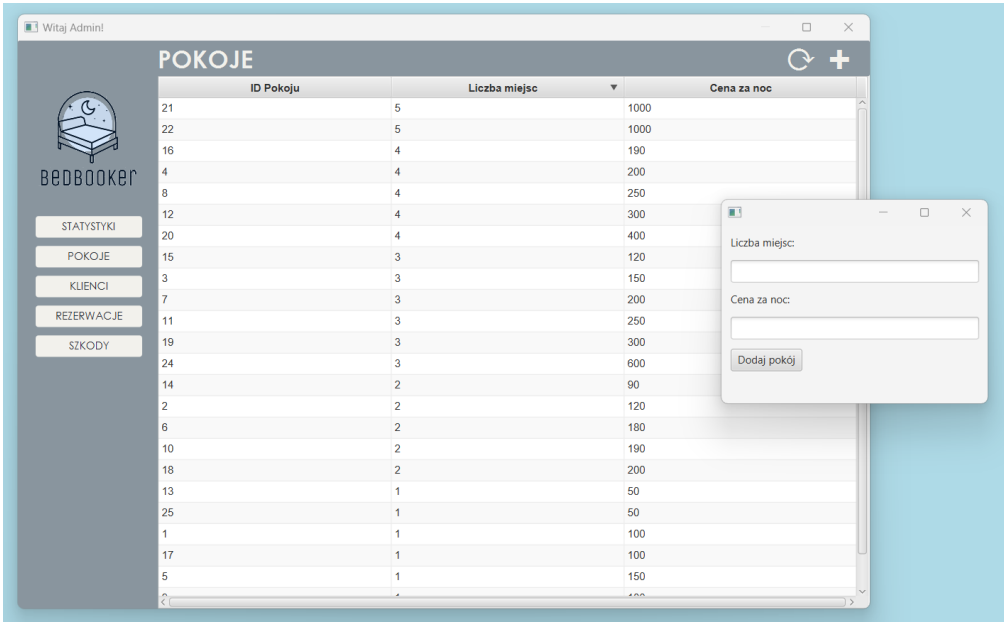
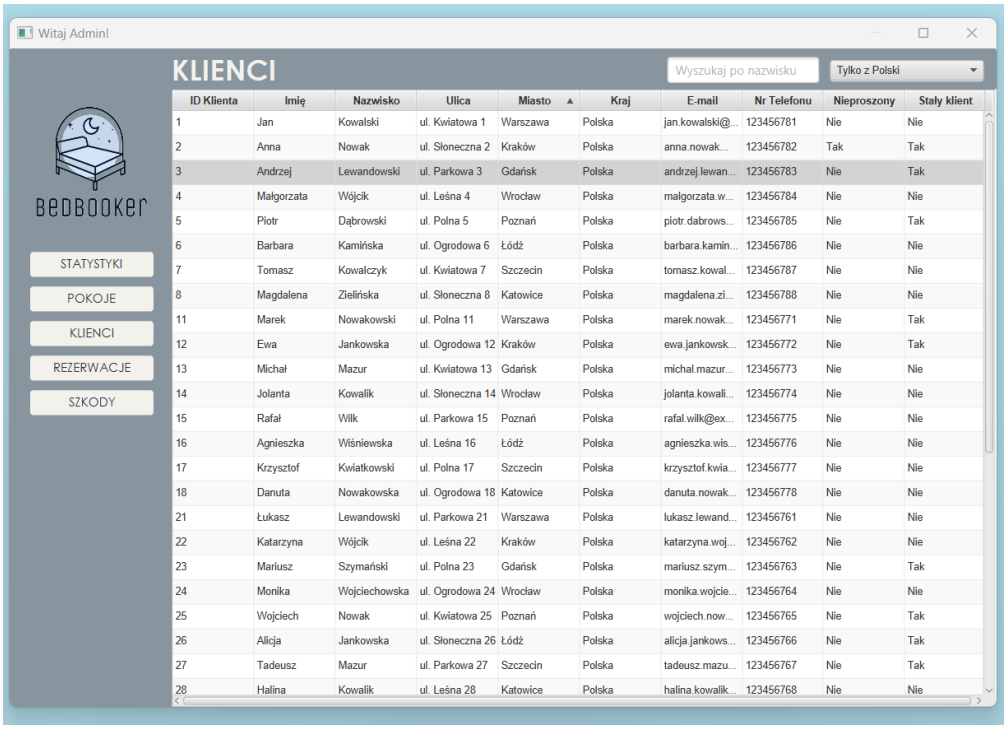


Tabela pokoi to zwykły SELECT z tabeli Rooms. Dodawawanie pokoju obsługuje procedura:

```
CREATE PROCEDURE AddRoom(IN p_NumberOfPlaces int, IN p_Price int)
  BEGIN
      INSERT INTO Rooms (NumberOfPlaces, Price)
      VALUES (p_NumberOfPlaces, p_Price);
  END;
```

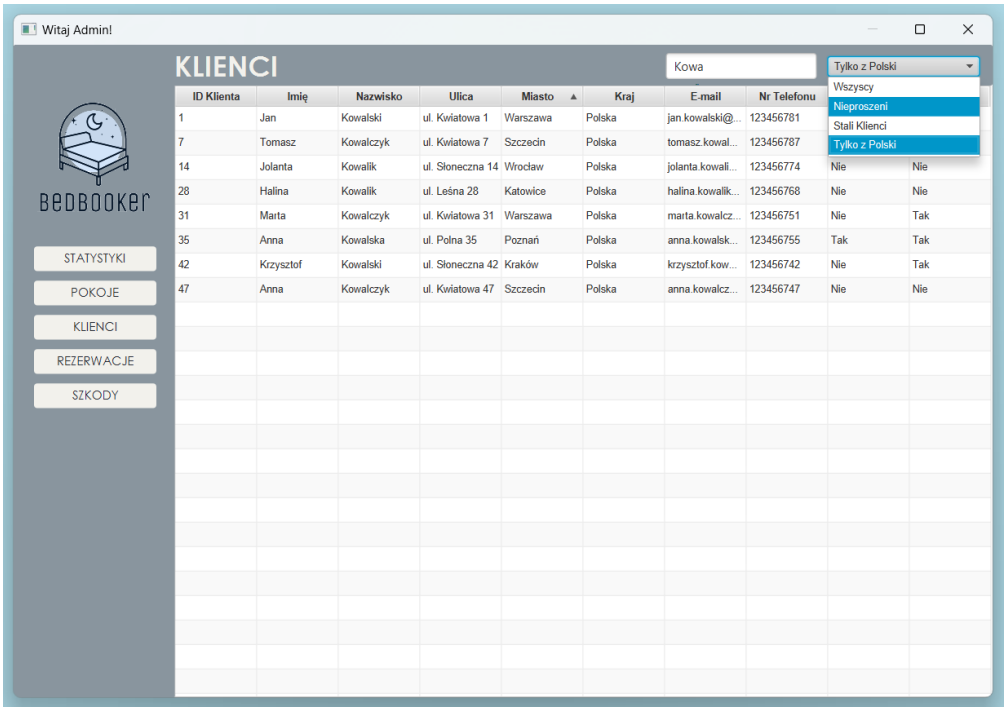
Klienci

Ekran z listą wszystkich klientów, którzy złożyli rezerwację w hotelu.



ID Klienta	Imię	Nazwisko	Ulica	Miasto	Kraj	E-mail	Nr Telefonu	Nieproszony	Stały klient
1	Jan	Kowalski	ul. Kwiatowa 1	Warszawa	Polska	jan.kowalski@...	123456781	Nie	Nie
2	Anna	Nowak	ul. Słoneczna 2	Kraków	Polska	anna.nowak...	123456782	Tak	Tak
3	Andrzej	Lewandowski	ul. Parkowa 3	Gdańsk	Polska	andrzej.lewan...	123456783	Nie	Tak
4	Małgorzata	Wójcik	ul. Leśna 4	Wrocław	Polska	malgorzata.w...	123456784	Nie	Nie
5	Piotr	Dąbrowski	ul. Polna 5	Poznań	Polska	piotr.dabrows...	123456785	Nie	Tak
6	Barbara	Kamińska	ul. Ogrodowa 6	Łódź	Polska	barbara.kamin...	123456786	Nie	Nie
7	Tomasz	Kowalczyk	ul. Kwiatowa 7	Szczecin	Polska	tomasz.kowal...	123456787	Nie	Nie
8	Magdalena	Zielińska	ul. Słoneczna 8	Katowice	Polska	magdalena.zi...	123456788	Nie	Nie
11	Marek	Nowakowski	ul. Polna 11	Warszawa	Polska	marek.nowak...	123456771	Nie	Tak
12	Ewa	Jankowska	ul. Ogrodowa 12	Kraków	Polska	ewa.jankowsk...	123456772	Nie	Tak
13	Michał	Mazur	ul. Kwiatowa 13	Gdańsk	Polska	michal.mazur...	123456773	Nie	Nie
14	Jolanta	Kowalik	ul. Słoneczna 14	Wrocław	Polska	jolanta.kowali...	123456774	Nie	Nie
15	Rafał	Wilk	ul. Parkowa 15	Poznań	Polska	rafal.wilk@ex...	123456775	Nie	Nie
16	Agnieszka	Wiśniewska	ul. Leśna 16	Łódź	Polska	agnieszka.wis...	123456776	Nie	Nie
17	Krzysztof	Kwiatkowski	ul. Polna 17	Szczecin	Polska	krzysztof.kwia...	123456777	Nie	Nie
18	Danuta	Nowakowska	ul. Ogrodowa 18	Katowice	Polska	danuta.nowak...	123456778	Nie	Nie
21	Łukasz	Lewandowski	ul. Parkowa 21	Warszawa	Polska	lukasz.lewand...	123456761	Nie	Nie
22	Katarzyna	Wójcik	ul. Leśna 22	Kraków	Polska	katarzyna.woj...	123456762	Nie	Nie
23	Mariusz	Szymański	ul. Polna 23	Gdańsk	Polska	mariusz.szym...	123456763	Nie	Tak
24	Monika	Wojciechowska	ul. Ogrodowa 24	Wrocław	Polska	monika.wojcie...	123456764	Nie	Nie
25	Wojciech	Nowak	ul. Kwiatowa 25	Poznań	Polska	wojciech.now...	123456765	Nie	Tak
26	Alicja	Jankowska	ul. Słoneczna 26	Łódź	Polska	alicja.jankows...	123456766	Nie	Tak
27	Tadeusz	Mazur	ul. Parkowa 27	Szczecin	Polska	tadeusz.mazu...	123456767	Nie	Tak
28	Halina	Kowalik	ul. Leśna 28	Katowice	Polska	halina.kowalik...	123456768	Nie	Nie

Można wyszukiwać ich po nazwisku, po kilku filtrach oraz sortując dowolną kolumnę rosnąco lub malejąco.



ID Klienta	Imię	Nazwisko	Ulica	Miasto	Kraj	E-mail	Nr Telefonu	Nieproszony	Stały klient
1	Jan	Kowalski	ul. Kwiatowa 1	Warszawa	Polska	jan.kowalski@...	123456781		
7	Tomasz	Kowalczyk	ul. Kwiatowa 7	Szczecin	Polska	tomasz.kowal...	123456787	Nie	Nie
14	Jolanta	Kowalik	ul. Słoneczna 14	Wrocław	Polska	jolanta.kowali...	123456774	Nie	Nie
28	Halina	Kowalik	ul. Leśna 28	Katowice	Polska	halina.kowalik...	123456768	Nie	Nie
31	Marta	Kowalczyk	ul. Kwiatowa 31	Warszawa	Polska	marta.kowalcz...	123456751	Nie	Tak
35	Anna	Kowalska	ul. Polna 35	Poznań	Polska	anna.kowalsk...	123456755	Tak	Tak
42	Krzysztof	Kowalski	ul. Słoneczna 42	Kraków	Polska	krzysztof.kow...	123456742	Nie	Tak
47	Anna	Kowalczyk	ul. Kwiatowa 47	Szczecin	Polska	anna.kowalcz...	123456747	Nie	Nie

```
CREATE VIEW CustomerFullInfo AS
  SELECT p.PersonID AS PersonID,
         p.Name     AS Name,
         p.Surname  AS Surname,
         p.StreetAddress AS Address,
         c.CountryName   AS Country,
         ci.CityName     AS City,
         p.PhoneNumber   AS PhoneNumber,
         p.Email         AS Email,
         (CASE WHEN rc.PersonID is not null
             THEN 'True' ELSE 'False' END)      AS IsRegular,
         (CASE WHEN bc.PersonID is not null
             THEN 'True' ELSE 'False' END)      AS IsBanned
  FROM Persons p join Countries c on p.CountryID = c.CountryID
                 join Cities ci on p.CityID = ci.CityID
                 left join RegularCustomers rc on p.PersonID = rc.PersonID
                 left join BannedCustomers bc on p.PersonID = bc.PersonID;
```

Klienci nieproszeni to tacy, którzy wykonali więcej niż 2 szkody lub na ponad 1000 złotych. Na tych klientów nie można złożyć rezerwacji.

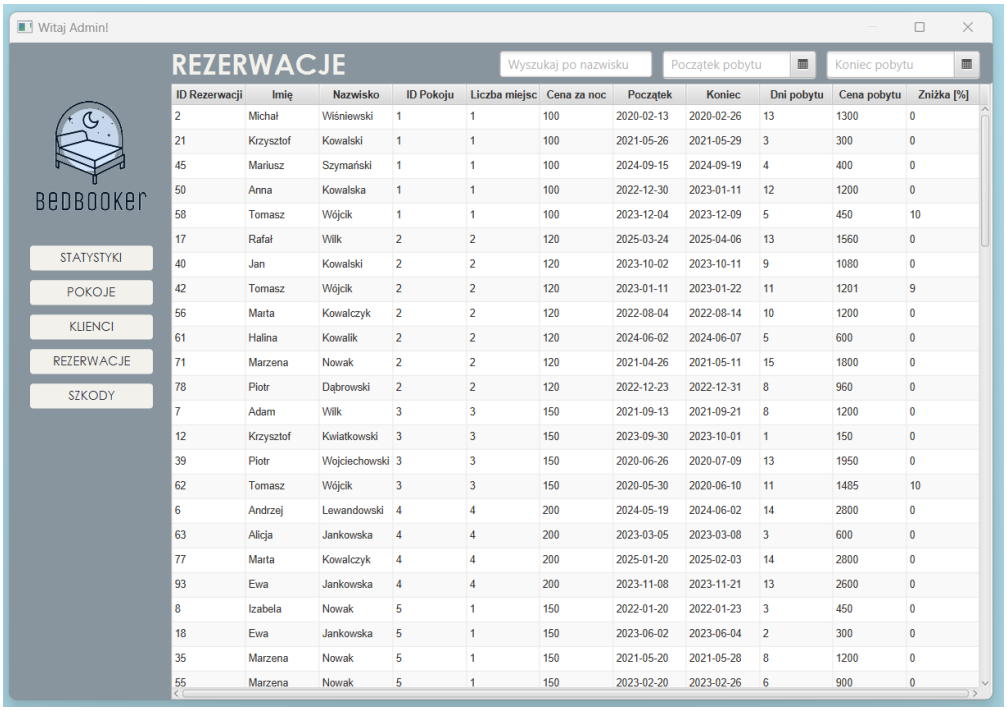
```
CREATE VIEW BannedCustomers AS
  SELECT P.PersonID        AS PersonID,
         P.Name            AS Name,
         P.Surname         AS Surname,
         COUNT(D.DamageID) AS DamageCount,
         SUM(D.Price)      AS TotalDamage
  FROM Persons P join Reservations R on P.PersonID = R.PersonID
                 join Damages D on R.ReservationID = D.ReservationID
    GROUP BY P.PersonID, P.Name, P.Surname
    HAVING (COUNT(D.DamageID) > 2) or (SUM(D.Price) > 1000);
```

Stali klienci złożyli więcej niż 4 rezerwacje lub wydali w hotelu więcej niż 3000 złotych. Naliczany jest dla nich rabat procentowy na kolejne rezerwacje.

```
CREATE VIEW RegularCustomers AS
  SELECT P.PersonID                         AS PersonID,
         P.Name                             AS Name,
         P.Surname                          AS Surname,
         COUNT(R.ReservationID)             AS ReservationCount,
         SUM(R.Price)                       AS TotalReservationPrice,
         (4 + floor((SUM(R.Price) / 1000))) AS Discount
  FROM Persons P JOIN Reservations R on P.PersonID = R.PersonID
    GROUP BY P.PersonID, P.Name, P.Surname
    HAVING (COUNT(R.ReservationID) > 4) OR (SUM(R.Price) > 3000);
```

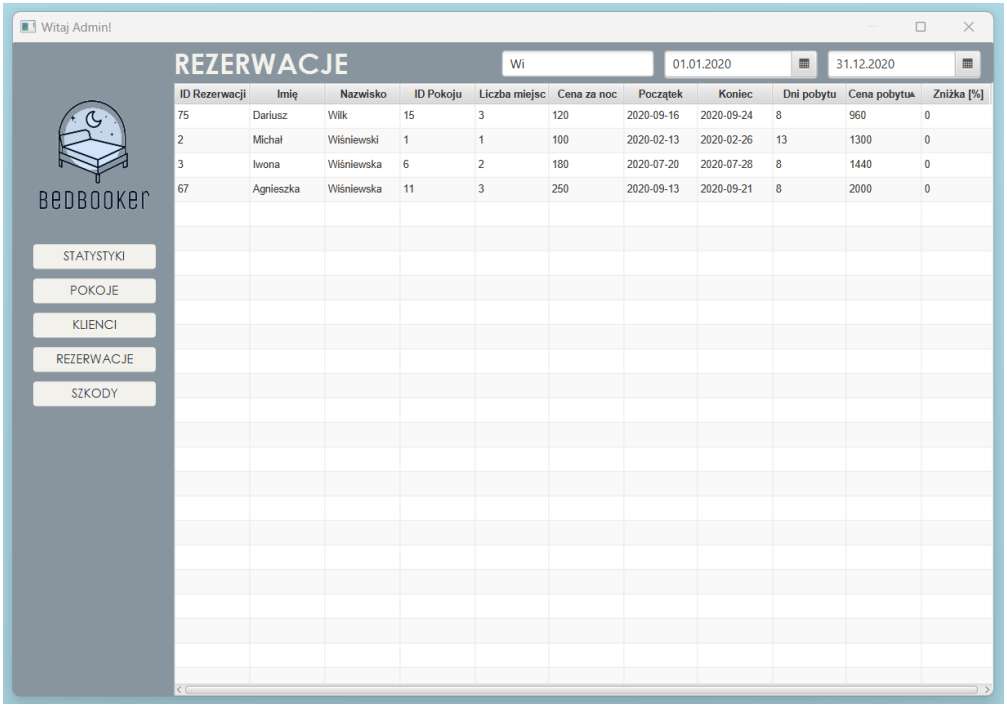
Rezerwacje

Widok na wszystkie hotelowe rezerwacje.



ID Rezerwacji	Imię	Nazwisko	ID Pokoju	Liczba miejsc	Cena za noc	Początek	Koniec	Dni pobytu	Cena pobytu	Zniżka [%]
2	Michał	Wiśniewski	1	1	100	2020-02-13	2020-02-26	13	1300	0
21	Krzysztof	Kowalski	1	1	100	2021-05-26	2021-05-29	3	300	0
45	Mariusz	Szymański	1	1	100	2024-09-15	2024-09-19	4	400	0
50	Anna	Kowalska	1	1	100	2022-12-30	2023-01-11	12	1200	0
58	Tomasz	Wójcik	1	1	100	2023-12-04	2023-12-09	5	450	10
17	Rafał	Wilk	2	2	120	2025-03-24	2025-04-06	13	1560	0
40	Jan	Kowalski	2	2	120	2023-10-02	2023-10-11	9	1080	0
42	Tomasz	Wójcik	2	2	120	2023-01-11	2023-01-22	11	1201	9
56	Marta	Kowalczyk	2	2	120	2022-08-04	2022-08-14	10	1200	0
61	Halina	Kowalik	2	2	120	2024-06-02	2024-06-07	5	600	0
71	Marzena	Nowak	2	2	120	2021-04-26	2021-05-11	15	1800	0
78	Piotr	Dąbrowski	2	2	120	2022-12-23	2022-12-31	8	960	0
7	Adam	Wilk	3	3	150	2021-09-13	2021-09-21	8	1200	0
12	Krzysztof	Kwiatkowski	3	3	150	2023-09-30	2023-10-01	1	150	0
39	Piotr	Wojciechowski	3	3	150	2020-06-26	2020-07-09	13	1950	0
62	Tomasz	Wójcik	3	3	150	2020-05-30	2020-06-10	11	1485	10
6	Andrzej	Lewandowski	4	4	200	2024-05-19	2024-06-02	14	2800	0
63	Alicja	Jankowska	4	4	200	2023-03-05	2023-03-08	3	600	0
77	Marta	Kowalczyk	4	4	200	2025-01-20	2025-02-03	14	2800	0
93	Ewa	Jankowska	4	4	200	2023-11-08	2023-11-21	13	2600	0
8	Izabela	Nowak	5	1	150	2022-01-20	2022-01-23	3	450	0
18	Ewa	Jankowska	5	1	150	2023-06-02	2023-06-04	2	300	0
35	Marzena	Nowak	5	1	150	2021-05-20	2021-05-28	8	1200	0
55	Marzena	Nowak	5	1	150	2023-02-20	2023-02-26	6	900	0

Można wyszukiwać je po nazwisku klienta, po dacie pobytu oraz sortować każdą kolumnę.

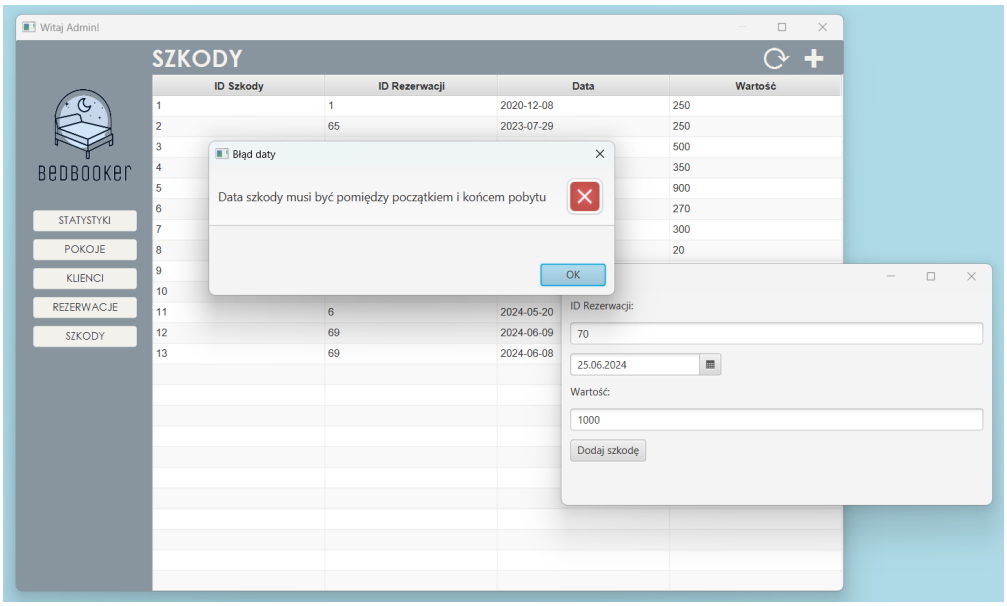


ID Rezerwacji	Imię	Nazwisko	ID Pokoju	Liczba miejsc	Cena za noc	Początek	Koniec	Dni pobytu	Cena pobytu	Zniżka [%]
75	Dariusz	Wilk	15	3	120	2020-09-16	2020-09-24	8	960	0
2	Michał	Wiśniewski	1	1	100	2020-02-13	2020-02-26	13	1300	0
3	Iwona	Wiśniewska	6	2	180	2020-07-20	2020-07-28	8	1440	0
67	Agnieszka	Wiśniewska	11	3	250	2020-09-13	2020-09-21	8	2000	0

```
CREATE VIEW ReservationDetails AS
SELECT R.ReservationID                                  AS ReservationID,
       P.Name                                           AS PersonName,
       P.Surname                                        AS PersonSurname,
       Ro.RoomID                                        AS RoomID,
       Ro.NumberOfPlaces                                AS NumberOfPlaces,
       Ro.Price                                         AS RoomPrice,
       R.StartDate                                      AS StartDate,
       R.EndDate                                        AS EndDate,
       (to_days(R.EndDate) - to_days(R.StartDate))      AS NumberOfDays,
       R.Price                                          AS ReservationPrice,
       R.Discount                                       AS Discount
FROM Reservations R JOIN Persons P on R.PersonID = P.PersonID
                    JOIN Rooms Ro on R.RoomID = Ro.RoomID;
```

Filtry nakładano poleceniem WHERE na tym widoku.

Szkody

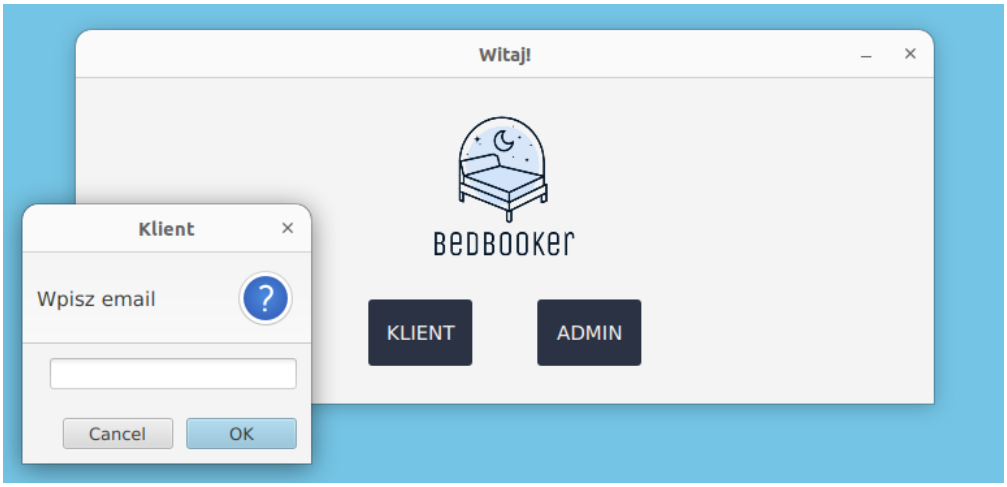


Panel szkód działa podobnie do panelu Pokoje. Również można dodawać nowe szkody, a następnie odświeżać listę. Jednak tym razem należało wprowadzić kontrolę wprowadzanych danych.

```
CREATE PROCEDURE AddDamage(IN p_ReservationID int, IN p_Date date,
                           IN p_Price int)
BEGIN
    DECLARE v_StartDate DATE;
    DECLARE v_EndDate DATE;

    SELECT StartDate, EndDate INTO v_StartDate, v_EndDate
    FROM Reservations
        WHERE ReservationID = p_ReservationID;

    IF v_StartDate IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Nie istnieje rezerwacja o podanym ReservationID';
    END IF;

    IF p_Date < v_StartDate OR p_Date > v_EndDate THEN
        SIGNAL SQLSTATE '45001'
        SET MESSAGE_TEXT = 'Data szkody musi być pomiędzy StartDate i EndDate
rezerwacji';
    END IF;

    INSERT INTO Damages (ReservationID, Date, Price)
    VALUES (p_ReservationID, p_Date, p_Price);

END;
```

Widok klienta

Widok klienta zawiera infomacje interesujące konkretnego klienta. Funckjonalności z wyjątkiem logowania zostaną omówione na przykładzie zalogowanego użytkownika.

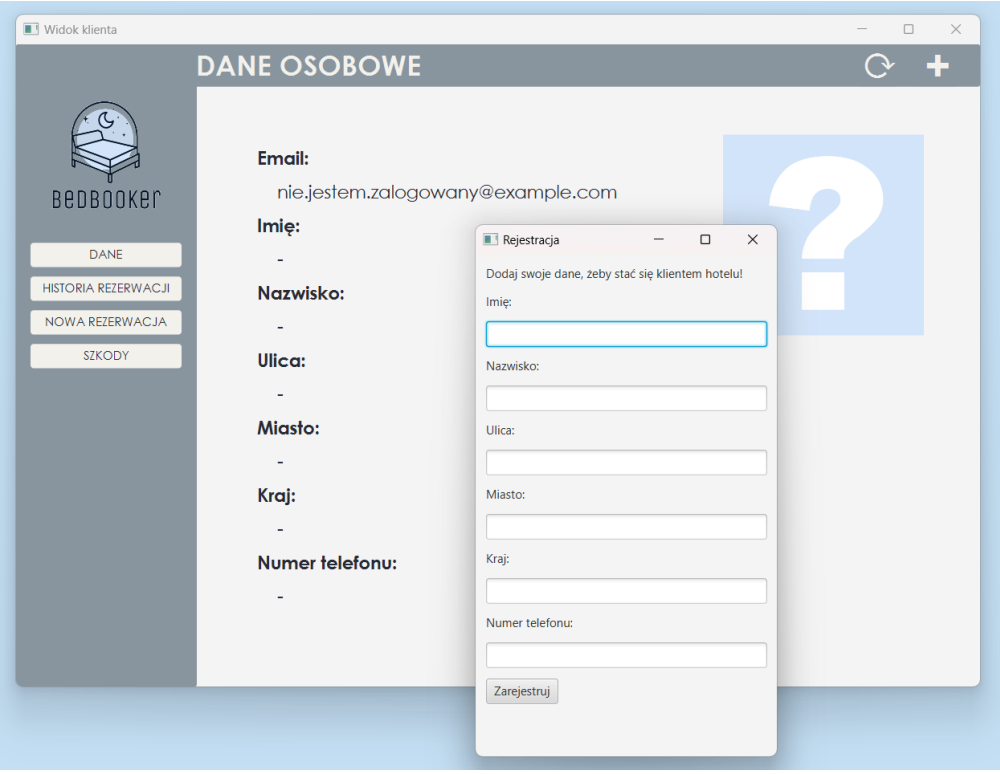


Logowanie do systemu

Logowanie do systemu zostało uproszczone i odbywa się jedynie poprzez podanie prawidłowego adresu e-mail (czyli takiego, który występuje w bazie danych). Możemy również podać e-mail, którego nie ma w bazie- wtedy jesteśmy traktowani jako niezalogowani.

Bez logowania

Jeżeli podamy e-mail nieistniejący w bazie, utracimy dostęp do pewnych funckjonalności. Jednak w zakładace Dane dostaniemy możliwość utworzenia konta.

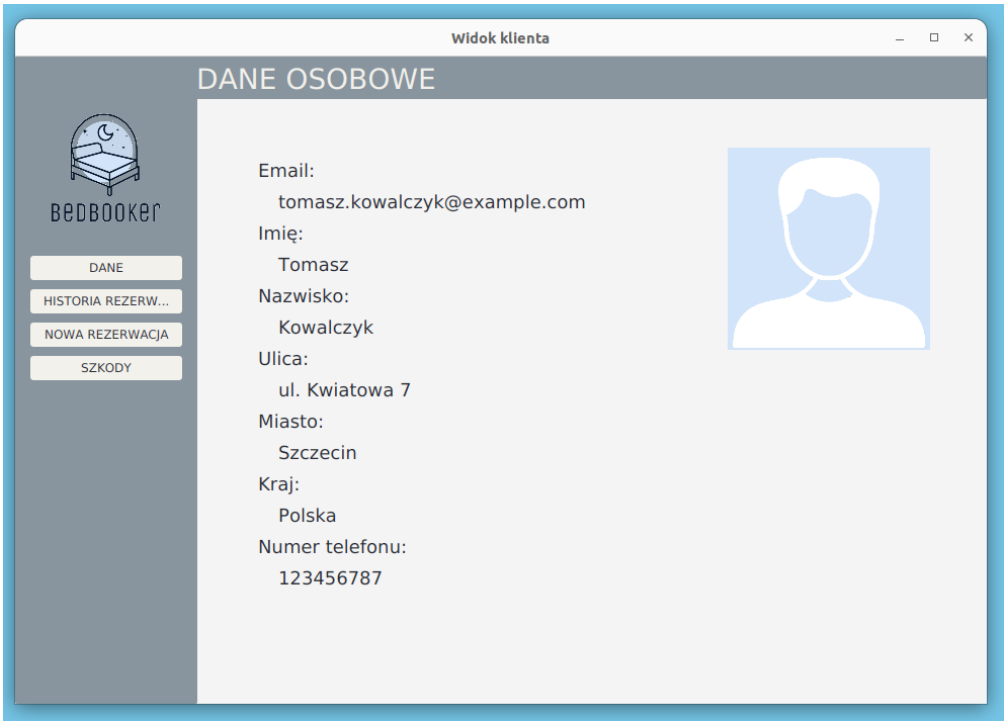


Jest to realizowane przez funckję `AddCustomer`:

```
create procedure AddCustomer(IN CustomerName varchar(20), IN CustomerSurname
varchar(20),
                                                    IN CustomerStreetAddress
varchar(50),
                                                    IN CustomerCityName
varchar(20),
                                                    IN CustomerCountryName
varchar(20),
                                                    IN CustomerPhoneNumber
varchar(10),
                                                    IN CustomerEmail varchar(30))
BEGIN
    DECLARE CityID INT;
    DECLARE CountryID INT;

    IF NOT EXISTS (SELECT 1 FROM Countries WHERE CountryName =
CustomerCountryName) THEN
        INSERT INTO Countries (CountryName) VALUES (CustomerCountryName);
    END IF;

    SELECT CountryID INTO CountryID FROM Countries WHERE CountryName =
CustomerCountryName;

    IF NOT EXISTS (SELECT 1 FROM Cities WHERE CityName = CustomerCityName) THEN
        INSERT INTO Cities (CityName) VALUES (CustomerCityName);
    END IF;

    SELECT CityID INTO CityID FROM Cities WHERE CityName = CustomerCityName;

    INSERT INTO Persons (Name, Surname, StreetAddress, CityID, CountryID,
PhoneNumber, Email)
    VALUES (CustomerName, CustomerSurname, CustomerStreetAddress, CityID,
CountryID,
            CustomerPhoneNumber, CustomerEmail);
END;
```

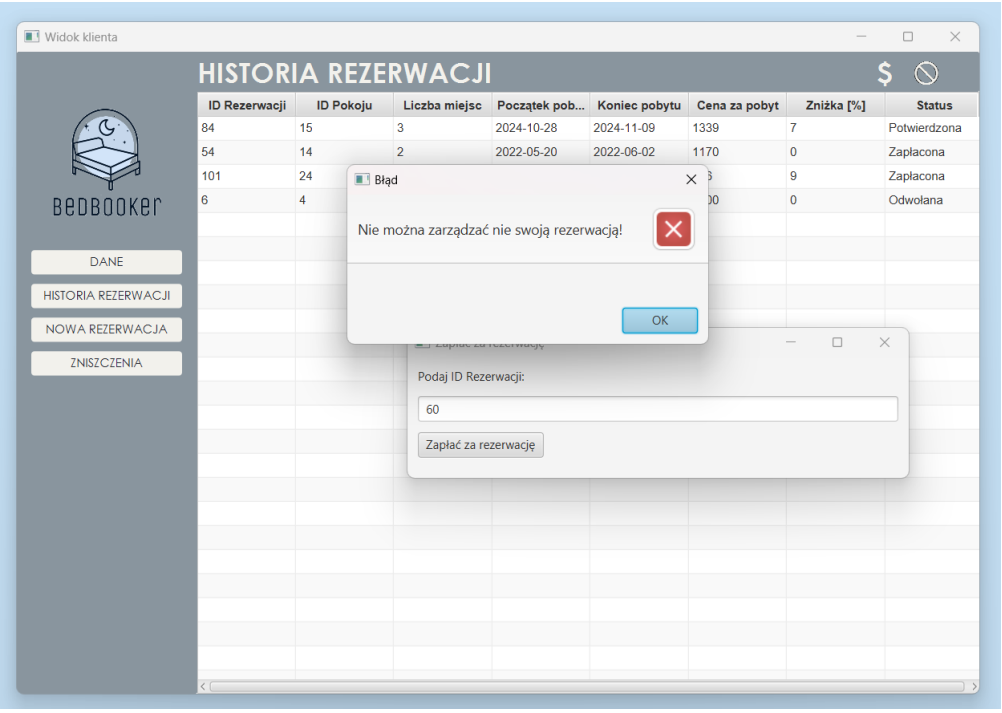
Poprawny e-mail

Zalogujmy się jako Tomasz Kowalczyk (tomasz.kowalczyk@example.com). Po zalogowaniu widzimy nasze dane.



Historia rezerwacji

Historia rezerwacji pozwala klientowi przeglądać swoje rezerwacje oraz zarządzać ich statusem- płacić i odwoływać.



System kontroluje, czy Klient aby na pewno podaje ID swojej Rezerwacji. Kontroluje również czy Klient nie próbuje zapłacić za rezerwację odwołaną- taka transakcja jest przerywana.

Płatność obsługuje następująca procedura:

```
CREATE PRODEDURE PayForReservation(IN v_reservationId int)
BEGIN
    DECLARE currentStatus INT;

    SELECT StatusID INTO currentStatus
    FROM Reservations
    WHERE ReservationID = v_reservationId;

    IF currentStatus = 4 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Nie można zapłacić za odwołaną rezerwację';
    ELSE
        CALL UpdateReservationStatus(v_reservationId, 3);
    END IF;
END;
```

Odwoływanie:

```
CREATE PROCEDURE CancelReservation(IN v_reservationId int)
BEGIN
    CALL UpdateReservationStatus(v_reservationId, 4);
END;
```

Funkcja aktualizująca status podanej rezerwacji uzupełnia również tabelę Logs:

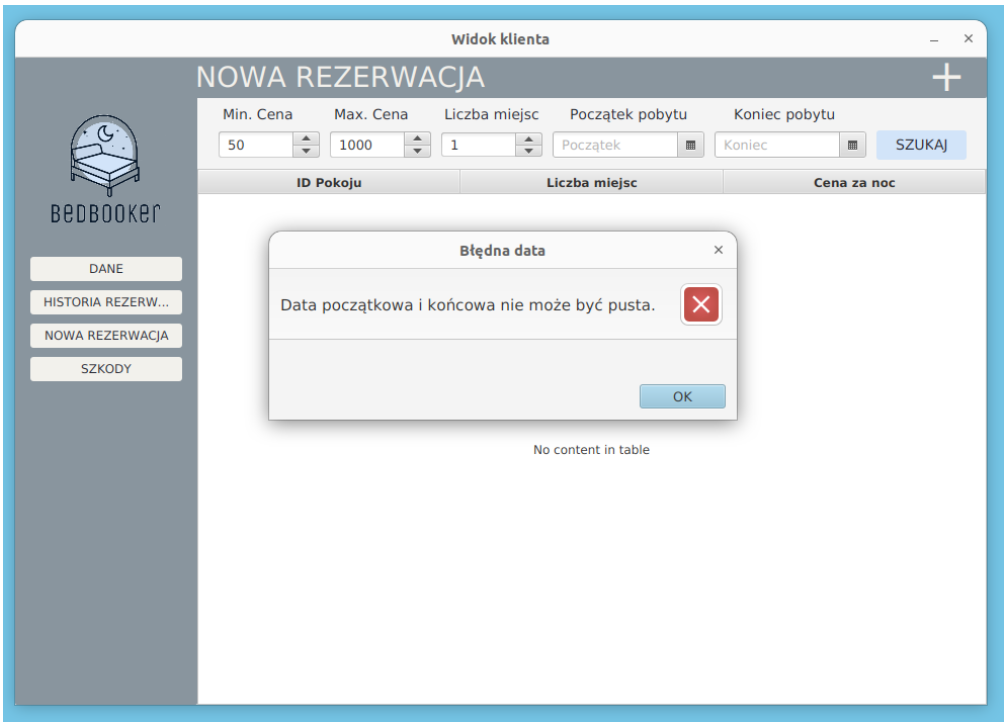
```
CREATE PROCEDURE UpdateReservationStatus(IN v_reservationId int, IN newStatusId
int)
BEGIN
    UPDATE Reservations
    SET StatusID = newStatusId
    WHERE ReservationID = v_reservationId;

    INSERT INTO Logs(ReservationID, StatusID, DateTime)
    VALUES (v_reservationId, newStatusId, NOW());
END;
```

Nowa rezerwacja

Możemy ustawić minimalną i maksymlną kwotę za noc jak i liczbę miejsc, która nas interesuje.

Osoba niezalogowana może jedynie przeglądać wolne pokoje, nie może jednak uworzyć rezerwacji.

Nie możemy dodać rezerwacji przed wybraniem daty początkowej i końcowej.



Po wypełnieniu dat, możemy kliknąć przycisk `Szukaj`.

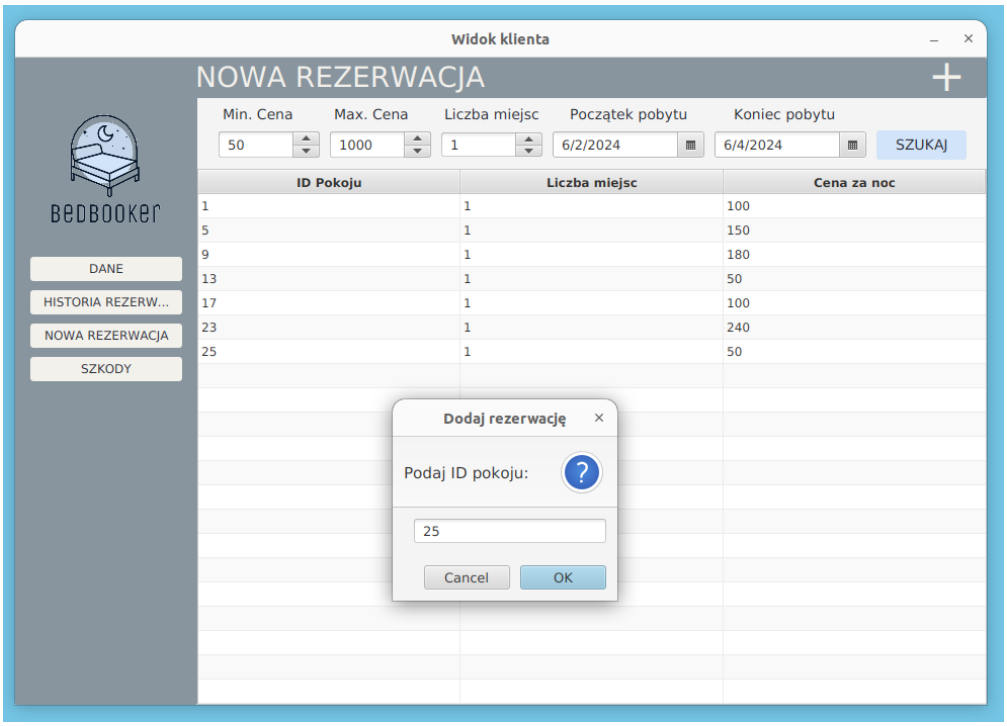
Otrzymujemy dostępne pokoje zgodne z podanymi kryteriami. Korzystamy tutaj z procedury `AvaiableRooms`:

```
create procedure AvailableRooms(IN StartDate date, IN EndDate date, IN
NumberOfPlaces int,
                                                IN MinPrice int, IN MaxPrice
int)
BEGIN
    SELECT r.RoomID, r.NumberOfPlaces, r.Price
    FROM Rooms r
    WHERE r.NumberOfPlaces = NumberOfPlaces
        AND r.Price BETWEEN MinPrice AND MaxPrice
        AND IsRoomAvailable(r.RoomID, StartDate, EndDate) = 1;
END;
```

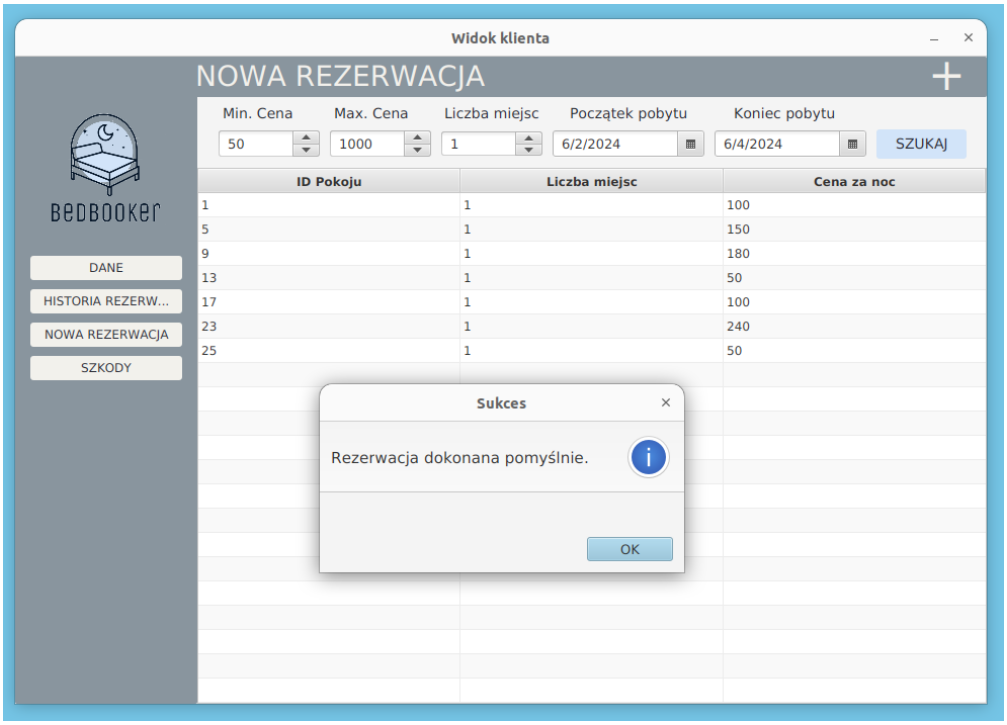
Ta procedura korzysta natomiast z procedury `IsRoomAvailable`, która szuka pokoi, które nie posiadają rezerwacji w danym terminie (mowa o nieodwołanych rezerwacjach).

```
create function IsRoomAvailable(RoomIDParam int, StartDateParam date, EndDateParam
date)
    returns int
BEGIN
    DECLARE RoomCount INT;

SELECT COUNT(*) INTO RoomCount
FROM Reservations
WHERE RoomID = RoomIDParam
  AND StartDate <= EndDateParam
  AND EndDate >= StartDateParam
  AND StatusID <> 4;

IF RoomCount > 0 THEN
        RETURN 0;
ELSE
        RETURN 1;
END IF;
END;
```

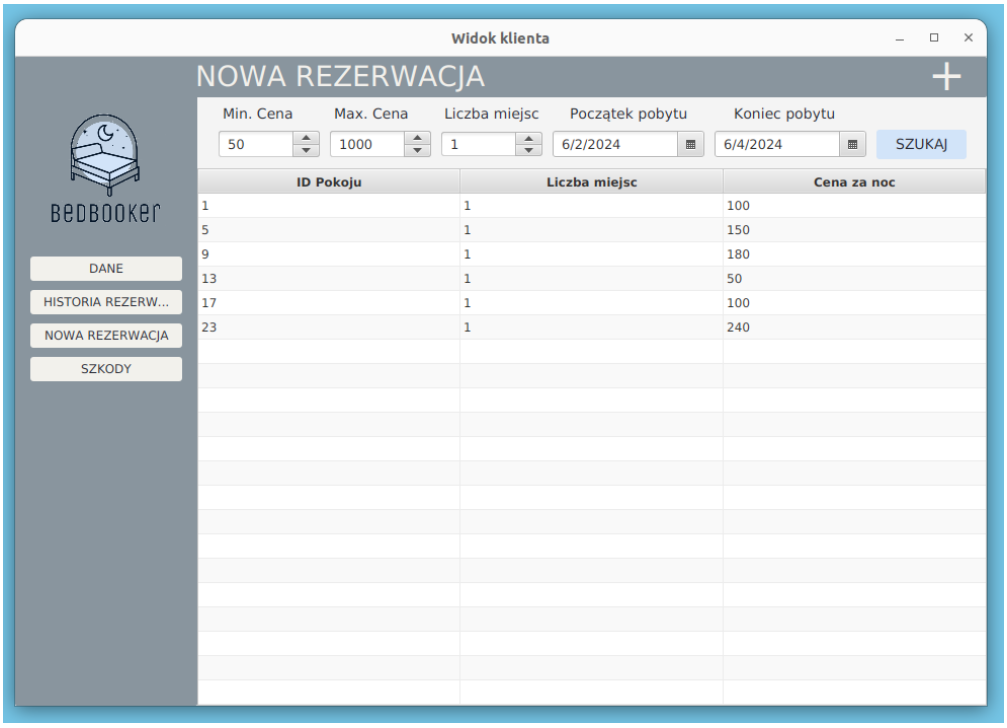
Możemy teraz dodać nową rezerwację. Zostaniemy poproszeni o ID interesującego nas pokokju:



Wybierzmy pokój 25. Po zatwierdzeniu, dostajemy komunikat o dodanej rezerwacji.



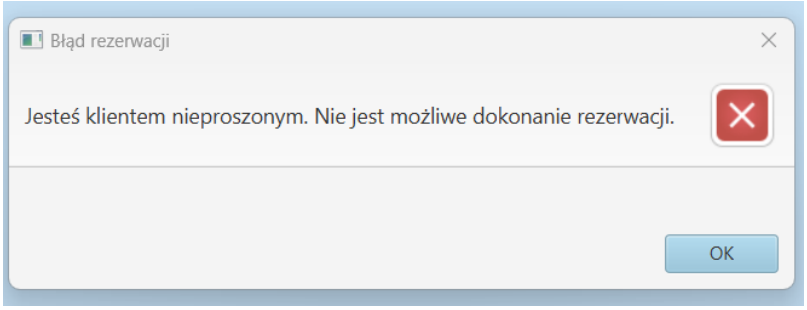
A na koniec widok dostępnych pokoi w danym terminie się aktualizuje:



Możemy również zobaczyć, że nowa rezerwacja pojawiła się w bazie danych:

	ReservationID	PersonID	RoomID	StartDate	EndDate	StatusID	Price	Discount
94	94	38	14	2021-07-23	2021-08-03	1	990	0.00
95	95	47	13	2022-01-30	2022-02-14	1	750	0.00
96	96	21	17	2024-09-22	2024-09-25	1	300	0.00
97	97	11	6	2020-06-01	2020-06-15	1	2520	0.00
98	98	31	15	2023-07-08	2023-07-17	1	994	8.00
99	99	5	12	2024-03-10	2024-03-20	1	2790	7.00
100	100	1	13	2024-06-25	2024-06-27	1	100	0.00
101	101	3	24	2024-06-17	2024-06-18	1	546	9.00
102	102	13	25	2024-06-04	2024-06-06	1	100	0.00
103	103	13	23	2024-06-04	2024-06-06	1	480	0.00
104	104	13	25	2024-06-12	2024-06-13	1	50	0.00
105	105	51	9	2024-06-19	2024-06-22	1	540	0.00
106	106	51	7	2024-06-19	2024-06-20	1	200	0.00
107	107	7	25	2024-06-02	2024-06-04	1	100	0.00

Jeżeli ktoś, jest na liście klientów nieproszonych, nie ma możliwości złożenia rezerwacji.



Szkody

Logując się przykładowo jako Andrzej Lewandowski (andrzej.lewandowski@example.com), gdy wejdziemy w zakładki `Szkody` zobaczymy wszystkie szkody spowodowane przez Andrzeja.

Korzystamy tutaj procedury `GetDamagesForPerson`:

```
create procedure GetDamagesForPerson(IN personID int)
BEGIN
    SELECT d.*
    FROM Damages d
             JOIN Reservations r ON d.reservationID = r.reservationID
    WHERE r.personID = personID;
END;
```

Dla osób nie mających szkód, pokazana zostanie pusta tabela: